Quantitative Foundations Project 1

Linear Feature Engineering

Instructor: Linwei Wang

Contents

1 Introduction to Solving Linear Systems

Example 1. Consider solving the linear system

$$2x = 4.$$

Obviously, this has the trivial solution $x = 2$.

Example 2. OK, consider instead solving the linear system

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

How to solve this? Well, the idea is to add two times the first equation to the second. Then we obtain

$$\begin{aligned} 2x - 2x + 4y + y &= 2 \cdot 3 + 4. \\ 5y &= 10 \\ y &= 2 \\ x &= -1. \end{aligned}$$

Example 3. Alrighty. Let's consider an even more complex system

$$\begin{aligned} x + 2y + z &= 3 \\ -2x + y - z &= 4 \\ x + y + 2z &= 7. \end{aligned}$$

How can we solve this? Let's try looking at the problem as a whole. First, add two copies of the first equation to the second, and subtract one copy of the first equation from the third. This gives us

$$\begin{aligned} x + 2y + z &= 3 \\ 0x + 5y + z &= 10 \\ 0x - 1y + z &= 4. \end{aligned}$$

Now, subtract $\frac{1}{5}$th of the second equation from the third.

$$\begin{aligned} x + 2y + z &= 3 \\ 0x + 5y + z &= 10 \\ 0x + 0y + \frac{6}{5}z &= 6. \end{aligned}$$

Now, we can read off the solution for z.

$$\begin{aligned} x + 2y + z &= 3 \\ 0x + 5y + z &= 10 \\ 0x + 0y + z &= 5. \end{aligned}$$

And we can start to attack the previous equations by "back-substitution". First solve for y by substituting in for z.

$$\begin{aligned} x + 2y + z &= 3 \\ 0x + 5y + 5 &= 10 \\ 0x + 0y + z &= 5. \end{aligned}$$

$$\begin{aligned} x + 2y + z &= 3 \\ 0x + 1y + 0 &= 1 \\ 0x + 0y + z &= 5. \end{aligned}$$

Now, solve for x by substituting in for y and z

$$\begin{aligned} x + 2 + 5 &= 3 \\ 0x + 1y + 0 &= 1 \\ 0x + 0y + z &= 5. \end{aligned}$$

We finally have the solution

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

1.1 Formalization

Now, suppose we wanted to solve a 4×4 or 10×10 system. Well, doing what we did above will become a little awkward. Thus, some generic notation has been introduced that makes things easier to work with. We would write the above equation in matrix forms as

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & 1 & -1 \\ 1 & 1 & 2 \end{bmatrix}$$

is a 3×3 **matrix**,

$$\mathbf{b} = \begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix}$$

is a length 3 **vector**, and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

is the solution we are looking for. (In this case, $x_1 = -4$, $x_2 = 1$, and $z = 5$.)

Writing

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

simply means the set of equations

$$\begin{aligned} 1x_1 + 2x_2 + 1x_3 &= 3 \\ -2x_1 + 1x_2 - 1x_3 &= 4 \\ 1x_1 + 1x_2 + 2x_3 &= 7. \end{aligned}$$

1.2 Gaussian Elimination

In this section, we will implement an algorithm called Gaussian Elimination to solve linear systems. This algorithm is just a formalization of the technique we used in the first section to solve linear systems.

To solve a linear system, we will use two steps:

1. Transform the linear system into an upper-triangular linear system that has the same solution.

2. Solve the upper-triangular linear system.

We will talk about step 2 first, step 1 second.

1.2.1 Solving Upper-Triangular Systems

If we want to solve the system $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$, we would call the system upper-triangular if all the entries of $\mathbf{A}$ below the diagonal are zero. So, for example, we would say that

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{bmatrix}$$

is upper-triangular, but

$$\mathbf{B} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 0 & 2 \\ 1 & 0 & 3 \end{bmatrix}$$

is not. In this section, we will further assume (for convenience) that the diagonal of $\mathbf{A}$ is entirely nonzero.

Now, suppose we have an upper-triangular system for A above, and we want to solve for $\mathbf{x}$ when

$$\mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix}.$$

Recall, from the definition of matrix multiplication that

$$b_i = \sum_{j=1}^{N} A_{ij} x_j.$$

However, if we have an upper triangular matrix, we know that $A_{ij} = 0$ whenever $i > j$. Thus, we can instead write

$$b_i = \sum_{j=i}^{N} A_{ij} x_j.$$

Thus, we propose the following algorithm, which is known as **back-substitution.**

Algorithm (Back-substitution v. 1)

1. Input $\mathbf{A}, \mathbf{b}$.
2. For $i = N, N-1, ..., 1$
   (a) Solve the equation $b_i = \sum_{j=i}^{N} A_{ij} x_j$. for x_i, using the previously identified values of x_{i+1}, x_{i+2}, ..., x_N.
3. Return $\mathbf{x}$.

Now, if we know x_{i+1}, x_{i+2}, ..., x_N, how do we find the value of x_i? A simple manipulation shows that

$$x_i = \frac{1}{A_{ii}}\big(b_i - \sum_{j=i+1}^{N} A_{ij} x_j\big)$$

when $A_{ii} \neq 0$.

This gives the following algorithm:

Algorithm (Back-substitution v. 2)

1. Input $\mathbf{A}, \mathbf{b}$.
2. For $i = N, N-1, ..., 1$
   (a) $x_i \leftarrow \frac{1}{A_{ii}}\big(b_i - \sum_{j=i+1}^{N} A_{ij} x_j\big)$
3. Return $\mathbf{x}$.

If we want to be really explicit, we might write this like so:

Algorithm (Back-substitution v. 3)

1. Input $\mathbf{A}, \mathbf{b}$.
2. For $i = N, N-1, ..., 1$
   (a) $c \leftarrow b_i$
   (b) For $j = i+1, i+2, ..., N$
      i. $c \leftarrow c - A_{ij} x_j$
   (c) $x_i \leftarrow c / A_{ii}$
3. Return $\mathbf{x}$.

This procedure has a complexity of $\mathcal{O}(N^2)$.

Exercise 4. Write an algorithm that implements back-substitution. Test that, on an input of

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{bmatrix}$$

and $\mathbf{b} = (2, 3, 6)$ it returns the correct value of $\mathbf{x} = (0, -1, 2)$.

Algorithm (Back-substitution v. 4) - vectorization implementation in MATLAB

1. Input $\mathbf{A}, \mathbf{b}$.
2. For $i = N, N-1, ..., 1$
   (a) $x_i \leftarrow b_i - \mathbf{A}_{i,i+1:N}\mathbf{x}_{i+1:N}$
   (b) $x_i \leftarrow x_i / A_{ii}$
3. Return $\mathbf{x}$.

1.2.2 Reducing to Upper-Triangular Systems

Now, suppose that we have some system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{A}$ is not assumed upper-triangular. What we want to do here is find a system

$$\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$$

where $\tilde{\mathbf{A}}$ is upper-triangular, *but has the same solution* $\mathbf{x}$.

How does this work? Well, consider the set of equations

$$\begin{aligned} ax + by &= d \\ ex + fy &= g. \end{aligned}$$

We claim that, for any c, it is true that

$$(e + ca)x + (f + cb)y = g + cd.$$

In particular, if we choose

$$c = -\frac{e}{a},$$

then we have the system

$$(f - \frac{eb}{a})y = g - \frac{e}{a}d,$$

where everything is constant except for y.

Example 5. Consider the linear system defined by

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix}.$$

We want to make this into an upper-triangular system.

First of all, we will multiply the first row by $-\frac{1}{2}$ and add it to the second row. The result is

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1-\frac{1}{2}2 & 1-\frac{1}{2}1 & 2-\frac{1}{2}1 \\ 1 & 1 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 3-\frac{1}{2}2 \\ 6 \end{bmatrix},$$

or, simplifying,

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & \frac{1}{2} & \frac{3}{2} \\ 1 & 1 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \\ 6 \end{bmatrix}.$$

Now, we will multiply the first row by $-\frac{1}{2}$ and add it to the third row. This gives us

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & \frac{1}{2} & \frac{3}{2} \\ 0 & \frac{1}{2} & \frac{5}{2} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \\ 5 \end{bmatrix}.$$

Finally, we multiply the second row by -1, and add it to the third row. This gives us

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & \frac{1}{2} & \frac{3}{2} \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}.$$

And we're done!

The general algorithm we use is the following

Algorithm (Reduce to Upper-Triangular v. 1)

1. Input **A**, **b**.
2. For $i = 1, 2, ..., N$
   (a) For $j = i + 1, i + 2, ..., N$

i. $c \leftarrow -A_{ji}/A_{ii}$
ii. For $k = 1, 2, ..., N$
    A. $A_{jk} \leftarrow A_{jk} + cA_{ik}$
iii. $b_j \leftarrow b_j + cb_i$

3. Return **A**, **b**

Or, if we follow the rule of vectorization implementation in MATLAB, we can write:

Algorithm (Reduce to Upper-Triangular v. 2)

1. Input **A**, **b**.
2. For $i = 1, 2, ..., N$
   (a) For $j = i + 1, i + 2, ..., N$
      i. $c \leftarrow -A_{ji}/A_{ii}$
      ii. $\mathbf{A}_{j\cdot} \leftarrow \mathbf{A}_{j\cdot} + c\mathbf{A}_{i\cdot}$
      iii. $b_j \leftarrow b_j + cb_i$
3. Return **A**, **b**

Exercise 6. Write a program in your favorite programming language that reduces a system to upper-triangular form. Test that, on an input of

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

and $\mathbf{b} = (2, 3, 6)$ it returns the correct values of

$$\tilde{\mathbf{A}} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & \frac{1}{2} & 1\frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\tilde{\mathbf{b}} = (2, 2, 3).$$

Now, what is the complexity of this algorithm? Well, from looking at the second version, we can see that we have three nested loops, each of which loops over N elements. This means the total complexity is $\mathcal{O}(N^3)$

1.2.3 Putting the Pieces Together

So, we have an algorithm for solving a linear system: First, reduce the linear system to an upper-triangular one with the same solution, and then solve that system by back-substitution.

Algorithm (Gaussian-Elimination)

1. Input $\mathbf{A}$, $\mathbf{b}$.
2. Find $\tilde{\mathbf{A}}$ and $\mathbf{b}$ by reducing to upper-triangular.
3. Solve the linear system $\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$ for $\mathbf{x}$ using back-substitution.
4. Return $\mathbf{x}$.

Exercise 7. Using the previous two routines, implement a program to solve a linear system. Test that on an input of

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

and $\mathbf{b} = (2, 3, 6)$, it returns the correct solution of

$$\mathbf{x} = (2, -5, 3).$$

1.3 More Information

An excellent review of basic linear algebra is Linear Algebra Review and Reference.