IFT 725 : Assignment 3
Individual work

Due date : November 9th, 8 :30am (**at the latest**)

In this assignment, you must implement in Python a restricted Boltzmann machine (RBM) and a denoising autoencoder, used to pretrain a neural network.

The implementation of the RBM and the autoencoder must be contained in classes named `RBM` and `Autoencoder`, that inherit from the class `Learner` of the MLPython library. The definition of the classes must be placed in files named `rbm.py` and `autoencoder.py` respectively. Thes classes support the use of the hyper-parameters :

- `lr` : learning rate of stochastic gradient descent (`float`)
- `hidden_size` : size of the hidden layer (`int`)
- `seed` : seed of the random number generator for initialization of the parameters (`int`)
- `n_epochs` : number of training iterations (`int`)

The `RBM` class must also support the following hyper-parameter :

- `CDk` : number of Gibbs step used by contrastive divergence (`int`)

On the other hand, the `Autoencoder` class must support the following hyper-parameter :

- `noise_prob` : the noise probabiliy of fixing an input to 0 (`float`)

A skeleton of the `RBM` and `Autoencoder` classes are provided in the files `rbm.py` and `autoencoder.py` available on the course's website. You only have to implement the method `train` in these files. **It is important to use the Numpy library in your implementation, so that it is efficient**.

To debug your implementations, the scripts `run_show_filters_rbm.py` and `run_show_filters_autoencoder.py` can be used to compare the learned filters (i.e. the connections of each hidden units) with those obtained by a correct implementation (see the files `rbm_filters.pdf` and `autoencoder_filters.pdf` available on the course's website).

Moreover, scripst `run_stacked_rbms_nnet.py` et `run_stacked_autoencoders_nnet.py` are available to pretrain a neural network using either the restricted Boltzmann machine or the denoising autoencoder (respectively) on the *OCR Letters* data set (the same as in the first assignment).

The script `run_stacked_rbms_nnet.py` requires the following hyper-parameters :

```
Usage: python run_stacked_rbms_nnet.py lr dc sizes pretrain_lr pretrain_n_epochs pretrain_CDk seed

Ex.: python run_stacked_rbms_nnet.py 0.01 0 [200,100] 0.01 10 1 1234
```

The script `run_stacked_autoencoders_nnet.py` requires the following hyper-parameters :

```
Usage: python run_stacked_autoencoders_nnet.py lr dc sizes pretrain_lr pretrain_n_epochs
       pretrain_noise_prob seed

Ex.: python run_stacked_autoencoders_nnet.py 0.01 0 [200,100] 0.01 10 0.1 1234
```

The scripts will print the 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 text files named

`results_stacked_rbms_nnet_ocr_letters.txt` and `results_stacked_autoencoders_nnet_ocr_letters.txt` (respectively). Each new execution of the script will append a new line. Pretraining uses the number of pretraining epochs (`pretrain_n_epochs`) specified by the user. Early stopping based on the classification error on the validation set determines the number of training iterations for fine-tuning (with a look ahead of 5).

Once your implementation is complete, you must generate results on this *OCR letters* data set 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** (don't report experiments only with the RBM or the autoencoder, try both at least once) ;
- 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 [1]. The document must be in the PDF format and must have been generated using LaTeX [2]. 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 the restricted Boltzmann machine :
    - what is the energy function and the definition of $p(\mathbf{x}, \mathbf{h})$?
    - what is the expression for $p(\mathbf{x})$?
    - what is the expression for the conditionnal distributions $p(\mathbf{h}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{h})$?
    - what is the training objective we'd like to optimize and how is *contrastive divergence* used to approximate its gradients ?
    - why can't we compute exactly the gradients of the training objective we'd like to optimize ?
  - a description of the autoencoder (with/without denoising)
    - what is forward propagation in the autoencoder ?
    - what is the training objective ?
    - what is backpropagation in the autoencoder ?
    - what is the difference between the basic autoencoder and the denoising autoencoder ?
    - what is the advantage of the denoising autoencoder over the basic autoencoder ?
  - a description of the pretraining procedure of a neural network using a restricted Boltzmann machine or an autoencoder ;
  - an explanation for why pretraining can be helpful and improve the generalization of a neural network.
- **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

1. See `http://nips.cc/PaperInformation/StyleFiles` for the necessary format files. You will need to add the command `\nipsfinalcopy` after the command `\author` to deanonymize your document.
2. 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.

- a **plot** showing the progression of the average 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) ;
- **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 code and your report using the **turnin** command :

```
turnin -c ift725 -p devoir_3 rbm.py autoencoder.py rapport.pdf
```

Good luck !