Université de Genève

Data Science

TP 1: Linear Algebra

Author: Joao Filipe Costa da Quinta

E-mail: Joao.Costa@etu.unige.ch

October 4, 2021

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

1 - Matrix

.1

$$\begin{cases} a+b+c=1 \\ 4a+2b+c=9 \\ 9a+3b+c=27 \end{cases} \Leftrightarrow \begin{cases} a=5 \\ b=-7 \\ c=3 \end{cases}$$

(1) We can start by writing the first equation: $\longrightarrow c = 1 - a - b$

(2) insert step (1) in equation 2 and write it: $\longrightarrow 4a + 2b + 1 - a - b = 9 \Leftrightarrow 3a + b = 8 \Leftrightarrow b = 8 - 3a$

(3) Rewrite $c = 1 - a - b$: $\longrightarrow c = 1 - a - (8 - 3a) \Leftrightarrow c = -7 + 2a$

(4) Insert equations from (3) and (2) into equation 3:$\longrightarrow 9a + 3 * (8 - 3a) + (-7 + 2a) = 27$

(5) Solve step (4): $\longrightarrow 9a + 24 - 9a - 7 + 2a = 27 \Leftrightarrow a = 5$

(6) Insert step (5) into step (2):$\longrightarrow b = 8 - 3 * 5 \Leftrightarrow b = -7$

(7) Insert steps (5) and (6) into step (1): $\longrightarrow c = 1 - 5 + 7 \Leftrightarrow c = 3$

.2

$$\begin{cases} x+ay=2 \\ bx+2y=3 \end{cases} \Leftrightarrow \begin{cases} x=2-ay \\ bx+2y=3 \end{cases}$$

We can now insert the 1^{st} equation into the 2^{nd} equation as follows:
$b(2-ay)+2y=3 \Leftrightarrow 2b-aby+2y=3 \Leftrightarrow 2b-y(ab-2)=3 \Leftrightarrow y=(-1)*\left(\frac{3-2b}{(ab-2)}\right)$
With given a and b values we can easily compute y, then we reinsert y into the first equation to get x.

2 - The importance of the mathematical concept behind a code

.1

$def\ project\_on\_first(u,v)$ receives two column vectors as an argument, and it projects v onto u, the projected vector is usually called v', in the image bellow it is represented by $\vec{w_1}$. Visually, it means that v' and u are collinear. This also means that: $\exists\ \alpha \in \mathbb{R}$ tq. $v' * \alpha = u$.

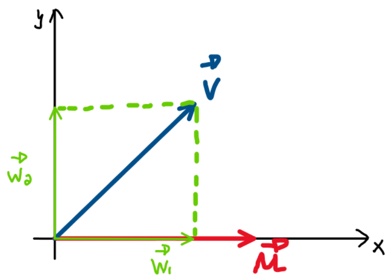

Figure 1: Projection of $\vec{v}$ onto $\vec{u} = \vec{w_1}$

.2

zip() function takes as argument two python lists of same size. It then merges one value from the first list, with another value from the second list (same index), creating a list of tuples.
Let's see an example:

x = zip([1,2], [3,4]) -> x = [(1,3),(2,4)]

This means that the three last lines of code perform a simple dot operation between the two vectors given as argument to zip().

It can be rewritten as: r = np.dot(u,v)

.3

Step 1: find the vector $\vec{w_2}$ orthogonal to $\vec{u}$
If we look at Figure 1, we can see that $\vec{w_1}$ is collinear to $\vec{u}$, and that $\vec{w_2}$ is orthogonal to $\vec{u}$. Moreover, $\vec{v}$ = $\vec{w_1}$ + $\vec{w_2}$, which means we can easily compute $\vec{w_2}$ if we have already computed $\vec{w_1}$.

$$\vec{w_2} = \vec{v} - \vec{w_1}$$

Step 2: Make it so the orthogonal vector $\vec{w_2}$ has the same norm as vector $\vec{u}$
We must first compute $||\vec{u}||$ as well as $||\vec{w_2}||$.
By multiplying $\vec{w_2}$ by a given real value α we can find a new vector $\vec{w_2'}$ that is collinear to $\vec{w_2}$, but of different norm.

$$\alpha = ||\vec{u}|| \ / \ ||\vec{w_2}||$$

$def\ orthogonal\_norm\_on\_first(u, v)$ is the function inside *some_script.py* that does this computation.

3 - Computing Eigenvalues, Eigenvectors, and Determinants

.1

$$Det(A) = Det\begin{pmatrix}1 & 1 & 3\\ 1 & 5 & 1\\ 3 & 1 & 1\end{pmatrix}$$

$1 * Det\begin{pmatrix}5 & 1\\ 1 & 1\end{pmatrix} - 1 * Det\begin{pmatrix}1 & 1\\ 3 & 1\end{pmatrix} + 3 * Det\begin{pmatrix}1 & 5\\ 3 & 1\end{pmatrix} = 1*(5-1) - 1*(1-3) + 3*(1-15) = 4 + 2 - 42 = -36$
The computer returned the same value.
To compute eigenvalues and eigenvectors we have to compute the characteristic polynomial, P(A):

$$P(A) = Det(A - I\lambda) = 0 <=> Det\begin{pmatrix}(1-\lambda) & 1 & 3\\ 1 & (5-\lambda) & 1\\ 3 & 1 & (1-\lambda)\end{pmatrix} = 0$$

$$(1-\lambda) * Det\begin{pmatrix}(5-\lambda) & 1\\ 1 & (1-\lambda)\end{pmatrix} - 1 * Det\begin{pmatrix}1 & 1\\ 3 & (1-\lambda)\end{pmatrix} + 3 * Det\begin{pmatrix}1 & (5-\lambda)\\ 3 & 1\end{pmatrix}$$
$= (1-\lambda)*(4 - 6\lambda + \lambda^2) - 1*(2+\lambda) + 3*(1 - 15 + 3\lambda) = -36 + 7\lambda^2 - \lambda^3 = -(\lambda+2)(\lambda-3)(\lambda-6)$
This means our eigenvalues are: $\{-2, +3, +6\}$, now we just have to compute the respective eigenvectors. This is done in the following way:

$$(A - I\lambda) = \begin{pmatrix}x\\ y\\ z\end{pmatrix} = \begin{pmatrix}0\\ 0\\ 0\end{pmatrix}, \lambda = \{-2, +3, +6\}$$

Once we solve it we get the respective eigenvectors:

$$\lambda = 6 \longrightarrow \begin{pmatrix}1\\ 2\\ 1\end{pmatrix}, \lambda = 3 \longrightarrow \begin{pmatrix}1\\ -1\\ 1\end{pmatrix}, \lambda = -2 \longrightarrow \begin{pmatrix}-1\\ 0\\ 1\end{pmatrix}$$

.2

.3

$tp\_1\_exercise\_3\_3$.*ipynb* is the file with all the code for this exercise.

4 - Computing Projection Onto a Line

.1

The distance is 3.61.
$def\ compute\_distance\_line\_a()$ is the function inside *some_script.py* that does this computation.

.2

We have a function $\alpha : 3x - 2y = -6$ that describes a line, and we have a point $A(5, 4)$. Let's see the steps required to compute the distance between A and f, figure 2 is a sketch of what is explained in the item bellow:

(1) We can rewrite it as $y = f(x) = \frac{-6-3x}{-2}$

(2) Choose two values for x, and compute the respective images: $f(x_1) = y_1$ and $f(x_2) = y_2$

(3) Compute a vector $\vec{u}$ that is collinear to our function f, $\vec{u} = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$

(4) Compute $\vec{v}$, vector between a point of the function f and our point A, $\vec{v} = \begin{bmatrix} A_x - x_1 \\ A_y - y_1 \end{bmatrix}$

(5) Compute $\vec{w_1}$ the projection of $\vec{v}$ onto $\vec{u}$, $\vec{w_1}$ and $\vec{u}$ are collinear

(6) Compute $\vec{w_2} = \vec{v}$ - $\vec{w_1}$, $\vec{w_2}$ is orthogonal to $\vec{u}$

(7) Compute $||\vec{w_2}||$, which is the distance between the point A and the line represented by the function f

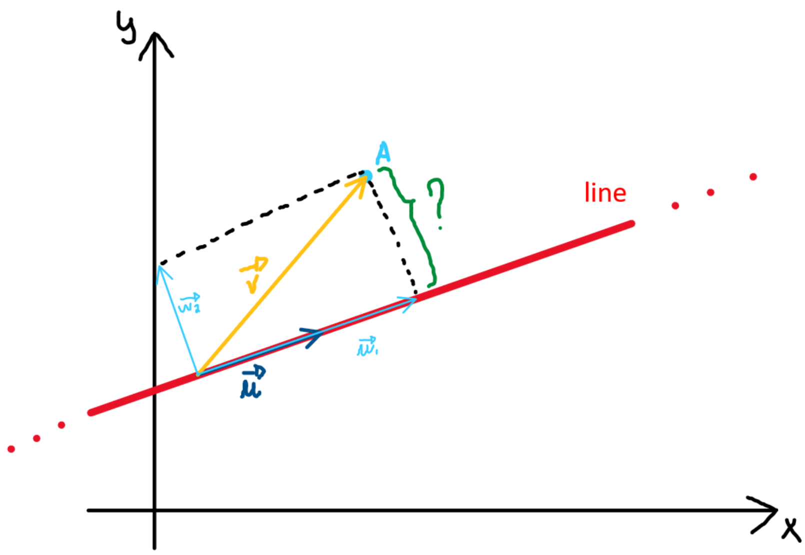

Figure 2: Distance between line and a point A