Université d'Orléans **2025/2026**
Graduate School Orléans Numérique (GSON)

Introduction to Deep Learning
Problem Sheet 2

1) Gradient Descent on a Simple Function

Visualize how gradient descent minimizes a scalar function.

Consider $f(x) = x^2 + 2x + 1$.

1. Derive the analytical gradient $f'(x)$.
2. Starting from $x_0 = 5$, perform 20 steps of gradient descent with learning rate $\eta = 0.2$.
3. Plot $f(x)$ and the successive x_k positions on the curve.
4. Repeat with $\eta = 0.01$ and $\eta = 1.0$. Discuss the effect of learning rate on convergence.

2) Linear Regression by Gradient Descent

Implement gradient descent manually for a simple regression model.

1. Given data points (x_i, y_i), model predictions as $\hat{y}_i = wx_i + b$.
2. Define the mean squared error loss:
$$E(w, b) = \frac{1}{n} \sum_i (wx_i + b - y_i)^2.$$
3. Derive analytical expressions for:
$$\frac{\partial E}{\partial w} = \frac{2}{n} \sum_i x_i(wx_i + b - y_i), \quad \frac{\partial E}{\partial b} = \frac{2}{n} \sum_i (wx_i + b - y_i).$$
4. Implement training in Python using NumPy only. Initialize $w = b = 0$, learning rate 10^{-3}, and train for 500 epochs.

3) Mini Neural Network: Manual Forward and Backward Pass

Implement a small fully connected neural network that performs regression on the Diabetes dataset using your own `dense` and `relu` classes provided on the course website.

1. Download the file `diabetes.csv` from the course website and load it using `np.loadtxt`. The dataset contains 10 input features (normalized medical measurements) and one continuous output (a diabetes progression score).
2. Split the dataset into:
   - Training set: 80%
   - Validation set: 10%
   - Test set: 10%
3. Build a small network using the provided Python classes:
$$\text{Dense}(10, 16) \to \text{ReLU}() \to \text{Dense}(16, 1).$$
   Use the `dense` and `relu` classes as given in the online material.

4. Implement the **forward pass**: propagate input data through each layer and compute predictions. Then compute the mean squared error (MSE) between predictions and true labels:

$$E = \frac{1}{n} \sum_i (\hat{y}_i - y_i)^2.$$

5. Implement the **backward pass**:

   (a) Compute the gradient of the MSE with respect to the predictions:

$$\frac{\partial E}{\partial \hat{y}} = \frac{2}{n}(\hat{y} - y).$$

   (b) Pass this error through the network in reverse order using each layer's `backprop` method.

6. Train your network for several epochs (e.g. 50000) and record the training and validation loss as a function of epoch number. Plot both curves.

5) Effect of Initialization

Observe the impact of weight initialization on convergence.

1. Repeat the training of your mini-network from question 3 using different initialization schemes:
   - all weights set to zero;
   - random values from $\mathcal{N}(0, 1)$;
   - Xavier initialization ($\text{Var}(w) = \frac{2}{n_{in}+n_{out}}$);
   - He initialization ($\text{Var}(w) = \frac{2}{n_{in}}$).
2. Plot training loss vs. epoch for each case.