Linear equations, matrices and vectors

Math and modelling for high school, Lecture 1

by Sergei Savin

Fall 2022

What is this course about?

In this course we try to do the following:

- Teach you basics of university-level linear algebra, basic introduction to calculus and differential equations and simulation;
- Focus on visual aspect of math: computer graphics, graphs, animation. We do it to use your strength - programming, to help you understand inherently abstract concepts.
- We try to keep the topics coherent and compact, avoiding topics that require deeper algebraic or calculus theory, which forces us to skip many interesting things - which you will surely learn in the university.

The following is a 3 by 3 matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{1}$$

Examples of 3 by 3 matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 6 & -5 \\ 0 & -3 & 3 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 10 & 2 & 0 \\ -4 & 5 & 0 \\ -4 & 5 & 2 \end{bmatrix}, \quad \text{etc} \tag{2}$$

The following is a 3-dimensional vector:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{3}$$

Examples 3-dimensional vectors:

$$\mathbf{v} = \begin{bmatrix} 5 \\ 5 \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 4 \\ -3 \\ -1 \end{bmatrix}, \quad \text{etc} \tag{4}$$

Recap: Norms

Given a vector: $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$, we can find its *norm*:

$$||\mathbf{v}|| = \sqrt{v_1^2 + v_2^2 + v_3^2} \tag{5}$$

You can think of it as *length* of a vector.

We can find matrix-vector product:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} a_{11}v_1 + a_{12}v_2 + a_{13}v_3 \\ a_{21}v_1 + a_{22}v_2 + a_{23}v_3 \\ a_{31}v_1 + a_{32}v_2 + a_{33}v_3 \end{bmatrix} \tag{6}$$

An example of a matrix-vector product:

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 6 & -5 \\ 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -10 \\ 6 \end{bmatrix} \tag{7}$$

Matrices don't need to be square. For the matrix-vector product to work, the matrix needs to have as many columns as there are elements in the vector.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} a_{11}v_1 + a_{12}v_2 + a_{13}v_3 \\ a_{21}v_1 + a_{22}v_2 + a_{23}v_3 \end{bmatrix} \tag{8}$$

An example of a matrix-vector product:

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

Read more

There are many resources on the topic:

- mathinsight.org/matrix_introduction
- mathinsight.org/matrix_vector_multiplication
- mathworld.wolfram.com/L2-Norm.html
- etc.

Matrices and vectors in Python

In Python we can use Numpy package to create matrices and vectors:

- matrix: `A = np.array([[1, -4, 5], [3, 1, -2], [2, -6, -9]])`
- vector: `v = np.array([1, 1, -2])`

Matrix-vector product is done as `y = A@v`.

Consider the system of equations:

$$\begin{cases} x_1 + 2x_2 = 10 \\ 2x_1 + 2x_2 = 0 \end{cases} \tag{10}$$

This is the same as:

$$\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \tag{11}$$

General form for this kind of problem is:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{12}$$

In this course we will solve it with
`x = np.linalg.solve(A, b)` functionality of numpy:

```
import numpy as np
A = np.array([[1, 2], [2, 2]])
b = np.array([10, 0])
x = np.linalg.solve(A, b)
print("solution:")
print(x)
print("residual:")
print(A@x - b)
```

Dimensions do not matter, we can deal with equations with 2, 3 and more variables the same way:

$$\begin{cases} x_1 - 4x_2 + 5x_3 = 1 \\ 3x_1 + x_2 - 2x_3 = 1 \\ 2x_1 - 6x_2 - 9x_3 = -2 \end{cases} \tag{13}$$

This is the same as

$$\begin{bmatrix} 1 & -4 & 5 \\ 3 & 1 & -2 \\ 2 & -6 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} \tag{14}$$

Again, it is $\mathbf{Ax} = \mathbf{b}$ and we solve it with `x = np.linalg.solve(A, b)`.

Dimensions: notation

When we want to note the dimension of a vector, we use the following notation:

$$\mathbf{b} \in \mathbb{R}^n \tag{15}$$

where $\mathbb{R}$ means space of real numbers and n is the dimensions.

For matrices it is similar:

$$\mathbf{A} \in \mathbb{R}^{n,n} \tag{16}$$

Why use this strange $\mathbb{R}$? Because later you might see something like $\mathbf{b} \in \mathbb{C}^n$, meaning it is a complex vector, or $\mathbf{b} \in \mathbb{N}^n$ meaning it is a vector of integers.

Degenerate eqs.: too many variables

Let us solve a degenerate system of equations:

$$x_1 + 2x_2 = 10 \tag{17}$$

This is the same as

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 10 \tag{18}$$

General form for this kind of problem is still:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{19}$$

We can solve it with `x, _, _, _ = np.linalg.lstsq(A, b, rcond=None)` fucntionality of numpy.

But what does it even mean to solve $x_1 + 2x_2 = 10$? Let us try to solve it by hand.

We know that $x_1 = 10 - 2x_2$. And we can pick any x_2. For example, $x_2 = 1$, then $x_1 = 10 - 2 = 8$. Or, $x_2 = 0$, then $x_1 = 10$. Or, $x_2 = 4$, then $x_1 = 10 - 8 = 2$. So, all the following $\mathbf{x}$ are solutions:

$$\mathbf{x} = \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}. \tag{20}$$

The numpy function `lstsq` will pick *smallest norm solution* out of all possible ones. In this case, it is, $\mathbf{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$.

Too many equations

Let us solve a degenerate system of equations:

$$\begin{cases} x_1 + 2x_2 = 2 \\ 3x_1 - 5x_2 = 0 \\ 2x_1 - 4x_2 = 5 \end{cases} \tag{21}$$

This is the same as

$$\begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 5 \end{bmatrix} \tag{22}$$

General form for this kind of problem is still:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{23}$$

Again, we just use `x, _, _, _ = np.linalg.lstsq(A, b, rcond=None)`.

```
A = np.array([[1, 2], [3, -5], [2, -4]])
b = np.array([[2], [0], [5]])
x, _, _, _ = np.linalg.lstsq(A, b, rcond=None)
e = A@x - b;
print("solution: " + str(x.T));
print("residual: " + str(e.T));
```

The output for this example is:

```
solution: [[1.07936508 0.14814815]]
residual: [[-0.62433862  2.4973545  -3.43386243]]
```

Notice the residual is **not zero**. Note that the function `lstsq` will pick solution with *smallest-norm residual*.

The following system is square (same number of variable and equations) but you still cannot solve it normally:

$$\begin{cases} x_1 - 2x_2 = 0 \\ 3x_1 - 6x_2 = 2 \end{cases} \tag{24}$$

This is the same as:

$$\begin{bmatrix} 1 & -2 \\ 3 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \tag{25}$$

And it does not have a zero-residual solution.

How do you tell if a system can be solved?

The test is - to find if the matrix *determinant* is not zero. For 2 by 2 matrices, the determinant is found as follows:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{26}$$

$$\det(\mathbf{A}) = ad - bc \tag{27}$$

For anything other than 2 by 2 we find determinant as `d = np.linalg.det(A)` fucntionality of numpy.

If a matrix has 0 determinant we call it *degenerate*. If the matrix has non-zero determinant, we call it *full-rank*.

Given a few vectors $\mathbf{v}_1$, $\mathbf{v}_2$, ..., $\mathbf{v}_n \in \mathbb{R}^n$, their linear combination is defined as:

$$\mathbf{w} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + ... + \alpha_n \mathbf{v}_n \tag{28}$$

where $\alpha_i \in \mathbb{R}$ are linear coefficients.

Example: Given $\mathbf{v}_1$ and $\mathbf{v}_2$:

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{29}$$

Then examples of their linear combinations are:

$$\mathbf{w} = 2\mathbf{v}_1 - \mathbf{v}_2 = 2 \begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \tag{30}$$

$$\mathbf{w} = -3\mathbf{v}_1 + 2\mathbf{v}_2 = -3 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ -7 \end{bmatrix} \tag{31}$$

$$\mathbf{w} = 5\mathbf{v}_2 = 5 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \tag{32}$$

If a set of vectors contain none that is a linear combination of the others - we call them *linearly independant*.

If one of your matrix's columns is a linear combination of the other - it will be degenerate. Lets look for an example.
Consider the matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{33}$$

Its columns are $\mathbf{a}_1$ and $\mathbf{a}_2$:

$$\mathbf{a}_1 = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}, \quad \mathbf{a}_2 = \begin{bmatrix} a_{21} \\ a_{22} \end{bmatrix} \tag{34}$$

If, for instance, $\mathbf{a}_2 = 2\mathbf{a}_1$, the matrix $\mathbf{A}$ will be degenerate.

An example of a matrix with linearly dependent columns:

$$\mathbf{a}_1 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, \quad \mathbf{a}_2 = -\mathbf{a}_1 = \begin{bmatrix} -2 \\ 3 \end{bmatrix} \tag{35}$$

Then matrix $\mathbf{A} = \begin{bmatrix} 2 & -2 \\ -3 & 3 \end{bmatrix}$ is degenerate, and $\det(\mathbf{A}) = 0$. Meaning, you can't solve a system like this one:

$$\begin{cases} 2x_1 - 2x_2 = 1 \\ -3x_1 + 3x_2 = 0 \end{cases} \tag{36}$$

Really though? Let us check. $-3x_1 + 3x_2 = 0$, therefore $x_1 = x_2$, and since $2x_1 - 2x_2 = 1$ we get $2x_1 - 2x_1 = 1$ and so $0 = 1$, which is clearly incorrect.

Read more

- mathinsight.org/matrices_determinants
- en.wikipedia.org/wiki/Determinant

Lecture slides are available via Moodle.

You can help improve these slides at:
github.com/SergeiSa/Extra-math-for-high-school

Check Moodle for additional links, videos, textbook suggestions.