MAC0460/MAC5832 – EP5

DCC / IME-USP — Primeiro semestre de 2020

1 Objectives

The aim of this EP is to give students the opportunity to experiment:

- `scikit-learn` (https://scikit-learn.org/)
- neural networks (https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html)
- model selection, based on cross-validation

2 Description

We will use again the MNIST dataset (see EP4 for more details)

More specifically, the goal is to train and validate multiple architectures of neural networks (of the multilayer perceptron type) on the training set of MNIST, then choose one model, and finally evaluate its performance on the test set. The main steps of the script are the following:

1. The MNIST dataset is already separated into training (60000 examples) and test (10000 examples) sets. Keep this separation.
2. This time we will use pixel intensities as features. The images must be flattened to vector format and the feature values must be scaled to the range [0,1] (as float).
3. To reduce computation time, let us create a reduced training set by randomly selecting 2000 examples of each class from the original training set. Thus our effective training set (reduced training set) will consist of 20000 examples.

   After this step, you will have the following sets: `X_train`, `y_train`, `X_test`, `y_test`.

   Shuffle the samples in the training set.
4. Model selection: a model must be chosen based on 5-fold cross-validation applied on the reduced training set; use stratified cross-validation. All models must be trained on the same set of folds.

   You should consider at least 3 different models (architectures). Among them, the one at https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/2.1-a-first-look-at-a-neural-network.ipynb (same architecture and whenever possible, same hyperparameters) is mandatory. The others may be networks with variations in architecture and/or hyperparameters. Fix the number of epochs for training (you may plot the learning curve to see how it is behaving and based on that choose the number of appropriate epochs).
5. After model selection, train again the selected model, now using the entire reduced training set, to get the final model. (Note that here there is no validation step)

6. Test the final model on the test set.

The items listed in the above script are mandatory. They must be clearly organized in a Python notebook, according to the following basic structure. For each part, specific information as specified below should be printed (with `print()` function) or plotted.

Part I: Data preparation (items 1 to 3)

Print the following properties of the sets `X_train`, `y_train`, `X_test`, `y_test`:

```
* dtype
* shape
* minimum value
* maximum value
```

At the end of this part, randomly choose at least 3 examples from the set `(X_train, y_train)` and display the images as well as the respective class labels.

Part II: Model selection (item 4)

With respect to the five folds, print the number of examples of each class at each fold.

For each model, print the accuracy with respect to each of the validation folds, and the final cross-validation accuracy.

Justify your choice of model based on the statistics you observed.

Part III: Selected model evaluation (items 5 and 6)

Plot the learning curve of the selected model (when retraining on `X_train`).

Print the number of examples of each class in the test set.

Plot the confusion matrix with respect to the test set.

3 Additional comments

Feel free to add any more sections, comments, plots or prints you find relevant or convenient to the above script organization.

The training and testing part of the above script must be done using `scikit-learn`; you can also use any of the functions available there (regarding cross-validation, model selection, learning curve, metric computation, and so on). Feel free also to use additional data reading/visualization libraries.

You should submit the notebook with the outputs.