IFT 725 : Assignment 1
Individual work

Due date : September 21st, 5pm (**at the latest**)

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 `int`s)
- `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 must generate results on the *OCR letters* data sets to assess the performance of your implementation. Specifically, you must :

- 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.

These results must be reported in a report following the format of the NIPS machine learning conference [2]. The document must be in the PDF format and must have been generated using LaTeX [3]. The report must contain the following sections :

- **Introduction :** gives a summary of the content and objective of the report ($^1/_2$ page).
- **Description of the approach :** mathematically describes the model and the learning algorithm (up to 4 pages), including
  - a description of forward propagation ;
  - a description of backpropagation ;
  - a description of the training objective ;
  - a description of the stochastic gradient descent algorithm in the context of a neural network ;
  - a description of all the hyper-parameters.
- **Experiments :** presents the results of your experiments, that is :
  - a **table** with the average training and validation errors of **at least 15 different configurations of hyper-parameters** ;
  - a **plot** showing the progression of the classification error rate on the training and validation sets for a configuration of hyper-parameters of your choice
  - a **plot** showing the progression of the average regularized negative log-likelihood on the training and validation sets for a configuration of hyper-parameters of your choice
  - the result on the test set **only** for the hyper-parameter configuration with the best performance on the validation set ;
  - the 95% confidence interval for the test set classification error.

The points are distributed as follows :

- **10 points** for the correctness of the implementation, as well as its quality (including the appropriate use of Numpy) ;

1. `http://www.dmi.usherb.ca/~larocheh/mlpython/tutorial.html#processed-datasets-mlproblems`
2. See `http://nips.cc/PaperInformation/StyleFiles` for the necessary format files
3. See `http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/GSWLaTeX.pdf` for an introduction and `http://www.andy-roberts.net/writing/latex/pdfs` for how to generate a PDF document from a LaTeXfile.

- **4 points** for the quality and accuracy of the **Introduction** and **Description of the approach** sections in the report ;
- **4 points** for reporting all the expected results in the **Experiments** section and for their validity.

The report may be written in English or French. All the work in this assignment most be done individually. No code, text or results may be shared between students. However, students are encouraged to discuss elements of their solutions orally with each other.

Please submit your implementation and your report using the **turnin** command :

```
turnin -c ift725 -p devoir_1 nnet.py rapport.pdf
```

Good luck !