Chapter 4: Numerical Matrix Methods

4.1 Introduction

In this chapter, I will be completely different that it was in the previous chapter. Here, $I_{n\times n}$ denotes the $n \times n$ identity matrix.

Recall the general linear model:

$$Y = X\beta + \varepsilon$$

where

- Y is an $n \times 1$ vector of observations;
- X is an $n \times m$ design matrix;
- β is an $m \times 1$ vector of parameters; and
- ε is an $n \times 1$ vector of iid $N(0, \sigma^2)$ errors.

The best linear unbiased estimate of β is derived by finding a (the) solution $\widehat{\beta}$ to the *normal equations*

$$X'Y = X'X\beta \quad .$$
$$\widehat{\beta} = (X'X)^{-1}X'Y$$

There's lots of good numerical stuff going on here! And how we approach the problem depends on how big n (and, of lesser general concern, m) is.

If n is small, then even the inverse is not such a big deal to calculate, so that computing $\widehat{\beta}$ in the straightforward way above is not so much worse than any of the other things presented in this chapter. However, if n is large, then there are enough arithmetic steps in doing the multiplications and inversion for *roundoff error* to be a serious concern!! So, this chapter presents a handful of approaches for solving the normal equations. All will have advantages; *e.g.*, few computations or ease of implementation. You can't have both, though!

Golden Rule of Numerical Matrix Computation: Don't ever explicitly compute an inverse unless you actually need to (and think about whether you really need to very carefully). For example, I don't need to compute an inverse to find $\widehat{\beta}$.

In order to compare methods, we will talk about computational effort spent in terms of the number of multiplications/divisions an algorithm will perform. Some resources also include the number of additions/subtractions, but these require very little computing time to perform in comparison to the multiplications. We will express these in big-O notation, and typically this will be in terms of the largest dimension n.

4.2 Gaussian Elimination with Backward Substitution

This is either a reminder of what you should have learned in an undergraduate linear algebra course, or a crash-course in matrix manipulation.

The most basic approach here is to look at the normal equations as simply a collection of m equations in m unknowns, which can be solved by Gaussian elimination:

$$\begin{aligned} X'Y &= X'X\beta \\ V_{m\times 1} &= A_{m\times m}\beta_{m\times 1} \\ a_{11}\beta_1 + a_{12}\beta_2 + \ldots + a_{1m}\beta_m &= v_1 \\ a_{21}\beta_1 + a_{22}\beta_2 + \ldots + a_{2m}\beta_m &= v_2 \\ &\vdots \\ a_{m1}\beta_1 + a_{2m}\beta_2 + \ldots + a_{mm}\beta_m &= v_m \end{aligned}$$

Of course, there is some computational expense in getting to that step, due to the matrix multiplications involved. I leave it as an exercise for you to think about how many individual multiplications are involved!

To solve the system of equations, we create an *augmented matrix*

$$[\, A \mid V \,] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \ldots & a_{1m} & v_1 \\ a_{21} & a_{22} & \ldots & a_{2m} & v_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mm} & v_m \end{array}\right]$$

and then manipulate the rows so that an upper-triangular matrix is formed:

$$\left[\begin{array}{cccc|c} a_{11}^* & a_{12}^* & \ldots & a_{1m}^* & v_1^* \\ 0 & a_{22}^* & \ldots & a_{2m}^* & v_2^* \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & a_{mm}^* & v_m^* \end{array}\right].$$

From this point, the solution can be further manipulated by *backward substitution*, which will essentially leave the identity matrix on the left-hand side of the line while the solution will be on the right-hand side.

The manipulations take the following form:

Gaussian Elimination

- Interchange two rows.
- Multiply a row by a nonzero scalar.
- Replace a row by itself plus a multiple of another row.

Backward Substitution

This is pretty obvious when you view the upper-triangular augmented matrix as a system of m equations in m unknowns again.

$$\begin{aligned} a_{11}^*\beta_1 + a_{12}^*\beta_2 + \ldots + a_{1m}^*\beta_m &= v_1^* \\ a_{22}^*\beta_2 + \ldots + a_{2m}^*\beta_m &= v_2^* \\ &\vdots \\ a_{mm}^*\beta_m &= v_m^* \end{aligned}$$

Just solve for $\beta_m, \ldots, \beta_1$ working from the bottom up, but using the row manipulations from above to accomplish that.

Example: Find a solution to

$$\begin{aligned} x + y + 2z &= 1 \\ x - z &= 1 \\ 2x + 3y + 5z &= 4 \end{aligned}$$

The matrix equivalent of this form is

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & -1 \\ 2 & 3 & 5 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 4 \end{pmatrix}$$

and the augmented matrix is

$$\left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 1 & 0 & -1 & 1 \\ 2 & 3 & 5 & 4 \end{array}\right]$$

Putting it through the row eliminations:

$$\left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 1 & 0 & -1 & 1 \\ 2 & 3 & 5 & 4 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 0 & -1 & -3 & 0 \\ 2 & 3 & 5 & 4 \end{array}\right] -R_1 + R_2 \to R_2$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 0 & 1 & 3 & 0 \\ 2 & 3 & 5 & 4 \end{array}\right] -R_2 \to R_2$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 0 & 1 & 3 & 0 \\ 0 & 1 & 1 & 2 \end{array}\right] -2R_1 + R_3 \to R_3$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & -2 & 2 \end{array}\right] -R_2 + R_3 \to R_3$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & -1 \end{array}\right] -\frac{1}{2}R_3 \to R_3$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right] -3R_3 + R_2 \to R_2$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right] -2R_3 + R_1 \to R_1$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right] -R_2 + R_1 \to R_1$$

So the unique solution to the system of equations is $x = 0,\ y = 3,\ z = -1$.

If there were not a unique solution, then the row reduction would have arrived at zeros in one row, with the rest determining the set of solutions.

If row reduction gives us a row with elements on the left all zero, but a non-zero element on the right, then that means there is no solution.

4.1.1 Pivoting

The *pivot element* is the element on the diagonal, particularly when all elements to the left of the diagonal in that row are zero. This is also the element that you'd divide the row by to achieve a 1 in the diagonal element.

Suppose, however, that during the Gaussian elimination calculations, the pivot element is zero. You can't divide through by that element! What to do?!

Simple: just swap that row with one of the rows that has a better potential pivot element. This sort of swapping rows is called *partial pivoting*. Partial pivoting is also necessary when the pivot element is very small. Dividing through by a very small number

yields a very large number, and very large numbers are subject to a great deal of roundoff error, since all of the smaller-order digits are lost.

Full pivoting is when you swap columns, and therefore also the rows. To understand why swapping the rows is also necessary when swapping columns, think about the system of equations again, and where the solution to those equations will be when the row reduction is complete. Full pivoting requires some bookkeeping, as one must keep up with the permutation of the solutions in this scenario. You don't have to do this bookkeeping if you only swap rows.

Note: You might see reference to *Gauss-Jordan* elimination if you look in the literature. Gauss-Jordan elimination differs slightly from Gaussian elimination with backward substitution. In Gauss-Jordan, the pivot element is used to zero out *all* elements in that column, rather than just those elements below the diagonal. Gaussian elimination with backward substitution is faster and more stable than Gauss-Jordan.

4.2 L-U Decomposition

Recall the normal equations:

$$X'Y = X'X\beta\,.$$
$$V = A\beta$$

Suppose we can write A as the product of two square matrices

$$A = L \cdot U$$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix} = \begin{bmatrix} l_{11} & & & 0 \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{m1} & l_{m2} & \dots & l_{mm} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1m} \\ & u_{22} & \dots & u_{2m} \\ & & \ddots & \vdots \\ 0 & & & u_{mm} \end{bmatrix}$$

where L is a lower-triangular matrix, and U is an upper-triangular matrix. Then solving the normal equations for β is just simple (and numerically stable) forward and backward substitution:

$$V = A\beta$$
$$V = L(U\beta)$$
$$V = L\gamma$$
$$\gamma = U\beta$$

We can first solve the third equation for γ, then solve that new equation $U\beta = \gamma$ for β.

Indeed, once we have such a decomposition of A, then V can change and the solution is still just as easy. This was *not* the case with Gaussian elimination with backward substitution! This is *very* handy for multiple experiments with the same design matrix.

And so it remains: Given the square matrix A, how can we determine the L-U decomposition of A? Essentially, it is a complex set of equations with some restrictions. Those equations may be written most generally as

$$l_{i1}u_{1j} + \ldots = a_{ij}$$

where the ending of the sum is determined, of course, by where the zeros in the row/column begin. Since we have an equation for each element of A, there are m^2 equations. Since each element of L and U is an unknown, there are $m^2 + m$ unknowns (which is clear when you see that the diagonal is represented twice). If we specify m of the unknowns, specifically

$$l_{ii} \equiv 1 \;\; i = 1, \ldots, m$$

then we have just as many equations as unknowns and we can solve (!) the system, which doesn't appear to be a straightforward proposition.

An ingenious solution is called *Crout's algorithm*, and it almost trivially solves this system of equations for all of the l's and u's just by arranging them in a certain order!

Here are what the equations look like when you multiply them out for a 4×4 example:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} =$$

$$\begin{bmatrix} l_{11}u_{11} & l_{11}u_{12} & l_{11}u_{13} & l_{11}u_{14} \\ l_{21}u_{11} & l_{21}u_{12} + l_{22}u_{22} & l_{21}u_{13} + l_{22}u_{23} & l_{21}u_{14} + l_{22}u_{24} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + l_{33}u_{33} & l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} \\ l_{41}u_{11} & l_{41}u_{12} + l_{42}u_{22} & l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44}u_{44} \end{bmatrix}$$

$=$

$$\begin{bmatrix} \underline{\boldsymbol{u_{11}}} & \underline{\boldsymbol{u_{12}}} & \underline{\boldsymbol{u_{13}}} & \underline{\boldsymbol{u_{14}}} \\ \underline{\boldsymbol{l_{21}}}u_{11} & l_{21}u_{12} + \underline{\boldsymbol{u_{22}}} & l_{21}u_{13} + \underline{\boldsymbol{u_{23}}} & l_{21}u_{14} + \underline{\boldsymbol{u_{24}}} \\ \underline{\boldsymbol{l_{31}}}u_{11} & l_{31}u_{12} + \underline{\boldsymbol{l_{32}}}u_{22} & l_{31}u_{13} + l_{32}u_{23} + \underline{\boldsymbol{u_{33}}} & l_{31}u_{14} + l_{32}u_{24} + \underline{\boldsymbol{u_{34}}} \\ \underline{\boldsymbol{l_{41}}}u_{11} & l_{41}u_{12} + \underline{\boldsymbol{l_{42}}}u_{22} & l_{41}u_{13} + l_{42}u_{23} + \underline{\boldsymbol{l_{43}}}u_{33} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + \underline{\boldsymbol{u_{44}}} \end{bmatrix}$$

The resulting sequence for solving for the unknown u's and l's, highlighted in the above matrix with underlined bold, is Crout's algorithm (though it should really be called *Crout's observation* since it is hardly an algorithm). Notice that the $u_{1\cdot}$'s are trivially solved. Then so are the $l_{\cdot 1}$'s. Then so are the $u_{2\cdot}$'s. Etcetera.

Crout's algorithm for the L-U decomposition of A.

Input: $A = [a_{ij}]$
Output: $L = [l_{ij}]$ and $U = [u_{ij}]$

1. Set
   $l_{kk} = 1,\ k = 1, \ldots, m$
   $k = 1$ (k is the pivot element at the moment)
2. For $j = k, \ldots, m$,

$$u_{kj} = a_{kj} - \sum_{n=1}^{k-1} l_{kn} u_{nj}$$

   where the sum is taken to be zero if $j = 1$.
3. For $i = k + 1, \ldots, m$,

$$l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{n=1}^{k-1} l_{in} u_{nk} \right)$$

4. $k = k + 1$
5. If $k > m$, stop.
6. Go to Step 2.

You may have to go through the algorithm yourself to see this, but you will have already solved for the l's and u's that you need once you find that you need them!

Notice also that every a_{ij} is used exactly once in solving for each l_{ij} and u_{ij}, and never again after that. So when considering storage, you can replace the matrix A with both the matrices L and U:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & \ldots & u_{1m} \\ l_{21} & u_{22} & u_{23} & u_{24} & \ldots & u_{2m} \\ l_{31} & l_{32} & u_{33} & u_{34} & \ldots & u_{3m} \\ l_{41} & l_{42} & l_{43} & u_{44} & \ldots & u_{4m} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{m1} & l_{m2} & l_{m3} & l_{m4} & \ldots & u_{mm} \end{bmatrix}$$

which we can do since we've already set all of the $l_{ii} = 1$.

Example: Let's find an LU decomposition of

$$A = \begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix}$$

For $k = 1$, get the u's:

$$\begin{bmatrix} \mathbf{6} & \mathbf{-2} & \mathbf{0} \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix}$$

and then get the l's:

$$\begin{bmatrix} 6 & -2 & 0 \\ \mathbf{\frac{3}{2}} & -1 & 1 \\ \mathbf{\frac{1}{2}} & 7 & 5 \end{bmatrix}$$

Then $k = 2$, and the u's are

$$\begin{bmatrix} 6 & -2 & 0 \\ \frac{3}{2} & \mathbf{2} & \mathbf{1} \\ \frac{1}{2} & 4 & 5 \end{bmatrix}$$

while the l and the final u (when $k = 3$) become

$$\begin{bmatrix} 6 & -2 & 0 \\ \frac{3}{2} & 2 & 1 \\ \frac{1}{2} & \mathbf{4} & \mathbf{1} \end{bmatrix}$$

So

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{1}{2} & 4 & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} 6 & -2 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

and to verify that their product is A:

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{1}{2} & 4 & 1 \end{bmatrix} \begin{bmatrix} 6 & -2 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix}$$

Replacing the elements of A as we go through the algorithm makes pivoting easier to accomplish. Pivoting is just as necessary for the numerical stability of Crout's algorithm as it was for Gaussian elimination. In this case, though, we can only do partial pivoting. This means that we don't actually decompose A into LU, but we decompose a row-equivalent version of A (and if we deep track of the row permutations [by permuting the identity matrix], this decomposition is just as useful as the original would have been).

Pivoting is accomplished in the following manner. Until we actually divide by a diagonal u, the arithmetic is all the same. So, until the actual division by u_{jj} is done in Step 3, we can switch row i with one beneath it for a better (bigger) pivot element. So when pivoting, first do the arithmetic, then choose the maximum in the column, from the pivot element down, and then divide by the new, pivoted u_{jj}.

To do partial pivoting, we need to modify Step 3 like this:

Because we have already performed this same addition on the pivot position k in Step 2, perform the addition on the elements below the pivot. Then compare the pivot with all elements velow it in that column. Switch as necessary, then complete the division.

3a. For $i = k+1, \ldots, m$,

$$l_{ik} = a_{ik} - \sum_{n=1}^{k-1} l_{in} u_{nk}$$

3b. Find i^*, the row in which $\max\{|u_{kk}|, |l_{ik}|, i > k\}$ is located.
3c. Swap row k with row i^*.
3d. $l_{ik} = l_{ik}/u_{kk}$

Let's go through an example with pivoting involved to see what difference is made. Recall that the matrix we're working with is

$$A = \begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix}$$

Focus on the first pivot element. There is no arithmetic to do on that first column, since the sum in Step 3a is zero.

Continuing on, in that first column, 9 is the largest, so swap rows 2 and 1.

$$\begin{bmatrix} \mathbf{9} & \mathbf{-1} & \mathbf{1} \\ \mathbf{6} & \mathbf{-2} & \mathbf{0} \\ 3 & 7 & 5 \end{bmatrix}$$

Now, we complete the l's in the first column by dividing by 9.

$$\begin{bmatrix} 9 & -1 & 1 \\ \mathbf{\frac{2}{3}} & -2 & 0 \\ \mathbf{\frac{1}{3}} & 7 & 5 \end{bmatrix}$$

Now $k = 2$, so find the u's in that row, and do the arithmetic on the u's in that row and the l's in that column.

$$\begin{bmatrix} 9 & -1 & 1 \\ \frac{2}{3} & \mathbf{-\frac{4}{3}} & \mathbf{0} \\ \frac{1}{3} & \mathbf{\frac{22}{3}} & 5 \end{bmatrix}$$

The largest element in absolute value below the pivot element is in the third row, so swap the second and third rows.

$$\begin{bmatrix} 9 & -1 & 1 \\ \mathbf{\frac{1}{3}} & \mathbf{\frac{22}{3}} & \mathbf{5} \\ \mathbf{\frac{2}{3}} & \mathbf{-\frac{4}{3}} & \mathbf{0} \end{bmatrix}$$

Finish by dividing by the pivot element.

$$\begin{bmatrix} 9 & -1 & 1 \\ \frac{1}{3} & \frac{22}{3} & 5 \\ \frac{2}{3} & \mathbf{-\frac{2}{11}} & 0 \end{bmatrix}$$

Finally, $k = 3$, so find that last u.

$$\begin{bmatrix} 9 & -1 & 1 \\ \frac{1}{3} & \frac{22}{3} & \mathbf{\frac{14}{3}} \\ \frac{2}{3} & -\frac{2}{11} & \mathbf{\frac{2}{11}} \end{bmatrix}$$

When we multiply L and U back out, we don't quite get A back:

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{2}{3} & -\frac{2}{11} & 1 \end{bmatrix} \begin{bmatrix} 9 & -1 & 1 \\ 0 & \frac{22}{3} & \frac{14}{3} \\ 0 & 0 & \frac{2}{11} \end{bmatrix} = \begin{bmatrix} 9 & -1 & 1 \\ 3 & 7 & 5 \\ 6 & -2 & 0 \end{bmatrix}$$

So, clearly, pivoting will scramble the rows. That is why you need to keep track of the row permutations!

The LU decomposition can be used to solve a system of linear equations via forward & backward substitution. Let's use these two decompositions to find x, y, and z for

$$\begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 14 \\ 21 \\ 9 \end{pmatrix}$$

Using the first LU decomposition,

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 14 \\ \frac{3}{2} & 1 & 0 & 21 \\ \frac{1}{2} & 4 & 1 & 9 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 14 \\ 0 & 1 & 0 & 0 \\ 0 & 4 & 1 & 2 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 14 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

and

$$\left[\begin{array}{ccc|c} 6 & -2 & 0 & 14 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 6 & -2 & 0 & 14 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 2 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

So $x = 2, y = -1, z = 2$

Using the other decomposition, we have to rearrange the rows in the same manner as we pivoted:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 21 \\ \frac{1}{3} & 1 & 0 & 9 \\ \frac{2}{3} & -\frac{2}{11} & 1 & 14 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 21 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & \frac{4}{11} \end{array}\right]$$

and

$$\left[\begin{array}{ccc|c} 9 & -1 & 1 & 21 \\ 0 & \frac{22}{3} & \frac{14}{3} & 2 \\ 0 & 0 & \frac{2}{11} & \frac{4}{11} \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

If we didn't do that permutation, then you get a very ugly, very wrong answer.

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 14 \\ \frac{1}{3} & 1 & 0 & 21 \\ \frac{2}{3} & -\frac{2}{11} & 1 & 9 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 14 \\ 0 & 1 & 0 & \frac{49}{3} \\ 0 & 0 & 1 & \frac{115}{33} \end{array}\right]$$

and

$$\left[\begin{array}{ccc|c} 9 & -1 & 1 & 14 \\ 0 & \frac{22}{3} & \frac{14}{3} & \frac{49}{3} \\ 0 & 0 & \frac{2}{11} & \frac{115}{33} \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & -\frac{37}{22} \\ 0 & 1 & 0 & -\frac{329}{33} \\ 0 & 0 & 1 & \frac{115}{6} \end{array}\right]$$

LU decompositions can be used for more than just solving linear systems of equations. They can also be used to determine inverses and determinants.

4.2.1 Using LU Decomposition to Find an Inverse

Once you have determined the LU decomposition, you can use it to find the inverse of a matrix by repeatedly solving the linear equations with each column of the identity matrix, resulting in the corresponding column of the inverse.

Picking on our toy matrix from the previous examples in this section, the inverse of

$$\begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix}$$

is then

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ \frac{3}{2} & 1 & 0 & 0 \\ \frac{1}{2} & 4 & 1 & 0 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -\frac{3}{2} \\ 0 & 0 & 1 & \frac{11}{2} \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ \frac{3}{2} & 1 & 0 & 1 \\ \frac{1}{2} & 4 & 1 & 0 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -4 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ \frac{3}{2} & 1 & 0 & 0 \\ \frac{1}{2} & 4 & 1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

and

$$\left[\begin{array}{ccc|c} 6 & -2 & 0 & 1 \\ 0 & 2 & 1 & -\frac{3}{2} \\ 0 & 0 & 1 & \frac{11}{2} \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -\frac{7}{2} \\ 0 & 0 & 1 & \frac{11}{2} \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 6 & -2 & 0 & 0 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & -4 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & \frac{5}{6} \\ 0 & 1 & 0 & \frac{5}{2} \\ 0 & 0 & 1 & -4 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 6 & -2 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & -\frac{1}{6} \\ 0 & 1 & 0 & -\frac{1}{2} \\ 0 & 0 & 1 & 1 \end{array}\right]$$

So the inverse is

$$\begin{bmatrix} -1 & \frac{5}{6} & -\frac{1}{6} \\ -\frac{7}{2} & \frac{5}{2} & -\frac{1}{2} \\ \frac{11}{2} & -4 & 1 \end{bmatrix}$$

which you can verify is correct on your own.

4.2.2 Using the LU Decomposition to Find the Determinant

The determinant of a matrix is very easy once you know its LU decomposition:

$$|A| = |L| \cdot |U| = \prod_{i=1}^{m} u_{ii}$$

since the determinant of an upper- or lower-triangular matrix is just the product of its diagonal elements.

4.3 Cholesky Decomposition

This one is pretty handy for statistics. First, a few reminders about matrices are necessary for the discussion.

A square matrix A is said to be *symmetric* if $A' = A$.

A square matrix A is *positive definite* if $x'Ax > 0 \, \forall \, x \in \mathbb{R}^m$. If the inequality sign is changed to $\geq$, then A is called *positive semidefinite* or *non-negative definite*.

If we take A to be as in the normal equations, then $A = X'X$ is certainly symmetric. In many cases, it is also positive definite.

In statistics, covariance matrices are generally symmetric and positive definite. (A covariance matrix is not symmetric if the random process is periodic in its second moments; and it is not positive definite if any of the random variables in the associated vector are degenerately constant.)

In the case of a symmetric and positive definite A, we can do a special version of the LU decomposition in which $U = L'$. This decomposition is called the *Cholesky decomposition*, and is about twice as fast as the LU decomposition, and very numerically stable. Let's look at what happens to $A = LL'$, in terms of Crout's solution:

In this case, we don't have to restrict the l_{ii}'s as we did before; indeed, we wouldn't even want to! And we have that $u_{ij} = l'_{ij} = l_{ji}$. Symmetry, which implies that $a_{ij} = a_{ji}$, also buys us a lot.

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} =$$

$$\begin{bmatrix} \boldsymbol{l_{11}^2} & & & \\ \boldsymbol{l_{21}} l_{11} & l_{21}^2 + \boldsymbol{l_{22}^2} & & \\ \boldsymbol{l_{31}} l_{11} & l_{31} l_{21} + \boldsymbol{l_{32}} l_{22} & l_{31}^2 + l_{32}^2 + \boldsymbol{l_{33}^2} & \\ \boldsymbol{l_{41}} l_{11} & l_{41} l_{21} + \boldsymbol{l_{42}} l_{22} & l_{41} l_{31} + l_{42} l_{32} + \boldsymbol{l_{43}} l_{33} & l_{41}^2 + l_{42}^2 + l_{43}^2 + \boldsymbol{l_{44}^2} \end{bmatrix}$$

No wonder the Cholesky is twice as fast as general LU! And, Cholesky's is so stable that pivoting is unnecessary! Positive definite means that you won't be seeing zero on the diagonal. So Crout's algorithm can be modified as follows:

Algorithm: Cholesky decomposition

Input: Symmetric, positive definite $A = [a_{ij}]$
Output: $L = [l_{ij}]$

1. Set
   $k = 1$ (k is the pivot element pointer)
2. $$l_{kk} = \left(a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2 \right)^{\frac{1}{2}}$$
   where the sum is taken to be zero if $k = 1$.
3. For $i = k + 1, \ldots, m$,
   $$l_{ik} = \frac{1}{l_{kk}} \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj} \right)$$
4. $k = k + 1$
5. If $k > m$, stop.
6. Go to Step 2.

Of course, you can use the Cholesky decomposition to solve linear systems, invert symmetric positive definite matrices, and find determinants as well. The code that you write especially for the linear systems and the inversion can be simplified over that using LU decomposition.

Note: If A is positive definite, then $|A| > 0$ as well.

4.4 QR Decomposition

A square matrix A can be decomposed into
fv

$$A = QR$$

where R us an upper-triangular matrix and Q is an *orthonormal* matrix (that is, $Q'Q = I$). This can be used to solve a linear system of equations in the following manner:

$$\begin{aligned} A\beta &= V \\ QR\beta &= V \\ Q'QR\beta &= Q'V \\ R\beta &= Q'V \end{aligned}$$

The last equation is easily solved since R is an upper-triangular matrix.

Any matrix, in fact, can be decomposed into an orthonormal matrix and an upper-triangular type of matrix, though there are some very nice properties for the decomposition of a square positive-definite matrix that we'll discuss later.

Let's start this discussion with the least squares equation:

$$Y = X\beta$$

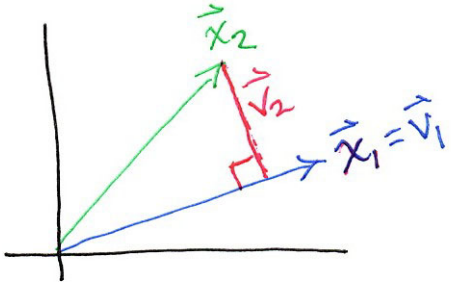
The $n \times m$ design matrix X which is of full rank can be factored as an $n \times m$ matrix Q, the columns of which form an orthonormal basis for the column space of X; and R, which is an $m \times m$ upper-triangular, invertible matrix with positive entries on its diagonal.

When we think about how we accomplish a QR decomposition, we must first ask how we construct the orthonormal basis for X.

4.4.1 The Gram-Schmidt Orthogonalization Process.

Let $\{x_1, x_2, \ldots, x_m\}$ be a basis for a subspace of $\mathbb{R}^n$. That implies that $x_j \in \mathbb{R}^n \; \forall$ $j = 1, \ldots, m$. Then we rotate the vectors until they are mutually orthogonal, forming the new orthogonal basis $\{v_1, \ldots, v_m\}$.

How do we accomplish that? Consider two vectors, x_1 and x_2, which are not already orthogonal. Fix x_1; that is now v_1, the first vector in the orthogonal basis. Next, we need to reconstruct x_2 so that it is orthogonal to $x_1 = v_1$. We need to find the *projection* of x_2 onto x_1.



Recall: The dot product of a vector $x \in \mathbb{R}^n$ with a vector $y \in \mathbb{R}^n$ is

$$x \cdot y = \sum_{i=1}^{n} x_i y_i = x'y,$$

which is a scalar.

The projection of x_2 onto x_1 is given by $\frac{x_2 \cdot x_1}{x_1 \cdot x_1} x_1$, which is just a re-scaling of x_1. (See picture.) If we subtract this projection from x_2, then the result is orthogonal to x_1:

$$\begin{aligned} x_1 \cdot \left(x_2 - \frac{x_2 \cdot x_1}{x_1 \cdot x_1} x_1 \right) &= x_1 \cdot x_2 - x_1 \cdot \left(\frac{x_2 \cdot x_1}{x_1 \cdot x_1} \right) x_1 \\ &= x_1 \cdot x_2 - \frac{x_2 \cdot x_1}{x_1 \cdot x_1} (x_1 \cdot x_1) \\ &= 0 \end{aligned}$$

The Gram-Schmidt orthogolization procedure is successive projections onto vectors already orthogonalized.

$$\begin{aligned} v_1 &= x_1 \\ v_2 &= x_2 - \frac{x_2 \cdot v_1}{v_1 \cdot v_1} v_1 \\ v_3 &= x_3 - \frac{x_3 \cdot v_1}{v_1 \cdot v_1} v_1 - \frac{x_3 \cdot v_2}{v_2 \cdot v_2} v_2 \\ &\vdots \\ v_p &= x_p - \frac{x_p \cdot v_1}{v_1 \cdot v_1} v_1 - \ldots - \frac{x_p \cdot v_{p-1}}{v_{p-1} \cdot v_{p-1}} v_{p-1} \end{aligned}$$

Then $\{v_1, \ldots, v_p\}$ is an orthogonal basis for $\text{sp}\{x_1, x_2, \ldots, x_m\}$.

You make the orthogonal basis an *orthonormal basis* simply by normalizing all of the elements of the basis.

Example: Find an orthonormal basis for the space spanned by the columns of the matrix

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

These vectors are clearly not orthogonal: their dot-products are not equal to zero. They are also not normal.

The Gram-Schmidt orthogonalization of these columns is then

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{\begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}{\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{3}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{3}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix}$$

$$v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \frac{x_3 \cdot v_1}{v_1 \cdot v_1} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{x_3 \cdot v_2}{v_2 \cdot v_2} \begin{bmatrix} -\frac{3}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \frac{2}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{\frac{2}{4}}{\frac{3}{4}} \begin{bmatrix} -\frac{3}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{2}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

It is easily verified that these are all mutually orthogonal. The orthonormal basis is then given by

$$\left\{ \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} -\frac{3}{2\sqrt{3}} \\ \frac{1}{2\sqrt{3}} \\ \frac{1}{2\sqrt{3}} \\ \frac{1}{2\sqrt{3}} \end{bmatrix}, \begin{bmatrix} 0 \\ -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{bmatrix} \right\}$$

So if that was our design matrix X, then

$$Q = \begin{bmatrix} \frac{1}{2} & -\frac{3}{2\sqrt{3}} & 0 \\ \frac{1}{2} & \frac{1}{2\sqrt{3}} & -\frac{2}{\sqrt{6}} \\ \frac{1}{2} & \frac{1}{2\sqrt{3}} & \frac{1}{\sqrt{6}} \\ \frac{1}{2} & \frac{1}{2\sqrt{3}} & \frac{1}{\sqrt{6}} \end{bmatrix}$$

and you can easily see that $Q'Q = I_{3\times 3}$, but since Q is not square $QQ' \neq I_{4\times 4}$.

Now, how do we find R? Well, if $X = QR$, then $Q'X = Q'QR = R$. Thus

$$Q'X = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{3}{2\sqrt{3}} & \frac{1}{2\sqrt{3}} & \frac{1}{2\sqrt{3}} & \frac{1}{2\sqrt{3}} \\ 0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$Q'X = \begin{bmatrix} 2 & \frac{3}{2} & 1 \\ 0 & \frac{3}{2\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & 0 & \frac{2}{\sqrt{6}} \end{bmatrix} = R$$

4.4.2 Givens Reduction a.k.a, Givens QR Decomposition; Givens Rotation.

This is a handy-dandy decomposition when you're doing least-squares estimation under forward stepwise regression or leave-one-out schemes.

A *Givens rotation matrix* is designed to zero-out a particular element of a matrix or vector. It does this by rotating that (column) vector through an angle θ until the desired component is zero.

For example, suppose we want to zero-out the lower element of the vector $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. The Givens matrix which will accomplish that is

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \equiv \begin{bmatrix} c & \kappa \\ -\kappa & c \end{bmatrix}.$$

So let's determine the values of c and κ that will accomplish our goal.

$$\begin{bmatrix} c & \kappa \\ -\kappa & c \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} cx_1 + \kappa x_2 \\ -\kappa x_1 + cx_2 \end{pmatrix}$$

Then we need to solve $-\kappa x_1 + cx_2 = 0$ subject to $c^2 + \kappa^2 = 1$, which yields

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}$$
$$\kappa = \frac{x_2}{\sqrt{x_1^2+x_2^2}}$$

which makes a lot of sense when you think of the vector x in terms of polar coordinates. When we make that multiplication,

$$\begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ -\frac{x_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sqrt{x_1^2+x_2^2} \\ 0 \end{pmatrix}$$

Note that this little matrix is orthonormal!

$$\begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ -\frac{x_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} \end{bmatrix} \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & -\frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ \frac{x_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} \end{bmatrix} = I_{2\times 2}$$

So, we can insert this little 2×2 matrix into a larger $n \times n$ identity matrix to creatively zero-out a chosen element below the diagonal of any $n \times m$ matrix! And that matrix, too, is orthonormal.

And the product of two orthonormal matrices is orthonormal...

Example: Transform $X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ into an upper-triangular matrix with Givens rotations.

The Givens rotation to zero-out the (4,1) element is the leftmost matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & 0 & 0 \end{bmatrix}$$

It just so-happened to zero-out the last element in the other columns as well, because those elements were identical.

The Givens rotation to zero-out the (3,1) element (as well as the (3,2) element) is the leftmost matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & -\frac{\sqrt{2}}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \sqrt{3} & \sqrt{3} & \frac{2}{\sqrt{3}} \\ 0 & 0 & \frac{\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & 0 \end{bmatrix}$$

One more rotation and we'll have something upper-triangular, though not square.

$$\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \sqrt{3} & \sqrt{3} & \frac{2}{\sqrt{3}} \\ 0 & 0 & \frac{\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & \frac{3}{2} & 1 \\ 0 & \frac{\sqrt{3}}{2} = \frac{3}{2\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & 0 & \frac{\sqrt{2}}{\sqrt{3}} = \frac{2}{\sqrt{6}} \\ 0 & 0 & 0 \end{bmatrix}$$

That resulting matrix should look very familiar!! The sub-matrix above the last row of zeros is identical to what we got using the Gram-Schmidt approach to QR decomposition!

So the Givens rotations are just another approach to QR decomposition, but with a different Q. Recall that before, we had $Q_{n\times m}R_{m\times m}$ and solved the system by computing

$$Q'Y = Q'X\beta = R\beta$$

and then doing backwards substitution. This time, we have $Q_{n\times n}R_{n\times m} = Q_{n\times n}\begin{bmatrix} R_{m\times m} \\ 0 \end{bmatrix}$, where $Q = (\text{product of the Givens rotations})'$. So, we *implicitly* find Q' by doing the Givens rotations, and come down to

$$Q'Y = Q'X\beta = \begin{bmatrix} R \\ 0 \end{bmatrix}\beta$$

giving us the same backwards substitution that we did before.

Note that using Givens rotations is more computationally expensive than doing Gram-Schmidt. However, Givens rotations come in handy for forward stepwise regression: Recall in that model selection scheme, you add variables to the model, testing their suitability along the way. Adding a variable to the model simply adds a column to the design matrix and a paramater to β; so, just add one more Givens rotation to find $\widehat{\beta}$.

Givens rotations are also becoming very popular among those who are into parallel programming, as they are clearly well-suited to parallellizing.

4.5 Singular-Value Decomposition (SVD)

This decomposition is very similar to the QR decomposition. In this instance, though, we are decomposing an $n \times m$ matrix X into

$$X = U_{n\times n}\mathcal{D}_{n\times m}V'_{m\times m} = U\begin{bmatrix} D_{m\times m} \\ 0 \end{bmatrix}V'$$

where U and V are both orthonormal matrices, and D is a diagonal matrix. An alternate way that the SVD is presented is: There exist orthonormal matrices $U_{n\times n}$ and $V_{m\times m}$ such that

$$U'XV = \mathcal{D}.$$

The elements on the diagonal are called the *singular values* of the matrix X.

Note: We can also construct and SVD that looks like $X = U_{n\times m}D_{m\times m}V'_{m\times m}$, with the difference being that the square matrices are the latter two, rather than the outside two.

The column vectors of U are the normed eigenvectors of XX', and the column vectors of V are the normed eigenvectors of $X'X$. They are also called the *left* and *right* singular vectors of X. The singular values of X then are the square roots of the eigenvalues of $X'X$ or XX'.

Recall: *eigenvalue* and *eigenvector*.

A square matrix A is said to have an *eigenvector* x corresponding to an *eigenvalue* λ if $Ax = \lambda x$. Multiples of eigenvectors, though satisfying the equation, are not considered to be eigenvectors, nor is the vector $x = 0$. The eigenvectors and eigenvalues together are called the *eigensystem*.

Finding the the SVD or the eigensystem of a matrix is very tricky business in general. There are several fairly complicated methods for doing it, and even *Numerical Recipes* advocates using a canned routine to do these! We'll stick to something reasonably simple, even though it isn't the fastest or most numerically stable approach available.

Algorithm Outline: The Singular Value Decomposition.

Let X be an $n \times m$ matrix. It doesn't have to be of full rank.

1. Find the eigenvalues of $X'X$ and arrange them in descending order.
2. Find the orthogonal eigenvectors of the matrix $X'X$ corresponding to the obtained eigenvalues and arrange them in the same order to form the column-vectors of the matrix V.
3. Form a diagonal matrix $\mathcal{D}$, placing the square roots of the eigenvalues of $X'X$ on the leading diagonal of it, in descending order.
4. Find the first m column vectors of the matrix U:

$$u_i = \frac{1}{d_i}Xv_i,\ i = 1, \ldots, m.$$

5. Complete U by the Gram-Schmidt orthogonalization procedure.

Now let's address finding the eigensystem. In order to do the SVD, we will need to find eigenvalues. To find eigenvalues, we must do a series of *Jacobi* rotations, which are very similar to a Givens rotation, with the same goal at each step: zero-out a particular element of the matrix. Jacobi rotations are specifically for square symmetric

matrices. Successive applications of the Jacobi rotations will converge to a diagonal matrix.

Suppose we want to zero out $a_{pq} = a_{qp}$. We do something similar to what we did for the Givens rotation as before, but with symmetry in mind.

$$c = \frac{1}{\sqrt{t^2+1}} \text{ and } \kappa = tc$$

where

$$t = \frac{\text{sgn}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}}$$
$$\theta = \frac{a_{qq} - a_{pp}}{2a_{pq}}$$

The c is placed in the qq and pp elements, and $\pm\kappa$ in the pq and qp elements as is appropriate. So rather than placing the rotation into the identity matrix as an intact block, we break it up. Call this matrix J.

BROKEN LINK***FIX (See *Numerical Recipes* as well as `http://beige.ucs.indiana.edu/ B673/node24.html` for where all of this comes from. It is the result of going through the plane rotations and solving a quadratic equation in θ.)

Then the Jacobi rotation is carried out by

FIX This notation is not consistent with following notation e.g. $V'AV$ on the next? page

$$JAJ'.$$

This will zero-out the pq and qp elements. Note that we don't have to do the actual multiplication every time - we can derive what will happen to the elements in rows p and q, as well as columns p and q:

$$a_{pp}^{new} = a_{pp} - ta_{pq}$$

$$a_{qq}^{new} = a_{qq} + ta_{pq}$$

$$a_{rp}^{new} = a_{rp} - \kappa\left(a_{rq} + \frac{1-c}{\kappa}a_{rp}\right) = a_{pr}^{new}, r \neq p, r \neq q$$

$$a_{rq}^{new} = a_{rq} + \kappa\left(a_{rp} - \frac{1-c}{\kappa}a_{rq}\right) = a_{qr}^{new}, r \neq p, r \neq q$$

$$a_{pq}^{new} = 0 = a_{qp}^{new}$$

At the end of all of this (*i.e.*, when convergence has occurred), we have

$$D = V'AV$$

where the columns of

$$V = J_1 \cdot J_2 \cdot J_3 \cdot \ldots \cdot J_n$$

are the eigenvectors of A. We can even update V without carrying out the multiplications: If V is initially the identity matrix,

$$v_{rs}^{new} = v_{rs}, s \neq p, s \neq q, r = 1, \ldots, m$$

$$v_{rp}^{new} = v_{rp} - \kappa\left(v_{rq} + \frac{1-c}{\kappa}v_{rp}\right), r = 1, \ldots, m$$

$$v_{rq}^{new} = v_{rq} + \kappa\left(v_{rp} - \frac{1-c}{\kappa}v_{rq}\right), r = 1, \ldots, m$$

We proceed in a systematic way, first zeroing-out the elements (1,2), (1,3), (1,4) ... then (2,3), (2,4), ... etc. One pass such as this through every element below the diagonal is called a *sweep*. When that sweep is completed, sweep again. Do that until the off-diagonal elements are close to 0.

Example: Find the eigenvalues and eigenvectors of $X'X$ when $X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

$$X'X = \begin{bmatrix} 4 & 3 & 2 \\ 3 & 3 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

To zero out the (1,2) = (2,1) elements: $p = 1, q = 2$

$$\theta = \frac{a_{qq} - a_{pp}}{2a_{pq}} = \frac{3-4}{2(3)} = -\frac{1}{6}$$

$$t = \frac{\text{sgn}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}} = \frac{-1}{\left|\frac{1}{6}\right| + \sqrt{\frac{1}{36} + 1}} = \frac{-6}{1 + \sqrt{37}}$$

$$c = \frac{1}{\sqrt{t^2 + 1}} = \frac{1}{\sqrt{\left(\frac{6}{1+\sqrt{37}}\right)^2 + 1}} = 0.76302$$

$$\kappa = tc = -0.6463749$$

Let's actually do the first rotation:

$$\begin{bmatrix} c & -\kappa & 0 \\ \kappa & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & 2 \\ 3 & 3 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} c & \kappa & 0 \\ -\kappa & c & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 4.9912047 & 4.2281847 & 2.8187898 \\ -0.2964396 & 0.3499353 & 0.2332902 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} c & \kappa & 0 \\ -\kappa & c & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 6.5413815 & 0.00000005 & 2.8187898 \\ 0.00000005 & 0.4586188 & 0.2332902 \\ 2.8187898 & 0.2332902 & 2 \end{bmatrix}$$

If we just used the pre-calculated substitutions:

$$\begin{bmatrix} 4-3t & 0 & 2-\kappa\left(2+\frac{1-c}{\kappa}2\right) \\ 0 & 3+3t & 2+\kappa\left(2-\frac{1-c}{\kappa}2\right) \\ 2-\kappa\left(2+\frac{1-c}{\kappa}2\right) & 2+\kappa\left(2-\frac{1-c}{\kappa}2\right) & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 6.5413815 & 0 & 2.8187898 \\ 0 & 0.4586187 & 0.2332902 \\ 2.8187898 & 0.2332902 & 2 \end{bmatrix}$$

Now, we can iterate! $p = 1, q = 3$

$$\theta = \frac{a_{qq} - a_{pp}}{2a_{pq}} = \frac{2 - 6.5413815}{2(2.8187898)} = -0.8055552$$
$$t = \frac{\text{sgn}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}} = -0.4785473$$
$$c = \frac{1}{\sqrt{t^2 + 1}} = 0.9020336$$
$$\kappa = tc = -0.4316658$$

$$D_2 = \begin{bmatrix} 6.5413815 - 2.8187898t & 0 - \kappa\left(0.2332902 + \frac{1-c}{\kappa}0\right) & 0 \\ 0 - \kappa\left(0.2332902 + \frac{1-c}{\kappa}0\right) & 0.4586187 & 0.2332902 + \kappa\left(0 - \frac{1-c}{\kappa}0.2332902\right) \\ 0 & 0.2332902 & 2 + 2.8187898t \end{bmatrix}$$

$$= \begin{bmatrix} 7.8903057 & 0.1007034 & 0 \\ 0.1007034 & 0.4586187 & 0.2104356 \\ 0 & 0.2104356 & 0.6510758 \end{bmatrix}$$

$$V_2 = \begin{bmatrix} 0.76302 & -0.6463749 & 0 \\ 0.6463749 & 0.76302 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c & 0 & \kappa \\ 0 & 1 & 0 \\ -\kappa & 0 & c \end{bmatrix}$$

$$= \begin{bmatrix} 0.76302 - \kappa\left(0 + \frac{1-c}{\kappa}0.76302\right) & -0.6463749 & 0 + \kappa\left(0.76302 + \frac{1-c}{\kappa}0\right) \\ 0.6463749 - \kappa\left(0 + \frac{1-c}{\kappa}0.6463749\right) & 0.76302 & 0 + \kappa\left(0.6463749 + \frac{1-c}{\kappa}0\right) \\ 0 - \kappa\left(1 + \frac{1-c}{\kappa}0\right) & 0 & 1 + \kappa\left(0 + \frac{1-c}{\kappa}1\right) \end{bmatrix}$$

$$= \begin{bmatrix} 0.76302c & -0.6463749 & 0.76302\kappa \\ 0.6463749c & 0.76302 & 0.6463749\kappa \\ -\kappa & 0 & c \end{bmatrix} = \begin{bmatrix} 0.6882697 & -0.6463749 & -0.3293696 \\ 0.5830519 & 0.76302 & -0.2790179 \\ 0.4316658 & 0 & 0.9020336 \end{bmatrix}$$

If we continue to iterate, eventually, we will achieve

$$D = \begin{bmatrix} 7.8916 & 0 & 0 \\ 0 & 0.7859 & 0 \\ 0 & 0 & 0.3225 \end{bmatrix} \text{ and } V = \begin{bmatrix} 0.6793 & -0.6312 & 0.3744 \\ 0.5932 & 0.1720 & -0.7864 \\ 0.4320 & 0.7563 & 0.4913 \end{bmatrix}$$

Example: Find the SVD of $X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

1. Find the eigenvalues of $X'X$ and arrange them in descending order.

Done, above. 7.8916, 0.7858823, 0.3225104

2. Find the orthogonal eigenvectors of the matrix $X'X$ corresponding to the obtained eigenvalues and arrange them in the same order to form the column-vectors of the matrix V.

$$\begin{bmatrix} 0.6793 & -0.6312 & 0.3744 \\ 0.5932 & 0.1720 & -0.7864 \\ 0.4320 & 0.7563 & 0.4913 \end{bmatrix}$$

3. Form a diagonal matrix $\mathcal{D}$, placing the square roots of the eigenvalues of $X'X$ on the leading diagonal of it, in descending order.

$$\mathcal{D} = \begin{bmatrix} 2.8092 & 0 & 0 \\ 0 & 0.8865 & 0 \\ 0 & 0 & 0.5679 \\ 0 & 0 & 0 \end{bmatrix}$$

4. Find the first m column vectors of the matrix U:

$$u_i = \frac{1}{d_i} X v_i,\ i = 1, \ldots, m.$$

$$u_1 = \frac{1}{2.8092} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{pmatrix} 0.6793 \\ 0.5932 \\ 0.4320 \end{pmatrix} = \begin{pmatrix} 0.2418 \\ 0.4530 \\ 0.6068 \\ 0.6068 \end{pmatrix}$$

$$u_2 = \begin{pmatrix} -0.7120 \\ -0.5180 \\ 0.3352 \\ 0.3352 \end{pmatrix}, u_3 = \begin{pmatrix} 0.6592 \\ 0.7256 \\ 0.1395 \\ 0.1395 \end{pmatrix}$$

5. Complete U by the Gram-Schmidt orthogonalization procedure.

$$u_4 = \begin{pmatrix} 0 \\ 0 \\ -0.7071 \\ 0.7071 \end{pmatrix}$$

That's all well and good, but how do we use it to solve the least squares problem?! Check it out:

$$\begin{aligned} \widehat{\beta} &= (X'X)^{-1}X'Y \\ &= (V\mathcal{D}'U'U\mathcal{D}V')^{-1}V\mathcal{D}'U'Y \\ &= (V\mathcal{D}'\mathcal{D}V')^{-1}V\mathcal{D}'U'Y \\ &= V(\mathcal{D}'\mathcal{D})^{-1}V'V\mathcal{D}'U'Y \\ &= V[\,D^{-1} \quad 0]U'Y \end{aligned}$$

So, once you identify the SVD, you can calculate the least squares estimate directly and easily!

Why is this preferred by many folks? Because it can pretty easily deal with sets of equations that are not of full rank (i.e., *singular*), or are very close to being so. Once $\mathcal{D}$ is determined, you can just check those diagonal elements: If any are zero or close to zero, then your original matrix is singular or close to it. And, the SVD can provide you with a solution where other methods can't because of the full-rank requirement.

Now, there is at least one other way to compute an SVD. Indeed, this is one of the triumphs of modern computing!

4.5.1 QR Method for Finding Eigensystems

This assumes that A is square and symmetric, so that all of the eigenvalues of A are real. This method converges to an upper-triangular matrix where the diagonal entries are the eigenvalues of A.

In addition, note that if Q is a square orthonormal matrix, by the definition of the inverse, $Q^{-1}Q = QQ^{-1} = I$, and by the definition of orthonormality, $Q'Q = I$. Therefore, since Q must be of full rank, the inverse is unique, we have $QQ' = I$.

Suppose A is an $n \times n$ matrix. Let $A = Q_0R_0$ be a QR factorization of A, and create $A_1 = R_0Q_0$. Then, let $A_1 = Q_1R_1$ be a QR factorization of A_1, and create $A_2 = R_1Q_1$.

Facts:

1. $A = Q_0A_1Q_0'$, since $A_1Q_0' = R_0Q_0Q_0' = R_0$.
2. $A = (Q_0Q_1)A_2(Q_0Q_1)'$
3. Q_0Q_1 is an orthogonal matrix.
4. A, A_1 and A_2 all have the same eigenvalues.

The QR method for Generating Eigensystems:

1. Let $A = Q_0R_0$ be a QR factorization of A; create $A_1 = R_0Q_0$
2. Let $A_1 = Q_1R_1$ be a QR factorization of A_1; create $A_2 = R_1Q_1$.
3. Continue this process: Once A_i has been created, let $A_i = Q_iR_i$ be a QR factorization of A_i, and create $A_{i+1} = R_iQ_i$.
4. Stop when all entries below the main diagonal are small, or stop if it appears that convergence will not happen.

Note: This method *can* fail to converge, and it can be extremely slow! But we can add a modification that will speed up convergence and is usually more stable:

Facts:

Let A be an $n \times n$ matrix, I be the $n \times n$ identity matrix, and c be a scalar constant.

1. Let λ be an eigenvalue of A. Then $Ax = \lambda x$ iff $(A - cI)x = (\lambda - c)x$.
2. Let $A - cI = QR$ be a QR factorization of $A - cI$ and define $A_1 = RQ + cI$. Then $A = QA_1Q'$.

The Modified QR Method for Finding Eigensystems:

1. Choose c to be the last diagonal entry in A. Let $A - cI = Q_0R_0$ be a QR factorization of $A - cI$; create $A_1 = R_0Q_0 + cI$.
2. Choose c_1 to be the last diagonal entry in A_1. Let $A_1 - c_1I = Q_1R_1$ be a QR factorization of $A_1 - c_1I$; create $A_2 = R_1Q_1 + c_1I$.
3. Continue this process: Once A_i has been created, choose c_i to be the last diagonal entry in A_i. Let $A_i - c_iI = Q_iR_i$ be a QR factorization of $A_i - c_iI$, and create $A_{i+1} = R_iQ_i + c_iI$.
4. When all entries on the *final row* of A_i, except the last entry, are close to zero, deflate A_i by removing the final row and column. The final entry in the row/column just removed is an eigenvalue.
5. Repeat 1-4 on the deflated matrix until all the eigenvalues have been found or until it appears that convergence to a triangular limit matrix will not happen.
6. To complete the eigensystem, use the collected eigenvalues λ to solve for the corresponding eigenvector x according to the definition: $(A - \lambda I)x = 0$.