Assignment 1 : Feedforward neural networks

IMPORTANT : Please do not share your solution to this assignment on the web or with anyone !

In this assignment, you must implement in Python a multi-layer feedforward neural network for classification.

The implementation of the neural network must be contained in a class named `NeuralNetwork`, that inherits from the class `Learner` of the MLPython library. The definition of the class must be placed in a file named `nnet.py`. Your implementation should support the use of the hyper-parameters :

- `lr` : learning rate of stochastic gradient descent (`float`)
- `dc` : decrease constant for the learning rate
- `sizes` : list of the number of neurons in each hidden layer, from the first to the last hidden layer (`list` of `ints`)
- `L2` : L2 regularization of the weight matrices of the neural network (`float`)
- `L1` : L1 regularization of the weight matrices of the neural network (`float`)
- `seed` : seed of the random number generator for initialization of the parameters (`int`)
- `tanh` : boolean indicating whether the hyperbolic tangent function must be used as the activation function of the hidden neurons (`True`) rather than the sigmoid function (`False`)
- `n_epochs` : number of training iterations (`int`)

A skeleton of the `NeuralNetwork` class is provided in the file `nnet.py` available on the course's website. The skeleton also specifies the signature of all methods that you must implement. **It is important to use the Numpy library in your implementation, so that it is efficient**.

A method called `verify_gradients` is already implemented. It compares the computation of the gradients through backpropagation with a finite difference approximation. It is important to use this method to test whether your implementation of the forward and backward propagation are correct. A script `run_verify_gradients.py` that verifies the gradients for different configurations of hyper-parameters is also provided. The reported differences between your implementation and the finite difference approximation should be smaller than 10^{-10}.

Moreover, a script `run_nnet.py` is available to train a neural network on the *OCR Letters* data set, using early stopping. The script's arguments are the values of the hyper-parameters, as follows :

```
Usage: python run_nnet.py lr dc sizes L2 L1 seed tanh

Ex.: python run_nnet.py 0.1 0 [20,10] 0 0 1234 False
```

The script will print the neural network errors on the training and validation sets after every epoch of training. At the end of training, the errors on the training, validation and test sets will also be appended into a text file named `results_nnet_ocr_letters.txt`. Each new execution of the script will append a new line. The errors that must be computed in your implementation are the classification errors and the regularized negative log-likelihood. Early stopping will use the classification error on the validation set to determine when to stop and will use a "look ahead" of 5.

For the script to work properly, you must first download the *OCR Letters* data set using MLPython's automatic procedure. Make sure to define the `MLPYTHON_DATASET_REPO` environment variable and use the script `mlpython_helper` to download the data set as follows :

```
mlpython_helper -datasets -download ocr_letters
```

You can find more information in MLPython's tutorial, including on how to handle `MLProblem` objects[1] (MLPython's data set objects).

Once your implementation is complete, you can generate results on the *OCR letters* data set to assess the performance of your implementation. Specifically, try to :

- report the classification error rates on the training and validation sets **for at least 15 different choices of hyper-parameter configurations** ;
- illustrate the **progression of the classification error on the training and validation sets**, for a configuration of your choice of the hyper-parameters ;
- also illustrate the **progression of the average negative log-likelihood on the training and validation sets**, for a configuration of your choice of the hyper-parameters ;
- report the classification error rate on the test set **only for the hyper-parameter configuration having the best performance on the validation set** ;
- specify a **95% confidence interval** of the test set classification error.

Good luck !

1. `http://www.dmi.usherb.ca/~larocheh/mlpython/tutorial.html#processed-datasets-mlproblems`