EXERCISE DAY 2
CPDE SUMMER SCHOOL:
A PRACTICAL INTRODUCTION TO CONTROL, NUMERICS AND MACHINE LEARNING

DANIËL VELDMAN

Recall from the lecture that the training of a deep (residual) neural network can be viewed as an optimal control problem in which the cost function

$$J(\mathbf{V}, \mathbf{b}) = \frac{1}{2} \sum_{i=1}^{I} |\mathbf{x}^i(T) - \mathbf{y}^i_{\text{out}}|^2 + \frac{w_1}{2} \sum_{i=1}^{I} \int_0^T |\mathbf{x}^i(t) - \mathbf{y}^i_{\text{out}}|^2 \, \mathrm{d}t + \frac{w_2}{2} \int_0^T (\|\mathbf{V}(t)\|_F^2 + |\mathbf{b}(t)|^2) \, \mathrm{d}t, \tag{1}$$

should be minimized subject to the dynamics (for $i = 1, 2, 3, \ldots, I$)

$$\mathbf{x}^i(0) = \mathbf{x}^i_{\text{in}}, \qquad \dot{\mathbf{x}}^i(t) = \mathbf{V}(t)\sigma(\mathbf{x}^i(t) + \mathbf{b}(t)). \tag{2}$$

Note: all files for this exercise work both in Matlab and Octave.

a. Compute the state

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_I(t) \end{bmatrix} \in \mathbb{R}^{NI}. \tag{3}$$

resulting from an input $(\mathbf{V}(t), \mathbf{b}(t))$ using the Forward Euler scheme by filling in the missing lines in `NN_compute_state`. Use a time grid with $N_T = 101$ points. Use the obtained solution to evaluate the cost functional J according to the scheme explained in the lecture by completing the missing line in `NN_loss_function`.

b. Compute to the adjoint state

$$\mathbf{\Phi}(t) = \begin{bmatrix} \boldsymbol{\varphi}_1(t) \\ \boldsymbol{\varphi}_2(t) \\ \vdots \\ \boldsymbol{\varphi}_I(t) \end{bmatrix} \in \mathbb{R}^{NI}, \tag{4}$$

by completing the missing lines in `NN_compute_adjoint`. Use this adjoint state to compute the gradients $\nabla_{\mathbf{V}} J(\mathbf{V}, \mathbf{b})$ and $\nabla_{\mathbf{b}} J(\mathbf{V}, \mathbf{b})$ by completing the file `NN_compute_gradients`. Follow the procedure discretize-then-optimize approach given in the lecture.
Validate the obtained gradients by comparing $\beta \mapsto J(\mathbf{V} - \beta\nabla_{\mathbf{V}} J(\mathbf{V}, \mathbf{b}), \mathbf{b})$ and $\beta \mapsto J(\mathbf{V}, \mathbf{b} - \beta\nabla_{\mathbf{b}} J(\mathbf{V}, \mathbf{b}))$ to the linear approximations you get from the computed gradients.

c. Use the results from parts a and b to develop and implement a gradient-based algorithm to minimize the functional J by completing the missing lines in the file `CPDESS_Exercise2`. Assure that the step size (learning rate) β is chosen such that J is decreasing in every step, i.e. such that

$$J(\mathbf{V} - \beta \nabla_{\mathbf{V}} J(\mathbf{V}, \mathbf{b}), \mathbf{b} - \beta \nabla_{\mathbf{b}} J(\mathbf{V}, \mathbf{b})) < J(\mathbf{V}, \mathbf{b}). \tag{5}$$

Note that you will need at least 1000 iterations (epochs), or maybe even 10,000, to get a reasonably good classification result.