Assignment 1: Linear Algebra, Differential Calculus
Vision and Image Processing

Søren Olsen, François Lauze

November 16, 2020

This is the first mandatory assignment on the course Vision and Image Processing. The goal for you is to get familiar with vector and matrix operations as well as some computations of derivatives and gradients.

This assignment must be solved individually. You have to pass this and the following mandatory assignments and quizzes in order to pass this course. If you do not pass the assignment, but you have made a SERIOUS attempt, you will get a second chance of submitting a new solution.

The deadline for this assignment is Wednesday 25/11, 2020 at 22:00. You must submit your solution electronically via the Absalon homepage. Go to the assignments list and choose this assignment and upload your solution prior to the deadline. Remember to include your name and KU user name (login for KUnet) in the solution.

1 Vector Matrix Calculations, pen and paper

1. Let $\mathbf{v}$ be the vector $[-2, 3, 4, 1, a]^T$. Compute $\mathbf{v}^T\mathbf{v}$ and $\mathbf{v}\mathbf{v}^T$.

2. Let A and B the following matrices

$$A = \begin{bmatrix} 1 & -6 & 1 \\ 2 & -4 & 5 \\ 8 & 6 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 3 \\ -3 & 7 \\ -1 & 1 \end{bmatrix}$$

   Compute $\mathbf{v} = B.[x, y]^T$, $\mathbf{w} = A.\mathbf{v}$. Then compute $C = A.B$ and $\mathbf{z} = C.[x, y]^T$.

3. Let A be the matrix:

$$A = \begin{bmatrix} x-3 & -x+3 & -x+5 \\ x-2 & -x+2 & -x+4 \\ -1 & 1 & -1 \end{bmatrix}$$

   Compute respectively $A^2 = A.A$ and $A^3 = A.A.A$. Such a matrix A is called *nilpotent*.

4. Let A be the matrix

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

Let $\mathbf{e}_1 = [1,0,0]^T$, $\mathbf{e}_2 = [0,1,0]^T$ and $\mathbf{e}_3 = [0,0,1]^T$. Compute respectively the solutions $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$ of the linear systems

$$A.\mathbf{f}_1 = \mathbf{e}_1, \quad A.\mathbf{f}_2 = \mathbf{e}_2, \quad A.\mathbf{f}_3 = \mathbf{e}_3.$$

Let B the matrix which columns are respectively $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$. Then compute $A.B$ and $B.A$. What does it mean?

5. **[optional]** Let A and B be square matrices of size (n,n). Show that $(A.B)^T = B^T.A^T$.

2 Vector Matrix Calculations, Python

In this exercise, you are supposed to use Python and the `numpy` package. If you know Matlab, you can adapt the calculations to it.

From your preferred Python programming environment, you should start by loading `numpy` by importing it. It is standard to shorten `numpy` as `np`. See the slides for array declaration.

```
>> import numpy as np
```

Just to get your hands in the computations!

- An inner and an outer product. Compute the inner product of

  `a = np.array([1,4,-5,3])` and `b = np.array([3,-1,0,2])`.

  Use `np.dot()` or, preferably, the infix notation `@`. Computing the outer product requires some extra work. Display the *shape* of `a`, `b`. and `b.T` (`b.shape` and `b.T.shape`). To force Python-`numpy` to distinguish between column and line vectors, 1D arrays can be reshaped: Try this piece of code interactively:

```
>> print(a@b)
>> a.shape = (4,1) # a matrix with 1 column (a column vector)
>> b.shape = (4,1) # a matrix with 1 column (a column vector)
>> print(a)
>> print(b)
>> print(b.shape)
>> print(b.T.shape)
>> print(a@b)
>> print(a.T@b)
>> print(float(a.T@b))
>> print(a@b.T)
```

  What do you get?

- Let $\mathbf{t} = [2, 0, -1]^T$. Compute the cross-product $\mathbf{t} \times \mathbf{t}$ of $\mathbf{t}$ and itself. (`numpy` has a cross-product operation, feel free to implement yours). Let $\mathbf{a} = [-2, 6, 1]^T$. Compute $\mathbf{b} = \mathbf{t} \times \mathbf{a}$ and $\mathbf{c} = \mathbf{a} \times \mathbf{t}$. Then compute the inner products $\mathbf{t}^T\mathbf{b}$ and $\mathbf{a}^T\mathbf{b}$.

To a 3D vector $\mathbf{t}$, one associates the matrix

$$\hat{\mathbf{t}} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

- Write a function that takes a 3D vector $\mathbf{t}$ and associates the corresponding matrix $\hat{\mathbf{t}}$. With $\mathbf{t}$ and $\mathbf{a}$ from the previous question, compute $\hat{\mathbf{t}}\mathbf{t}$ (matrix vector product) as well as $\hat{\mathbf{t}}\mathbf{a}$. What do you get?

- For a 3D vector $\mathbf{v}$, the operation $\mathbf{v} \mapsto \mathbf{t} \times \mathbf{v}$ is *linear*. So it can be represented by a matrix. Which one?

- Check that $\hat{\mathbf{t}}^T = -\hat{\mathbf{t}}$. Compute $\hat{\mathbf{t}}^2 = \hat{\mathbf{t}}.\hat{\mathbf{t}}$ (matrix-matrix product). What is the transpose of $\hat{\mathbf{t}}^2$? What about $\hat{\mathbf{t}}^3 = \hat{\mathbf{t}}.\hat{\mathbf{t}}.\hat{\mathbf{t}}$? You may want first to perform numerical computations in Python!

3 Derivatives

Functions of one variable.

- Compute the derivative $f'(x)$ of $f(x) = e^{-\frac{x^2}{2}}$ (hint: use the chain rule for the exponential, see the tables below.)

- Compute its second derivative, $f''(x)$. (hint: reuse the previous rule and Leibniz rule).

- Plot on the same graph $f(x)$, $f'(x)$ and $f''(x)$, say for $x \in [-7, 7]$ (easy with Python's `matplotlib`).

- Compute the derivative $g'(t)$ of $g(t) = \cos(t)/\sin(t)$.

Functions of several variables. This exercise uses many of the derivation rules!

- Let $f(x, t)$ be the Gaussian distribution function of variance t,

$$f(x, t) = \frac{1}{\sqrt{2\pi t}} e^{-\frac{x^2}{2t}}.$$

- Compute $\frac{\partial f}{\partial x}(x, t)$, the first partial derivative of $f(x, t)$ with respect to x (remember that you have to assume that t is fixed and you use the rules of derivation for a function of one variable). It is also called the x-derivative of $f(x, t)$.

- Compute $\frac{\partial f}{\partial t}(x, t)$, the t-derivative of $f(x, t)$.

- Compute $\frac{\partial^2 f}{\partial x^2}(x, t)$, the second partial derivative of f with respect to x, this is the x-derivative of $\frac{\partial f}{\partial x}(x, t)$.

- Then show that
$$\frac{\partial f}{\partial t}(x,t) = \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(x,t)$$
This is an extremely important property of this function, especially for many computer vision applications, where t will play the role of a *spatial scale parameter*.
- Plot these functions for $t = 1, 2, 4$, with $x \in [-15, 15]$.

4 Some derivation rules and classical derivatives

Standard derivation rules: $f(x)$, $g(x)$ are functions of the real variable x, λ is a real number. The first derivative is denoted by $f'(x)$, the second by $f''(x)$ (the third is often denoted by by $f'''(x)$ while, in general, for higher order, say n, the n-th derivative is denoted by $f^{(n)}(x)$).

function	derivative	rule name
$\lambda f(x)$	$\lambda f'(x)$	scalar multiplication rule
$f(x) + g(x)$	$f'(x) + g'(x)$	sum rule
$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$	Leibniz rule
$\frac{f(x)}{g(x)}$	$\frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$	quotient rule
$f(g(x))$	$f'(g(x))g'(x)$	chain rule
$e^{f(x)}$	$f'(x)e^{f(x)}$	exponentiation rule (chain rule!)
$\ln \lvert f(x) \rvert$	$\frac{f'(x)}{f(x)}$	logarithm rule
$(f(x)g(x))''$	$f''(x)g(x) + 2f'(x)g'(x) + g''(x)$	iterated Leibniz rule

Some classical derivatives

function	derivative	domain/remark
x^α	$\alpha x^{\alpha-1}$	if α is not an integer, x should be > 0
e^x	e^x	$x \in \mathbb{R}$
$\ln x$	$\frac{1}{x}$	$x > 0$
$\sqrt{x}$	$\frac{1}{2\sqrt{x}}$	special case of first rule with $\alpha = \frac{1}{2}$, $x > 0$
$\cos x$	$-\sin x$	$x \in \mathbb{R}$
$\sin x$	$\cos x$	$x \in \mathbb{R}$
$\tan x$	$\frac{1}{\cos^2 x}$	$x \neq k \pm \frac{\pi}{2}, k \in \mathbb{Z}$
$\arcsin x$	$\frac{1}{\sqrt{1-x^2}}$	$-1 < x < 1$
$\arccos x$	$-\frac{1}{\sqrt{1-x^2}}$	$-1 < x < 1$
$\arctan x$	$\frac{1}{1+x^2}$	$x \in \mathbb{R}$

5 A few things about Python and Maltab

In Python with `numpy`, the `numpy.dot()` function can be use for inner products, matrix vector products, as well as matrix matrix products. From Python 3.6 at least and the corresponding `numpy` package, one can also use the *infix* notation @. In Python-`numpy`, the transpose of a array m is $m.T$. The submodule `linalg` of `numpy` contains many standard linear algebra operations.

In Matlab almost all variables are matrices. Column vectors are matrices of size $(n, 1)$ while line vectors are matrices of size $(n, 1)$. The matrix-matrix multiplication simply uses "$*$". Transposition is denoted with an apostrophe, the transpose of m is m'.

A note on relevant software

We recommend that you select python 3.6 or higher as your programming language, preferably from Anaconda, with the `numpy` and `scipy` packages. Matlab, C, C++ may be other possibilities. Still we recommend that you select the language you are most confident in. We however assume that you will use Python and `numpy`. If you still wish to use another programming language, you should have some knowledge about linear algebra libraries for this language. With Matlab, this is straightforward, the name itself stands for "MATrix LABoratory".

The focus should be on learning the methods and not learning a new programming language. If you wish to use Matlab, you may download and install this from the "Softwarebiblioteket" available at KUnet.