Project description

Mathematics for Machine Learning MA4029

The project can be done in groups of two people.

In this project, you are going to adopt your Multivariate Calculus knowledge to train a Multilayer Perceptron (MLP).

You are supposed to define a function that receives the following list of parameters. The purpose of this function is to design the architecture of your network.

MLP architecture function parameters

1- Number of layers
2- Number of neurons in each layer
3- activation function in each layer

For example, if you want to have 3 layers with 10 neurons in the first layer, 5 neurons in the second layer, and 1 neuron in the last layer, one way would be to pass a list as follows:

[10,5,1] and for activation functions, an example would be [‘sigmoid’, ‘sigmoid’, ‘linear’]

You need to have another function for training your designed neural network. The arguments for this function are as follows:

MLP training function parameters

1- learning rate
2- number of iterations
3- X-train, y-train

Each row of X-train contains a data point, and each column is a feature value. y-train contains the corresponding output (target) for each data point.

To be able to train MLP, you need to define a loss function. It can be Mean Square Error between your MLP prediction and actual output, for example. Then, you need to take the derivative of the loss function with respect to all parameters of the MLP. However, you need to do it systematically using recursion and chain rule, accompanied by Matrix algebra.

During each epoch, the error on the training set should be printed (ideally, it should be on another dataset, technically called validation set; however, for simplicity, you can skip that). If MLP works properly, the error should be decreased during the training.

Finally, you need to have another function that receives the test data and provides the prediction on the test data and returns the error.

MLP test function parameters

- X-test, y-test

You should train your MLP on the following functions:

- f(x) = exp(x)
- f(x) = sin(x)/x
- f(x,y) = sin(x) + cos(y)

For a detailed explanation of MLP and how you can train it, use the following reference.
As a suggestion, to have an intermediate step, instead of a flexible architecture, you can start with a fixed architecture and train it. This will lead to some simplification; once you did that, you will have a better vision of how to proceed with the flexible architecture.

Reference:

Neural Network Design (2nd Edition) - Chapter 11
Martin T Hagan , Howard B Demuth , Mark H Beale , Orlando De Jesús