# Coursework 1
# Mathematics for Machine Learning (CO-496)

## Instructions

This coursework has both writing and coding components. The coding part must be done in python. The code will be run on the standard CSG installation and will be tested on the labTS [1] system. On starting the course you will be given a gitlab repository. Every commit you make to this repository can be marked automatically.

You are not permitted to use any symbolic manipulation libraries (e.g. `sympy`) or automatic differentiation tools (e.g. `tensorflow`, `pytorch`) for your submitted code (though, of course, you may find these useful for checking your answers). You should not need to import anything other than `numpy` for the submitted code for this assignment.

The writing assignment requires plots, which you can create using any method of your choice. You should not submit the code used to create these plots.

No aspect of your submission may be hand-drawn. You are strongly encouraged to use LaTeX to create the written component.

You are required to submit the following:

- A `.pdf` file for your written answers, submitted through CATe
- A `.py` file which implements all the methods for the coding exercises, submitted through gitlab.

[1] https://teaching.doc.ic.ac.uk/labts

# 1 Differentiation

In this question, we define the following constants:

$$\mathbf{B} = \begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } \quad \mathbf{b} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

We define also the following functions, which are all $\mathbb{R}^2 \to \mathbb{R}$

$f_1(\mathbf{x}) = \mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{B}\mathbf{x} - \mathbf{a}^T\mathbf{x} + \mathbf{b}^T\mathbf{x}$
$f_2(\mathbf{x}) = \sin\left((\mathbf{x}-\mathbf{a})^T(\mathbf{x}-\mathbf{a})\right) + (\mathbf{x}-\mathbf{b})^T\mathbf{B}(\mathbf{x}-\mathbf{b})$
$f_3(\mathbf{x}) = 1 - \left(\exp\left(-(\mathbf{x}-\mathbf{a})^T(\mathbf{x}-\mathbf{a})\right) + \exp\left(-(\mathbf{x}-\mathbf{b})^T\mathbf{B}(\mathbf{x}-\mathbf{b})\right) - \frac{1}{10}\log\left|\frac{1}{100}\mathbf{I} + \mathbf{x}\mathbf{x}^T\right|\right)$

a) Write $f_1(\mathbf{x})$ in the completed square form $(\boldsymbol{x}-\boldsymbol{c})^\top \boldsymbol{C}(\boldsymbol{x}-\boldsymbol{c}) + c_0$.

3 marks for

- correct $\boldsymbol{C}$
- correct $\boldsymbol{c}$
- correct $c_0$

lose a mark for any incorrect notation

b) Explain how you can tell that $f_1$ has a minimum point. State the minimum value of $f_1$ and find the input which achieves this minimum.

2 marks for

- the appropriate eigenvalues and accompanying explanation
- the minimum value stated and the input which achieves this value

c) Write a python function `grad_f1(x)` that return the gradient for $f_1$.

4 marks

d) Write a python function `grad_f2(x)` that return the gradient for $f_2$.

6 marks

e) Optional: Write a python function `grad_f3(x)` that return the gradient for $f_3$. If you don't do this question you still need the gradient for the next part, but you can use an automatic differentiation package. `autograd` is recommended as a good lightweight option, but you could also use e.g. `pytorch`, `tensorflow`, `mxnet` or `chainer`.

There are no marks for this question, but you can still submit the code to labts to verify your answer.

f) Use your gradients to implement a gradient descent algorithm with 50 iterations to find a local minimum for both $f_2$ and $f_3$. Show the steps of your algorithm on a contour plot of the function. Start from the point $(1, -1)$ and state the step size you used. Produce separate contour plots for the two functions, using first component of $\mathbf{x}$ on the $x$ axis and the second on the $y$.

4 marks for:

- sensible scales, plots labeled, contours visible (use at least 20)
- correct functions shown
- gradient descent for first plot with 50 points, converging to minimum
- gradient descent for second plot, converging to one of the minima

g) For the two functions $f_2$ and $f_3$, discuss the qualitative differences you observe when performing gradient descent with step sizes varying between 0.1 and 1, again starting the point $(1, -1)$. Briefly describe also what happens in the two cases with grossly mis-specified step-sizes (i.e. greater than 1), with a reason to explain the difference in behaviour.

6 marks for:

- observation about number of minima for $f_2$
- observation about large stepsizes for $f_2$
- observation about number of minima for $f_3$
- observation about different behaviour for different moderate stepsizes for $f_3$
- observation about large stepsizes for $f_3$
- explanation for the difference in behaviour