Midterm Project: Training Feedforward Neural Networks

Purposes

In the project, you are required to implement a feedforward neural network class which is equipped with the basic operations for supervised learning.

Implementation Tasks

1. (20 points) Write a Python class for Feedforward Neural Network (FNN). You need to discuss with your teammates on the data structures. Some hints are given as follows:

   - (5 points) We know that the parameters of an FNN are the number of layers, the number of neurons in each layer, the weights on the edges and the biases on the neurons. Then, how can we design the data structures such that the information that we need to keep is minimized. For example, we may first define a Python class for layer and then build the FNN class on that. In the implementation, you are also required to "absorb" the biases to the weight matrices.
   - (3 points) Write a member function named `forward` which performs a forward propagation on the FNN. It takes an input vector (a NumPy array) and returns the output of the FNN.
   - (3 points) Write a member function named `gd` which performs a gradient descent on the FNN parameters according to the computed partial derivatives (gradients).
   - (9 points) Write a member function named `backward` which performs a backpropagation on the FNN. The purpose of backpropagation is to compute the partial derivatives of the loss function with respect to all parameters (weights). Please discuss with your teammates on the data structure that is used for keeping the partial derivatives. Here, we do not fix the type of the loss function. Instead, the function takes the loss function as a parameter.

2. (20 points) Test your implementation using the following case studies.

   - (5 points) **Case 1: Simple Regression.** Write a program that trains an FNN which approximates the function $\sin(x)$ based on the samples randomly generated in the range of $[-3, 3]$ for x. The task is similar to the PyTorch program `regression.py` except that the samples are required to generated randomly. You may use the MSE loss function in this task.
   - (5 points) **Case 2: Data-Driven Model.** Write a program that trains an FNN for approximating the one-step (0.5 s) reachability relation of the Van der Pol system whose ODE is given below

     $$\dot{x}_1 = x_2,\ \dot{x}_2 = -x_1 + (1 - x_2^2)x_2$$

     You may use the code in the file `vanderpol.py` for generating the samples in the state space of $x_1 \in [-3, 3]$, $x_2 \in [-3, 3]$. You may also use an MSE loss function however a *mini-batch stochastic gradient descent method* should be used. To do so, you may first set up a batch size, randomly select samples to the batch, and then perform the standard gradient descent method to update the parameters. Please discuss with your teammates on the details.
   - (10 points) **Case 3: Handwriting Numbers.** Write a program that trains an FNN for recognizing the handwriting numbers in the MNIST data set. You may use either PyTorch or scikit-learn for obtaining the training and testing data sets. Your FNN is required to have a LogSoftmax output layer[1] and the loss function is required to be an NLLLoss[2]. You may use the same test method given in the `handwritingnumbers.py` file. In this task, you are also required to use the mini-batch stochastic gradient descent method.

     In this task, your FNN should have 784 (28×28) input values each of which represents a pixel in the image.

[1] `https://pytorch.org/docs/stable/generated/torch.nn.LogSoftmax.html#torch.nn.LogSoftmax`
[2] `https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html#torch.nn.NLLLoss`

You may use randomly generated testing data in the evaluations of the first two cases. Notice that, in Case 2, your testing data may just be a set of initial states. You may compare the trajectories produced by the data-driven model and the original ODE.

3. (10 points) Write a report for your implementation and experiments. Although you are not required to use a particular template, the report should cover the following content:
   - (3 points) Using figures or pseudocode to explain the data structures for FNN.
   - (2 points) Explaining the data organization in backpropagation.
   - (5 points) A summary for the experiments along with your observations and thoughts.