Powered by Linear Algebra

The central role of matrices and vector spaces in Data Science

Norman Matloff

2025-02-27

Table of contents

1

Preface

Welcome to my magnum opus! :-) I've written a number of books, but consider this one to be the most important.

Subtlety in the Title

Let's start with the title of this book, "Linear Algebra *Rules* Data Science." It would be more typical of "math support for field X" books to use "for" rather than "in," but the use of the latter aims to emphasize the fact that:

Linear algebra is absolutely fundamental to the Data Science field. For us data scientists, it is "our" branch of math. Mastering this branch, which is definitely within the reach of all, pays major dividends.

Philosophy

This is an unconventional linear algebra textbook, doing everything "backwards." The presentation of each concept begins with a problem to be solved, almost always from Data Science, leading up to a linear algebra solution. Basically, the math sneaks up on the reader, who suddenly realizes they've just learned a new general concept!

Or, the reader suddenly sees a connection to a familiar topic. For instance, after introducing the Cauchy-Schwarz Inequality, the book shows that it implies our familiar notion that statistical correlation is bounded between -1 and 1.

In an excellent review of book on Principal Components Analysis, the reviewer writes

I bought the acclaimed *Linear Algebra Done Right*...A classic...It was rather dull, one definition after another. It is not difficult

to follow the logic, and to see how one theorem follows from another. But…it felt random. "Follow these steps and you get the result you are looking for. I can prove that it works because of the following …". But why does it work? What possessed you to follow this path as opposed to any other? How might I have come up with this myself? The book provided no answers and I was left feeling deeply unsatisfied.

The fundamental philosophy of my book here is to avoid such reader frustration.

Who is this book for?

Of course the book should work well as a classroom textbook. The "applications first" approach should motivate students, and the use of Quarto enables easy conversion to Powerpoint by instructors.

But I also hope the book's emphasis on the Why? and How? especially appeals to do-it-yourselfers, those whose engagement in self-study is motivated by intellectual curiosity rather than a course grade.

Prerequisite background

Basic data science:

- Calculus.
- Some exposure to R is recommended, but the text can be read without it.
- Basics of random variables, expected value and variance.

For a quick, painless introduction to R, see my fasteR tutorial, say the first 8 lessons.

The role of theory

> ! This book is "mathematical but not very theoretical."
>
> Theorems are mainly limited to results with practical importance, and proofs are sometimes rather informal. But the subject matter is indeed mathematical. The goal is to develop in the reader mathematical skill and intuition into this powerful tool, rather than coding of linear algebra methods.

R Libraries Used

- **dsld**
- **igraph**
- **lars**
- **networkdata**
- **pracma**
- **qeML**

2 Matrix Multiplication

i Goals of this chapter:

The two main structures in linear algebra are *matrices* and *vector speaces*. We begin the book with the former, introduced in this chapter, inroducing matrix multiplication and presenting several applications.

2.1 A Random Walk Model

Let's consider a *random walk* on {1,2,3,4,5} in the number line. Time is numbered 1,2,3,... Our current position is termed our *state*. The notation $X_k = i$ means that at time k we are in state/position i.

Our rule will be that at any time k, we flip a coin. If we are currently at position i, we move to either i+1 or i-1, depending on whether the coin landed heads or tails. The exceptions are $k = 1$ and $k = 5$, in which case we stay put if tails or move to the adjacent position if heads.

We can summarize the probabilities with a *matrix*, a two-dimensional array:

$$P_1 = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{pmatrix}$$

For instance, look at Row 2. There are 0.5 values in Columns 1 and 3, meaning there is a 0.5 chance of a move $2 \to 1$, and a 0.5 chance of a move $2 \to 3$.

Note that each row in a transition matrix must sum to 1. After, from state i we must go *somewhere*.

We use a subscript 1 here in P_1, meaning "one step." We go from, say, state 2 to state 1 in one step with probability 0.5. P_1 is called the *one-step transition matrix* (or simply the *transition matrix*) for this process.

What about the two-step transition matrix P_2? From state 3, we could go to state 1 in two steps, by two tails flips of the coin. The probability of that is $0.5^2 = 0.25$. So the row 3, column

1 element in P_2 is 0.25. On the other hand, if from state 3 we flip tails then heads, or heads then tails, we are back to state 3. So, the row 3, column 3 element in P_2 is 0.25 + 0.25 = 0.5.

The reader should verify the correctness here:

$$P_2 = \begin{pmatrix} 0.5 & 0.25 & 0.25 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 \\ 0.25 & 0 & 0.5 & 0 & 0.25 \\ 0 & 0.25 & 0 & 0.5 & 0.25 \\ 0 & 0 & 0.25 & 0.25 & 0.5 \end{pmatrix}$$

Well, finding two-step transition probabilities would be tedious in general, but it turns out that is a wonderful shortcut: Matrix multiplication. We will cover this in the next section, but first a couple of preliminaries.

The above random walk is a *Markov chain*. The Markov Property says that the system "has no jemory." If say we land at position 2, we will go to 1 or 3 with probability 1/2 *no matter what the previous history of the system was*; it doesn't matter *how* we got to state 3. That in turn comes from the independence of the successive coin flips.

Notation: Individual elements of a matrix are usually written with double subscripts. For instance, a25 will mean the row 2, column 5 element of the matrix A.

2.2 Matrix Multiplication

This is the most fundamental operation in linear algebra. It is defined as follows:

Given matrix A of k rows and m columns and matrix B of m rows and r columns, the product C = AB is an kxm matrix, whose row i, column j element is

$$a_{i1}b_{i1} + a_{i2}b_{i1} + ... + a_{m1}b_{m1}$$

This is the "dot product" of row i of A and column j of B: Find the products of the paired elements in the two vectors, then sum.

For example, set

$$A = \begin{pmatrix} 5 & 2 & 6 \\ 1 & 1 & 8 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 5 & -1 \\ 1 & 0 \\ 0 & 8 \end{pmatrix}$$

Let's find the row 2, column 2 element of C = AB. Again, that means taking the dot product of row 2 of A and column 2 of B, which we've highlighted below.

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

and

$$B = \begin{pmatrix} 5 & -1 \\ 1 & 0 \\ 0 & 8 \end{pmatrix}$$

The value in question is then

1 (-1) + 1 (0) + 1 (8) = 7

Let's check it, with R:

```
a <- rbind(c(5,2,6),c(1,1,1))
b <- cbind(c(5,1,0),c(-1,0,8))
a %*% b
```

Be sure to remember that the number of rows of B must match the number of columns of A. In this case, we say thst the matrices are *conformable*.

.The **rbind** and **cbind** functions ("row bind" and "column bind") are very handy tools for creating matrices.

```
     [,1] [,2]
[1,]   27   43
[2,]    6    7
```

The reader should make sure to check the other elements by hand.

> **Tip**
>
> Always keep in mind that in the matrix product AB, the number of rows of B must equal the number of columns of A. The two matrices are then said to be *conformable*.

2.3 The Identity Matrix

The *identity matrix* I of size n is the nxn matrix with 1s on the diagonal and 0s elsewhere. $IB = B$ and $AI = A$ for any conformable A and B.

2.4 Vectors

A matrix that has only one row or only one column is called a *vector*. Depending on which of those two shapes it has, we may refer to it as a *row vector* or *column vector*. Usually we will simply say "vector," in which case it will be meant as a column vector.

2.5 Application to Markov Chain Transition Matrices

Now let's return to the question of how to easily compute P_2, the two-step transition matrix. It turns out that:

Let P denote the transition matrix of a (finite-state) Markov chain. The k-step transition matrix is P^k.

At first, this may seem amazingly fortuitous, but it makes sense in light of the "and/or" nature of the probability computations involved. Recall our computation for the row 1, column 2 element of P_2 above. From state 1, we could either stay at 1 for one flip, then move to 2 on the second flip, or we could go to 2 then return to 1. Each of these has probability 0.5, so the total probability is

$$(0.5)(0.5) + (0.5)(0.5)$$

But this is exactly the form of our "dot product" computation in the definition of matrix multiplication,

$$a_{i1}b_{i1} + a_{i2}b_{i1} + ... + a_{m1}b_{m1}$$

Statisticians and computer scientists like to look at the *asymptotic* behavior of systems. Let's see where we might be after say, 6 steps:

```
matpow <- function(m,k) {
   nr <- nrow(m)
   tmp <- diag(nr)  # identity matrix
   for (i in 1:k) tmp <- tmp %*% m
   tmp
}

p1 <- rbind(c(0.5,0.5,0,0,0), c(0.5,0,0.5,0,0), c(0,0.5,0,0.5,0),
   c(0,0,0.5,0,0.5), c(0,0,0,0.5,0.5))
matpow(p1,6)
```

```
        [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 0.312500 0.234375 0.234375 0.109375 0.109375
[2,] 0.234375 0.312500 0.109375 0.234375 0.109375
[3,] 0.234375 0.109375 0.312500 0.109375 0.234375
[4,] 0.109375 0.234375 0.109375 0.312500 0.234375
[5,] 0.109375 0.109375 0.234375 0.234375 0.312500
```

So for instance if we start at position 2, there is about an 11% chance that we will be at position 3 at time 6. What about time 25?

```
matpow(p1,25)
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.2016179 0.2016179 0.1993820 0.1993820 0.1980001
[2,] 0.2016179 0.1993820 0.2016179 0.1980001 0.1993820
[3,] 0.1993820 0.2016179 0.1980001 0.2016179 0.1993820
[4,] 0.1993820 0.1980001 0.2016179 0.1993820 0.2016179
[5,] 0.1980001 0.1993820 0.1993820 0.2016179 0.2016179
```

So, no matter which state we start in, at time 25 we are about 20% likely to be at any of the states. In fact, as time n goes to infinity, this probability vector becomes exactly (0.20,0.20,0.20,0.20,0.20), as we will see in the next chapter.

2.6 Network Graph Models

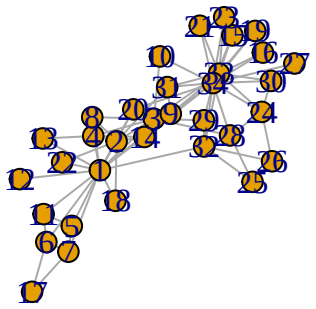
There has always been lots of analyses of "Who is connected to who," but activity soared after the advent of Facebook and the film, *A Social Network.* See for instance *Statistical Analysis of Network Data with R* by Eric Kolaczy and Gábor Csárdi. As the authors say,

The oft-repeated statement that "we live in a connected world" perhaps best captures, in its simplicity…From on-line social networks like Facebook to the World Wide Web and the Internet itself, we are surrounded by examples of ways in which we interact with each other. Similarly, we are connected as well at the level of various human institutions (e.g., governments), processes (e.g., economies), and infrastructures (e.g., the global airline network). And, of course, humans are surely not unique in being members of various complex, inter-connected systems. Looking at the natural world around us, we see a wealth of examples of such systems, from entire eco-systems, to biological food webs, to collections of inter-acting genes or communicating neurons.

And of course, at the center of it all is a matrix! Here is why:

Let's consider the famous Karate Club dataset:

```
# remotes::install_github("schochastics/networkdata")
library(networkdata)
data(karate)
library(igraph)
plot(karate)
```



There is a link between node 13 and node 4, meaning that club members 13 and 4 are friends.

Specifically, the *adjacency matrix* has row i, column j element as 1 or 0, according to whether a link exists between nodes i and j.

This graph is *undirected*, as friendship is mutual. Many graphs are *directed*, but we will assume undirected here.

```
adjK <- as_adjacency_matrix(karate)
adjK
```

```
34 x 34 sparse Matrix of class "dgCMatrix"

 [1,] . 1 1 1 1 1 1 1 1 . 1 1 1 1 . . . 1 . 1 . 1 . . . . . . . . . 1 . .
 [2,] 1 . 1 1 . . . 1 . . . . . 1 . . . 1 . 1 . 1 . . . . . . . . 1 . . .
 [3,] 1 1 . 1 . . . 1 1 1 . . . 1 . . . . . . . . . . . . . . . 1 1 . . . 1 .
 [4,] 1 1 1 . . . . 1 . . . . 1 1 . . . . . . . . . . . . . . . . . . . . .
 [5,] 1 . . . . . 1 . . . 1 . . . . . . . . . . . . . . . . . . . . . . . .
 [6,] 1 . . . . . 1 . . . 1 . . . . . 1 . . . . . . . . . . . . . . . . . .
 [7,] 1 . . . 1 1 . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . .
 [8,] 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

```
 [9,] 1 . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . 1 1
[10,] . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
[11,] 1 . . . 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
[12,] 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
[13,] 1 . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
[14,] 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
[15,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1
[16,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1
[17,] . . . . . 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . .
[18,] 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
[19,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1
[20,] 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
[21,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1
[22,] 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
[23,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1
[24,] . . . . . . . . . . . . . . . . . . . . . . . . . 1 . 1 . 1 . . 1 1
[25,] . . . . . . . . . . . . . . . . . . . . . . . . . 1 . 1 . . . 1 . .
[26,] . . . . . . . . . . . . . . . . . . . . . . . 1 1 . . . . . . 1 . .
[27,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1
[28,] . . 1 . . . . . . . . . . . . . . . . . . . . 1 1 . . . . . . . . 1
[29,] . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . 1
[30,] . . . . . . . . . . . . . . . . . . . . . . . 1 . . 1 . . . . . 1 1
[31,] . 1 . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . 1 1
[32,] 1 . . . . . . . . . . . . . . . . . . . . . . . 1 1 . . 1 . . . 1 1
[33,] . . 1 . . . . . 1 . . . . . 1 1 . . 1 . 1 . 1 1 . . . . . 1 1 1 . 1
[34,] . . . . . . . . 1 1 . . . 1 1 1 . . 1 1 1 . 1 1 . . 1 1 1 1 1 1 1 .
```

```
adjK[13,4]
```

```
[1] 1
```

Accordingly, row 13, column 4 does have a 1 entry.

As is the case with Markov transition matrices, powers of an adjacency matrix can yield valuable information. In the Markov case, multiplication gives us sums of paired products, computing probabilities. What about the network graph case?

Here products are of the form 0x0, 0x1, 1x0 or 1x1. If there is a nonzero entry m in row i, column j of the square of the

adjacency matrix, that means there were m 1x1 products in that sum, which would correspond to m paths. Let's look into this.

```
adjK2 <- adjK %*% adjK
```

We see that **adjK2[11,1]** is 2. Inspection of **adjK** shows that its row 11, columns 6 and 7 are 1s, and that rows 6 and 7, column 1 are 1s as well. So there are indeed two two-hop paths from node 11 to node 1, specifically 11 → 6 → 1 and $ 11 → 7 → 1.

Actually, what is typically of interest is *connectivity* rather than number of paths. For any given pair of nodes, is there a multihop path between them? Or does the graph break down to several "islands" of connected nodes?

Again consider the karate club data.

```
u <- matpow(adjK,33)
sum(u == 0)
```

```
[1] 0
```

So, in this graph, no pair of nodes has 0 paths between them. The graph is connected.

Making this kind of analysis fully correct requires paying attention to things such as cycles. The details are beyond the scope of this book.

2.7 Recommender Systems

If you inquire about some item at an online store, the software will also present you with some related items that it thinks would be of interest to you. How does the software make this guess?

Clearly, the full answer is quite complex. But we can begin to see the process by looking at some real data.

```
site <- 'http://files.grouplens.org/datasets/movielens/ml-100k/u.data'
q <- read.table(site)
names(q) <- c('user','movie','rating','userinfo')
head(q)
```

```
  user movie rating  userinfo
1  196   242      3 881250949
2  186   302      3 891717742
3   22   377      1 878887116
4  244    51      2 880606923
5  166   346      1 886397596
6  298   474      4 884182806
```

We see for instance that user 22 gave movie 242 a rating of 1. If we want to know some characteristics of this user, his/her ID is 878887116, which we can find in the file **u.user** at the above URL. Other files tell us more about this movie, e.g. its genre, and so on.

Let's explore the data a bit:

How many users and movies are in this dataset?

```
length(unique(q$user))
```

```
[1] 943
```

```
length(unique(q$movie))
```

```
[1] 1682
```

How many other users rated movie number 242?

```
sum(q$movie == 242)
```

```
[1] 117
```

Did user 22 rate movie 234, for instance?

```
which(q$user == 22 & q$movie == 234)
```

```
integer(0)
```

Now we can begin to see a solution to the recommender problem. Say we wish to guess whether user 22 would like movie 234. We could look for other users who have rated many of the same movies as user 22, then focus on the ones who rated movie 234. We could average those ratings to obtain a predicted rating for movie 234 by user 22.

We could also incorporate the characteristics of user 22 and the others, which may improve our prediction accuracy, but we will not pursue that here.

In order to assess interuser similarity of the nature described above, we might form a matrix S, as follows. There would be 943 rows, one for each user, and 1682 columns, one for each moview. The element in row i, column j would be the rating user i gave to movie j. Most of the matrix would be 0s.

The point of constructing S is that determining the similarity of users becomes a matter of measuring similarity of rows of S. This paves the way to exploiting the wealth of matrix-centric methodology we will develop in this book.

Your Turn: Write a function with call form

```
makeSimilarityMatrix(m)
```

that takes as input a matrix or data frame of user-movie-rating data, and returns a matrix as described above, i.e. one row per user, one column per movie, and so on.

2.8 Matrix Algebra

2.8.1 Other basic operations

Matrix multiplication may seem odd at first, but other operations are straightforward.

Addition: We just add corresponding elements. For instance,

$$A = \begin{pmatrix} 5 & 2 & 6 \\ 1 & 2.6 & -1.2 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 20 & 6 \\ 3 & 5.8 & 1 \end{pmatrix}$$

$$A + B = \begin{pmatrix} 5 & 22 & 12 \\ 4 & 8.4 & -0.2 \end{pmatrix}$$

We do have to make sure the addends match in terms of numbers of rows and columns, 2 and 3 in the example here.

Scalar multiplication: Again, this is simply elementwise. E.g. with A as above,

$$1.5A = \begin{pmatrix} 7.5 & 3 & 9 \\ 1.5 & 3.9 & -1.8 \end{pmatrix}$$

Distributive property:

For matrices A, B and C of suitable conformability (A and B match in numbers of rows and columns, and their common number of columns matches the number of rows in C), we have

(A+B) C = AC + BC

2.8.2 Matrix transpose

This is a very simple but very important operation: We merely exchange rows and columns of the given matrix. For instance, with A as above, its transpose (signified with "'"), is

$$A' = \begin{pmatrix} 5 & 1 \\ 2 & 2.6 \\ 6 & -1.2 \end{pmatrix}$$

The R function for transpose is `t()`.

It can be shown that if A and B are conformable, then

(AB)' = B'A'

For some matrices C, we have C' = C. C is then termed *symmetric*.

We will often write a row vector in the form (a,b,c,...). So (5,1,88) means the 1x3 matrix with those elements. If we wish this to be a column vector, we use transpose, so that for instance (5,1,88)' means a 3x1 matrix.

2.8.3 Trace of a square matrix

The *trace* of a square matrix A is the sum of its diagonal elements, $tr(A) = \sum_{i=1}^{n} A_{ii}$. This measure has various properties, some obvious (trace of the sum is sum of the traces), and some less so, such as:

Suppose A and B are square matrices of the same size. Then

$$tr(AB) = tr(BA) \tag{2.1}$$

Proof: See Your Turn problem.

Your Turn: Prove Equation 2.1. Hint: Write out the left-hand side as a double sum. Reverse the order of summation, and work toward the right-hand side.

And furthermore:

It can be shown that trace is invariant under *circular shifts*, e.g. UVW, VWU and WUV all have the same trace.

3 Partitioned Matrices: an Invaluable Visualization Tool

Here, "visualization" is not a reference to graphics but rather to highlighting certain submatrices.

> Crucial Tool
>
> The techniques introduced in this section will be used throughout the book. Readers should spend extra time here, devising some of their own examples.

3.1 How It Works

Consider a matrix-vector product Mv. Of course, that means that v is a column vector whose length is equal to the number of columns of M. If M is of size rxs, then v is sx1.

Let's denote column j of M by Cj. Then we will see later in this chapter that

$$Mv = v_1 C_1 + v_2 C_2 + ... + v_s C_s$$

For instance, take

$$A = \begin{pmatrix} 5 & 2 & 6 \\ 1 & 2.6 & -1.2 \end{pmatrix}$$

and

$$v = \begin{pmatrix} 10 \\ 2 \\ 1 \end{pmatrix}$$

The reader should check that

$$10 \begin{pmatrix} 5 \\ 1 \end{pmatrix} + 2 \begin{pmatrix} 2 \\ 2.6 \end{pmatrix} + 1 \begin{pmatrix} 6 \\ -1.2 \end{pmatrix} = Av$$

where the latter is

$$\begin{pmatrix} 60 \\ 14 \end{pmatrix}$$

Note that the above expression,

$$10 \begin{pmatrix} 5 \\ 1 \end{pmatrix} + 2 \begin{pmatrix} 2 \\ 2.6 \end{pmatrix} + 1 \begin{pmatrix} 6 \\ -1.2 \end{pmatrix},$$

is a sum of scalar products of vectors, which is called a *linear combination* of those vectors. The quantities 10, 2 and 1 are the *coefficients* in that linear combination.

In other words, we have that:

The product Av of a matrix times a column vector is equal to a linear combination of the columns of the matrix, with coefficients equal to the column vector.

Similarly,

The product wA of a row vector and a matrix is equal to a linear combination of the rows of the matrix, with the coefficients coming from the row vector.

To further illustrate partitioned matrices, write the above matrix A as

$$A = \begin{pmatrix} A_{11} & A_{21} \end{pmatrix}$$

where

$$A_{11} = \begin{pmatrix} 5 & 2 \\ 1 & 2.6 \end{pmatrix}$$

and

$$A_{12} = \begin{pmatrix} 6 \\ -1.2 \end{pmatrix}$$

Symbolically, A now looks like a 1x2 "matrix." Similarly, writing

$$v = \begin{pmatrix} v_{11} \\ v_{21} \end{pmatrix}$$

where

$$v_{11} = \begin{pmatrix} 10 \\ 2 \end{pmatrix}$$

and $v_{21} = 1$ (a 1x1 matrix), v looks to be 2x1.

So, again pretending, treat the product Av as the multiplication of a 1x2 "matrix" and 2x1 "vector", yielding a 1x1 result,

$$A_{11}v_{11} + A_{12}v_{21}$$

But all that pretending actually does give the correct answer!

$$A_{11}v_{11}+A_{12}v_{21} = \begin{pmatrix} 5 & 2 \\ 1 & 2.6 \end{pmatrix} \begin{pmatrix} 10 \\ 2 \end{pmatrix} + \begin{pmatrix} 6 \\ -1.2 \end{pmatrix} 1 = \begin{pmatrix} 60 \\ 14 \end{pmatrix}$$

We can extend that reasoning further. Say A and B are matrices of sizes mxn and nxk, and consider the product AB. Partition B by its columns,

$$B = (B^{(1)}, B^{(2)}, ..., B^{(k)})$$

Now pretending that A is a 1x1 "matrix" and B is a1xk "matrix", we have

$$AB = (AB^{(1)}, AB^{(2)}, ..., AB^{(k)})$$

In other words,

In the product AB, column j is a linear combination of the columns of A, and the coefficients in that linear combination are the elements of column j of B.

A similar result holds for the row of the product.

Your Turn: Write out the details of that "similar result."

4 A Further Look at Markov Chains

Suppose X_0, our state at time 0, is random. Let f denote its distribution, i.e. its list of probabilities: $f_i = P(X_0 = i)$, i = 1,…,k, where k is the number of states in the chain. What about X_1, the state at time 1? Let's find an expression for g, the distribution of X_1.

This section will be longer than previous ones, but will bring together many of the concepts. The reader's patience here will be an investment paying great dividends in the sequel.

$$g_j = P(X_1 = j) = \sum_{i=1}^{k} P(X_0 = i)P(X_1 = j|X_0 = i) = \sum_{i=1}^{k} f_i a_{ij}$$

where a_{ij} is the row i, column j element of the chain's transition matrix P.

For example, consider g_5. How could we be at state 5 at time 1? We could start in state 1, probability f_1, then move to state 5, probability a_{15}, for a total probability of $f_1 a_{15}$. Or, we could start in state 2, probability f_2, then move to state 5, probability a_{25}, for a total probability of $f_2 a_{25}$. And so on.

So,

$$g_j = \sum_{i=1}^{k} f_i a_{ij}$$

Putting this is more explicit matrix terms,

$$g = \begin{pmatrix} g_1 \\ g_2 \\ \dots \\ g_k \end{pmatrix} = \begin{pmatrix} f_1 a_{11} + f_2 a_{21} + \dots + f_k a_{k1} \\ f_1 a_{12} + f_2 a_{22} + \dots + f_k a_{k2} \\ \dots \\ f_1 a_{1k} + f_2 a_{2k} + \dots + f_k a_{kk} \end{pmatrix}$$

That last expression is

$$f'P$$

so we have the nice compact relation for the distribution of X_1 in terms of the distribution of X_0.

By the way, it won't be used here, but just for practice, note that the right-hand side, Pf, here is a linear combination of the columns of P, from our above material on partitioning.

$$g' = f'P$$

And setting h to the distribution of X_2, the same reasoning gives us

$$h' = g'P$$

Then since $g' = f'P$, we have

Note that we used the Markov property, "memorylessness." Once we reach time 1, "time starts over," regradless of the previous history, i.e. regardless of where we were at time 0.

$$h' = f'P^2$$

and so on. Iterating, we obtain

$$d'_j = d'_0 P^j$$

where d_i is the distribution of X_i.

Similarly,

$$d'_j = d'_{j-1} P$$

For convenience, let's take transposes:[Recall that $(AB)' = B'A'$.

$$d_j = P' d_{j-1}$$

Now suppose our chain as a long-run distribution, as in {Section 2.5}. Let's call that distribution ν. By lettting $j \to \infty$ above, we have

$$\nu = P'\nu$$

Since P is known, this provides us with a way to compute ν. Yes, finding a high power of P would do this too, but that would involve a lot of computation, and even then it would not yield the exact answer. We will return to this in the next chapter.

4.1 Application: Google PageRank

When Google was first formed, its key internal component was a method to rank Web sites in terms of popularity. They developed such a method, and named it PageRank, a pun combining the term *Web page* (i.e. Web site) and the name of one of the founders, Larry Page. It's based on a Markov model.

The transition matrix is modeled as follows. Row i has o_i nonzero entries, each of which is equal to $1/o_i$. They define popularity as the resulting long-run distribution, i.e. ν in Section 2.5.

4.2 Random Vectors

You are probably familiar with the concept of a random variable, but of even greater importance is random *vectors*.

Say we are jointly modeling height, weight, age, systolic blood pressure and cholesterol, and are especially interested in relations between these quantities. We then have the random vector

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{pmatrix} = \begin{pmatrix} \text{height} \\ \text{weight} \\ \text{age} \\ \text{bp} \\ \text{chol} \end{pmatrix}$$

4.2.1 Sample vs. population

We may observe n realizations of X in the form of sample data, say on $n = 100$ people. In the statistics world, we treat this data as a random sample from some population, say all Americans. Usually, we are just given the data rather then having actual random sampling, but this view recognizes that there are a lot more people out there than our data.

We speak of estimating population quantities. For instance, we can estimate the population value \$E(X1), i.e. mean of X_1 throughout the population, by the sammple analog,

$$\frac{1}{n}\sum_{i=1}^{n} X_{1j}$$

where X_{ij} denotes the value of X_i for the j^{th} person in our sample.

By contrast, this view is rarely taken in the machine learning community. The data is the data, and the fact that it is a small subset of a much larger group is irrelevant. They will often allude to the randomness of the data by mentioning the "data generating mechanism."

4.3 Covariance Matrices

Recall the notion in statistics of *covariance*: Given a pair of random variables U and V, their covariance is defined by

$$Cov(U,V) = E[(U - EU)(V - EV)]$$

Loosely speaking, this measures the degree to which the two random variables vary together. Consider for instance human height H and W. Taller people tend to also be heavier. Say we sample many people from a population. Most of those who are taller than average, i.e. $H > EH$ will also be heavier than average, $W > EW$, making $(H-EH)(W-EW) > 0$. Similarly, shorter people tend to be lighter, but then we still have $(H -$

$EH)(W - EW) > 0$. So, usually $(H - EH)(W - EW) > 0$, and though there will be a number of exceptions, they will be rare enough so that $E[(U - EU)(V - EV)] > 0$.

Of course, the *magnitude* of $(H - EH)(W - EW)$ plays a role too.

V is usually large – meaning above its mean EV – when U is large (i.e. above *its* mean), and they are usually both small together. Then $U - EU$ and $V - EV$ are usually of the same sign, thus have a positive product. Then $Cov(U, V) > 0$. If on the other hand, one is usually small when the other is large and vice versa, $Cov(U, V) < 0$. This will later lead to the concept of correlation, but that intuition will serve us now.

Note some properties of scalar covariance.

- Cov is bilinear, i.e. $Cov(aU, bV) = abCov(U, V)$.
- $Cov(U, U) = Var(U)$.
- $Var(U + V) = Var(U) + Var(V) + 2Cov(U, V)$.

The relations between the various components of X are often characterized by the *covariance matrix* of X, whose entries consist of scalar covariances between pairs of components of a random vector. It is defined as follows for a k-component random vector. The covariance matrix, denoted by $Cov(X)$, is a k x k matrix, and for $1 \leq i, j \leq k$,

The definition is soewhat overloaded. "Cov" refers both to the covariance between two random variables, say height and weight, and to the covariance of a random vector, which is a matrix. But it will always be clear from context which one is being discussed.

$$Cov(X_i, X_j) = E[(X_i - EX_i)(X_j - EX_j)]$$

As an example, here is data on major league baseball players:

```
library(qeML)
data(mlb1)
head(mlb1)
```

```
        Position Height Weight   Age
1        Catcher     74    180 22.99
2        Catcher     74    215 34.69
3        Catcher     72    210 30.78
4  First_Baseman     72    210 35.43
5  First_Baseman     73    188 35.71
6 Second_Baseman     69    176 29.39
```

```
hwa <- mlb1[,-1]
cov(hwa)
```

```
           Height    Weight        Age
Height  5.3542814  25.61130 -0.8239233
Weight 25.6113038 433.60211 12.9110576
Age    -0.8239233  12.91106 18.6145019
```

```
cor(hwa)
```

```
            Height    Weight         Age
Height  1.00000000 0.5315393 -0.08252974
Weight  0.53153932 1.0000000  0.14371113
Age    -0.08252974 0.1437111  1.00000000
```

Again, we'll be discussing more of this later, but what about that negative correlation between height and age? It's near 0, and this could be a sampling artifact, but another possibility is that in this sport, shorter players do not survive as well.

Properties of the matrix version of covariance:

- For statistically independent random vectors Q and W of the same length,

$$Cov(Q+W) = Cov(Q) + Cov(W) \tag{4.1}$$

- For any nonrandom scalar c, and Q a random vector, we have $Cov(cQ) = c^2 Cov(Q)$.
- Say we have a random vector X, of length k, and a nonrandom matrix A of sise mxk. Then AX is a new random vector Y of m components. It turns out that

$$Cov(Y) = ACov(X)A' \tag{4.2}$$

  The proof is straightforward but tedious, and it will be omittted.

- $Cov(X)$ is a symmetric matrix. This follows from the symmmetry of the definition.
- The diagonal elements of $Cov(X)$ are the variances of the random variables X_i. This follows from the definition of the variance of a random variable.
- If X is a vector of length 1, i.e. a number, then

$$Cov(X) = Var(X)$$

- For any length-k column vector a,

$$Var(a'X) = a'\ Cov(X)\ a$$

- Thus $Cov(X)$ is *nonnegative definite*, meaning that for any length-k column vector a

$$a'Cov(X)a \geq 0$$

4.4 Your Turn

Your Turn: The long-run probabilities here turned out to be uniform, with value 0.20 for all five states. In fact, that is usually not the case. Make a small change to P_1 – remember to keep the row sums to 1 – and compute a high power to check whether the long-run distribution seems nonuniform.

Your Turn: Not every Markov chain, even ones with finitely many states, have long-run distributions. Some chains have *periodic* states. It may be, for instance, that after leaving state i, once can return only after an even number of hops. Modify our example chain here so that states 1 and 5 (and all the others) have that property. Then compute P^n for various large values of n and observe oscillatory behavior, rather than long-run convergence.

Your Turn: Consider the following model of a discrete-time, single-server queue:

- Model parameters are p (probability of job completion), q (probability of new job arriving) and m (size of the buffer).
- Jobs arrive, are served (possibly after queuing) and leave.
- Only one job can be in service at a time.
- At each time epoch:
  - The job currently in service, if any, will complete with probability p.
  - Slightly after a possible job completion, a job in the queue, if any, will start service.

  a Slightly after that, anew job will arrive with probability q. If the queue is empty, this job starts service. If not, and if the queue is not full, it will join the queue. Otherwise, the job is discarded.
- The system is memoryless.
- The current state is the number of jobs in the system, taking on the values 0,1,2,..,m+1; that last state means m jobs in the queue and 1 in service.

For instance, say p = 0.4, q = 0.2, m = 5, Suppose the current state is 3, so there is a job in service and two jobs in the queue. Our next state will be 2 with probability (0.4) (0.8); it will be 3 with probability (0.4) (0.2), and so on.

Analyze this system for the case given above.‘ Find the approximate long-run distribution, and also the proportion of jobs that get discarded.

5 Matrix Inverse

i Goals of this chapter:

Central to matrix operations is the matrix *inverse*, which is somewhat analogous to reciprocals in arithmetic. This is a fundamental operation in linear algebra (though ironically, related quantities are often used instead).

5.1 Example: Computing Long-Run Markov Distribution

At the end of the last chapter, we found that for a Markov chain with transition matrix P and stationary distribution ν,

$$\nu = P'\nu \tag{5.1}$$

This suggests a method for computing ν, by solving the above equation.

Rewrite it using the identity matrix:

$$(I - P')\nu = 0$$

Recall that for any square matrix C and identity matrix I of the same size, $IC = CI = C$.

For the random walk chain in Chapter 1, we had

$$P = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{pmatrix}$$

In that particular model, P' = P, but for most chains this is not the case.

With $\nu = (\nu_1, \nu_2, \nu_3, \nu_4, \nu_5)'$, the equation to be solved, $(I - P')\nu = \nu$, is

$$\begin{pmatrix} 0.5 & -0.5 & 0 & 0 & 0 \\ -0.5 & 1 & -0.5 & 0 & 0 \\ 0 & -0.5 & 1 & -0.5 & 0 \\ 0 & 0 & -0.5 & 1 & -0.5 \\ 0 & 0 & 0 & -0.5 & 0.5 \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

If we perform the matrix multiplication, we have an ordinary system of linear equations:

$$\begin{aligned} 0.5\nu_1 - 0.5\nu_2 &= 0 \\ -0.5\nu_1 + \nu_2 - 0.5\nu_3 &= 0 \\ -0.5\nu_2 + \nu_3 - 0.5\nu_4 &= 0 \\ -0.5\nu_3 + \nu_4 - 0.5\nu_5 &= 0 \\ -0.5\nu_4 + 0.5\nu_5 &= 0 \end{aligned}$$

This is high school math, and we could solve the equations that way. But this is literaally what linear algebra was invented for, solving systems of equations! We will use matrix inverse.

But first, we have a problem to solve: The only solution to the above system is with all $\nu_i = 0$. We need an equation involving a nonzero quantity.

But we do have such an equation. The ν is a stationary distribution for a Markov chain, and thus must sum to 1.0. Let's replace the last row by that relation:

$$\begin{pmatrix} 0.5 & -0.5 & 0 & 0 & 0 \\ -0.5 & 1 & -0.5 & 0 & 0 \\ 0 & -0.5 & 1 & -0.5 & 0 \\ 0 & 0 & -0.5 & 1 & -0.5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

More compactly,

$$B\nu = d$$

Many square matrices A have a multiplicative inverse, denoted by A^{-1}, with the property that

$$A^{-1}A = AA^{-1} = I$$

If our matrix B is invertible, we can premultiply both sides of our equation above, yielding

$$B^{-1}d = B^{-1}B\nu = \nu$$

So, we have obtained our solution for the stationary distribution. We can evaluate it numerically via the R **solve** function, which finds matrix inverse:

```
B <-
rbind(c(0.5,-0.5,0,0,0), c(-0.5,1,-0.5,0,0), c(0,-0.5,1,-0.5,0),
   c(0,0,-0.5,1,-0.5), c(1,1,1,1,1))
B
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]  0.5 -0.5  0.0  0.0  0.0
[2,] -0.5  1.0 -0.5  0.0  0.0
[3,]  0.0 -0.5  1.0 -0.5  0.0
[4,]  0.0  0.0 -0.5  1.0 -0.5
[5,]  1.0  1.0  1.0  1.0  1.0
```

```
Binv <- solve(B)
# check the inverse
Binv %*% B  # yes, get I (of course with some roundoff error)
```

```
              [,1]          [,2]          [,3]          [,4]          [,5]
[1,]  1.000000e+00 1.942890e-16 -1.387779e-16  1.942890e-16  8.326673e-17
[2,]  1.942890e-16 1.000000e+00  3.053113e-16 -2.775558e-17  8.326673e-17
[3,] -2.775558e-17 0.000000e+00  1.000000e+00  0.000000e+00  2.775558e-17
[4,] -1.665335e-16 4.996004e-16 -3.608225e-16  1.000000e+00 -2.775558e-17
[5,] -1.665335e-16 7.216450e-16 -4.996004e-16  5.551115e-17  1.000000e+00
```

```
nu <- Binv %*% c(0,0,0,0,1)
nu
```

```
     [,1]
[1,]  0.2
[2,]  0.2
[3,]  0.2
[4,]  0.2
[5,]  0.2
```

This confirms our earlier speculation based on powers of P'.

5.2 Matrix Algebra

Several properties to note:

- If the inverses of A and B exist, and A and B are conformable, then $(AB)^{-1}$ exists and is equal to $B^{-1}A^{-1}$.
- $(A')^{-1}$ exists and is equal to $(A^{-1})'$.
- If A is invertible and symmetric, then $(A^{-1})'$ is also symmetric.$

5.3 Computation of the Matrix Inverse

Finding the inverse of a large matrix – in data science applications, the number of rows and columns n can easily be hundreds or more – can be computationally challenging. The run time is proportional to n^3, and roundoff error can be an issue. Sophisticated algorithms have been developed, such as QR and Choleski decompositions. So in R, we should use, say, **qr.solve** rather than **solve** if we are working with sizable matrices.

The classic "pencil and paper" method for matrix inversion is instructive, and will be presented here.

5.3.1 Pencil-and-paper computation

The basic idea follows the pattern the reader learned for solving systems of linear equations, but with the added twist of involving some matrix multiplication.

Let's take as our example

$$A = \begin{pmatrix} 4 & 7 \\ -8 & 15 \end{pmatrix}$$

We aim to transform this to the 2x2 identity matrix, via a sequence of row operations.

5.3.2 Use of elementary matrices

Let's multiply row 1 by 1/4, to put 1 in the first element:

$$\begin{pmatrix} 1 & \frac{7}{4} \\ -8 & 15 \end{pmatrix}$$

In matrix terms, that operation is equivalent to premultiplying A by

$$E_1 = \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & 1 \end{pmatrix}$$

We then add 8 times row 1 to row 2, yielding

$$\begin{pmatrix} 1 & \frac{7}{4} \\ 0 & 29 \end{pmatrix}$$

The premultiplier for this operation is

$$E_2 = \begin{pmatrix} 1 & 0 \\ 8 & 1 \end{pmatrix}$$

Multiply row 2 by 1/29:

$$\begin{pmatrix} 1 & \frac{7}{4} \\ 0 & 1 \end{pmatrix}$$

corresponding to

$$E_3 = \begin{pmatrix} 1 & 0 \\ 8 & 1/29 \end{pmatrix}$$

And finally, add -7/4 row 2 to row 1.

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

The premultiplier is

$$E_4 = \begin{pmatrix} 1 & \frac{7}{4} \\ 0 & 1 \end{pmatrix}$$

Now, how does that give use A^{-1}? The method your were taught probably set up the partioned matrix (A, I). The row operations that transformed A to I also transformed I to A^{-1}. Here's why:

As noted, the row operations are such that

$$E_4 E_3 E_2 E_1 A$$

give us the final transformed result, i.e. I:

$$E_4 E_3 E_2 E_1 A = I$$

Then

$$A = (E_4 E_3 E_2 E_1)^{-1} I = E_4^{-1} E_3^{-1} E_2^{-1} E_1^{-1}$$

Recall that the inverse of a product (if it exists is the reverse product of the inverses.

So this accomplishes our goal of computing A^{-1}, provided we have the inverses of the elementary matrices E_i, But those are easy, as each simply "undoes" its partner. E_1, for instance, multiplies the row 1, column 1 element by 1/4, which is "undone" by multiplying that element by 4,

$$E_1^{-1} = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$$

Also, to undo the operation of adding 8 times row 1 to row 2, we add -8 times row 1 to row 2:

$$E_2^{-1} = \begin{pmatrix} 1 & 0 \\ -8 & 1 \end{pmatrix}$$

5.4 Nonexistent Inverse

Suppose our matrix A had been slightly different:

$$A = \begin{pmatrix} 4 & 7 \\ -8 & -14 \end{pmatrix}$$

This would have led to

$$\begin{pmatrix} 1 & \frac{7}{4} \\ 0 & 0 \end{pmatrix}$$

This cannot lead to i, indicating that A^{-1} does not exist, and the matrix is said to be *singular*. And it's no coincidence that row 2 of A is double row 1. This has many implications, as will be seen in our chapter on vector spaces.

5.5 Determinants

This is a topic that is quite straightforward and traditional, even old-fashioned. Yet the theme of a book by Sheldon Axler„ *Linear Algebra Done Right* is that determinants are overemphasized. He relegates the topic to the very end of the book. Yet determinants do appear often in applied linear algebra settings. Moreover, they will be convenient to use in explaing very concepts in this book.

But why place the topic in this particular chapter? The answer lies in the fact that earlier in this chapter we had the proviso "If $(A'A)^{-1}$ exists." The following property of determinants is then relevant:

A square matrix G is invertible if and only if $det(G) \neq 0$.

There are better ways to ascertain invertibility than this, but it is conceptually helpful. Determinants play a similar role in the topic of eigenvectors in Chapter 5.

5.5.1 Definition

The standard definition is one of the ugliest, in all of mathematics. Instead we will define the term using one of the methods for calculating determinants.

Consider an rxr matrix G. For $r = 2$, write G as

$$G = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

and define $\det(G)$ to be $ad - bc$. For $r > 2$, define submatrices as follows.

G_i is the $(r-1)$x$(r-1)$ submatrix obtained by removing row 1 and column j from G. Then $\det(G)$ is defined recursively as

$$\sum_{i=1}^{r} (-1)^{i+1} \det(G_i)$$

For instance, consider

$$M = \begin{pmatrix} 5 & 1 & 0 \\ 3 & -1 & 7 \\ 0 & 1 & 1 \end{pmatrix}$$

The $\det(M) = 5(-8) - 3 + 0 = -43$.

A glimpse at the classical definition:

Using the same approach as in the last computation, we would find that the determinant of a general 3 x 3 matrix

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

is

$$aei + bfg + cdh - afh - bdi - ceg$$

Each term here is involves a product of 3 of the elements of the matrix. The formation of the terms in general, and the deterimination of + and - signs, is done in complex but precise manner that we will not present here. But the reader should at least keep in mind that each term is a product of n elements of the matrix, a fact that will be relevant in the sequel.

It involves sums and differences of permuted products of distinct elements of the matrix.

5.5.2 Properties

We state these without proof:

- G^{-1} exists if and only if $\det(G) \neq 0$
- $\det(GH) = \det(G)\det(H)$

5.6 The Multivariate Normal Distribution Family

The familiar "bell-shaped curve" refers to the *normal* (or *Gaussian*) family, whose densities have the form

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-0.5(\frac{t-\mu}{\sigma})^2}$$

The values of μ and σ are the mean and standard deviation. But what if we have a random vector, say of length k? Is there a generalized normal family?

5.6.1 Example: k = 2

The answer is yes. Here is an example for $k = 2$:

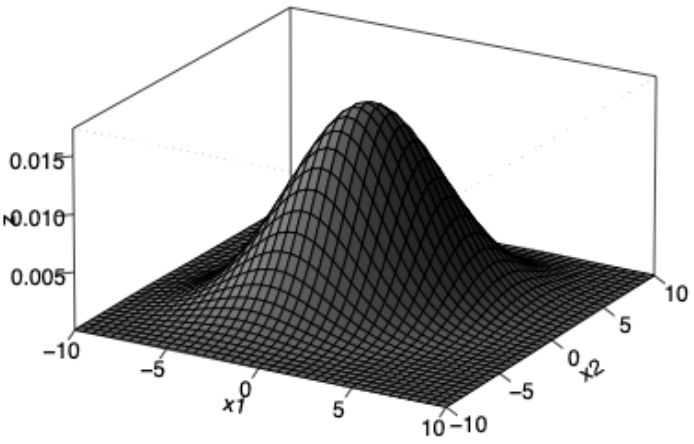


Figure 5.1: 3D bell density

5.6.2 General form

Well, then, what is the form of the k-dimensional density function? Just as the univariate normal family is parameterized by mean and variance, the multivariate one is parameterized via mean vector μ and covariance matrix Σ. The form is

$$(2\pi)^{-k/2} \det(\Sigma)^{-k/2} e^{-0.5(t-\mu)'(\Sigma)^{-1}(t-\mu)}$$

Note the intuition:

- Instead of $1/\sigma^2$, i.e. instead of dividing by variance, we "divide by Σ," intuitively viewing matrix inverse as a "reciprocal" of a matrix.
- In other words, covariance matrices operate roughly like generalized variances.
- Instead of squaring the scalar $t-\mu$, we "square" it in the vector case by peforming a $w'w$ operation, albeit with Σ^{-1} in the middle.

Clearly, we should not stretch these analogies very far, but they do help our intuition here.

5.6.3 Properties

Theorem 5.1. *if a random vector is MV normally distributed, then the conditional distribution of any one of its components Y, given the others $X_{others} = t$ (note that t is a vector if $k > 2$) has the following properties:*

- *It has a (univariate) normal distribution.*
- *Its mean $E(Y|X_{others}) = t$ linear in t.*
- *Its variance $Var(Y|X_{others}) = t$ does not involve t.*

These of course are the classical assumptions of linear regression models. They actually come from the MV normal model.

Proof. This comes out of writing down the conditional density (overall density divided by marginal), and then doing some algebra.

□

□

Theorem 5.2. *If X is a multivariate-normal random vector, then so is AX for any conformable nonrandom matrix A.*

Proof. Again, perform direct evaluation of the density.

□

□

Theorem 5.3. *A random vector X has a multivariate normal distribution if and only if $w'X$ has a univariate normal distribution for all conformable nonrandom vectors w.*

Theorem 5.4 (The Multivariate Central Limit Theorem)**.** *Let $X_1, X_2, ...$ be a sequence on statistically independent random vectors, with common distribution multivariate normal with mean vector μ and covariance matrix Σ. Write*

$$\bar{X} = \frac{X_1 + ... + X_n}{n}$$

Then the distribution of the random vector

$$W_n = \sqrt{n}(\bar{X} - \mu)$$

goes to multivariate normal with the 0 vector as mean and covariance matrix Σ.

The usual form would involve the "square root" of a matrix, but we will not discuss that concept until our chapter on inner product spaces.

‘ ::: {.proof}

Theorem 6.3 reduces the problem to the univariate case, where we know the Central Limit Theorem holds.

□

:::

5.7 Multinomial Random Vectors Have Approximate Multivariate Normal Distributions

Recall that a *multinomial* random vector is the mathematial analog of an R factor variable – a categorical variable with k levels/categories. Just as a binomial random variable represents the number of "successes" in n "trials," a multinomial random vector represents the numbers of successes in each of the k categories.

Let's write such a random vector as

$$X = \begin{pmatrix} N_1 \\ ... \\ N_k \end{pmatrix}$$

Let p_i be the probability of a trial having outcome $i - 1, ..., k$. Note the following:

- $N_1 + ... + N_k = n$
- The marginal distribution of N_i is binomial, with success probability p_i and n trials.

So for instance if we roll a fair die 10 times, then N_i is the number of trials in which the roll's outcome was i dots and $p_i = 1/6$, $i = 1, 2, 3, 4, 5, 6$.

Let's find $Cov(X)$. Define the *indicator* vector

$$I_i = \begin{pmatrix} I_{i1} \\ ... \\ I_{ik} \end{pmatrix}$$

Here I_{ij} is 1 or 0, depending on whether trial i resulted in category j. (A 1 "indicates" that caregory j occurred.)

The key is that

$$X = \sum_{i=1}^{n} I_i$$

For instance, in the die-rolling example, the first component on the right-hand side is the number of rolls in which we got 1 dot, and that is by definition the same as N_1, the first component of X.

So there we have it – X is a sum of independent, identically distributed random vectors, so by the Multivariate Central Limit Theorem, X has an approximate multivariate normal distribution. Now, what are the mean vector and covariance matrix in that distribution?

From our discussion above, we know that

$$EX = \begin{pmatrix} np_1 \\ ... \\ np_k \end{pmatrix}$$

What about $Cov(X)$?

5.8 The Delta Method

This is one of the most useful simple tools in statistics.

To set the stage, let's review the statistical concepts of *confidence interval* and *standard error*. Say we have an estimator $\hat{\theta}$ of some population parameter θ, e.g. $\bar{X}$ for a population mean μ.

- Roughly speaking, the term *standard error* of is our estimate of $\sqrt{\hat{\theta}}$. More precisely, suppose that $\hat{\theta}$ is asymptotically normal. The standard error is an estimate of the standard deviation of that normal distribution.

  This can be used to form a confidence interval (see below), but also stands on its own as an indication of the accuracy of $\hat{\theta}$.

- A, say 95%, confidence interval for μ is then

$$\hat{\theta} \pm 1.96 \text{ s.e.}(\hat{\theta})$$

The 95% figure means that of all possible samples of the given size from the population, 95% of the resulting confidence intervals will contain θ.

Now, for the delta method, as a first example, say we are estimating a population mean μ and are also interested in estimating $\log(\mu)$.

We will probably use the sample mean $\bar{X}$ to estimate μ, and thus use $W = \log \bar{X}$ to estimate $\log(\mu)$. But how do we obtain a standard error for W?

If we just need to form a confidence interval for $\log(\mu)$, we can form a CI for μ and then take the log of both endpoints. But again, standard errors are of interest in their own right.

The Central Limit Theorem tells as that $\bar{X}$ is asymptotically normally distributed. But what about $\log \bar{X}$?

From calculus, we know that a smooth function f can be written as a Taylor series,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)(x - x_0)^2/2 + ...$$

where here "'" denotes derivative.

In our case here, setting $f(t) = \log t$, $x_0 = \mu$ and $x = \bar{X}$, we have

$$W = \log \mu + \log'(\mu)(\bar{X} - \mu) + \log''(\mu)(\bar{X} - \mu)^2/2 + ...$$

and $\log'(t) = 1/t$ and so on.

The key point is that as n grows, $\bar{X} - \mu$ goes to 0, and $(\bar{X} - \mu)^2$ goes to 0 even faster. Using theorems from probability theory, one can show that, in the sense of distribution,

$$W \approx log(\mu) + log'(\mu)(\bar{X} - \mu)$$

In other words, W has an approximate normal distribution that has mean $log(\mu)$ and variance

$$\frac{1}{\mu^2}\sigma^2/n$$

where σ^2 is the population variance. We estimate the latter by the usual S^2 quantity, and thus have our standard error,

$$\text{s.e.}(W) = \frac{S}{\bar{X}\sqrt{n}}$$

Now, what if the function f has two arguments instead of one? The above linear approximation is now

$$f(v,w) \approx f(v_0,w_0) + f_1(v_0,w_0)(v-v_0) + f_2(v_0,w_0)(w-w_0) \tag{5.2}$$

where f_1 and f_2 are partial derivatives,

A *partial derivative* of a function of more than one variable is the derivative with respect to one of those variables. E.g. $\partial/\partial v\ vw^2 = w^2$ and $\partial/\partial w\ vw^2 = 2vw$.

$$f_1(v,w) = \frac{\partial}{\partial v} f(v,w)$$

$$f_2(v,w) = \frac{\partial}{\partial w} f(v,w)$$

So if we are estimating, for instance, a population quantity $(\alpha,\beta)'$ by $(Q,R)'$, its standard error is

$$\sqrt{f_1^2(Q,R)Var(Q) + f_2^2(Q,R)Var(R) + 2f_1(Q,R)f_2(Q,R)Cov(Q,R)}$$

where again quantities like $Var(Q)$ are understood to be in the context of the asymptotic normal distribution.

As usual, use of matrix notation can help clean up messy expressions like this. The *gradient* of f, say in the two-argument case as above, is the vector

$$\nabla f = \begin{pmatrix} f_1(v,w) \\ f_2(v,w) \end{pmatrix}$$

so that Equation 5.2 can be written as

$$f(v,w) = f(v_0,w_0) + (\nabla f)' \begin{pmatrix} v - v_0 \\ w - w_0 \end{pmatrix}$$

5.9 Your Turn

Your Turn: Prove the assertions in Section 5.2. Note that for the identity matrix I, $I' = I$.

Your Turn: In Section 5.3.2, find the inverses of E_3 and E_4 using similar reasoning, and thus find A^{-1}.

Your Turn: If you are familiar with recursive calls, write a function `dt(a)` to compute the determinant of a square matrix A.

Your Turn: The determinant of a 3 x 3 matrix

$$M = \begin{pmatrix} a & b & d \\ d & e & f \\ g & h & i \end{pmatrix}$$

is

$$aei + bfg + cdh - ceg - bdi - afh$$

Suppose the elements of M are independent random variables with uniform distributions on (0,1). Argue that P(M is invertible) = 1.

Your Turn: Show that in the scalar context,

$$Cov(X,Y) = E(XY) - EX\ EY$$

Your Turn: Show that in the vector context,

$$Cov(X) = E(XX') - (EX)(EX)'$$

6 Covariance Matrices, Multivariate Normal Distribution, Delta Method

i Goals of this chapter:

A central entity in multivariate analysis is that of the *covariance matrix*. In this chapter we define the term, list its many properties, and show its role in the multivariate analog of the normal distribution family. We close the chapter with an application to the *delta method*, one of the most useful simple tools in statistics.

6.1 Covariance Matrices

Recall the notion in statistics of *covariance*: Given a pair of random variables U and V, their covariance is defined by

$$Cov(U, V) = E[(U - EU)(V - EV)]$$

Loosely speaking, this measures the degree to which the two random variables vary together. Consider for instance human height H and W. Taller people tend to also be heavier. Say we sample many people from a population. Most of those who are taller than average, i.e. $H > EH$, will also be heavier than average, $W > EW$, making $(H-EH)(W-EW) > 0$. Similarly, shorter people tend to be lighter, i.e. we often have $H < EH$ and $W < EW$, but then we still have $(H-EH)(W-EW) > 0$. So, one way or the other, usually $(H - EH)(W - EW) > 0$, and though there will be a number of exceptions, they will be rare enough so that

$$E[(H - EH)(W - EW)] > 0.$$

Of course, the *magnitude* of $(H - EH)(W - EW)$ plays a role too.

In other words, $Cov(H, W) > 0$. Similarly if U is often large when V is small, and vice versa, we will likely have $Cov(H, W) < 0$.

If this sounds like correlation to you, then your hunch is correct. Covariance will indeed later lead to the concept of correlation, but that intuition will serve us now.

Note some properties of scalar covariance.

- Cov is bilinear, i.e. $Cov(aU, bV) = abCov(U, V)$

- $Cov(U,U) = Var(U)$
- $Var(U+V) = Var(U) + Var(V) + 2Cov(U,V)$
- $Cov(U,V) = E(UV) - (EU)(EV)\{\#\text{eq-covuv}\}$

The relations between the various components of X are often characterized by the *covariance matrix* of X, whose entries consist of scalar covariances between pairs of components of a random vector. It is defined as follows for a k-component random vector. The covariance matrix, denoted by $Cov(X)$, is a k x k matrix, and for $1 \leq i, j \leq k$,

The definition is somewhat overloaded. "Cov" refers both to the covariance between two random variables, say height and weight, and to the covariance of a random vector, which is a matrix. But it will always be clear from context which one is being discussed.

$$Cov(X_i, X_j) = E[(X_i - EX_i)(X_j - EX_j)]$$

As an example, here is data on major league baseball players:

```
library(qeML)
data(mlb1)
head(mlb1)
```

```
        Position Height Weight   Age
1        Catcher     74    180 22.99
2        Catcher     74    215 34.69
3        Catcher     72    210 30.78
4  First_Baseman     72    210 35.43
5  First_Baseman     73    188 35.71
6 Second_Baseman     69    176 29.39
```

```
hwa <- mlb1[,-1]
cov(hwa)
```

```
           Height    Weight        Age
Height  5.3542814  25.61130 -0.8239233
Weight 25.6113038 433.60211 12.9110576
Age    -0.8239233  12.91106 18.6145019
```

```
cor(hwa)
```

```
            Height    Weight         Age
Height  1.00000000 0.5315393 -0.08252974
Weight  0.53153932 1.0000000  0.14371113
Age    -0.08252974 0.1437111  1.00000000
```

Again, we'll be discussing more of this later, but what about that negative correlation between height and age? It's near 0, and this could be a sampling artifact, but another possibility is that in this sport, shorter players do not survive as well.

Properties of the matrix version of covariance:

- Matrix form of definition:

$$Cov(X) = E[(X - X)(X - EX)'] \tag{6.1}$$

(Note the dimensions: X is a column vector, say k x 1, so $(X - X)(X - EX)'$ is q x q. The expected value is then of that size as well.)

- For statistically independent random vectors Q and W of the same length,

$$Cov(Q + W) = Cov(Q) + Cov(W) \tag{6.2}$$

- For any nonrandom scalar c, and Q a random vector, we have $Cov(cQ) = c^2 Cov(Q)$.
- Say we have a random vector X, of length k, and a nonrandom matrix A of sise mxk. Then AX is a new random vector Y of m components. It turns out that

$$Cov(Y) = ACov(X)A' \tag{6.3}$$

  The proof is straightforward but tedious, and it will be omittted.

- $Cov(X)$ is a symmetric matrix. This follows from the symmmetry of the definition.

- The diagonal elements of $Cov(X)$ are the variances of the random variables X_i. This follows from the definition of the variance of a random variable.
- If X is a vector of length 1, i.e. a number, then

$$Cov(X) = Var(X)$$

- For any length-k column vector a,

$$Var(a'X) = a' \; Cov(X) \; a \tag{6.4}$$

- Thus $Cov(X)$ is *nonnegative definite*, meaning that for any length-k column vector a

$$a'Cov(X)a \geq 0$$

6.2 The Multivariate Normal Distribution Family

The familiar "bell-shaped curve" refers to the *normal* (or *Gaussian*) family, whose densities have the form

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-0.5(\frac{t-\mu}{\sigma})^2}$$

The values of μ and σ are the mean and standard deviation. But what if we have a random vector, say of length k? Is there a generalized normal family?

6.2.1 Example: k = 2

The answer is yes. Here is an example for $k = 2$:

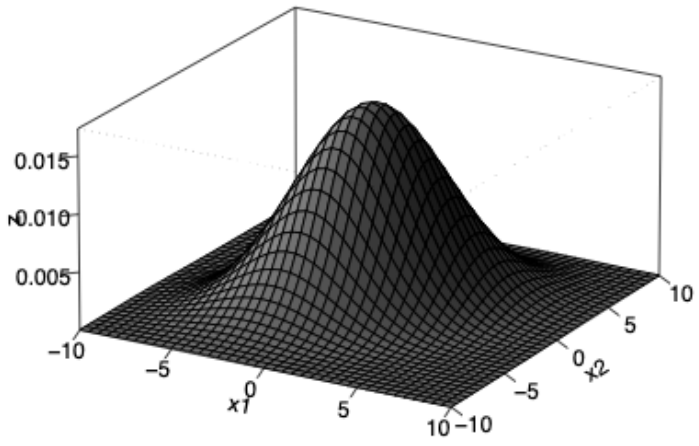


Figure 6.1: 3D bell density

6.2.2 General form

Well, then, what is the form of the k-dimensional density function? Just as the univariate normal family is parameterized by mean and variance, the multivariate one is parameterized via mean vector μ and covariance matrix Σ. The form is

$$(2\pi)^{-k/2} \det(\Sigma)^{-k/2} e^{-0.5(t-\mu)'(\Sigma)^{-1}(t-\mu)}$$

Note the intuition:

- Instead of $1/\sigma^2$, i.e. instead of dividing by variance, we "divide by Σ," intuitively viewing matrix inverse as a "reciprocal" of a matrix.
- In other words, covariance matrices operate roughly like generalized variances.
- Instead of squaring the scalar $t-\mu$, we "square" it in the vector case by peforming a $w'w$ operation, albeit with Σ^{-1} in the middle.

Clearly, we should not stretch these analogies very far, but they do help our intuition here.

6.2.3 Properties

Theorem 6.1. *If a random vector is MV normally distributed, then the conditional distribution of any one of its components Y, given the others $X_{others} = t$ (note that t is a vector if $k > 2$) has the following properties:*

- *It has a (univariate) normal distribution.*
- *Its mean $E(Y|X_{others}) = t$ linear in t.*
- *Its variance $Var(Y|X_{others}) = t$ does not involve t.*

These of course are the classical assumptions of linear regression models. They actually come from the MV normal model.

Proof. This comes out of writing down the conditional density (overall density divided by marginal), and then doing some algebra.

□

□

Theorem 6.2. *If X is a multivariate-normal random vector, then so is AX for any conformable nonrandom matrix A.*

Proof. Again, perform direct evaluation of the density.

□

□

Theorem 6.3. *A random vector X has a multivariate normal distribution if and only if $w'X$ has a univariate normal distribution for all conformable nonrandom vectors w.*

Theorem 6.4 (The Multivariate Central Limit Theorem)**.** *Let $X_1, X_2, ...$ be a sequence on statistically independent random vectors, with common distribution multivariate normal with mean vector μ and covariance matrix Σ. Write*

$$\bar{X} = \frac{X_1 + ... + X_n}{n}$$

Then the distribution of the random vector

$$W_n = \sqrt{n}(\bar{X} - \mu)$$

goes to multivariate normal with the 0 vector as mean and covariance matrix Σ.

The usual form would involve the "square root" of a matrix, but we will not discuss that concept until our chapter on inner product spaces.

‘ ::: {.proof}

Theorem 6.3 reduces the problem to the univariate case, where we know the Central Limit Theorem holds.

□

:::

6.3 Multinomial Random Vectors Have Approximate Multivariate Normal Distributions

Recall that a *multinomial* random vector is the mathematial analog of an R factor variable – a categorical variable with k levels/categories. Just as a binomial random variable represents the number of "successes" in n "trials," a multinomial random vector represents the numbers of successes in each of the k categories.

Let's write such a random vector as

$$X = \begin{pmatrix} N_1 \\ ... \\ N_k \end{pmatrix}$$

Let p_i be the probability of a trial having outcome $i - 1, ..., k$. Note the following:

- $N_1 + ... + N_k = n$
- The marginal distribution of N_i is binomial, with success probability p_i and n trials.

So for instance if we roll a fair die 10 times, then N_i is the number of trials in which the roll's outcome was i dots and $p_i = 1/6$, $i = 1, 2, 3, 4, 5, 6$.

Let's find $Cov(X)$. Define the *indicator* vector

$$I_i = \begin{pmatrix} I_{i1} \\ ... \\ I_{ik} \end{pmatrix}$$

Here I_{ij} is 1 or 0, depending on whether trial i resulted in category j. (A 1 "indicates" that caregory j occurred.)

The key is that

$$X = \sum_{i=1}^{n} I_i \tag{6.5}$$

For instance, in the die-rolling example, the first component on the right-hand side is the number of rolls in which we got 1 dot, and that is by definition the same as N_1, the first component of X.

So there we have it – X is a sum of independent, identically distributed random vectors, so by the Multivariate Central Limit Theorem, X has an approximate multivariate normal distribution. Now, what are the mean vector and covariance matrix in that distribution?

From our discussion above, we know that

$$EX = \begin{pmatrix} np_1 \\ ... \\ np_k \end{pmatrix}$$

What about $Cov(X)$? Again, recognizing that Equation 6.5 is a sum of independent terms, Equation 6.2 tells us that

$$Cov(X) = Cov(\sum_{i=1}^{n} I_i) = \sum_{i=1}^{n} Cov(I_i) = nCov(I_1),$$

that last equality reflecting that the I_i are identically distributed (the trials all have the same probabilistic behavior).

Now to evaluate that covariance matrix, consider two specific elements I_{1j} and I_{im} of I_1. Recall, that element is equal to 1 or 0, depending on whether the first trial results in Categories j and m, respectively. Then what is $Cov(I_{1j}, I_{1m})$? From **?@eq-covuv**, we have

$$Cov(I_{1j}, I_{1m}) = E(I_{1j} I_{1m}) - (EI_{1j})(EI_{1m}) \tag{6.6}$$

Consider the two cases:

- $i = j$: Here $E(I_{1j}^2) = E(I_{1j}) = p_j$. Thus

$$Cov(I_{1j}, I_{1m}) = p_j(1 - p_j)$$

- $i \neq j$: Each I_s consists of one 1 and $k-1$ 0s. Thus $E(I_{1j}I_{1m}) = 0$ and

$$Cov(I_{1j}, I_{1m}) = -p_j p_m$$

6.4 The Delta Method

This is one of the most useful simple tools in statistics.

6.4.1 Review: confidence intervals, standard errors

To set the stage, let's review the statistical concepts of *confidence interval* and *standard error*. Say we have an estimator $\hat{\theta}$ of some population parameter θ, e.g. $\bar{X}$ for a population mean μ.

- Loosely speaking, the term *standard error* of is our estimate of $\sqrt{Var(\hat{\theta})}$. More precisely, suppose that $\hat{\theta}$ is asymptotically normal. The standard error is an estimate of the standard deviation of that normal distribution. For this reason, It is customary to write $AVar(\hat{\theta})$ rather than $Var(\hat{\theta})$.

  This can be used to form a confidence interval (see below), but also stands on its own as an indication of the accuracy of $\hat{\theta}$.

- A, say 95%, confidence interval for μ is then

$$\hat{\theta} \pm 1.96 \text{ s.e.}(\hat{\theta})$$

The 95% figure means that of all possible samples of the given size from the population, 95% of the resulting confidence intervals will contain θ.

6.4.2 Delta method: motivating example

Now, for the delta method, as a first example, say we are estimating a population mean μ and are also interested in estimating $\log(\mu)$.

We will probably use the sample mean $\bar{X}$ to estimate μ, and thus use $W = \log \bar{X}$ to estimate $\log(\mu)$. But how do we obtain a standard error for W?

If we just need to form a confidence interval for $\log(\mu)$, we can form a CI for μ and then take the log of both endpoints. But again, standard errors are of interest in their own right.

6.4.3 Use of the Central Limit Theorem

The Central Limit Theorem tells as that $\bar{X}$ is asymptotically normally distributed. But what about $\log \bar{X}$?

From calculus, we know that a smooth function f can be written as a Taylor series,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)(x - x_0)^2/2 + ...$$

where here "'' denotes derivative.

In our case here, setting $f(t) = \log t$, $x_0 = \mu$ and $x = \bar{X}$, we have

$$W = \log \mu + \log'(\mu)(\bar{X} - \mu) + \log''(\mu)(\bar{X} - \mu)^2/2 + ...$$

and $\log'(t) = 1/t$ and so on.

The key point is that as n grows, $\bar{X} - \mu$ goes to 0, and $(\bar{X} - \mu)^2$ goes to 0 even faster. Using theorems from probability theory, one can show that, in the sense of distribution,

$$W \approx log(\mu) + log'(\mu)(\bar{X} - \mu)$$

In other words, W has an approximate normal distribution that has mean $log(\mu)$ and variance

$$\frac{1}{\mu^2}\sigma^2/n$$

where σ^2 is the population variance. We estimate the latter by the usual S^2 quantity, and thus have our standard error,

$$\text{s.e.}(W) = \frac{S}{\bar{X}\sqrt{n}}$$

6.5 Use of the Multivariate Central Limit Theorem

Now, what if the function f has two arguments instead of one? The above linear approximation is now

$$f(v,w) \approx f(v_0,w_0) + f_1(v_0,w_0)(v-v_0) + f_2(v_0,w_0)(w-w_0) \tag{6.7}$$

where f_1 and f_2 are partial derivatives,

A *partial derivative* of a function of more than one variable is the derivative with respect to one of those variables. E.g. $\partial/\partial v\ vw^2 = w^2$ and $\partial/\partial w\ vw^2 = 2vw$.

$$f_1(v,w) = \frac{\partial}{\partial v} f(v,w)$$

$$f_2(v,w) = \frac{\partial}{\partial w} f(v,w)$$

So if we are estimating, for instance, a population quantity $(\alpha,\beta)'$ by $(Q,R)'$, its standard error is

$$\sqrt{f_1^2(Q,R)AVar(Q) + f_2^2(Q,R)AVar(R) + 2f_1(Q,R)f_2(Q,R)ACov(Q,R)}$$

As usual, use of matrix notation can help clean up messy expressions like this. The *gradient* of f, say in the two-argument case as above, is the vector

$$\nabla f = \begin{pmatrix} f_1(v_0, w_0) \\ f_2(v_0, w_0) \end{pmatrix}$$

so that Equation 6.7 can be written as

$$f(v, w) = f(v_0, w_0) + (\nabla f)' \begin{pmatrix} v - v_0 \\ w - w_0 \end{pmatrix}$$

Then from Equation 6.4,

$$AVar[f(v, w)] = (\nabla f)' AV(v, w)(\nabla f) \tag{6.8}$$

Example: Ratio of Two Means

Say our sample data consists of mother-daughter pairs,

$$\begin{pmatrix} M \\ D \end{pmatrix}$$

representing the heights of mother and daughter. Denote the population mean vector by

$$\nu = \begin{pmatrix} \mu_M \\ \mu_D \end{pmatrix}$$

We might be interested in the ratio $\omega = \mu_D / \mu_M$. Our estimator will be $\hat{\omega} = \bar{D}/\bar{M}$, the ratio of the sample means.

Then in Equation 6.8, with $f(q, r) = q/r$

$$\nabla f = \begin{pmatrix} 1/r \\ -q/r^2 \end{pmatrix}$$

which in our application here we would approximate by

$$\nabla f = \begin{pmatrix} 1/\bar{M} \\ -\bar{D}/\bar{M}^2 \end{pmatrix}$$

As to $AVar(v, w)$ in Equation 6.8, we would use the multivariate analog of the usual S^2 in the univeriate case, taking advantage of Equation 6.1:

$$\widehat{\Sigma} = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})'$$

So now we have the estmated mean and variance, and can obtain a standard error for $\hat{\omega}$, from which we can form a confidence interval.

6.6 Your Turn

Your Turn: Show that in the scalar context,

$$Cov(X, Y) = E(XY) - EX \; EY$$

Your Turn: Show that in the vector context,

$$Cov(X) = E(XX') - (EX)(EX)'$$

7 Linear Statistical Models

> **i** Goals of this chapter:
>
> Here we bring together the concepts of previous chapters by presenting the *linear model*, one of the most fundamental techniques in statistics. It relies heavily on linear algebra, notably matrix inverse.

This chapter could have been titled, say, "Optimization, Part I," since many applications of linear algebra involve minimization or maximization of some kind, and this chapter will involve calculus derivatives. But the statistical applications of linear algebra are equally important.

7.1 Linear Regression through the Origin

Let's consider the **Nile** dataset built-in to R. It is a time series, one measurement per year.

```
head(Nile)
```

```
[1] 1120 1160  963 1210 1160 1160
```

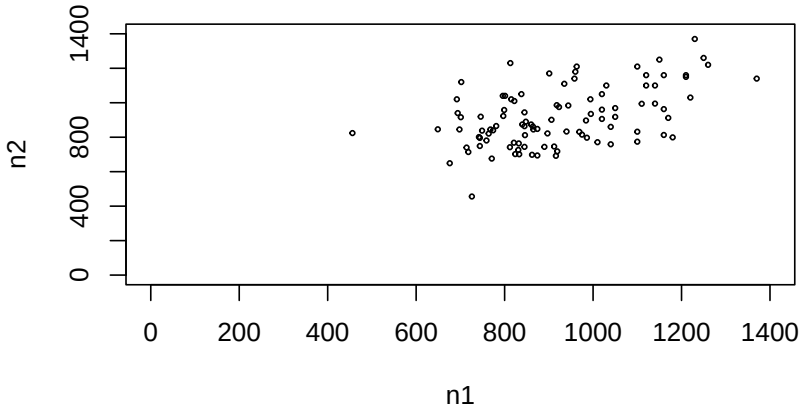
```
# predict current year from previous year?
n1 <- Nile[-(length(Nile))]
head(n1)
```

```
[1] 1120 1160  963 1210 1160 1160
```

```
n2 <- Nile[-1]
head(n2)
```

```
[1] 1160  963 1210 1160 1160  813
```

```
plot(n1,n2,cex=0.4,xlim=c(0,1400),yli=c(0,1400))
```



We would like to fit a straight line through that data point cloud. We might have two motivations for doing this:

- The line might serve as nice summary of the data.
- More formally, let C and V denote the current and previous year's measurements..Then the model

$$E(C|V) = \beta V$$

may be useful. Here the slope β is an unknown value to be estimated from the data.

Readers with some background in linear regression models should note that this assumption of a linear trend through the origin is the only assumption we are making here. Nothing on normal distributions etc.

> **!** Model Validity
>
> The great statistician George Box once said, "All models are wrong but some are useful." All data scientists should keep this at the forefronts of their minds.

7.1.1 Least squares approach

We wish to estimate β from our data, which we regard as a sample from the data generating process. Denote our data by $(C_i, V_i), i = 1, ..., 100$. Let $\widehat{\beta}$ denote our estimate. How should we obtain it?

This "box" is nothing all statistical scientists should keep at the forefronts of their minds. We are always working with sample data, subject to intersample variations. The quantity $\widehat{\beta}$ is a random variable.

Pretend for a moment that we don't know, say, C_{28}. Using our estimated β, our predicted value would be $\widehat{\beta}V_{28}$. Our squared prediction error would then be $(C_{28} - \widehat{\beta}W_{28})^2$.

Well, we actually do know C_{28} (and the others in our data), so we can answer the question:

In our search for a good value of $\widehat{\beta}$, we can ask how well we would predict our known data, using that candidate value of $\widehat{\beta}$ in our data. Our total squared prediction error would be

$$\sum_{i=1}^{100}[C_i - \widehat{\beta}V_i)^2$$

A natural choice for b would be the value that minimizes this quantity.

Why not look at the absolute value instead of the square? The latter makes the math flow well, as will be seen shortly.

7.1.2 Calculation

As noted, our choice for b will be the minimizer of

$$\sum_{i=1}^{100}(C_i - bV_i)^2$$

over all possible values of b. We then set $\widehat{\beta}$ to that minimizing value of b.

This is a straightforward calculus problem. Setting

$$0 = \frac{d}{db}\sum_{i=1}^{100}(C_i - bV_i)^2 = -2\sum_{i=1}^{100}(C_i - bV_i)V_i$$

and solving b, we find that

$$b = \frac{\sum_{i=1}^{n} C_i V_i}{\sum_{i=1}^{n} V_i^2}$$

7.1.3 R code

```
lm(n2 ~ n1-1)
```

```
Call:
lm(formula = n2 ~ n1 - 1)

Coefficients:
  n1
0.98
```

This says, "Fit the model $E(C|V) = \beta V$ to the data, with the line constrained to pass through the origin." The constraint is specified by the -1 term.

We see that the estimate regression line is

$$E(C|V) = 0.98V$$

7.2 Linear Regression Model with Intercept Term

Say we do not want to constrain the model to pass the line through the origin. Our model is then

$$E(C|V) = \beta_0 + \beta_1 V$$

where we now have two unknown parameters to be estimated.

7.2.1 Least-squares estimation, single predictor

Our sum of squared prediction errors is now

$$\sum_{i=1}^{100}[O_i - (b_0 + b_1 V_i)]^2$$

This means setting two derivatives to 0 and solving. Since the derivatives involve two different quantities to be optimized, b_0 and b_1, the derivatives are termed *partial*, and the ∂ symbol is used instead of 'd'.

$$0 = \frac{\partial}{\partial b_0}\sum_{i=1}^{100}[C_i - (b_0 + b_1 V_i)]^2 \tag{7.1}$$

$$= -2\sum_{i=1}^{100}[C_i - (b_0 + b_1 V_i)] \tag{7.2}$$

and

$$0 = \frac{\partial}{\partial b_1}\sum_{i=1}^{100}[C_i - (b_0 + b_1 V_i)]^2 \tag{7.3}$$

$$= -2\sum_{i=1}^{100}[C_i - (b_0 + b_1 V_i)]V_i \tag{7.4}$$

We could then solve for the b_i, but let's go straight to the general case.

7.3 Least-Squares Estimation, General Number of Predictors

7.3.1 Nile example

As we have seen, systems of linear equations are natural applications of linear algebra. The equations setting the derivatives to 0 can be written in matrix terms as

$$\begin{pmatrix} \sum_{i=1}^{n} C_i \\ \sum_{i=1}^{n} C_i V_i \end{pmatrix} = \begin{pmatrix} 100 & \sum_{i=1}^{n} V_i \\ \sum_{i=1}^{n} V_i & b_1 \sum_{i=1}^{n} V_i^2 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$$

Actually, that matrix equation can be derived more easily by using matrices to begin with:

Define S and T:

$$S = \begin{pmatrix} C_1 \\ C_2 \\ ... \\ C_{100} \end{pmatrix}$$

and

$$T = \begin{pmatrix} V_1 \\ V_2 \\ ... \\ V_{100} \end{pmatrix}$$

Then our linear assumption, $E(C|V) = \beta_0 + \beta_1 V$, applied to S and T, is

$$E(S|T) = A\beta$$

where

$$A = \begin{pmatrix} 1 & V_1 \\ 1 & V_2 \\ ... & ... \\ 1 & V_{100} \end{pmatrix}$$

and $\beta = (\beta_0, \beta_1)'$.

Our predicted error vector, using our candidate estimate b of β, is very simply expressed:

$$S - Ab$$

And since for any column vector u, the sum of its squared elements is

$$u'u$$

our sum of squared prediction errors is

$$(S - Ab)'(S - Ab) \tag{7.5}$$

Now how we will minimize that matrix expression with respect to the vector b? That is the subject of the next section.

7.3.2 Matrix derivatives

The (column) vector of partial derivatives of a scalar quantity is called the *gradient* of that quantity. For instance, with

$$u = 2x + 3y^2 + xy$$

we have that its gradient is

$$\begin{pmatrix} 2 + y \\ 6y + x \end{pmatrix}$$

With care, we can compute gradients entirely at the matrix level, using easily derivable properties, without ever resorting to returning to the scalar expressions. Let's apply them to the case at hand in the last section,

$$(S - Ab)'(S - Ab) \tag{7.6}$$

7.3.3 Differentiation purely in matrix terms

It can be shown that for a column vector a,

$$\frac{d}{da}a'a = 2a \tag{7.7}$$

Equation 7.6 is indeed of the form $a'a$, but the problem here is that a in turn is a function of b, This calls for the Chain Rule, which does exist at the matrix level:

For example if $u = Mv + w$, with M and w constants (i.e. not functions of v, then

$$\frac{d}{dv}u'u = 2M'u$$

We must keep in mind that we are working with vectors and matrices, so that $M'u$, say rxs times s x1, is conformable matrix multiplication while uM' is mathematical nonsense.

In our case at hand, we have $M = -A$ and $w = S$, so that

$$\frac{d}{db}[(S - Ab)'(S - Ab) = -2A'(S - Ab)$$

So, set

$$0 = A'(S - Ab) = A'S - A'Ab$$

yield our minimizing b:

$$\hat{\beta} = (A'A)^{-1}A'S \tag{7.8}$$

providing the inverse exists (more on this in the next chapter).

Let's check this with the Nile example:

```
A <- cbind(1,n1)
S <- n2
Ap <- t(A)  # R matrix transpose
solve(Ap %*% A) %*% Ap %*% S  # R matrix inverse
```

```
          [,1]
   452.7667508
n1   0.5043159
```

```
# check via R
lm(n2 ~ n1)
```

```
Call:
lm(formula = n2 ~ n1)

Coefficients:
(Intercept)           n1
   452.7668       0.5043
```

```
det(Ap %*% A)
```

```
[1] 277463876
```

Nonzero! So $(A'A)^{-1}$ does exist, as we saw.

7.3.4 The general case

Say our data consists of n points, each of which is of length p. Write the j^{th} element of the i^{th} data point as X_{ij}. Then set

$$A = \begin{pmatrix} 1 & X_{11} & ... & X_{p1} \\ 1 & X_{21} & ... & X_{p2} \\ ... & ... & ... & ... \\ 1 & X_{n1} & ... & X_{np} \end{pmatrix} \tag{7.9}$$

Continue to set S to the length-n column vector of our response variable. Our model is

$$E(S|A) = A\beta$$

for an unknown vector β of length $p+1$. Our estimated of that vector based on our data will be denoted

$$b = (b_0, b_1, ..., b_p)'$$

Then, using the same reasoning as before, we have the minimizing value of b:

$$\hat{\beta} = (A'A)^{-1}A'S \tag{7.10}$$

again providing that the inverse exists.

7.3.5 Example: mlb1 data

As an example, let's take the **mlb1** from my qeML ('Quick and Easy Machine Learning package. The data is on major league baseball players. We will predict weight from height and age.

Dataset kindly provided by the UCLA Dept. of Statistics

```
library(qeML)
data(mlb1)
head(mlb1)
```

```
         Position Height Weight   Age
1         Catcher     74    180 22.99
2         Catcher     74    215 34.69
3         Catcher     72    210 30.78
4  First_Baseman     72    210 35.43
5  First_Baseman     73    188 35.71
6 Second_Baseman     69    176 29.39
```

```
ourData <- as.matrix(mlb1[,-1]) # must have matrix to enable %*%
head(ourData)
```

```
  Height Weight   Age
1     74    180 22.99
2     74    215 34.69
3     72    210 30.78
```

```
4     72    210 35.43
5     73    188 35.71
6     69    176 29.39
```

```
A <- cbind(1,ourData[,c(1,3)])
Ap <- t(A)
S <- as.vector(mlb1[,3])
solve(Ap %*% A) %*% Ap %*% S
```

```
                [,1]
       -187.6381754
Height    4.9235994
Age       0.9115326
```

```
# check via R
lm(Weight ~ .,data=mlb1[,-1])
```

```
Call:
lm(formula = Weight ~ ., data = mlb1[, -1])

Coefficients:
(Intercept)       Height          Age
  -187.6382       4.9236       0.9115
```

So, if we have a player of known height and age, we would predict the weight to be

-187.6382 + 4.9236 x height + 0.9115 x age

7.3.6 Homogeneous variance case

In addition to

$$E(S|A) = A\beta$$

it is often assumed that

In some applications, A is actually chosen by an experimenter, so that it is not random. But even in the random case, it is standard to condition on A. By Equation 10.2, a conditional confidence interval, say, at the 95% level also has the level unconditionally.

$$Cov(S|A) = \sigma^2 I$$

where σ is an unknown constant to be estimated from the data.

So $Var(S_i) = \sigma^2$ for all i. It is usually not a realistic assumption. Say for instance we are predicting human weight from height. There should be more variation in weight among taller people than above shorter people. But it's a simplifying assumption without really good alternatives, so it is commonly used.

And in that setting, our formulas from {**?@sec-covar**} come in handy, as follows.

Recall that $\widehat{\beta}$ is our estimate of the unknown population parameter β, based on our random sample data. But that means that $\widehat{\beta}$ is a random vector, and thus has a covariance matrix. Using Equation 6.3 and setting $R = (A'A)^{-1}A'\$$, we have

$$Cov(\widehat{\beta}|A) = RCov(S|A)R' = R\sigma^2 IR' = \sigma^2 RR' = \sigma^2(A'A)^{-1} \tag{7.11}$$

Classical statistical formulas use this relation to find standard errors for $\widehat{\beta}_i$ etc.

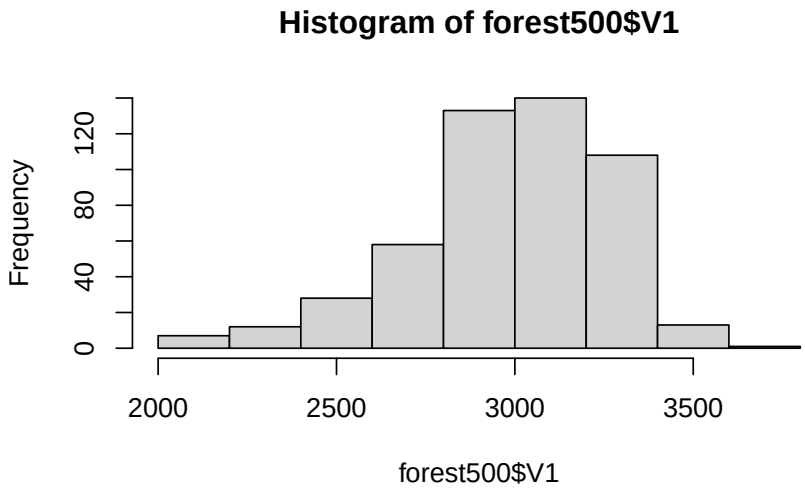
7.4 Update Formulas

One important theme in developing prediction models (linear regression, neural networks etc.) is the avoidance of *overfitting*, meaning that we fit an overly elaborate model to our data. We simply are estimating too many things for the amount of data we have, "spreading our data too thin."

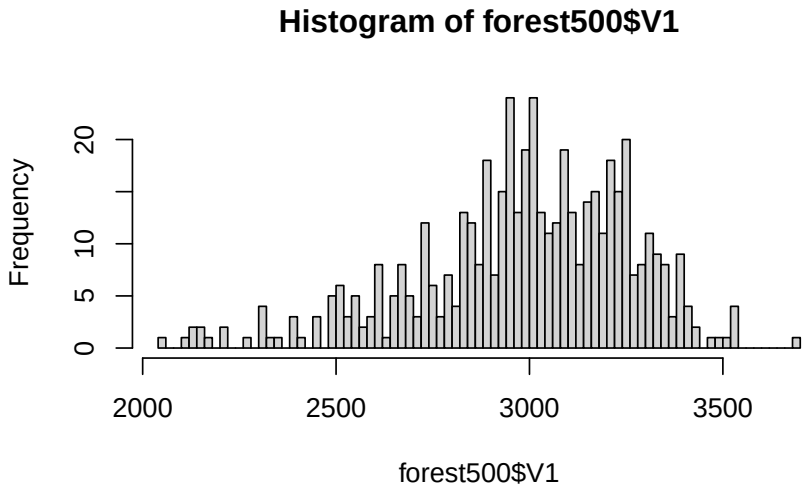
A common example is using too many predictor variables, so that we are estimating a large number of coefficients β_i.

Or we may draw a histogram with too many bins:

```
library(qeML)
data(forest500)  # data on forest ground cover
hist(forest500$V1,breaks=10)
```




```
hist(forest500$V1,breaks=100)
```



In a histogram, we are estimating the heights of the bins. With 10 bins we obtained a smooth graph, but with 100 bins it became choppy. So again, we are estimating too many things, given the capacity of the data.

A histogram is an estimate of the probability density function of the observed random variable. Having more, thus narrower, bins reduces bias but increases variance.

With larger datasets, we can use more predictor variables, more histogram bins, and so on. The question then arises is, for instance, *How many* predictors, *how many* bins and so on, can we afford to use with our given data?

The typical solution is to fit several models of different complexity, then choose the one that predicts the best. But we must do this evaluation on "fresh" data; we should not predict on the same data on which we fitted our model.

We thus rely on partitioning our data, into a *training set* to which we fit our model, and a *test set*, on which we predict using the fitted model. We may wish to do this several times.

A special case is the Leaving One Out method, in which the holdout set size is 1. It might go like this, for a dataset d:

```
sumErrs = 0
for i = 1,...,n  # dataset has n datapoints
   fit lm to d[-i,]
   use result to predict d[i,]
   add prediction error to sumErrs
return sumErrs
```

This can become computationally challenging, as we would need to refit the model each time. Each call to **lm** involves a matrix inversion (equivalent), and we must do this n times.

It would be nice if we could have an "update" formula that would quickly recalculate the model found on the full dataset. Then e would need to perform matrix inversion just once. In the case of linear models, such a formula exists, in the Sherman-Morrison-Woodbury relation:

Given an invertible matrix B and row vectors u and v having lengths equal to the number of columns of B. form the matrix

$$C = B + uv'$$

Then C^{-1} exists and is equal to

Note that the quantity uv' is a square matrix the size of B, so the sum and product make sense.

$$B^{-1} - \frac{1}{1 + v'B^{-1}u} B^{-1}(uv')B^{-1}$$

Now, how can we apply this to the Leave One Out method? In the matrix A in Equation Equation 7.9, we wish to remove row i; call the result A_{-i}. Our new version of $A'A$ is then

$$A'_{-i}A_{-i}$$

So our main task is to obtain

$$(A'_{-i}A_{-i})^{-1}$$

by updating $(A'A)^{-1}$, which we already have from our computation in the full dataset.

We can do this as follows. Denote row i of A by a_i, and set

$$u = -a_i, v = a_i$$

To show why these choices for u and v work, consider the case in which we delete the last row of A. (The analysis would be similar for other cases.) Write the latter as a partitioned matrix,

$$\begin{pmatrix} A_{(-n)} \\ a_n \end{pmatrix}$$

We pretend it is a 2x1 "matrix," and A' is then "1x2":

$$A' = \begin{pmatrix} A'_{(-n)} & a'_n \end{pmatrix}$$

Thus

$$A'A = A'_{-i}A_{-i} + a_n a'_n$$

yielding

$$A'_{-i}A_{-i} = A'A - a_n a'_n$$

just what we need for Sherman-Morrison-Woodbury: With $B = A'A$, we have

$$[A'_{-i}A_{-i}]^{-1} = B^{-1} + \frac{1}{1 - a'_i B^{-1} a_i} B^{-1}(a_i a'_i)B^{-1}$$

Let's check it:

```
a <- rbind(c(1,3,2),c(1,0,5),c(1,1,1),c(1,9,-3))
apa <- t(a) %*% a
apai <- solve(apa)
a2 <- a[-2,]
apa2 <- t(a2) %*% a2
apa2i <- solve(apa2)
# prepare for S-M-W
adel <- matrix(a[2,],ncol=1)
w1 <- 1/(1 - t(adel) %*% apai %*% adel)
w1 <- as.numeric(w1)
uvt <- adel %*% t(adel)
w2 <- apai %*% uvt %*% apai
# S-M-W says this will be apa2i
apai + w1 * w2
```

```
           [,1]       [,2]      [,3]
[1,]  3.4140625 -0.7109375 -1.015625
[2,] -0.7109375  0.1640625  0.234375
[3,] -1.0156250  0.2343750  0.406250
```

```
apa2i
```

```
           [,1]       [,2]      [,3]
[1,]  3.4140625 -0.7109375 -1.015625
[2,] -0.7109375  0.1640625  0.234375
[3,] -1.0156250  0.2343750  0.406250
```

7.5 Your Turn

Your Turn: Show Equation 7.7. Write $a = (a_1, ..., a_k)'$ and find the gradient "by hand." Compare to $2a$.

Your Turn: Show that

$$\frac{d}{du} u'Qu = 2Qu$$

for a constant symmetric matrix Q and a vector u. ($u'Qu$ is called a *quadratic form*.)

8 Matrix Rank

> **Goals of this chapter:**
>
> Many matrices are noninvertible. The subject of this chapter. matrix *rank* is closely tied to the invertibility issue, and is central to understanding matrix applications.

In our computations in the latter part of the last chapter, we added a proviso that $(A'A)^{-1}$ exists. In this chapter, we'll present a counterexample, which will naturally lead into our covering matrix rank, and in the next chapter,the basics of vector spaces.

8.1 Example: Census Data

This dataset is also from **qeML**. It is data for Silicon Valley engineers in the 2000 Census. Let's focus on just a few columns.

```
data(svcensus)
head(svcensus)
```

```
       age     educ occ wageinc wkswrkd gender
1 50.30082 zzzOther 102   75000      52 female
2 41.10139 zzzOther 101   12300      20   male
3 24.67374 zzzOther 102   15400      52 female
4 50.19951 zzzOther 100       0      52   male
5 51.18112 zzzOther 100     160       1 female
6 57.70413 zzzOther 100       0       0   male
```

```
svc <- svcensus[,c(1,4:6)]
head(svc)
```

```
       age wageinc wkswrkd gender
1 50.30082   75000      52 female
2 41.10139   12300      20   male
3 24.67374   15400      52 female
4 50.19951       0      52   male
5 51.18112     160       1 female
6 57.70413       0       0   male
```

```
lm(wageinc ~ .,data=svc)
```

```
Call:
lm(formula = wageinc ~ ., data = svc)

Coefficients:
(Intercept)          age      wkswrkd   gendermale
   -29384.1        496.7       1372.8      10700.8
```

So, we estimate that, other factors being equal, men about paid close to $11,000 more than women. This is a complex issue, but for our purposes here, how did **gender** become **gendermale**, no explicit mention of women?

Let's try to force the issue:

```
svc$man <- as.numeric(svc$gender == 'male')
svc$woman <- as.numeric(svc$gender == 'female')
svc$gender <- NULL
head(svc)
```

```
       age wageinc wkswrkd man woman
1 50.30082   75000      52   0     1
2 41.10139   12300      20   1     0
3 24.67374   15400      52   0     1
4 50.19951       0      52   1     0
5 51.18112     160       1   0     1
6 57.70413       0       0   1     0
```

```
lm(wageinc ~ .,data=svc)
```

```
Call:
lm(formula = wageinc ~ ., data = svc)

Coefficients:
(Intercept)          age      wkswrkd          man        woman
   -29384.1        496.7       1372.8      10700.8           NA
```

Well, we couldn't force the issue after all. Why not? We hinted above that $A'A$ may not be invertible. Let's take a look.

```
A <- cbind(1,svc[,-2])
A <- as.matrix(A)
ApA <- t(A) %*% A
ApA
```

```
               1        age  wkswrkd      man    woman
1        20090.0   794580.7   907240  15182.0   4908.0
age     794580.7 33956543.6 35869770 600860.8 193719.9
wkswrkd 907240.0 35869770.5 45252608 692076.0 215164.0
man      15182.0   600860.8   692076  15182.0      0.0
woman     4908.0   193719.9   215164      0.0   4908.0
```

Is this matrix invertible? Let's apply the elementary row operations introduced in Section 5.3.1:

```
library(pracma)
rref(ApA)
```

```
        1 age wkswrkd man woman
1       1   0       0   0     1
age     0   1       0   0     0
wkswrkd 0   0       1   0     0
man     0   0       0   1    -1
woman   0   0       0   0     0
```

Aha! Look at that row of 0s! The row operations process ended prematurely. This matrix has no inverse. We say that the matrix has *rank* 4 – 4 nonzero rows – when it needs to be 5; we also say that the matrix is not of *full rank*. We will return to this point shortly, but first we formalize introduce the row operations process.

8.2 Reduced Row Echelon Form (RREF) of a Matrix

We formalize and extend Section 5.3.1.

8.2.1 Elementary row operations

These are:

- Multiply a row by a nonzero constant.
- Add a multiple of one row to another.
- Swap two rows.

Again, each operation can be implemented via pre-multiplying the given matrix by a corresponding elementary matrix. The latter is the result of applying the given operation to the identity matrix I. For example, here is the matrix corresponding to swapping rows 2 and 3:

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

For example:

```
e <- rbind(c(1,0,0),c(0,0,1),c(0,1,0))
e
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    0    1
[3,]    0    1    0
```

```
a <- matrix(runif(12),nrow=3)
a
```

```
          [,1]      [,2]      [,3]      [,4]
[1,] 0.5994517 0.1708624 0.4385588 0.6426890
[2,] 0.8465690 0.2906282 0.9271512 0.5402207
[3,] 0.7098337 0.8616611 0.6161337 0.7487169
```

```
e %*% a  # matrix mult. is NOT e * a!
```

```
          [,1]      [,2]      [,3]      [,4]
[1,] 0.5994517 0.1708624 0.4385588 0.6426890
[2,] 0.7098337 0.8616611 0.6161337 0.7487169
[3,] 0.8465690 0.2906282 0.9271512 0.5402207
```

Yes, rows 2 and 3 were swapped.

8.2.2 The RREF

By applying these operations to a matrix A, we can compute its *reduced row echelon form* A_{rref}, which is defined by the following properties:

- Each row, if any, that consists of all 0s is at the bottom of the matrix.
- The first nonzero entry in a row, called the *pivot*, is a 1.
- Each pivot will be to the right of the pivots in the rows above it.
- All entries below a pivot are 0.
- Each pivot is the only nonzero entry in its column.

The reader should verify that the matrix at the end of Section 8.1 has these properties.

8.2.3 Review of Section 5.3.1

- Each row operation can be performed via premultiplication by an elementary matrix. Each such matrix is invertible, and its inverse is an elementary matrix.
- Thus

$$B_{rref} = E_k ... E_2 E_1 B \tag{8.1}$$

for a sequence of invertible elementary matrices.

- And

  $B = (E_1)^{-1}(E_2)^{-1}...(E_k)^{-1}B_{rref}$,

  where each $(E_i)^{-1}$ is itself an elementary row operation.

8.3 Rank andLinear Dependent/Independent Vectors

8.3.1 Motivation

Recall that in the nonfull rank example we presented in Section 5.4, one row was double another,

row2 + 2 row1 = 0

In the census example,

```
head(A)
```

```
  1      age wkswrkd man woman
1 1 50.30082      52   0     1
2 1 41.10139      20   1     0
3 1 24.67374      52   0     1
4 1 50.19951      52   1     0
5 1 51.18112       1   0     1
6 1 57.70413       0   1     0
```

the sum of the last two columns of A was equal to the first column:

column1 - column4 - column5 = 0

Write this more fully as a formal linear combination of the columns of this matrix:

1 column1 + 0 column2 + 0 column3 + (-1) column4 + (-1) column5 = 0

The vector of coefficients is (1,0,0,-1,-1).

8.3.2 Formal definitions

Definition: We say a set of vectors is *linearly dependent* if some linear combination of them (excluding the trivial case in which all the coefficients are 0) equals 0, as in both cases above. If no nontrival linear combination of the vectors is 0, we say the vectors are *linearly independent*.

So, here is the formal definition of rank:

Definition The *rank* of a matrix B is its maximal number of linearly independent rows.

8.3.3 Relevance

> **!** A Central Concept
>
> As we will see, linear dependence of the rows or columns is the culprit in non-full rank matrices. This is a major issue for linear models, in which an inverse is necessary. Moreover, in many cases a matrix will technically be invertible, but "approximately" nonfull rank. This can wreak havoc in linear models.
> The reader can see, then, that understanding of matrix rank is key. That will be our initial motivation for learning about vector spaces in the next chapter, which will in turn open the door to many central concepts and methods of linear algebra.

8.4 Some Properties of Rank

In Section 8.1, we found the matrix $A'A$ to be noninvertible, as its RREF had a row of 0s. We mentioned a relation to rank, but didn't formalize it, which we will do now.

8.4.1 Yes, one can tell the rank of a matrix by looking at its RREF

This is an important property:

Theorem 8.1 (Rank Is Preserved in Pre-/Post-multiplication by Invertible Matrices)**.** *Let A be any matrix, and let V and W be square, invertible matrices with sizes conformable with the products below. Then the ranks of VA and AW are equal to that of A.*

Proof. Say A has rank r. Then there exist r linearly independent columns of A. For convenience of notation, say these are the first r columns, and call them $c_1, ..., c_r$. Then the first r columns of WA are $Wc_1, ..., Wc_r$, by matrix partitioning.

Note that for a vector x, $Vx = 0$ if and only if $x = 0$. (Just multiply $Vx = 0$ on the left by V^{-1}.) So a linear combination $\lambda_1 c_1 + ... + \lambda_r c_r$ is 0 if and only if the corresponding linear combination $\lambda_1 Vc_1 + ... + \lambda_r Vc_r$ is 0. Thus VA has rank r.

The argument for the case of AW is identical, this time involving rows of A.

□

□

This then gives us:

Corollary: The rank of a matrix B is equal to the rank of its RREF, B_{rref}. Thus we can determine the rank of B by counting the nonzero rows of its RREF.

Proof:

The Theorem 8.1 above theorem says that pre- or postmultiplying B will not change its rank. Then invoke Equation 8.1.

8.4.2 Row rank and column rank

We have defined the rank of a matrix to be the number of maximally linearly independent rows. We'll now call that the *row rank*, and define the *column rank* to be the number of maximally linearly independent columns.

Theorem: The column rank of a matrix B is equal to that of its RREF.

Proof: Simply follow the steps in the proof for the row rank case above. (Or apply that case to B'.)

8.4.3 Rank is bounded above by the "narrow" dimension of the matrix

In Section 8.1, we found the matrix $A'A$ to not be of full rank. It is surprising that so was A:

```
qr(ApA)$rank  # qr is a built-in R function
```

```
[1] 4
```

```
qr(A)$rank
```

```
[1] 4
```

This is rather startling. A has over 20,000 rows — yet only 4 linearly independent ones? But it follows from this fact:

There are many subsets of 4 rows that are linearly independent. But no sets of 5 or more are linearly independent.

For any rxs matrix G, the rank of G is less than or equal to $\min(r, s)$.

But first, a more far-reaching theorem that will also imply the above:

Theorem: The row rank and column rank of a matrix B are equal.

Proof: First, it's clear from looking at the form of the RREF that both the row rank and the column rank of an RREF are equal to the number of rows with leading 1s. (Again, see the example at the end of Section 8.1 to picture this.) In other words,

$$rowrank(B_{rref}) = colrank(B_{rref})$$

But we found earlier that

$$rowrank(B) = rowrank(B_{rref})$$

and

$$colrank(B) = colrank(B_{rref})$$

The result follows.

And thus:

Corollary: The (common value of) row and column ranks of an mxn matrix B must be less than or equal to the minimum of the number of rows and columns of B .

8.5 Your Turn

Your Turn: Consider an mxn matrix A with $m \geq n$. Then consider the partitioned matrix

$$B = \begin{pmatrix} A \\ I \end{pmatrix}$$

where $I is the nxn identity matrix. Prove that B is of full rank.

Your Turn: Consider the three basic elementary matrices discussed here: Swap rows i and j; multiply row i by a constant

b; adding c times row i to row j, for a constant c. Find general formulas for the determinants of the three matrices.

Your Turn: Prove that if the matrix A has a 0 row, then $det(A) = 0$.

9 Vector Spaces

$$x^2$$

> **Goals of this chapter:**
>
> As noted earlier, the two main structures in linear algebra are matrices and vector spaces. We now introduce the latter, a bit more abstract than matrices but even more powerful. We lay the crucial foundation in this chapter, then reap the benefits in the applications in the remaining chapters.

9.1 Review of Matrix Rank Properties

In the last chapter, we presented the concepts of matrix row and column rank, defined to be the maximal number of linearly independent combinations of the rows or columns of the matrix, respectively. We proved that

For any matrix B,

$$\text{rowrank}(B) = \text{colrank}(B) = \text{rowrank}(B_{rref}) = \text{colrank}(B_{rref})$$

We can say something stronger:

Theorem: Let V denote the *span* of the rows of B,i.e. the set of all possible linear combinations of rows of B. Define V_{rref} similarly. Then

$$V = V_{rref}$$

The analogous result holds for columns.

Proof: Actually, we already proved this in the proof regarding rank in Section 8.4.1, in which we wrote, "any nonzero linear combination of rows in B_{rref} will correspond to a nonzero linear combination of the rows of B,'' and vice versa. This showed a one-to-one correspondence between the two sets of linear combinations.

So, not only do the two matrices have the same maximal numbers of linearly independent rows, they also generate *the same linear combinations* of those rows.

The sets V and V_{rref} are called the *row spaces* of the two matrices, and yes, they are examples of vector spaces, as we will now see.

9.2 Vector Space Definition

A set of objects W is called a *vector space* if it satisfies the following conditions:

- Some form of addition between vectors u and v, denoted $u + v$, is defined in W, with the result that $u+v$ is also in W. We describe that latter property by saying W is *closed* under addition.
- There is a unique element called "0' ' such that $u + 0 = 0 + u = u$.
- Some form of scalar multiplication is defined, so that for any number c and and u in W, cw exists and is in W. We describe that latter property by saying W is *closed* under scalar multiplication
- This being a practical book with just a dash of theory, we'll skip the remaining conditions, involving algebraic properties such as commutativity of addition ($u + v = v + u$).

9.2.1 Examples

9.2.2 R^n

In the vast majority of examples in this book, our vector space will be R^n.

Here R represents the set of all real numbers, and R^n is simply the set of all vectors consisiting of n real numbers. In an mxk matrix the rows are members of so R^m and the columns are in R^k.

9.2.3 The set C(0,1) of all continuous functions on the interval [0,1]

No surprises here. Vector addition and scalar multiplication are done as functions. If say u is the squaring function and v\$ is the sine function, then

$$3u = 3x^{0.5}$$

and

$$u + v = x^{0.5} + \sin(x)$$

9.2.4 The set $RV(\Omega)$ of all random variables defined on some probability space Ω

Consider the example in {Section 7.3.5} on major league baseball players. We choose a player at random. Denote weight, height and age by W, H and A.

Vector addition and scalar multiplication are defined in a straightforward manner. For instance, the sum of H and A is simply height + age. This may seem like a rather nonsensical sum, but it fits the technical definition, and moreover, we have already been doing things like this! This after all is what is happening in our prediction expression from that section,

$$\text{predicted weight} = -187.6382 + 4.9236H + 0.9115A$$

In fact, in this vector space, the above is a linear combination of the random variables 1, H and A. Note that random variables such as $HW^{1.2}$ and so on are also members of this vector space, essentially any function of W, H and A.

9.3 Subspaces

Say W_1 a subset of a vector space W, such that W_1 is closed under addition and scalar multiplication. W_1 is called a *subspace* of W. Note that a subspace is also a vector space in its own right.

9.3.1 Examples

R^3 and R^n:

For instance, take W_1 to be all vectors of the form (a,b,0). Clearly, W_1 is closed under addition and scalar multiplication.

Another subspace of R^3 is the set of vectors (a,a,b), i.e. those vectors whose first two element are equal. What about vectors of the form (a,b,a+b)? Yes.

We saw earlier that the *row space* of an mxn matrix A, consisting of all linear combinations of the rows of A, is a subspace of R^m. Similarly, the *column space* is a subspace of R^n.

$C(0,1)$:

One subspace is the set of all polynomial functions. Again, the sum of two polynomials is a polynomial, and the same holds for scalar multiplication, so the set of polynomials is closed under those operations, and is a subspace.

$RV(\Omega)$:

The set of all random variables that are functions of H, say, is a subspace. Another subspace is the set of all random variable with mean 0.

9.3.2 Span of a set of vectors

The *span* of a set of vectors $G = v_1, ..., v_k$ is the set of all linear combinations of those vectors. It's a subspace of the main space.

9.4 Basis

Consider a set of vectors $u_1, ...u_r$ in a vector space W. Recall that the span of these vectors is defined to be the set of all linear combinations of them. In verb form, we say that $u_1, ...u_r$ *spans* W if we can generate the entire vector space from those vectors via linear combinations. It's even nicer if the vectors are linearly independent:

We say the vectors $u_1, ...u_r$ in a vector space W form a *basis* for W if they are linearly independent and span W.

9.4.1 Examples

R^3:

The vectors (1,0,0), (0,1,0) and (0,0,1) are easily seen to be a basis here. They are linearly independent, and clearly span R^3. For instance, to generate (3,1,-0.2), we use this linear combination:

(3,1,-0.2) = 3 (1,0,0) + 1 (0,1,0) + (-0.2) (0,0,,1)

Our convention has been that vectors are considered in matrix terms as column vectors by default. However, in nonmatrix contexts, it will be convenient to write R^n vectors as rows.

But bases are not unique; for instance, the set (1,0,0, (0,1,0), (0,1,1) works equally well as a basis for this space, as are (infinitely) many others..

A basis for the subspace of vectors of the form (a,a,b) is (1,1,0) and (0,0,1).

$C(0,1)$:

Alas, there is no finite basis here. Even infinite ones have issues in their mathematical formulation.

For those with a background in mathematical analysis, here is how the problem is handled. One starts with a subspace that is *dense* in the full space, i.e. any vector in the full space can be approximated to any precision by some linear combination of vectors in the dense set. For instance, the famoua Stone-Weierstrauss Theorem says that the set of all polynomials is dense in $C(0,1)$. One then defines the dense set to be a "basis.'' The use of trig functions as the dense set is much more common than use of polynomials, in the familiar *Fourier series*.

$RV(\Omega)$:

The situation here is the same as for C(0,1).

9.5 Dimension

Geometrically, we often refer to what is called R^3 here as "3-dimensional.'' We extend this to general vector spaces as follows:

The *dimension* of a vector space is the number of vectors in any of its bases.

There is a bit of a landmine in that definition, as it presumes that all the bases do consist of the same number of vectors. This is true, but must be proven. Here we follow the elegant proof AF Beardon (*Algebra and Geometry*, 2005, Cambridge).

Theorem: The number of vectors is the same in every basis.

Proof: Consider two bases, $b_u = u_1, ..., u_m$ and $b_v = v_1, ..., v_n$ By definition, each u_i in b_u can be represented by b_v and vice versa:

$$u_i = r_{i1}v_1 + ... + r_{in}v_n = \sum_{q=1}^{n} r_{iq}v_q \tag{9.1}$$

$$v_j = s_{j1}u_1 + ... + s_{jm}u_m \sum_{w=1}^{m} s_{jw}u_w \tag{9.2}$$

Substituting the second equation into the first, we have

$$u_k = \sum_{q=1}^{n} r_{kq} \sum_{w=1}^{m} s_{qw}u_w = \sum_{q=1}^{n} \sum_{w=1}^{m} r_{kq}s_{qw}u_w$$

Since the u_w form a basis, then the below Your Turn problem on the uniqueness of coefficients in a basis representation says we can equate coefficients on both sides of the last equation. The coefficient of u_k on the right side is

$$\sum_{q=1}^{n} r_{kq}s_{qk}$$

whereas on the left side, it is 1. So,

$$1 = \sum_{q=1}^{n} r_{kq} s_{qk}$$

Summing over k, we obtain

$$n = \sum_{k=1}^{m} \sum_{q=1}^{n} r_{kq} s_{qk}$$

But going through the same steps as above, but now substituting Equation 9.1 into Equation 9.2, we would have

$$m = \sum_{p=1}^{m} \sum_{q=1}^{m} r_{pq} s_{qp}$$

The sums in the last two equations are the same!

Therefore $m = n$, i.e. the two bases are of the same size.

9.6 Your Turn

Your Turn: Say U and V are subspaces of W. Show that $U \cap V$ is also a subspace, and that its dimension is at most the minimum of the dimensions of U and V. Show by counterexample tha result does not hold for union.

Your Turn: Citing the properties of expected value, E(), show that the set of all random variable with mean 0 latter set is indeed a subspace of $RV(\Omega)$.

Your Turn: Prove that the coefficients in basis representations are unique. In other words, in a representation of the vector x in terms of a basis $u_1, ..., u_n$,

10 Inner Product Spaces

Goals of this chapter:

The usefulness of vector spaces is greatly enhanced with the addition of an *inner product* structures. We motivate and define such structures here, and present applications. Among other things, we will analyze a method for removing racial, gender etc. bias in machine learning algorithms.

10.1 Geometric Aspirations

You may recall from your high school geometry course the key concept of perpendicularity, represented by the symbol. You may also recall that in 2-dimensional space, given a point P and a line L, the line drawn from point P to the closest point P' within L is perpendicular to L. The same is true if L is a plane. The point P' is called the *projection* of P onto L.

This was shown in this book's cover, shown here:

The point at the end of the green vector is projected onto the mustard-colored plane, producing the red vector. It in turn is projected onto the blue line. There are right angles in each case.

The early developers of linear algebra wanted to extend such concepts to abstract vector spaces. This aids intuition, and has very powerful applications.

10.2 Definition

You may have seen dot products in a course on vector calculus or physics. For instance, the dot product of the vectors (3,1,1.5)' and (0,5,6)' is

3x0 + 1x5 + 1.5x6 = 14

This in fact is a standard inner product on R^3, but the general definition is as follows.

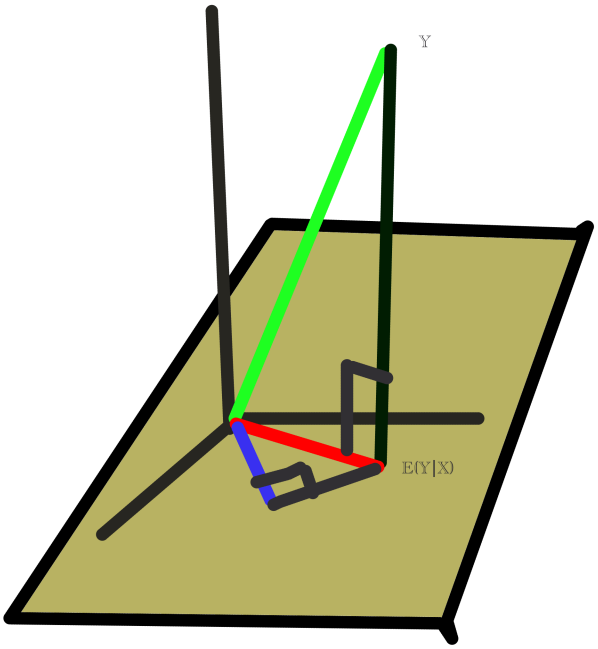


Figure 10.1: Projections

An *inner product* on a vector space V, denoted by the "angle brackets'' notation $< u, v >$, is a function with two vectors as arguments and a numerical output, with the following properties:

- $< u, v >=< v, u >$
- The function is bilinear:

$$< u, av + bw >= a < u, v > +b < u, w >$$

- $< u, u > \geq 0$, with equality if and only if $u = 0$.

10.3 Examples

R^n:

As noted, ordinary dot product is the most common inner product on this space.

$< (a_1, ..., a_n), (b_1, ..., b_n >= a_1 b_1 + ... + a_n b_n$

An n x n symmetric matrix M is called *positive definite* if

$$w' M w > 0$$

for all nonzero vectors w. (And the term used is *nonnegative definite* if $>$ is replaced by \$ \$.) Show that $< u, v >= u' M v$ is an inner product on R^n.)

C(0,1):

One inner product on this space is

$$< f, g >= \int_0^1 f(t) g(t)\ dt$$

For instance, with $f(t) = t^2$ and $g(t) = \sin(t)$, the inner product can be computed with R:

```r
f <- function(t) t^2
g <- function(t) sin(t)
fg <- function(t) f(t) * g(t)
integrate(fg,0,1)
```

```
0.2232443 with absolute error < 2.5e-15
```

This clearly fits most requirements for inner products, but what about $< f, f >= 0$ only if $f = 0$? A non-0 f will have $f^2(t) > 0$ for at least one t, and by continuity, $f^2(t) > 0$ on an interval containing that t, thus making a nonzero contribution to the integral and thus to the inner product.

Note that the 0 vector in this space is the function that is identically 0, not just 0 at some points

$RV(\Omega)$:

From here on, we'll restrict to the vector space of all random variables on Ω having finite variance. We define

$$< X, Y >= E(XY)$$

The properties of expected value, e.g. linearity, show most of the requirements for an inner product hold. But again, we need to show that

$$< X, X >= E(X^2) > 0$$

for any nonzero X, i.e. any X such that $P(X = 0) < 1$.

10.4 Norm of a Vector

This concept extends the notion of a the length of a vector, as we know it in R^2 and R^3.

Definition:

The *norm* of a vector x, denoted $||x||$, is

$$(< x, x >)^{0.5}$$

The *distance* from a vector x to a vector y is

$$||y - x||$$

10.5 Important Inequalities

These relations are highly useful, as we will see in the sequel.

10.5.1 The Cauchy-Schwarz Inequality

Theorem 10.1 (Cauchy-Schwarz Inequality)**.** *Say u and v are vectors in an inner product space. Then*

$$| < u, v > | \leq ||u||\ ||v||$$

Proof. See the Your Turn problem below.

□

10.5.2 Application: Correlation

Correlation coefficients are ubiquitous in data science. It is well known that their values fall into the interval [-1,1]. Let's prove that.

Recall from Section 6.1 the notion of the covariance between two random variables (from which the covariance matrix of a random vector is formed). We remarked that covariance is intuitively like correlation,but that latter is a scaled form. Formally,

$$\rho(X, Y) = \frac{E[(X - EX)(Y - EY)]}{\sqrt{Var(X)}\sqrt{Var(Y)}}$$

By dividing the covariance by the product of the standard deviations, we obtain a unitless quantity, i.e. free of units such as centimeters and degrees.

Now, X and Y are in $RV(\Omega)$. To simplify the algebra, consider the case $EX = EY = 0$.

Actually, we should say, "Make the transformation $X \to X - EX$, and note that it leaves both sides of the above correlation formula unchanged. We can thus assume $EX = EY = 0$.'' This is a common strategy, and should be kept in mind.

Recalling our inner product for this space, we have

$$< X, Y >= E(XY)$$

and

$$||X||^2 =< X, X >= E(X^2) = Var(X)$$

with the analogous relations for Y.

Cauchy=Schwarz then says

$$|E(XY)| \leq \sqrt{Var(X)}\sqrt{Var(Y)}$$

which says the correlation is between -1 and 1 inclusive.

10.5.3 The Triangle Inequality

In the world of ordinary physical geometry, we know the following

The distance from A to B is less than or equal to the sum of the distances from A to C and C to B.

Theorem 10.2 (Triangle Inequality)**.** *In a general inner product space,*

$$||x - z|| \leq ||x - y|| + ||y - z||$$

::: {.proof}

See the Your Turn problem below.

10.6 Projections

As mentioned, the extension of classical geometry to abstract vector spaces has powerful applications. There is no better example of this than the idea of *projections*.

10.6.1 Theorem

We say that vectors u and v are *orthogonal* if $< u, v >= 0$. This is the general extension of the notion of perpendicularity in high school geometry. Then we have the following:

Theorem 10.3 (Projection Theorem)**.** *Consider an inner product space V, with subspace W. Then for any vector x in W, there is a unique vector z in V, such that z is the closest vector to x in W.*

Furthermore, $x - z$ is orthogonal to any vector r in W.

The full proof is beyond the scope of this book, as it requires background in real analysis. Indeed, even the statement of the theorem is not mathematically tight.

For readers who do have such background, this is the Hilbert Projection Theorem. "Closest" is defined in terms of infimum, and W needs to be a topologically closed set. See for example the Wikipedia entry.l

For example, in the case of R^n, It will be seen shortly that for each W in the theorem, there is a matrix P_W that implements the projection, i.e.

$$z = P_W x$$

Note that projection operators are *idempotent*, meaning that if you apply a projection twice, the effect is the same as applying it once. In the matrix equation above, this means $P_W^2 = P_W$. This makes sense; once you drop down to the subspace, there is no further dropping down to that same space.

The case of $RV(\Omega)$ deserves its own section, coming up next.

10.6.2 The Pythagorean Theorem

That this ancient theorem in geometry still holds in general inner product spaces is a tribute to the power of abstraction.

The Pythagorean Theorem

If vectors X and Y are orthogonal, then

$$||X+Y||^2 = ||X||^2 + ||Y||^2 \tag{10.1}$$

10.7 Projections in $RV(\Omega)$

Here is what this very abstract vector space becomes useful in practical applications, such as will be presented in Section 10.11. We will need to build a bit of infrastructure.

The material in this section is rather involved, but it is needed for the section on racial, gender etc. fairness in machine learning, Section 10.11, which hopefully makes it worthwhile.

10.7.1 Conditional expectation

One of the Your Turn problems at the end of this chapter covers this setting:

Say we roll a die once, producing X dots. If $X = 6$, we get a bonus roll, yielding B additional dots; otherwise, $B = 0$. Let $Y = X + B$.

The basic form:

Now, what is $E(Y|B = 2)$? If $B = 2$, then we got the bonus roll, so $X = 6$ and $Y = X + B = 8$:

$$E(Y|B = 2) = 8$$

More generally,

$$E(Y|B = i) = \begin{cases} 3 & i = 0 \\ 6 + i & i = 1, 2, 3, 4, 5 \end{cases}$$

The random variable form:

The quantity $E(Y|B = i)$, a number, can be converted to a random variable, in the form of a function of B, which we will call Q, and denoted $E(Y|B)$, where

$$Q = \begin{cases} 3 & B = 0 \\ 6 + B & B = 1, 2, 3, 4, 5 \end{cases}$$

B is random, so $r(B)$ is also random, and here is its distribution:

Q takes on the values $3, 7, 8, 9, 10, 11$ with probabilities

$$1/6, 1/36, 1/36, 1/36, 1/36, 1/36$$

We need one more thing:

The Law of Iterated Expectation:

For random variable U and V, set

$$R = E[V|U]$$

Then

$$E(R) = E(V)$$

More concisely:

$$E[E(V|U)] = E(V) \tag{10.2}$$

Intuitive explanation: Say we wish to compute the mean height $E(H)$ of all students at a university. We might ask each department D to measure their own students, and report to us the resulting mean $E(H|D)$. We could then average all those departmental means to get the overall mean for the university:

$$E[E(H|D)] = E(H)$$

Note, though that that outer $E()$ (the first 'E') is a weighted average, since some departments are larger than others. The weights are the distribution of D.

10.7.2 Projections in $RV(\Omega)$: how they work

Consider random variables X and Y. Let W be the set of all functions of X with finite variance, which is a subspace of the vector space $RV(\Omega)$. The above theorem talks of a closest vector C in W to Y. Let's see what form C might take in this vector space.

Remember, the (squared) distance from Y to C is

$$||Y - C||^2 =< Y - C, Y - C >= E[(Y - C)^2]$$

That last term is

$$E[E((Y - C)^2|X)]$$

For any random variable Q of finite variance, the minimum value of $E[(Q - d)^2]$ over all constants d is attained by $d = E(Q)$. (See Your Turn problem below.) So, the minimum of $E((Y - C)^2|C)$, for all random variables C, is attained by the conditional mean,

Note that since we are conditioning on the random variable C, it becomes a constant, like d above.

$$C = E(Y|X)$$

In other words:

Projections in $RV(\Omega)$ are conditional means.

Moreover:

Since the difference between a vector and its projection onto a subspace is orthogonal to that subspace we have:

The vector $Y - E(Y|X)$ is uncorrelated with $E(Y|X)$. In other words, the prediction error (also called the *residual* has 0 correlation with the prediction itself.

10.8 Projections in the Linear Model

The case of the linear model will deepen our understanding, and will lead to a method for outlier detection that is commonly used in practice.

10.8.1 The least-squares solution is a projection

Armed with our new expertise on inner product spaces, we see that Equation 7.5 is

$$< S - Ab, S - Ab >$$

in the vector space R^n, where n is the number of our data points. Since we are minimizing that quantity with respect to b, the solution, $A\hat{\beta}$, is the projection of Y onto the subspace.

Of course, "data points" means rows in the data frame. In the statistics realm, people often speak of "observations."

But wait – *what* subspace? Well, it is the subspace consisting of all vectors of the form Ab:

The linear model projects the vector Y onto the column space of A.

10.8.2 Application: identifying outliers

An *outlier* is a data point that is rather far from the others. It could be an error, or simply an anomalous case. Even in the latter situation, such a data point could distort our results, so in both cases, identifying outliers is important.

Recall Equation 7.8, the general solution to our linear regression model:

$$\hat{\beta} = (A'A)^{-1}A'S$$

The projection itself is then

$$A\hat{\beta} = A(A'A)^{-1}A'S = HS$$

where the matrix

$$H = A(A'A)^{-1}A'$$

which projects S onto the column space of A, is called the *hat matrix*.

As a projection, H is idempotent, which one can easily verify by multiplication. H is also symmetric.

Let h_{ii} denote element i of the diagonal of H, with x_i denoting row i of A. One can show that

$$h_{ii} = x_i(A'A)^{-1}x_i' \tag{10.3}$$

The quantity h_{ii} is called the *leverage* for datapoint i.

Ah, so we know $h_{ii} \geq 0$. But also, using the material on circular shifts in Section 2.8.3, we have

$$tr(H) = tr[\underbrace{A(A'A)^{-1}}\, A'] = tr[A' \underbrace{A(A'A)^{-1}}] = tr(I) = p$$

for A of size nxp.

Thus the average value of h_{ii} is p/n. Accordingly, we might suspect an outlier if h_{ii} is considerably larger than p/n.

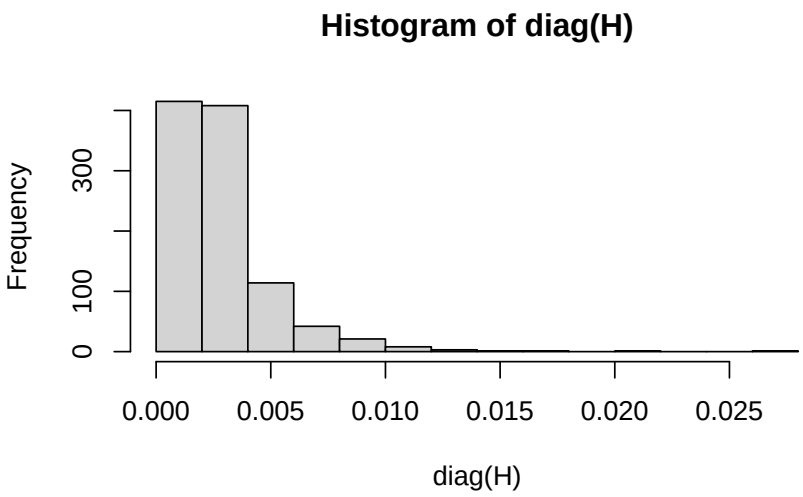
For example, let's look at the census data we've seen earlier:

```
library(qeML)
data(mlb1)
ourData <- as.matrix(mlb1[,-1]) # must have matrix to enable %*%
A <- cbind(1,ourData[,c(1,3)])
dim(A)
```

```
[1] 1015    3
```

```
S <- as.vector(mlb1[,3])
H <- A %*% solve(t(A) %*% A) %*% t(A)
```

```
hist(diag(H))
```



The ratio p/n here is 3/1015, about 0.003. We might take a look at the observations having h_{ii} above 0.01, say.

10.9 Orthogonal Bases

It turns out that a basis for a vector space is especially useful if its members are orthogonal to each other. We'll see why, and see how to generate such a basis from a nonorthogonal one.

An orthogonal basis in which every vector has length 1 is called *orthonormal*. Recall that $x \neq 0$ then $x/||x||$ has length 1, so that any orthogonal basis can easily be converted to orthonormal.

10.9.1 Motivation

Say we have a vector space V, a subspace W, and a vector x in V. We know x has a projection in W; call it z. But how do we find z?

Let $u_1, ..., u_k$ be a basis for W. Then there exist $a_1, ..., a_k$ such that

So we are assuming W (but not necessarily V is finite-dimensional. Using proper math, this could be extended.

$$z = a_1 u_1 + ..., + a_k u_k \tag{10.4}$$

So we can find z by finding the a_i.

10.9.2 The virtues of orthogonality

We do have a hint to work from: We know that $x - z$ is orthogonal to every vector in W – including the u_i. So

$$0 =< x - z, u_i >=< x, u_i > - < z, u_i >$$

Thus

$$< x, u_i >=< z, u_i >$$

Now, say the u_i are orthogonal to each other, and let's also say they are length 1. Then $< z, u_i >= a_i$,

so

$$< x, u_i >= a_i$$

So, we're done! We want to determine the a_i, and now we see that we can easily obtain it by calculating $< x, u_i >$. So, we have:

Given: a vector space V; a subspace W with orthonormal basis $u_1, ..., u_k$; and a vector x in V. Then the projection of x onto V is equal to

$$p =< x, u_1 > u_1 + ... + < x, u_k > u_k. \tag{10.5}$$

And, as a projection, we have that $x - p$ is orthogonal to all the u_i.

But how do we obtain an orthogonal basis, if we only have a nonorthogonal one? That's next…

10.9.3 The Gram-Schmidt Method

As seen in the last section, it is desirable to have an orthogonal basis, and it's even more convenient if its vectors have length 1 (an *orthonormal* basis). Converting to length 1 is trivial – just divide the vector by its length.

But if we start with a basis b_i and generate, say, orthogonal, length 1 vectors $u_1, ..., u_m$, how do we get u_{m+1} from b_{m+1}? the answer lies in Equation 10.5: Take the projection q of b_{m+1} onto the subspace generated by $u_1, ..., u_m$; $b_{m+1} - q$ will be orthogonal to $u_1, ..., u_m$, so we can take u_{m+1} to be $b_{m+1} - q$! We then normalize it.

So, here is the process for conversion:

The Gram-Schmidt Method

Say we have a basis $b_1, ..., b_k$ for some vector space. Convert it to an orthonormal basis as follows.

1. Set $u_1 = b_1/||b_1||$.
2. For each $i = 2, ..., k$, find the projection q of b_i onto the subspace generated by $u_1, ..., u_{i-1}$. Set u_i to $b_i - q$, and normalize.

10.10 Orthogonal Complements and Direct Sums

Consider a subspace W of an inner product space V. The set of vectors having inner product 0 with vectors in W is denoted $W^\perp$, known as the *orthogonal complement* of W. It too is a subpace, and jointly W and $W^\perp$ span all of V.

From Section 10.6, we know that for any x in V, one can uniquely write

$$x = x_1 + x_2$$

where x_1 and x_2 are in W and $W^\perp$, respectively.

Say $u_1, ..., u_r$ and $v_1, ..., v_s$ are bases for W and $W^\perp$. Then together they form a basis for all of V. Typically they are chosen to be orthonormal.

Finally, we say that V is the *direct sum* of W and $W^\perp$, denoted $V = W \oplus W^\perp$.

10.11 Application: Racial, Gender Etc. Fairness in Algorithms

COMPAS is a software tool designed to aid judges in determining sentences in criminal trials, by assessing the probability that the defendant would recividate. It is a commercial product by Northpointe.

COMPAS came under intense scrutiny after an investigation by *ProPublica*, which asserted evidence of racial bias against black defendants compared to white defendants with similar profiles. Northpointe contested these findings, asserting that their software treated black and white defendants equally. While this book does not take a position on the specific dispute, this case highlights the critical importance of addressing fairness in machine learning.

10.11.1 Setting

We consider prediction of a variable Y from a feature vector X and a vector of sensitive variables S. The target Y may be either numeric (in a regression setting) or dichotomous (in a two-class classification setting where $Y = 1$ or $Y = 0$). The m-class case can be handled using m dichotomous variables. We will consider only the numeric case here.

Our goal is to reduce or eliminate the influence of the sensitive variable S. One might ask, why not simply remove S from our model-fitting process? The problem is that X may include variables that are related to S. In the COMPAS example, race has some correlation with another variable, number of prior convictions.

10.11.2 The method of Scutari *et al*

The basic assumption (BA) amounts to (Y, X, S) having a multivariate Gaussian distribution, with Y scalar and X being a vector of length p. For convenience, assume here that S is scalar. As in Section 10.5.2, all variables are assumed centered, i.e. mean 0.

Let's review the material in Section 6.2: Say we have W with a multivariate normal distribution, and wish to predict one of its components, Y, from a vector X consisting of one or more of the other components, or linear combinations of them. Then

- the distribution of $Y|X$ is univeriate normal
- E(Y|X=t) is a linear function of t
- Var(Y|X=t) is independent of t

Let's add:

- though having 0 correlation does not in general imply independence, it does so in the multivariate normal case

One first applies a linear model in regressing X on S,

$$E(X|S) = S\gamma$$

where γ is a length-p coefficient vector. Here we are predicting the predictors (of Y), seemingly odd, but a first step in ridding ourselves from the influence of S.

Now consider the prediction errors (*residuals)*,

$$U = X - S\gamma$$

U can be viewed as the part of X that is unrelated to S; think of U as "having no S content." Note that U is a vector of length p.

Note the following:

- $E(X|S)$ is the projection of X onto the subspace of all functions of S.

- $X - E(X|S)$ (original vector minus the projection) is orthogonal to S.
- That is,

$$0 =< S, X - E(X|S) >= E[S(X - E(X|S))] = E(SU)$$

- Thus S and U are uncorrelated.
- Due to the BA, that means S and U are independent.
- In other words, our intution above that U "has no S content" was mathematically correct.
- Bottom line: Instead of predicting Y from X, use U as the predictor vector. This will enable truly S-free prediction.

Goal achieved.

10.12 Your Turn

Your Turn: Consider the space $RV(\Omega)$. Show that if $< X, X >= 0$, then X must be the 0 vector. Make use of the facts that $Var(X) = E(X^2) - [E(X)]^2$ and $Var(X) \geq 0$ with equality only of $X = c$ for some constant c.

Your Turn: Derive the Cauchy-Schwarz Inequality, using the following algebraic outline:

- The inequality

$$0 \leq < (au + v), (au + v) >$$

  holds for any scalar a.
- Expand the right-hand side (RHS), using the bilinear property of inner products.
- Minimize the resulting RHS with respect to a.

- Collect terms to yield

$$< u, v >^2 \leq ||u||^2 ||v||^2$$

Your Turn: Consider a set of vectors $W = v_1, ..., v_k$ in an inner product space V. Let U be another vector in V. Show that there exist scalars $a_1, ..., a_k$ and a vector v such that

$$u = a_1 v_1 + ... + a_k v_k + v$$

with

$$< v, v_i >= 0 \text{ for all i}$$

Your Turn: Say in $C(0, 1)$ we want to approximate functions by polynomials. Specifically, for any f in $C(0, 1)$, we want to find the closest polynomial of degree m. Write functions to do this, with the following call forms:

```
gsc01(m)  # performs Gram-Schmidt and returns the result
bestpoly(f,gsout)  # approx. f by output from gsc01
```

Hint: Since the vectors here are functions, you'll need a data structure capable of storing functions. An R **list** will work well here.

Your Turn: Show that the hat matrix is symmetric.

Your Turn: Use the Cauchy-Schwarz Inequality to prove the Triangle Inequality, using the following algebraic outline.

- Start with

$$||u + v||^2 =< (u + v, u + v >$$

- Expand the RHS algebraically.
- Using Cauchy-Schwarz to make the equation an inequality.
- Collect terms to yield the Triangle Inequality.

Your Turn: Consider the space $C(0,1)$. For function f and g of your own choosing, verify that the Cauchy-Schwarz and Triangle Inequalities hold in that case.

Your Turn: In the die rolling example, verify that

$$E[E(Y|B)] = E(Y)$$

Your Turn: Say we roll a die once, producing X dots. If $X = 6$, we get a bonus roll, yielding B additional dots; otherwise, $B = 0$. Let $Y = X + B$...

Your Turn: Say we roll a die once, producing X dots. If $X = 6$, we get a bonus roll, yielding B additional dots; otherwise, $B = 0$. Let $Y = X+B$. Verify that X and B satisfy the Cauchy-Schwarz and Triangle Inequalities, and also find $\rho(X, B)$.

Your Turn: Show that any set of orthogonal vectors is linearly independent.

Your Turn: Prove Equation 10.1 by expanding the inner product and simplifying algebraically.

Your Turn: Prove Theorem 10.2.

Your Turn: Prove that the vector $Y - E(Y|X)$ is uncorrelated with $E(Y|X)$

Your Turn: For X and Y in $RV(\Omega$, prove that

$$Var(Y) = E[Var(Y|X)] + Var[E(Y|X)],$$

first algebraically using Equation 10.2 and the relation $Var(R) = E(R^2) - (E(R))^2$, and then using the Pythagorean Theorem for a much quicker proof. As before, assume X and Y are centered.

your turn: show that Equation 10.3 holds. holds. holds.

Your Turn: Say V is R^n. Form the matrix A whose columns are the u_i, and let $P = AA'$. Show that

$$Px =< x, u_1 > u_1 + ...+ < x, u_k > u_k$$

so that P thereby implements the projection.

Your Turn: Consider the quadratic form $x'Px$. Show that if P is a projection matrix, the form equals $||Px||^2$.

11 Shrinkage Estimators

Goals of this chapter:

In the previous chapter, we introduced the norm of a vector. In many data science applications, solutions with smaller norms may be more accurate. This point is explored here.

11.1 Ridge Regression

In seminal paper, Hoerl and Kennard discussed the problem of *multicollinearity* of predictor variables in a linear model.

Technometrids, Februaru1970

11.1.1 Multicollinearity

The term *multicollinearity* refers to settings in which the following concerns arise:

- One column of the matrix A in Equation 7.8 is nearly equal to some linear combination of the others.
- Thus A is nearly not of full rank.
- Thus $A'A$ is nearly not of full rank.
- Thus $\hat{\beta}$ is unstable, in the form of high variance.

(Technically, the above conditions refer to *approximate* multicollinearity, with the exact version consisting of the first three bullet points, minus the word "nearly." However, informally, the term *multicollinearity* is usually taken to mean the approximate condition.)

That latter point is often quantified by the *Variance Inflation Factor*. To motivate it, consider the usual "R-squared" value from linear regression analyis, which is the squared correlation between "Y" and predicted "Y". Let R_j^2 denote that measure in the case of predicting column j of A from the other columns. The quantity

$$VIF_j = \frac{1}{1 - R_j^2}$$

then measures the negative impact due to multicollinearity on estimating $\hat{\beta}_j$.

Needless to say, the word "nearly" above, e.g. in "nearly not of full rank," is vague, and leaves open the question of "How near is 'near'?" and "What can we do about it?" Ridge regression is Hoerl's and Kennards's answer to the latter question, and it is based on a parameter that can be used to explore the former one.

11.1.2 The ridge solution

Their solution is simple; Add some quantity to the diagonal of $A'A$. Specifically, Equation 7.8 now becomes

$$\hat{\beta} - (A'A + \lambda I)^{-1} A'S \tag{11.1}$$

where λ is a positive number chosen by the analyst.

Here A has dimensions n x p, and I is the p x p identity matrix.

11.1.3 Matrix formulation

Using partitioned matrices helps understand ridge. Replace A and S by

$$A_{new} = \begin{pmatrix} A \\ \lambda I \end{pmatrix}$$

and

$$S_{new} = \begin{pmatrix} S \\ 0 \end{pmatrix}$$

where 0 means p 0s. In essence, we are adding artificial data here, consisting of p new rows to A, and p new elements to S. So Equation 11.1 is just the result of applying Equation 7.8 to A_{new} and S_{new}.

Loosely speaking, we can think of the addition of λI to $A'A$ makes the latter "larger", and thus its inverse smaller. In other words, we are "shrinking" $\widehat{\beta}$ towards 0. This effect is made even stronger by the fact that we added 0s data to S. This will be made more precise below.

11.1.4 Modern view

There are many ways to deal with multicollinearity, which we must remind the reader only concerns *approximate* linear dependence pf the columns of A. These days, ridge is often used for situations in which there is *exact* linear dependence.

We saw such a setting in Section 8.1. There we deliberately induced exact linear dependence by inclusion of both male and female dummy variables. Let's apply ridge:

```
library(qeML)
data(svcensus)
svc <- svcensus[,c(1,4:6)]
svc$man <- as.numeric(svc$gender == 'male')
svc$woman <- as.numeric(svc$gender == 'female')
svc$gender <- NULL
a <- svc[,-2]
lambda <- 0.1
a <- as.matrix(a)
tmp1 <- solve(t(a) %*% a + lambda * diag(4))
tmp1 %*% t(a) %*% as.matrix(svc$wageinc)
```

```
              [,1]
age        496.6716
wkswrkd   1372.7052
man     -18677.8751
woman   -29378.3086
```

The results essentially are the same as what we obtained by having only one dummy, thus no linear dependence: Men still enjoy about an $11,000 advantage. But ridge allowed us to avoid deleting one of our dummies. Such deletion is easy in

this case, but for large p, say in the hundreds or even more, some analysts prefer the convenience of ridge.

Of course, there is still the issue of how to choose the value of λ. This is beyond the scope of this book, which focuses on linar algebra.

11.1.5 Formalizing the notion of shrinkage

In the early 1980s, the statistical world was shocked by research by James and Stein that found, in short that:

Say W has q-dimensional normal distribution with mean vector μ and independent components having variance σ^2, each. We have a random sample of size n, i.e. n independent observations on W. Then if $q \geq 3$, in terms of Mean Squared Estimation Error, the best estimator of μ is NOT the sample mean $\bar{W}$. Instead, it's

$$\left(1 - \frac{(q-2)\sigma^2/n}{||\bar{W}||^2}\right) \bar{W}$$

The quantity within the parentheses is typically smaller than 1, giving us the shrinkage property. Note, though, that with larger n, the amount of shrinkage is minor.

In the case of linear regression, shrinkage works there too, with q being the number of columns in the A matrix.

11.1.6 Shrinkage through length penalization

Say instead of minimizing Equation 7.5, we minimize

$$(S - Ab)'(S - Ab) + \lambda||b||^2 \tag{11.2}$$

The larger b is, the harder it is to minimize the overall quantity Equation 11.2. We say that we *penalize* large values of b, an indirect way of pursuing shrinkage. Now take the derivative and set to 0:

$$0 = A'(S - Ab) + \lambda b$$

i.e.

$$(A'A + \lambda I)b = A'S$$

and thus

$$\hat{\beta} = (A'A + \lambda I)^{-1}A'S$$

It's ridge! So here is formalization of our "almost singular" etc. language above to justify ridge as a shrinkage estimator..

11.1.7 Shrinkage through length limitation

Instead of penalizing $||b||$, we could simply constrain it, i.e. we could set our optimization problem to:

minimize $(S - Ab)'(S - Ab)$, subject to the constraint $||b||^2 \leq \gamma$

We say that this new formulation is the *dual* of the first one. One can show that they are typically equivalent.

11.2 The LASSO

The LASSO (Least Absolute Shrinkage and Selection Operator) was developed by Robert Tibshirani in 1996, following earlier work by Leo Breiman. It takes $\hat{\beta}$ to be the value of b that minimizes

$$(S - Ab)'(S - Ab) + \lambda||b||_1 \tag{11.3}$$

where the "l1 norm" is

$$||b|| = \sum_{i=1}^{p} |b_i|$$

We will write our original norm as $||b||_2$.

This is a seemingly minor change, but with important implications. What Breiman and Tibshirani were trying to do was to obtain a *sparse* $\widehat{\beta}$, i.e. a solution with lots of 0s, thereby providing a method for predictor variable selection. This is important because so-called "parsimonious" prediction models are desirable.

11.2.1 Properties

To that end, first note that it can be shown that, under some technical conditions, that the ridge solution minimizes

$$(S - Ab)'(S - Ab)$$

subject to the constraint

$$||b||_2 \leq \gamma$$

while in the LASSO case the constraint is

$$||b||_1 \leq \gamma$$

As with λ in the original formulation, γ is a positive number chosen by the analyst.

11.2.2 Geometric view

Comparison between the ridge and LASSO concepts is often done via this graph depicting the LASSO setting:

Here $p = 2$, with $b = (b_1, b_2)'$. The horizontal and vertical axes represent b_1 and b_2.

- The constraint $||b||_1 \leq \gamma$ then takes the form of a diamond, with corners at $(\gamma, 0)$, $(0, \gamma)$, $(-\gamma, 0)$ and $(0, -\gamma)$. The constraint $||b||_1 \leq \gamma$ requires us to choose a point b somewhere in the diamond, including the boundary.

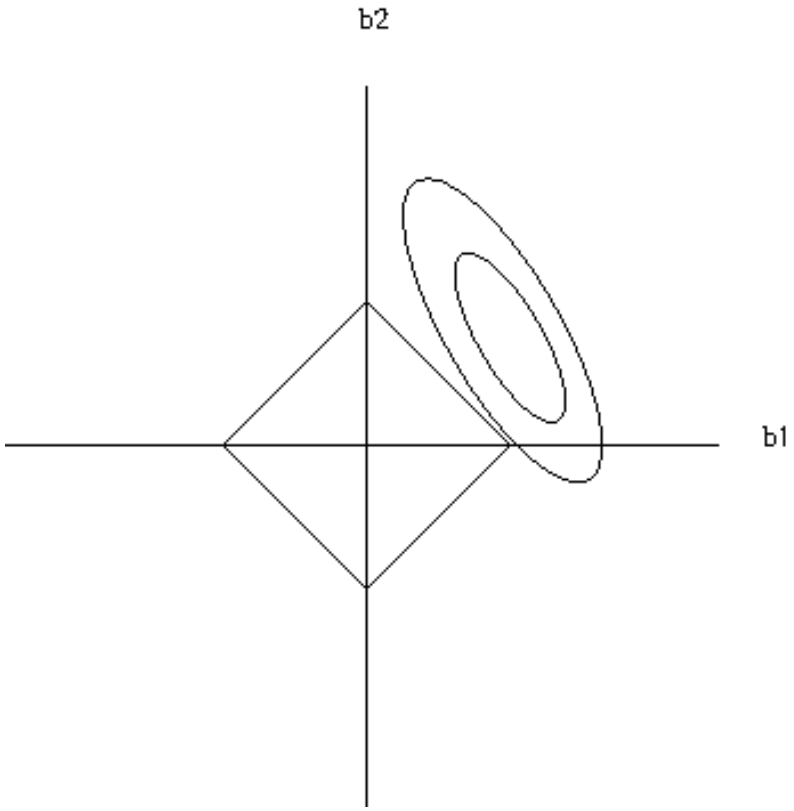


Figure 11.1: LASSO sparsity

- The concentric ellipses depict the values of $c(b) = (S - Ab)'(S - Ab)$, as follows.
- Consider one particular value of $c(b)$, say 1.68.
- Many different points b in the graph will have $c(b) = 1.68$; in fact, the locus of all such points is an ellipse.
- There is one ellipse for each possible value of $c(b)$. So, there are infinitely many ellipses, though only two are shown here.
- Larger values of $c(b)$ yield larger ellipses.
- On the one hand, we want to choose a b for which $c(b)$ – our total squared prediction error – is small, thus a smaller ellipse.
- But on the other hand, we need at least one point on the ellipse to be in common with the diamond.
- The solution is then a point b in which the ellipse just barely touches the diamond, respectively.

The key point is that that "barely touching" point will be one of the four corners of the diamond, points at which either $b_1 = 0$ or $b_2 = 0$ – *hence a sparse solution.*

This will not happen with ridge. The diamond would now be a circle (not shown). The "barely touching point" will almost certainly will be at a place in which both b_1 and b_2 are nonzero. Hence no sparsity.

11.2.3 Use of LASSO for dimension reduction

In prediction applications, one goal is to avoid *overfitting*, meaning using using such a complex model that it actually has poorer predictive power than a simpler one. An example is using too many predictors.

The LASSO, in producing a sparse estimated coefficient vector, reduces the number of predictors. This would seem to achieve our goal, but a problem is that we must still choose λ.

A common way to choose among models is *cross-validation*: We set aside a subset of the data for use in testing predictive accuracy. The remaining data is the *training set*. For each of our competing models, we fit the model on the training set, then use the result to predict the test set. We then use whichever model does best in the test set.

Example: Census data

Let's look once again at our census dataset (Section 8.1). With $n = 20090$ and $p = 14$, we probably would not have to worry about overfitting even if we used the full dataset, it is convenient to use as an example here.

CRAN's **glmnet** does automatic cross-validation, but let's use the **lars** package here for simplicity, just to see the effects of using different values of λ.

```
library(lars)
# lars pkg needs numeric inputs
svcdumm <- factorsToDummies(svcensus)
colnames(svcdumm)
```

```
 [1] "age"           "educ.14"       "educ.16"       "educ.zzzOther"
 [5] "occ.100"       "occ.101"       "occ.102"       "occ.106"
 [9] "occ.140"       "occ.141"       "wageinc"       "wkswrkd"
[13] "gender.female" "gender.male"
```

```
z <- lars(svcdumm[,-11],svcdumm[,11],'lasso')
z$lambda
```

```
 [1] 2.871366e+06 1.188844e+06 7.682765e+05 6.999891e+05 6.736743e+05
 [6] 6.052531e+05 4.046995e+05 3.193028e+05 2.888678e+05 1.472984e+05
[11] 5.748624e+04 4.184999e+04 2.579800e-07 2.392628e-07
```

```
z$beta
```

```
          age educ.14   educ.16 educ.zzzOther    occ.100   occ.101   occ.102
0     0.00000       0    0.0000         0.000      0.000     0.000    0.0000
1     0.00000       0    0.0000         0.000      0.000     0.000    0.0000
2     0.00000       0    0.0000     -6513.825      0.000     0.000    0.0000
3    40.99988       0    0.0000     -7513.808      0.000     0.000    0.0000
4    58.30503       0    0.0000     -7848.929      0.000     0.000  373.3673
5   103.15770       0    0.0000     -8692.806      0.000     0.000 1335.7650
6   234.27216       0    0.0000    -11078.501  -1828.656     0.000 3535.6762
7   285.98439       0    0.0000    -11982.103  -3510.927 -1699.833 3591.5225
8   303.90057       0  734.4989    -12199.888  -4106.353 -2294.723 3619.9295
9   384.75160       0 4150.1932    -13194.204  -5881.118 -4065.294 4750.4844
10 436.63083       0 6251.9663    -13787.583  -9448.929 -7619.247 3046.4329
11 445.33929       0 6619.8262    -13889.769  -9871.243 -8039.293 2948.3846
12 469.83727       0 7597.2521    -14167.389 -10791.016 -8952.465 2897.2855
13 469.83727       0 7597.2521    -14167.389 -10791.016 -8952.465 2897.2855
14 469.83727       0 7597.2521    -14167.389 -10791.016 -8952.465 2897.2855
     occ.106  occ.140  occ.141   wkswrkd gender.female gender.male
0      0.000   0.0000    0.000    0.0000         0.000           0
1      0.000   0.0000    0.000  813.0166         0.000           0
2      0.000   0.0000    0.000 1006.5238         0.000           0
3      0.000   0.0000    0.000 1038.1494         0.000           0
4      0.000   0.0000    0.000 1049.7984         0.000           0
5      0.000   0.0000    0.000 1078.5891     -1026.856           0
6      0.000   0.0000    0.000 1162.7454     -3848.567           0
7      0.000   0.0000    0.000 1196.6997     -4947.064           0
8      0.000   0.0000    0.000 1208.6741     -5327.782           0
9      0.000   0.0000 1471.035 1264.4784     -7040.585           0
10 -6312.259   0.0000    0.000 1300.2826     -7988.713           0
11 -7212.912   0.0000    0.000 1306.5163     -8150.998           0
12 -9410.990 941.5679    0.000 1323.1999     -8595.583           0
13 -9410.990 941.5679    0.000 1323.1999     -8595.583           0
14 -9410.990 941.5679    0.000 1323.1999     -8595.583           0
attr(,"scaled:scale")
 [1] 1590.60915   58.23770   26.47059   61.47839   59.51806   59.08411
 [7]   67.11780   22.10218   27.74858   49.64874 2069.48097   60.90133
[13]   60.90133
```

Again, we are not doing cross-validation here, but it appears to Occupation 141 is an expendable predictor, and we might consider dropping, say, the **educ.14** variable.

11.3 A Warning

Many statistical quantities now have *regularized*, i.e. shrunken versions. It is also standard practice in neural networks. This may be quite helpful in prediction contexts. However, note the following:

> **!** No Statistical Inference on Shrinkage Estimators
>
> Shrinkage produces a bias, of unknown size. Thus classical statistical inference (confidence intervals, hypothesis tests), e.g. those based on Equation 7.11 for linear models, is not possible.

11.4 Your Turn

Your Turn: Show that A_{new} in Section 11.1.3 is of full rank, p.

12 Eigenanalysis: Properties

Goals of this chapter:

The concept of eigenvalues and eigenvectors is one of the most important of all appplications of linear algebra to data science. We introduce the basic ideas and properties in this chapter.

12.1 Definition

The concept itself is simple:

Consider an m x m matrix M. If there is a nonzero vector x and a number λ such that

$$Mx = \lambda x \tag{12.1}$$

we say that λ is an *eigenvalue* of M, with *eigenvector* x.

12.2 A First Look

Here are a few properties to start with:

- The definition is equivalent to

  $$(M - \lambda I)x = 0$$

  which in turn implies that $M - \lambda I$ is noninvertible. That then implies that

  $$det(M - \lambda I) = 0$$

- The left-hand side of this equation is a polynomial in λ. So for an n x n matrix M, there are n roots of the equation and thus n eigenvalues. Note that some roots may be repeated; if M is the zero matrix, it will have an eigenvalue 0 with multiplicity n.

- In principle, that means we can solve the above determinant equation to find the eigenvalues of the matrix. The comments in Section 5.5.1 regarding the "ugly" definition of determinants imply that the equation involves a degree-m polynomial in λ.
- There are much better ways to calculate the eigenvalues than this, but a useful piece information arising from that analysis is that M as m eigenvalues (not necessarily distinct).

12.3 The Special Case of Symmetric Matrices

Many matrices in data science are symmetric, such as covariance matrices and $A'A$ in Equation 7.8.

12.3.1 Eigenvalues are real

Some eigenvalues may be complex numbers, i.e. of the form $a + bi$, but it can be shown that if M is symmetric, its eigenvalues are guaranteed to be real. This is good news for data science, as many matrices in that field are symmetric, such covariance matrices.

12.3.2 Eigenvectors are orthogonal

Eigenvectors corresponding to distinct eigenvalues of a symmetric matrix M are othogonal.

Proof: Let u and v be such eigenvectors, corresponding to eigenvalues μ and ν.

$$\mu' M\nu = (\mu' M\nu)' = \nu' M'\mu = \nu' M\mu$$

(in that first equality, we used the fact that the transpose of a number is that number). Substitute for $M\nu$ and $M\mu$, then subtract.

12.3.3 Symmetric matrices are diagonalizable

A square matrix R is said to be *diagonalizable* if there exists an invertible matrix P such that $P^{-1}RP$ is equal to some diagonal matrix D. We see that symmetric matrices fall into this category.

One application of this is the computation of powers of a diagonal matrix:

$$R^k = P^{-1}RP\ P^{-1}RP\ ...\ P^{-1}RP = P^{-1}D^kP$$

D^k is equal to the diagonal matrix with elements d_k^k, so the computation of M^k is easy.

Theorem: Any symmetric matrix M has the following properties

- M is diagonalizable.
- In fact, the matrix P is equal to the matrix whose columns are the eigenvectors of M, chosen to have length 1.
- The associated diagonal matrix has as its diagonal elements the eigenvalues of M.
- The matrix P has the property that $P^{-1} = P'$.
- Moreover, the rank of M is equal to the number of nonzero eigenvalues of M.

Proof: Use partitioned matrices.

Let $D = diag(d_1, ..., d_m)$, where the d_i are the eigenvalues of M, corresponding to eigenvectors P_i. Set the latter to have length 1, by dividing by their lengths.

Recall that the eigenvectors of M, i.e. the columns of P, are orthogonal. That means the rows of P' are orthogonal. Again using partitioning, we see that

$$P'P = I$$

so that $P^{-1} = P'$.

By the way, any square matrix Q such that $Q'Q = I$ is said to be *orthogonal*.

Now write

$$MP = M(P_1|...|P_m) = (MP_1|...|MP_n) = (d_1P_1|...|(d_mP_m) = PD$$

Multiply on the left by P', and we are done.

Regarding rank, Theorem 8.1 tells us that pre- or postmultiplying by an invertible matrix does not change rank, and clearly the rank of a diagonal matrix is the number of nonzero elements.

Example:

Let's illustrate all this with the data in Section 11.1.4. We will form the matrix $A'A$ in Section 7.3.3, which as mentioned, is symmetric.

Recall that in that example, A is not of full rank. Thus we should expect to see a 0 eigenvalue.

```
library(qeML)
data(svcensus)
svc <- svcensus[,c(1,4:6)]
svc$man <- as.numeric(svc$gender == 'male')
svc$woman <- as.numeric(svc$gender == 'female')
svc$gender <- NULL
a <- as.matrix(svc[,-2])
a <- cbind(1,a)  # add the column of 1s
m <- t(a) %*% a
eigs <- eigen(m)
eigs
```

```
eigen() decomposition
$values
[1] 7.594762e+07 3.292955e+06 7.466971e+03 1.293594e+03 2.801489e-12

$vectors
             [,1]          [,2]          [,3]         [,4]          [,5]
```

```
[1,] -0.015881850  0.004389585 -0.035995373  0.81553633  5.773503e-01
[2,] -0.649660696  0.760031134  0.004461047 -0.01654550  1.561251e-17
[3,] -0.759952974 -0.649864420  0.004991922 -0.01108122 -4.061024e-17
[4,] -0.012070494  0.002130704 -0.724401141  0.37650952 -5.773503e-01
[5,] -0.003811356  0.002258881  0.688405767  0.43902681 -5.773503e-01
```

```
m %*% eigs$vectors[,1]
```

```
               [,1]
         -1206188.6
age      -49340181.0
wkswrkd -57716616.5
man       -916725.2
woman     -289463.4
```

```
eigs$values[1] %*% eigs$vectors[,1]
```

```
         [,1]      [,2]      [,3]      [,4]      [,5]
[1,] -1206189 -49340181 -57716617 -916725.2 -289463.4
```

Yes, that first column is indeed an eigenvector, with the claimed eigenvalue.

Note that the expected 0 eigenvalue shows up as 2.801489e-12, quite small but nonzero, due to roundoff error.

12.4 Computation: the Power Method

One way to compute eigenvalues and eigenvectors is the *power method*, a simple iteration. We begin with an initial guess, x_0 for an eigenvector. Substituting in Equation 12.1, we have the next guess:

$$x_1 = Mx_0$$

We keep iterating until convergence, generating x_2 from x_1 and so on. However, the x_i may grow, so we normalize to length 1:

$$x_i \leftarrow \frac{x_i}{||x_i||}$$

12.5 Application: Computation of Long-Run Distribution in Markov Chains

We showed in Section 5.1 how matrix inverse can be used to compute the long-run distribution ν in a Markov chain. However, this is inefficient for very large transition matrices. For instance, in Google PageRank, there is a Markov state for every page on the Web!

Instead, we exploit the fact that Equation 5.1 says that the transition matrix has an eigenvector ν with eigenvalue 1. Due to the typical huge size of the matrix, the power method or a variant is often used.

12.6 Your Turn

Your Turn: Show that the diagonalizing matrix P above must have determinant ± 1.

Your Turn: Show that if x is an eigenvector of M with eigenvalue $\lambda \neq 0$, then for any nonzero number c, cx will also be an eigenvector with eigenvalue λ.

Your Turn: Show that if a matrix M has a 0 eigenvalue, M must be singular. Also prove the converse. (Hint: Consider the column rank.)

Your Turn: Consider a projection matrix P_W. Show that the only possible eigenvalues are 0 and 1. Hint: Recall that projection matrices are idempotent.

Your Turn: Say A is an invertible matrix with eigenvalue λ and eigenvector v. Show that v is also an eigenvector of A^{-1}, with eigenvalue $1/\lambda$.

Your Turn: Show that if A is nonnegative-definite, its eigenvalues must be nonnegative.

13 Principal Components

Goals of this chapter:

Today's large datasets being so common, we need a way to "cut things down to size," i.e. *dimension reduction*. It turns out that eigenanalysis can be quite useful for this purpose, by determining the *principal components* of our data. This is the focus of the current chapter.

13.1 Example: African Soils Data

To get things started, let's consider the African Soils dataset.

Let's take a look around:

```
load('data/AfricanSoil.Rd')
dim(AfricanSoil)
```

```
[1] 1157 3600
```

```
names(AfricanSoil)[1:25]
```

```
 [1] "PIDN"     "m7497.96" "m7496.04" "m7494.11" "m7492.18" "m7490.25"
 [7] "m7488.32" "m7486.39" "m7484.46" "m7482.54" "m7480.61" "m7478.68"
[13] "m7476.75" "m7474.82" "m7472.89" "m7470.97" "m7469.04" "m7467.11"
[19] "m7465.18" "m7463.25" "m7461.32" "m7459.39" "m7457.47" "m7455.54"
[25] "m7453.61"
```

```
names(AfricanSoil)[3576:3600]
```

```
 [1] "m605.545" "m603.617" "m601.688" "m599.76"  "BSAN"     "BSAS"
 [7] "BSAV"     "CTI"      "ELEV"     "EVI"      "LSTD"     "LSTN"
[13] "REF1"     "REF2"     "REF3"     "REF7"     "RELI"     "TMAP"
[19] "TMFI"     "Depth"    "Ca"       "P"        "pH"       "SOC"
[25] "Sand"
```

Let's try predicting **pH**, the acidity. But that leaves 3599 possible predictors. There is an old rule of thumb that one should have $p < \sqrt{n}$, for p predictors and n data points, to avoid overfitting, a rule which in our setting of $n = 1157$ is grossly violated. We need to do dimension reduction. One way to accomplish this is to use PCA.

(Note, though, that this presumes our goal is prediction, rather than effect estimation. In the latter case, our new variables will be principal components, and their regression coefficients may not be meaningful to us. If we are doing prediction, regression coefficients may not be of interest.)

13.2 Overall Idea

The goal is to find a few important linear combinations of our original predictor variables–important in the sense that they roughly summarize our data. These new variables are called the *principal components* (PCs) of the data. Let's see how this is done.

13.2.1 The first PC

Let $\mathbf{X}$ denote a set of variables of interest (not necessarily in a prediction context). In searching for good linear combinations, we want to aim for ones with high variance. We certainly don't want ones with low variance; after all, a random variable with 0 variance is a constant. So we wish to find linear combinations

$$Xu$$

which maximize

$$Var(Xu) = u'Cov(X)u$$

where we have invoked Equation 6.3.

But that goal is ill-defined, since we could take larger and larger vectors u, thus larger and larger vectors Xu, no maximum. So, let's constrain it to vectors u of length 1:

$$u'u = 1$$

The method of *Lagrange multipliers* is used to solve maximum/minimum problems that have constraints, by adding a new variable corresponding to the constraint. In our case, we maximize

$$u'Cov(X)u + \gamma(u'u - 1)$$

with respect to u and γ.

Setting derivatives to 0, we have

$$0 = 2Cov(X)u + 2\gamma u$$

and

$$0 = u'u - 1$$

In other words,

$$Cov(X)u = -\gamma u \tag{13.1}$$

Aha! The vector u is an eigenvector of the matrix $Cov(X)$ with eigenvalue $-\gamma$. We've discovered that our quest for dimension reduction can be couched as an eigenanalysis problem!

13.2.2 The second, third etc. PCs

There are as many principal components as there are columns in columns in X. So, how are the second, third and so on defined and computed?

The key issue is that we want our PCs to be orthogonal, because as we have seen orthogonal random vectors are statistically uncorrelated, and independent in the multivariate normal case. So if we summarize our data with say, the first few PCs, the fact that they are (at least loosely speaking) uncorrelated makes for a neater summary.

So with u and v denoting the first and second PCs, we want v to maximize

$$v'Cov(X)v + \gamma(v'v - 1)$$

subject to

$$u'v = 0$$

For the latter condition, let's add another Lagrange multiplier term:

$$v'Cov(X)v + \omega(v'v - 1) + \eta u'v$$

Setting derivatives (with respect to v) to 0, we have

$$0 = 2Cov(X)v + 2\omega v + \eta u \tag{13.2}$$

Multiplying on the left by u', we have

$$0 = 2u'Cov(X)v + 2\omega u'v + \eta u'u \tag{13.3}$$

But from before we have

$$u'Cov(X) = -\gamma u$$

so that

$$u'Cov(X)v = 0$$

in Equation 13.3. Moreover, $u'v = 0$ there as well, so that we have $\eta = 0$, which reduces Equation 13.3 to a statement that v too must be an eigenvector of $Cov(X)$.

13.3 Properties

The full set of eigenvectors is then known as the *principal components* (PCs). They have some further important properties:

- The PCs are orthogonal to each other. This follows from our earlier finding that eigenvectors of a symmetric matrix, in this case $Cov(X)$, are orthogonal. We say that a symmetric is *orthogonal* if its columns are orthogonal as vectors.
- This in turn implies that the PCs are uncorrelated, and in the multivariate normal case, statistically independent.
- The eigenvalues are the variances of the PCs. The PCs are in order of decreasing (or at least nonincreasing) variance.

  This first statement follows from pre-mutiplying Equation 13.1 by u', giving us $Var(u)$ on the left and $-\lambda u'u = -\lambda$ on the right. The second statement follows from the fact that as we go from the first PC to the second, third and so on, we are maximizing under more and more constraints, hence smaller maxima.

By the way, though we used the covariance matrix here, it is also common to simply do eigenanalysis on the original data matrix.

13.4 Eigenanalysis Relation between A and $Cov(X)$ in Linear Regression

Consider our usual linear regression setting, in which the matrix A contains our data for our predictor variables X. If we center and scale our data, then '

$$Cov(X) = A'A$$

Now say the PCA of X has a 0 eigenvalue. That implies that some linear combination $u'X$ has 0 variance. Then

$$0 = Var(u'X) = u'Cov(X)u = u'A'Au = (Au)'Au$$

Thus

$$Au = 0$$

Since Au is a linear combination of the columns of A, we see that the columns of A are linearly dependent.

The same argument works in reverse. If $Au = 0$, we fine that $Var(u'X) = 0$, and since the eigenvalues of $Cov(X)$ are variances of linear combinations of X, we see that one of those variances is 0.

13.5 Back to the Example

So, we remove the "Y'' variable, pH, number 3598 as seen above, as proceed. We will also remove the nonnumeric columns, **PIDN** and **Depth**. We could use R's **prcomp** function here, but to better illustrate the concepts, we do things from scratch.

```
x <- AfricanSoil[,-c(1,3595,3598)]
dim(x)
```

```
[1] 1157 3597
```

```
xcov <- cov(x)
eigs <- eigen(xcov)
eigs$values[1:100]  # don't print out all 3597!
```

```
  [1] 7.600042e+01 9.553189e+00 6.979629e+00 4.209555e+00 2.884526e+00
  [6] 2.175115e+00 1.954537e+00 1.321853e+00 9.375446e-01 6.453537e-01
 [11] 6.360017e-01 5.484400e-01 4.535480e-01 3.981618e-01 3.665122e-01
 [16] 3.496067e-01 2.597679e-01 2.035621e-01 1.498785e-01 1.248452e-01
 [21] 1.119365e-01 9.545515e-02 8.083867e-02 6.415968e-02 6.079547e-02
 [26] 5.737305e-02 4.468220e-02 3.902647e-02 3.641190e-02 3.056959e-02
 [31] 2.541331e-02 2.037622e-02 1.819728e-02 1.532509e-02 1.297067e-02
 [36] 1.242883e-02 1.073280e-02 9.420125e-03 8.687909e-03 7.980043e-03
 [41] 7.178366e-03 5.445802e-03 4.893268e-03 4.027980e-03 3.903172e-03
 [46] 3.526363e-03 3.341049e-03 3.062258e-03 2.847189e-03 2.723814e-03
 [51] 2.586813e-03 2.315871e-03 2.142482e-03 1.977798e-03 1.771873e-03
 [56] 1.506873e-03 1.474780e-03 1.302539e-03 1.173394e-03 1.141518e-03
 [61] 1.004309e-03 9.438253e-04 8.794233e-04 8.217583e-04 8.137159e-04
 [66] 7.269301e-04 7.187134e-04 6.454513e-04 6.261121e-04 5.256722e-04
 [71] 4.603372e-04 4.256952e-04 4.116083e-04 3.873680e-04 3.802244e-04
 [76] 3.421665e-04 3.144433e-04 3.013745e-04 2.650906e-04 2.561422e-04
 [81] 2.553842e-04 2.436702e-04 2.110897e-04 1.937904e-04 1.851444e-04
 [86] 1.766814e-04 1.569518e-04 1.484314e-04 1.437515e-04 1.263429e-04
 [91] 1.242947e-04 1.154861e-04 1.135908e-04 1.113247e-04 1.003160e-04
 [96] 9.908341e-05 9.180042e-05 8.670191e-05 8.415325e-05 7.907993e-05
```

Ah, the eigenvalues fall off rapidly after the first few. So, let's say we decide to use the first 9 u vectors.

```
eigvecs <- eigs$vectors[,1:9]
dim(eigvecs)
```

```
[1] 3597    9
```

So, recalling the sequence of equations leading to Equation 13.1, we are ready to replace our old variables by the new:

```
x <- as.matrix(x)
dim(x)
```

```
[1] 1157 3597
```

```
xnew <- x %*% eigvecs
dim(xnew)
```

```
[1] 1157    9
```

```
ph <- AfricanSoil[,3595]
lmout <- lm(AfricanSoil$pH ~ xnew)
# and so on
```

13.6 PCA in Detection of Multicollinearity

Recall the itemized list in Section 11.1.1 of various conditions under which we have multicollinearity. We can add the following item to that list:

- A has an eigenvalue that is nearly 0.

The reasons are straightforward. If $Ax = 0$ for some nonzero $x, then the material in **?@sec-partitioned** on partitioning says that the coefficients in x form a linear combination of the columns of A that results in the 0 vector, i.e. A is not of full rank.

In fact, PCA can warn us of specific linear combinations of $Cov(A'A)$ that are nearly 0, as follows.

- Say a PC corresponds to a small eigenvalue.
- Then from Section 13.2.1 that linear combination has small variance.
- Random variables with small variance are nearly constant.

- Thus with high probability, the linear combination is near to c for some number c.
- If our design matrix A includes a 1s column, then the above linear combination, together with this 1s column, gives us a linear combination that is usually close to the 0 vector, thus multicollinear.

In other words, PCA can point out to us specific linear combinations of our variables that may be problematic. If we are considering solving the problem by removing one or more predictor variables, this analysis could suggest which ones to remove.

13.7 How Many Principal Components Should We Use?

We chose to use the first 9 principal components in our example here, but just for illustration purposes. There are no formal rules for how many to use. Many "rules of thumb" exist.

If we are doing prediction, there is a very natural way to choose our number of PCs–do cross-validation (Section 11.2.3), and use whichever number gives the most accurate prediction.

13.7.1 Example: New York City taxi trips

This is a well-known dataset, a version of which is included in the **qeML** package. The object is to predict trip time, given pickup and dropoff locations, trip distance and day of the week.

In the raw form of the data, there are just 5 columns, thus 4 predictors. But the pickup and dropoff locations are R factors, coding numerous locations. These must be decoded to dummy variables (values 1 or 0, coding whether or not, say, a given trip began at pickup location 121. If for instance one calls the R linear regression function **lm**, that function will do the conversion internally, but in our case we will perform the conversion ourselves, using **regtools::factorsToDummies**. (The **regtools** package is included by **qeML**.)

```
library(qeML)
data(nyctaxi)
dim(nyctaxi)
```

```
[1] 10000     5
```

```
nyc <- factorsToDummies(nyctaxi,dfOut=T)
dim(nyc)
```

```
[1] 10000   357
```

Wow! That's quite a lot of predictors, and well in excess of the common rule of thumb that one should have no more than $\sqrt{n}$ predictors, 100 in this case.

So, we might try dimension reduction via PCA, then use the PCs as predictors. The function **qeML::qePCA** combines these two operations. Let's see how it works.

```
args(qePCA)
```

```
function (data, yName, qeName, opts = NULL, pcaProp, holdout = floor(min(1000,
    0.1 * nrow(data))))
NULL
```

Here **qeName** indicates which function is desired for prediction, e.g. **qeLin** for a linear model. (This function wraps **lm**.) But a key argument here is **pcaProp**. Recall that:

- The PCs come in order of decreasing variance.
- We are mainly interested in the first few PCs. The later ones have small variance, which makes them approximately constant and thus of no use to us.

The name 'pcaProp' stands for "proportion of total variance." If we set this to, say, 0.25, we are saying "Give us whatever number of the first few PCs that have a total variance of at least 25% of the total." Let's give that a try:

```
qePCA(nyc,'tripTime','qeLin',pcaProp=0.25)$testAcc
[1] 311.9159
```

We asked to predict the column 'tripTime' in the dataset **nyc** using the **qeLin** function, based on as many of the PCs that will give us 25% of the total variance. As seen above, most of the **qeML** prediction functions split the data into a training set and a holdout set. The model is fit to the training set, and then applied to prediction of the holdout set. The output value is the mean absolute prediction error (MAPE).

However, since the holdout set is randomly generated, the MAPE value is random, so we should do multiple runs. Some experimentation showed that MAPE here is highly variable, so we decided to perform 500 runs, e.g.

```
mean(sapply(1:500,function(i) qePCA(nyc,'tripTime','qeLin',pcaProp=0.1)$testAcc))
[1] 359.663
```

Table 13.1: Mean Absolute Predictive Error

pcaProb	MAPE
0.1	359.6630
0.2	359.3068
0.3	403.9878
0.4	397.3783
0.5	430.8958
0.6	423.7299
0.7	517.7917
0.8	969.1304
0.9	2275.091

Among other things, this shows the dangers of overfitting, in this case using too many PCs in our linear regression model. It seems best here to use only the first 10 or 20% of the PCs.

13.8 Your Turn

Your Turn: Modify the code for **qePCA** for the case of linear regression by addig a component **xCoeffs** to its return value. This will give the regression coefficients in terms of the original X predictors.

Your Turn: Say the symmetric matrix A has *block diagonal* form

$$A = \begin{pmatrix} A_1 & 0 & 0... \\ 0 & A_2 & 0... \\ ... & & \end{pmatrix}$$

where A_i $(i = 1, ..., r)$

- is symmetric
- is of size k_i x k_i
- has eigenvalues $\gamma_1, ..., \gamma_k$
- has eigenvectors $u1, j, ..., uk, j$, where $j = 1, ..., k_i$

State the form of the eigenvalues and eigenvectors of A.

14 Singular Value Decomposition

Goals of this chapter:

We've seen the notion of eigenanalysis for square matrices. Well, it turns out this can be extended usefully for non-square matrices, via *Singular Value Decomposition* (SVD), the topic of this chapter.

14.1 Basic Idea

Here is what we are aiming for.

Our target relation:

Given an $m \times n$ matrix A, we wish to find orthogonal matrices U and V, and a matrix $\Sigma = diag(\sigma_1, ..., \sigma_n)$ whose nonzero elements are positive, such that

$$A = U\Sigma V' \tag{14.1}$$

By convention the ordering of columns is set so that the σ_i occur in nonincreasing order.
Note that, by matrix partitioning, Equation 14.1 also says that

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i' \tag{14.2}$$

Note the dimensions:

- U is $m \times n$
- V is $n \times n$
- Σ is $n \times n$

Let r denote the rank of A. Recall that $r \leq \min(m, n)$. It will turn out that $\sigma_i = 0$ for $i = r + 1, ..., n$.

14.2 Solution

There are various derivations, e.g. along the lines of Section 13.2.1 (one maximizes the quantity UAV'), but let's go quickly to the answer:

1. Compute the eigenvalues $\sigma_1, ..., \sigma_n$ and eigenvectors $q_1, ..., q_n$ of $A'A$.
2. Since $A'A$ is symmetric and positive-semidefinite, its eigenvalues will be nonnegative and its length-1 eigenvectors will be orthogonal.
3. Set Σ to $diag(\sqrt{\sigma_1}, ..., \sqrt{\sigma_n})$, the nonincreasing list of eigenvalues. Set the columns of V to the corresponding eigenvectors (recall Section 12.3.3).
4. Set

$$u_i = \frac{1}{\sigma_i} A v_i, \ i = 1, ..., r$$

Using the Gram-Schmidt Method (Section 10.9.3), we can compute (if $r < m$ necessitates it) vectors $u_{r+1}, ...u_m$ so that $u_1, ..., u_m$ is an orthonormal basis for $\mathcal{R}^m$. Set U, decribed in partitioning terms: to

$$U = (u_1|...|u_m)$$

14.3 Checking the Formula

Are U and V really orthogonal, and does $U\Sigma V'$ really work out to be A?

- U is orthogonal.

  This follows immediately from the fact that the columns of U are an orthonormal basis from $\mathcal{R}^m$.

- V is orthogonal, by Section 12.3.2.

- The product evaluates to A, as claimed.

  Using matrix partitioning, we have

  $$U\Sigma = (\sigma_1 u_1 | ... | \sigma_n u_n)$$

  So, again using partitioning,

  $$U\Sigma V' = (\sigma_1 u_1 | ... | \sigma_n u_n) \begin{pmatrix} v_1' \\ ... \\ v_n' \end{pmatrix} = \sum_{i=1}^{n} \sigma_i u_i v_i'$$

  That last expression is equal to A, by Equation 14.2.

14.4 Uniqueness

The SVD can be shown to be unique, if there are no repeated eigenvalues. If the latter exist, one can permute the corresponding columns of U and V, but otherwise uniqueness holds.

14.5 The Fundamental Theorem of Linear Algebra

One can define four fundamental subspaces for any matrix A. We will need the following:

- row space$(A), \mathcal{R}(A)$: $\{x'A\}$ (all linear combinations of rows of A)
- column space$(A), \mathcal{C}(A)$: $\{Ax\}$ (all linear combinations of columns of A)
- null space$(A), \mathcal{N}(A)$: $\{x : Ax = 0\}$
- left null space$(A) = \mathcal{N}(A')$: $\{x : x'A = 0\}$

These subspaces have various properties, which may be verified via SVD. Equation 14.2 will come in very handy.

Consider the column space first. From Equation 14.2 write

$$Ax = \sum_{i=1}^{n} \sigma_i u_i v_i' x \tag{14.3}$$

Note that $v_i' x$ is a scalar, so we have

$$\begin{aligned} Ax &= \sum_{i=1}^{n} [\sigma_i (v_i' x)] u_i \\ &= \sum_{i=1}^{r} [\sigma_i (v_i' x)] u_i \end{aligned} \tag{14.4}$$

So

$$\mathcal{C}(A) \subset \text{ span}(u_1, ..., u_r) \tag{14.5}$$

Now, is the left side equal to all of the right side? Say a member of the right side is $c_1 u_1 + ... + c_r u_r$. Then, using the fact that $\{v_1 ...., v_r\}$ is an orthonormal set, take x to be

$$x = \sum_{j=1}^{r} \frac{c_j}{\sigma_j} v_j$$

Then for each i, we have

$$\sigma_i v_i' x = c_i$$

and

$$Ax = \sum_{i=1}^{r} c_i u_i$$

as desired, so Equation 14.5 is actually an equality.

We have thus characterized one of the four fundamental subspaces:

$\mathcal{C}(A)$ has dimension r, with basis $u_1, ..., u_r$.

Now look at the null space. $Ax = 0$ means that the vector x is orthogonal to each row of A. Thus

$$\begin{aligned} \mathcal{N}(A) &= \mathcal{R}(A)^\perp \\ &= \mathcal{C}(A')^\perp \end{aligned}$$

We already found above a characterization of the column space of a matrix, which we now apply to A': $\mathcal{C}(A')$ has dimension r, with basis $v_1, ..., v_r$. Applying $\perp$, and noting that orthogonality of $v_1, ..., v_n$, we can now characterize a second, and even a third, of the four subspaces:

$\mathcal{N}(A)$ is of dimension $n - r$, with basis $v_{r+1}, ..., v_n$.

$\mathcal{R}(A)$ is of dimension r, with basis $v_1, ..., v_r$.

We leave the fourth subspace as a Your Turn problem.

14.6 The Data Scientist's Swiss Army Knife

There are a great many pieces of information and sources of tools packed into this innocuous-looking factorization. Let's take a look at some:

14.6.1 Rank of A

This is the number of nonzero values in Σ. This follows from Theorem 8.1, since U and V, as orthogonal matrices, are invertible.

It should be noted, though, that if there are some really tiny elements there, one might also entertain the thought that the true rank is less than this size, as these tiny values may be due to roundoff error from 0.

14.6.2 SVD as matrix pseudoinverse

Recall the example in Section 8.1. We could not have dummy-variable columns for both male and female, as their sum would be a column of all 1s, in addition to a column the X data matrix already had. The three columns would then have a nonzero linear combination that evaluates to the 0 vector. Then in Equation 7.8, A (i.e. X) would not be of full rank, and $(A'A)^{-1}$ would not exist.

And yet the equation from which that comes,

$$A'Ab = A'S \tag{14.6}$$

is still valid. We could, as in that example, remove one of the gender columns, thus solving the problem of less than full rank, but the use of *pseudoinverses* (also known as *generalized inverses*) solves the problem directly. If A has hundreds or thousands of columns, say, the use of pseudoinverses may be more convenient.

We omit the formal definition of pseudoinverses and their properties. We will note the use of the latter as the need arises. For now, we state without proof that:

- One of the most famous forms of pseudoinverse, Moore-Penrose, is based on SVD. Given the SVD of a matrix M,

  $$M = U_M \Sigma_M V_M'$$

  its Moore-Penrose inverse, denoted by M^{-1} is

  $$M^- = V_M \Sigma_M^- U_M'$$

  where Σ_M^- is the diagonal matrix obtained form Σ_M by replacing each nonzero element by its reciprocal.

- The Moore-Penrose solution of $Mz = w$ for vectors z and w, is $M^- w$.

By the way, the R function **MASS::ginv** performs the necessary computation for us; we need not call **svd()**.

14.6.3 SVD in linear models

Now apply this to our linear model problem Equation 14.6. The claim is that

$$b = A^{-}S = V\Sigma^{-}U'S$$

solves the equation. Let's check (warning: this will be messy):

As in Section 5.2, where we found that the inverse of a product is the reverse product of the inverses, the same holds for pseudoinverses.

$$\begin{aligned} A'Ab &= (U\Sigma V')'(U\Sigma V')(U\Sigma V')^{-}S \\ &= (V\Sigma U')(U\Sigma V')(V\Sigma^{-}U')S \\ &= V\Sigma^2 V'V\Sigma^{-}U'S \\ &= V\Sigma^{-}U'S \end{aligned}$$

But that is exactly the expression we found for $A^{-1}S$ above.

□

14.7 Example: Census Data

```
library(qeML)
data(svcensus)
# have only 1 categorical/dichotomous variable, for simple example
head(svcensus)
```

```
       age     educ occ wageinc wkswrkd gender
1 50.30082 zzzOther 102   75000      52 female
2 41.10139 zzzOther 101   12300      20   male
3 24.67374 zzzOther 102   15400      52 female
4 50.19951 zzzOther 100       0      52   male
5 51.18112 zzzOther 100     160       1 female
6 57.70413 zzzOther 100       0       0   male
```

```
svc <- svcensus[,-c(2,3)]
svc <- factorsToDummies(svc)
head(svc)
```

```
          age wageinc wkswrkd gender.female gender.male
[1,] 50.30082   75000      52             1           0
[2,] 41.10139   12300      20             0           1
[3,] 24.67374   15400      52             1           0
[4,] 50.19951       0      52             0           1
[5,] 51.18112     160       1             1           0
[6,] 57.70413       0       0             0           1
```

```
x <- cbind(1,svc[,-2])
head(x)
```

```
           age wkswrkd gender.female gender.male
[1,] 1 50.30082      52             1           0
[2,] 1 41.10139      20             0           1
[3,] 1 24.67374      52             1           0
[4,] 1 50.19951      52             0           1
[5,] 1 51.18112       1             1           0
[6,] 1 57.70413       0             0           1
```

```
xminus <- MASS::ginv(x)
bhat <- xminus %*% svc[,2]
bhat
```

```
           [,1]
[1,] -16022.454
[2,]    496.747
[3,]   1372.756
[4,] -13361.634
[5,]  -2660.821
```

```
lm(wageinc ~ .,svcensus[,-c(2,3)])$coef
```

```
(Intercept)         age     wkswrkd  gendermale
 -29384.088     496.747    1372.756   10700.813
```

The two approaches are consistent with each other (though internally they are solving slightly different problems). Note that

$$-13361.634 - (-2660.821) = -10700.81$$

14.8 Application: SVD as the Best Low-Rank Approximation

14.9 Your Turn

Your Turn: Show that in the SVD factorization, U consists of the eigenvectors of AA'.

Your Turn: Find a characterization of the left null space of a matrix.

Your Turn: Write an R function with a matrix as its sole argument, with return value consisting of an R list containing four matrices, representing the four fundamental subspaces of the input argument. Each matrix will consist of columns equal to a basis for the given subspace.

15 Matrix Pseudoinverse

Goals of this chapter:

15.1 SVD as the Minimum-Norm Solution

In an overdetermined linear system such as Equation 14.6, there are many solutions. However, an advantage of Moore-Penrose is that it gives us the *minimum norm* solution. We'll discuss the significance of this shortly, but let's prove it first.

We will need this:

Theorem 15.1 (Multiplication by an Orthogonal Matrix Preserves Norm)**.** *For an orthogonal matrix M and a vector w, $||Mw|| = ||w||$.*

Proof. See the Your Turn problem below. □

Theorem 15.2 (The Moore-Penrose Solution Is Min-Norm)**.** *Consider a matrix B and vector q. Of all solutions x to*

$$Bx = q \tag{15.1}$$

the Moore-Penrose solution minimizes $||x||$.

Proof. Again, write

Adapted from a derivation by Carlo Tomasi

$$B = U\Sigma V',$$

and consider the residual sum of squares

$$||Bx - q||^2 = ||U(\Sigma V'x - U'q)||^2 \tag{15.2}$$

since $UU' = I$.

Thus we can remove the factor U in Equation 15.2, yielding

$$||Bx - q||^2 = ||\Sigma V'x - U'q||^2 \tag{15.3}$$

Rename

$$V'x$$

to y:

$$||Bx - q||^2 = ||\Sigma y - U'q||^2 \tag{15.4}$$

Now bring in the fact that $\sigma_i = 0$ for $i > r$, by writing everything in partitioned fashion. Break U into r and $n - r$ rows,

$$U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

and partition other objects similarly:

$$V = (V_1|V_2),$$

$$\Sigma = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix},$$

and

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

Substituting into Equation 15.4, we have

$$||Bx - q||^2 = || \begin{pmatrix} \Sigma_1 y_1 \\ 0 \end{pmatrix} - \begin{pmatrix} U_1 q \\ U_2 q \end{pmatrix} ||^2 \tag{15.5}$$

This means that y_2 can be anything at all, without changing the residual sum of squares, confirming that there are infinitely many equally-effective solutions!

But if we want x to be of minimum length, we need y to be of minimum length (the shortest y gives us the shortest $x = Vy$, again by the Your Turn problem). And,

$$||y||^2 = ||y_2||^2 + ||y_2||^2$$

Thus we take $y_2 = 0$.

Now, reviewing Equation 15.6, note that the equation

$$\Sigma_1 y_1 = U_1 q$$

has the solution

$$y_1 = \Sigma_1^{-1} U_1 q$$

An interesting sidelight of this is that it means that

$$||Bx - q||^2 = || \begin{pmatrix} 0 \\ U_2 q \end{pmatrix} ||^2, \tag{15.6}$$

i.e. the residual sum of squares depends only on U_2, not U_1. But returning to our quest for the minimum-norm solution, we map back to $x = Vy$, and have

$$x_1 = V_1 \Sigma_1^{-1} U_1' q$$

which is the SVD solution! Thus the SVD solution does have minimum norm.

□

□

to add:

possible reasons:

at interp, min norm suddenly gives more than 1 choice; min norm is like min Var; maybe show unbiased by arguing A'A b = A'S for all A, thus A'E(estreg) = A'beta'Y for all A etc.; this of course holds only for estimable x'beta, meaning x is in row space of A (or equiv, A'A)

condition number is max at interpolation

(size of?) second U depends on min nonzero eigenvalue

my empirical example

```
ovrf <- function(nreps,n,maxP) { load('YearData.save') # yr <- yr[,1:2] nas <- rep(NA,nreps*(maxP-1)) outdf <- data.frame(p=nas,mape=nas) rownum <- 0 for (i in 1:nreps) { idxs <- sample(1:nrow(yr),n) trn <- yr[idxs,] tst <- yr[-idxs,] for (p in 2:maxP) { rownum <- rownum + 1 # out<-qePolyLin(trn[,1:(p+1)], # 'V1',2,holdout=NULL) out <- qeLin(trn[,1:(p+1)],'V1',2,holdout=NULL) preds <- predict(out,tst[,2:(p+1)]) mape <- mean(abs(preds - tst[,1])) outdf[rownum,1] <- p outdf[rownum,2] <- mape print(outdf[rownum,]) } } outdf #run through tapply() for the graph }
```

15.2 Application: Explaining the Mystery of "Double Descent"

Around 2018-2019, one of the statistics field's most deeply-held notions was thrown off its pedestal, largely by some researchers in machine learning. (This is arguably when the idea first became widespread, but for earlier instances, see Marco Loog *et al*, *A brief prehistory of double descent*, PNAS, 2020.) And many of the explanations given have been related to SVD.

15.2.1 Motivating example

Let's consider the Million Song Dataset, varous versions of which are on the Web. Ours is the one with 515345 rows and 91 columns. The first column is the year of release, followed by 90 columns of various audio measurements. The goal is to predict the year from the audio variables.

15.2.2 Overfitting and BEYOND

Recall that a well-known rule of thumb is that the number p of predictors should be less than $\sqrt{n}$, where n is the number of data points. We will abandon this and see what happens.

We will add predictors one at a time, to form models of increasing complexity. As usual, let p denote the number of predictors in a given model. We will also use only a small number of rows, say $n = 30$. We start with $p = 1$, then $p = 2$, eventually reaching $p = n - 1$ (accounting for the 1s column in the A matrix in Equation 7.10), and not stopping even there. We go to $p = n$, $p = n + 1$ and so on.

When we reach $p = n+2$, the matrix A will have more columns than rows, and $(A'A)^{-1}$ will not exist. (See Your Turn problem below.) We can still use SVD to solve for b, but intuitively some kind of major change may happen at that point.

Here is the code and output

```
ovrf <- function(nreps,n,maxP)
 {
    load('YearData.save')
    # yr <- yr[,1:2]
    nas <- rep(NA,nreps*(maxP-1))
    outdf <- data.frame(p=nas,mape=nas)
    rownum <- 0
    for (i in 1:nreps) {
       idxs <- sample(1:nrow(yr),n)
       trn <- yr[idxs,]
       tst <- yr[-idxs,]
       for (p in 2:maxP) {
          rownum <- rownum + 1
          # don't use lm or qeLin, since no SVD; instead, fit polynomial
          # of degree 1, i.e. linear model
          out <- qePolyLin(trn[,1:(p+1)],'V1',1,holdout=NULL)
          preds <- predict(out,tst[,2:(p+1)])
          mape <- mean(abs(preds - tst[,1]))
          outdf[rownum,1] <- p
          outdf[rownum,2] <- mape
          print(outdf[rownum,])
       }
    }
    outdf  #run through tapply() for the graph
 }
```

```
> set.seed(111111); o1 <- ovrf(1,30,40)
  p     mape
1 2 7.865117
  p     mape
2 3 7.876853
  p     mape
3 4 7.984233
  p     mape
4 5 8.631492
  p     mape
5 6 8.608561
  p     mape
6 7 8.734537
  p    mape
7 8 8.74647
  p     mape
8 9 8.746996
   p     mape
9 10 9.042233
    p     mape
10 11 8.795864
    p    mape
11 12 8.88431
    p     mape
12 13 10.06747
    p     mape
13 14 10.17324
    p     mape
14 15 10.68381
    p     mape
15 16 10.41553
    p     mape
16 17 9.892437
    p    mape
17 18 9.71409
    p     mape
18 19 11.25314
    p     mape
19 20 11.88647
```

```
    p     mape
20 21 17.94607
    p     mape
21 22 18.63847
    p     mape
22 23 15.26327
    p     mape
23 24 19.50255
    p     mape
24 25 24.53056
    p     mape
25 26 23.60238
    p     mape
26 27 25.35561
    p    mape
27 28 36.1927
    p     mape
28 29 57.97567
    p     mape
29 30 754.9635
P > N. With polynomial terms and interactions, P is 31.


    p    mape
30 31 741.573
P > N. With polynomial terms and interactions, P is 32.


    p  mape
31 32 477.3
P > N. With polynomial terms and interactions, P is 33.


    p     mape
32 33 525.8362
P > N. With polynomial terms and interactions, P is 34.


    p     mape
33 34 465.0518
```

```
P > N. With polynomial terms and interactions, P is 35.


    p     mape
34 35 761.3692
P > N. With polynomial terms and interactions, P is 36.


    p    mape
35 36 689.829
P > N. With polynomial terms and interactions, P is 37.


    p     mape
36 37 673.8049
P > N. With polynomial terms and interactions, P is 38.


    p     mape
37 38 728.8918
P > N. With polynomial terms and interactions, P is 39.


    p     mape
38 39 666.2301
P > N. With polynomial terms and interactions, P is 40.


    p     mape
39 40 621.9375
```

MAPE here is mean absolute prediction error (calculated on the holdout set).

Here $n = 30$, so $p = 30, 31, ...$ uses SVD, and is overfitting. Indeed, much smaller values of p seem to have overfit as well. But the point is that once we got to the point at which there is no unique solution, MAPE actually improved, a huge shock to the field. It was always assumed that there is no point going beyond the values of p that gives us 0 for the sum of squares.

The graph of MAPE is typically a U-shape. In this case, we seem to have only the right half of a U, but still a U. What was shocking was that sometimes there is a *second* U-shape after we reach 0 sum of squares. Even more shocking, in some case the low point of the second U is lower than that of the first–it pays to radically ovefit.

15.2.3 The classic and modern views

In other words, here is the sea change that occurred in the field around 2018-2019.

Let κ denote the complexity of a model. In the case of a linear model, κ would be the number of predictor columns in our dataset, and there are ways of defining it for other methods.

We might try several values of κ, as we did in the table in **?@eq-nyc**, and then choose the one with smallest MAPE or other accuracy measure.

Before 2018-2019, the view was:

In plotting MAPE against κ, the curve will generally be roughly U-shaped, up to the point at which κ gives us a 0 value of RSS in the training set. That value of κ, called the *interpolation* point, will give us "perfect" prediction in the training set, but very poor prediction in the test set, which is what counts.

We choose the value of κ at which the curve is lowest. There is no point in trying values of κ past the interpolation point.

This was taken for granted throughout the statistics and machine learning fields. But around 2018-2019, machine learning engineers were routinely analyzing dataset of extraordinarily large sizes, using unprecedently large values of κ. And they discovered that, bizarrely, as κ moved past the interpolation point, the curve often went *down*–the second "descent"–often tracing its own second U-shape. And most significantly, the low point of the second U was sometimes below that of the first U! Overfitting–grossly so–may pay off!

So the modern view is:

The first U-shaped curve is as in the classic view. But the curve should be plotted past κ, and the overall minimum may be in that second U.

This is a rare sea change in classic quantitative analysis.

15.2.4 How can this bizarre effect occur?

Let's look further. If at least some our predictors are numeric (with Million Song, they all are), then for $p = n-1$, the (square) matrix A itself will very likely be invertible, and setting

$$b = A^{-1}S$$

will minimize Equation 7.5–with the value of that expression being 0–a perfect fit to the data!

Now again consider $p = n$. Let A_{new} denote our new A (it will be equal to the old one with a new column added on the right), and b_{new} denote a new version of b. We say "*a* new version" here, because there are now infinitely many versions; recall that the SVD version has minimum norm among them, a point we will return to shortly. Let's use A_{old} and b_{ol} to denote the quantities we got for $p = n - 1$.

We can write

$$A_{new} = (A_{old}|c_{n+1})$$

and

$$b_{new} = (b_{old}|b_{n+1})'$$

Thus

$$A'_{new}A_{new}b_{new} = (A'_{old}A_{old} + c_{n+1}c'_{n+1}) \begin{pmatrix} b_{old} \\ b_{n+1} \end{pmatrix}$$

As noted, we now must use SVD. We will still get a perfect fit. To see this, set $b_{n+1} = 0$ above, which gives us the same

fit we got for $n = p - 1$. In fact, there will be infinitely many values of b that achieve that perfect fit. But remember, the SVD solution has minimum norm among them all, a point we will return to shortly.

In considering the effectiveness of an estimator in statistics, we often look at bias and variance. Let's consider bias first.

In our setting here, the quantities of interest are of the form $w'\beta$ the predicted value for an individual who has the characterics w. We don't know β, so we use $w'b$ instead. , estimated by $w'b$. In the non-full rank setting, not all such quantities are physically estimable, but the theory says that any w in the row space of A will be fine. Moreover, it says the least squares estimate of $w'\beta$ will have 0 bias.

Now concerning variance, we have

$$Var(w'b) = w'Cov(b)w$$

In general, shorter random vectors will have less variability, which though not the same as variance/covariance at least gives us the feeling that such an estimator has low variance. Recall that this is the rationale for shrinkage estimators – accept a small amount of bias in return for a reduction in variance.

Thus it is at least plausible that we might get a second U-shape, as the impact of the minimum-norm nature of SVD kicks in. On the other hand, as p grows further, b has more and more components, and likely that minmum-norm b will grow in length at some point, hence the typical later turn back upward of the second U.

15.3 Your Turn

Your Turn: Show that for an orthogonal matrix M and a vector w, $||Mw|| = ||w||$.

Your Turn: Using properties of matrix rank, show that if $p + 1 > n$ in Equation 7.10, the inverse will not exist.

16 Recommender Systems

Goals of this chapter:

17 Neural Networks

Goals of this chapter:

17.1 Tensors

17.2 SGD

17.3 SGD and Double Descent

17.4 Low Rank Approximation of Weight Matrices

17.5 Role of Matrix Condition Number

https://math.stackexchange.com/questions/4362666/relationship-between-condition-number-of-a-matrix-and-eigenvalues