

Assignment 2 : Conditional random fields

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 linear chain conditional random field for classification.

The implementation of the CRF must be contained in a class named `LinearChainCRF`, that inherits from the class `Learner` of the `MLPython` library. The definition of the class must be placed in a file named `crf.py`. This class supports the use of the hyper-parameters :

- `lr` : learning rate of stochastic gradient descent (`float`)
- `dc` : decrease constant for the learning rate
- `L2` : L2 regularization of the weight matrices of the CRF (`float`)
- `L1` : L1 regularization of the weight matrices of the CRF (`float`)
- `n_epochs` : number of training iterations (`int`)

A skeleton of the `LinearChainCRF` class is provided in the file `crf.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 with a finite difference approximation. It is important to use this method to test whether your implementation of belief propagation inference and gradients computation 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_crf.py` is available to train a linear chain CRF on the *OCR Letters* data set (sequential classification version), using early stopping. The script's arguments are the values of the hyper-parameters, as follows :

```
Usage: python run_crf.py lr dc L2 L1
```

```
Ex.: python run_crf.py 0.1 0 0 0
```

The script will print the CRF 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_crf_ocr_letters_sequential.txt`. The standard deviation of these average errors (i.e. the standard error) will also be given (required to compute confidence intervals). 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 (sequential classification version) using the script `download_ocr_letters_sequential.py` available on the course website. Make sure to define the `MLPYTHON_DATASET_REPO` environment variable and use the script as follows :

```
python download_ocr_letters_sequential.py
```

Once your implementation is complete, you can generate results on this *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 !