

# Linear Algebra, Four Fundamental Subspaces, part 2

Computational Intelligence, Lecture 3

by Sergei Savin

Fall 2020

- Projection
  - How to find a projector
  - Use case
- Orthogonal compliment
- Column space
- Left null space
- Row space and left null space bases
- Example
- Homework

# Projection

Previously we used the term *projection*. What does it mean?

## Definition 1

For linear space  $\mathcal{L} \subset \mathbb{R}^n$ , an orthogonal projector  $\mathbf{P}$  onto it has properties:

- $\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{P}\mathbf{x} \in \mathcal{L}$
- $\forall \mathbf{x} \in \mathcal{L}, \mathbf{P}\mathbf{x} = \mathbf{x}$
- $\forall \mathbf{y} \in \mathcal{L}, \mathbf{y}^\top (\mathbf{I} - \mathbf{P})\mathbf{x} = 0$

It follows that  $\mathbf{P}\mathbf{P} = \mathbf{P}$ . Also notice that a projection always maps the space onto itself:  $\mathbf{P} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ .

In other words, an orthogonal projector takes the part of a vector that lies in the linear space, and cuts the rest off. We can refer to it as *projection*.

# Projection

## How to find a projector

As long as we know an orthonormal basis  $\mathbf{B}$  in the linear space  $\mathcal{L}$ , we can find a projector  $\mathbf{P}$  on that space as follows:

$$\mathbf{P} = \mathbf{B}\mathbf{B}^\top \quad (1)$$

You can think of it as follows:  $\mathbf{B}\mathbf{B}^\top \mathbf{x}$  first finds a decomposition of  $\mathbf{x}$  on the orthonormal set of vectors, where all parts of  $\mathbf{x}$  not in the span of  $\mathbf{B}$  will be mapped to zero; next, it maps this decomposition back into the original coordinates, but the part not-in-the-span-of- $\mathbf{B}$  is lost.

In general, if linear space  $\mathcal{L}$  is given as a span of the columns of some matrix  $\mathbf{L}$ , the projector  $\mathbf{P}$  on that space can be generated as follows:

$$\mathbf{P} = \mathbf{L}\mathbf{L}^+ = \mathbf{L}(\mathbf{L}^\top \mathbf{L})^{-1} \mathbf{L}^\top \quad (2)$$

# Projection

## Use case

Remember the problem: find all solutions to the system of equations  $\mathbf{Ax} = \mathbf{y}$ . Assume we found a single solution  $\mathbf{x}^s$ . If we know an orthonormal basis  $\mathbf{N}$  in the null space of  $\mathbf{A}$  (which we get by calling `null()` in MATLAB, for example), then we can find a projector  $\mathbf{P}$  onto that space and find which part of  $\mathbf{x}^s$  lies outside it:

$$\mathbf{P} = \mathbf{N}\mathbf{N}^\top \quad (3)$$

$$\mathbf{x}^p = (\mathbf{I} - \mathbf{P})\mathbf{x}^s \quad (4)$$

Thus, all solutions are found as  $(\mathbf{I} - \mathbf{N}\mathbf{N}^\top)\mathbf{x}^s + \mathbf{N}\mathbf{z}$ ,  $\forall \mathbf{z}$

# Orthogonal compliment

Remember how we defined row space: *Row space* of  $\mathbf{A}$  is the set of all inputs to  $\mathbf{A}$  that have a zero projection onto its null space.

Now we can observe that if  $\mathbf{P}$  is a projector onto the null space of  $\mathbf{A}$ , then the row space of  $\mathbf{A}$  is the set of all vectors  $\mathbf{x}$ , such that  $\mathbf{P}\mathbf{x} = \mathbf{0}$ .

We can observe that  $(\mathbf{I} - \mathbf{P})$  is a projector onto the row space. In a sense, row space contains everything left by the null space, and together then span all inputs on the matrix  $\mathbf{A}$ . They compliment each other, and they are orthogonal to each other. In other words, row space is an *orthogonal compliment* of null space, and vice versa.

Consider the problem. Current state of the system is given as  $\mathbf{x}_i \in \mathbb{R}^n$ , and we do not know it. At the next point in time, the state of the system is given as  $\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i$ . What are all possible next states that we can expect?

To solve it, we need to find all possible outputs of the linear operator  $\mathbf{A}$ . This is called *column space* of  $\mathbf{A}$ .

In MATLAB it can be constructed by calling function `orth()`. Both `orth()` and `null()` (as well as `rank()` and `pinv()`) simply call `svd()` and perform minimal computations on the resulting decomposition. You can check it by typing `open orth` in MATLAB command window.

# Left null space

Consider a dual problem. Current state of the system is given as  $\mathbf{x}_i \in \mathbb{R}^n$ , and we do not know it. At the next point in time, the state of the system is given as  $\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i$ . Which values the next states will not assume?

Notice, that this is the same as asking what is the orthogonal complement of the column space of  $\mathbf{A}$ . It is called *left null space*.

Let  $\mathbf{P}$  be the projector onto the column space of  $\mathbf{A}$ :  $\mathbf{P} = \mathbf{A}\mathbf{A}^+$ . Then  $\mathbf{Q} = (\mathbf{I} - \mathbf{P}) = (\mathbf{I} - \mathbf{A}\mathbf{A}^+)$  is a projector onto the left null space of  $\mathbf{A}$ .



# Row space and left null space bases

While MATLAB does not provide tools to directly find orthonormal bases in the row space and left null space of a matrix, it can be done by tweaking the code for `orth()` and `null()`.

Your HW: write down formulas for `orth()`, `null()` and their orthogonal compliments, and implement those in MATLAB or your language of choice that provides an SVD decomposition implementation.

We will define operators `row()` and `lnull()` to refer to bases in those two spaces.

# Example

Now that we have such powerful tools, we can solve difficult problems easily. Consider this one. Linear time-invariant (LTI) dynamical system is described as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{u}_1 + \mathbf{B}_2\mathbf{u}_2 \quad (5)$$

Find such control inputs  $\mathbf{u}_1^*$ ,  $\mathbf{u}_2^*$  that state  $\mathbf{x}^*$  becomes a fixed point. Additionally, assume that  $\mathbf{u}_1^*$  is free to use, while  $\mathbf{u}_2^*$  should be used as sparingly as possible.

This can be formulated in the language of optimization as follows:

$$\begin{aligned} & \underset{\mathbf{u}_1, \mathbf{u}_2}{\text{minimize}} && ||\mathbf{u}_2||, \\ & \text{subject to} && \mathbf{A}\mathbf{x}^* + \mathbf{B}_1\mathbf{u}_1 + \mathbf{B}_2\mathbf{u}_2 = \mathbf{0} \end{aligned} \quad (6)$$

# Example

## part 2

In order to check that the problem has at least one solution we need to make sure that there are such inputs  $\mathbf{u}_1^*$ ,  $\mathbf{u}_2^*$  that state  $\mathbf{x}^*$  becomes a fixed point. In other words, vector  $\mathbf{A}\mathbf{x}^*$  should lie in the span of the columns of the matrix  $[\mathbf{B}_1 \ \mathbf{B}_2]$ . Which is the same as saying that its projection on the complement on this column space (left null space of  $[\mathbf{B}_1 \ \mathbf{B}_2]$ ) is zero:

$$(\mathbf{I} - [\mathbf{B}_1 \ \mathbf{B}_2][\mathbf{B}_1 \ \mathbf{B}_2]^+) \mathbf{A}\mathbf{x}^* = \mathbf{0} \quad (7)$$

All solutions to the problem  $\mathbf{A}\mathbf{x}^* + \mathbf{B}_1\mathbf{u}_1 + \mathbf{B}_2\mathbf{u}_2 = \mathbf{0}$  can be found as:

$$\mathbf{u} = [\mathbf{B}_1 \ \mathbf{B}_2]^+ \mathbf{A}\mathbf{x}^* + \text{null}([\mathbf{B}_1 \ \mathbf{B}_2])\mathbf{z}, \quad \forall \mathbf{z} \quad (8)$$

# Example

## part 3

Now we need to pick one solution out of all of them, based on the criteria that it minimizes  $\mathbf{u}_2$ . We can solve it as an optimization (by thinking about the derivatives of the objective/cost function), but it can also be solved as a projection.

Let us define projector  $\mathbf{P} = \mathbf{B}_1\mathbf{B}_1^+$ . We can prove that  $\mathbf{PB}_2\mathbf{u}_2^* = \mathbf{0}$ . Assume  $\mathbf{PB}_2\mathbf{u}_2^* = \mathbf{a}$ , then  $\mathbf{u}_2^* = \mathbf{u}_2^0 + \mathbf{u}_2^a$ , where  $\mathbf{u}_2^0$  is in the null space of  $\mathbf{PB}_2$  and  $\mathbf{u}_2^a$  is in the row space of  $\mathbf{PB}_2$ , or equivalently  $\mathbf{PB}_2\mathbf{u}_2^0 = \mathbf{0}$ ,  $(\mathbf{I} - \mathbf{P})\mathbf{B}_2\mathbf{u}_2^0 = \mathbf{b}$ , and  $\mathbf{PB}_2\mathbf{u}_2^a = \mathbf{a}$ ,  $(\mathbf{I} - \mathbf{P})\mathbf{B}_2\mathbf{u}_2^a = \mathbf{0}$ . This gives us solution:

$$\mathbf{Ax}^* + \mathbf{B}_1\mathbf{u}_1^* + \mathbf{B}_2\mathbf{u}_2^0 + \mathbf{B}_2\mathbf{u}_2^a = \mathbf{0} \quad (9)$$

# Example

## part 4

But since  $\mathbf{a} \in \text{span}(\mathbf{B}_1)$  we can also provide an alternative solution:

$$\mathbf{A}\mathbf{x}^* + \mathbf{B}_1\mathbf{u}^* + \mathbf{B}_2\mathbf{u}_2^0 = \mathbf{0} \quad (10)$$

where  $\mathbf{u}^* = \mathbf{u}_1^* + \mathbf{B}_1^+ \mathbf{a}$ .

Since  $\mathbf{u}_2^0$  and  $\mathbf{u}_2^a$  belong to orthogonal subspaces,  $\|\mathbf{u}_2^0 + \mathbf{u}_2^a\| \geq \|\mathbf{u}_2^0\|$ . Therefore the alternative solution provides smaller norm  $\mathbf{u}_2$ , hence  $\mathbf{a} = \mathbf{0}$  and  $\mathbf{u}_2^a = \mathbf{0}$  and  $\mathbf{P}\mathbf{B}_2\mathbf{u}_2^* = \mathbf{0}$ .

Since  $\mathbf{P}\mathbf{B}_2\mathbf{u}_2^* = \mathbf{0}$ ,  $(\mathbf{I} - \mathbf{P})\mathbf{B}_2\mathbf{u}_2^* = \mathbf{u}_2^*$ , and while  $(\mathbf{I} - \mathbf{P})\mathbf{B}_1 = \mathbf{0}$ .

# Example

part 5

Multiplying our constraint by  $(\mathbf{I} - \mathbf{P})$  we attain:

$$(\mathbf{I} - \mathbf{P})\mathbf{A}\mathbf{x}^* + (\mathbf{I} - \mathbf{P})\mathbf{B}_2\mathbf{u}_2^* = \mathbf{0} \quad (11)$$

Which has an exact solution:

$$\mathbf{u}_2^* = -((\mathbf{I} - \mathbf{P})\mathbf{B}_2)^+(\mathbf{I} - \mathbf{P})\mathbf{A}\mathbf{x}^* \quad (12)$$

Which is the solution to the original problem;  $\mathbf{u}_1^*$  can be obtained as follows:

$$\mathbf{u}_1^* = -\mathbf{B}_1^+(\mathbf{A}\mathbf{x}^* + \mathbf{B}_2\mathbf{u}_2^*) \quad (13)$$

# Homework

- Prove the following for an orthogonal projector  $\mathbf{P}$ :  
 $(\mathbf{I} - \mathbf{P})(\mathbf{I} - \mathbf{P}) = (\mathbf{I} - \mathbf{P})$
- Write down formulas for `orth()`, `null()` and their orthogonal compliments, and implement those in MATLAB or your language of choice that provides an SVD decomposition implementation.
- Prove that (12) will always have a solution that turns  $((\mathbf{I} - \mathbf{P})\mathbf{B}_2)\mathbf{u}_2^* = -(\mathbf{I} - \mathbf{P})\mathbf{A}\mathbf{x}^*$  into equality.
- Prove that (13) will always have a solution that turns  $\mathbf{B}_1\mathbf{u}_1^* = -\mathbf{A}\mathbf{x}^* - \mathbf{B}_2\mathbf{u}_2^*$  into equality.

Lecture slides are available via Moodle.

You can help improve these slides at:

[github.com/SergeiSa/Computational-Intelligence-Slides-Fall-2020](https://github.com/SergeiSa/Computational-Intelligence-Slides-Fall-2020)

Check Moodle for additional links, videos, textbook suggestions.