11753 Computational Intelligence
Master in Intelligent Systems
Universitat de les Illes Balears

Handout #2: **Feed-forward Neural Networks (FFNN)**
(graded assignment)

Tasks T1 – T8 below require loading the datasets corresponding to each group. You can find the datasets for all group numbers in the course web page.

You have to load files `ds1-gg-k-nn-tr.csv` and `ds1-gg-k-nn-te.csv`, for, respectively, training and test splittings, where `gg` is the group number and `k` is the number of split. There are three disjoint splits per group and dataset, which can be loaded as follows:

```python
from pandas import read_csv
g = 1 # assuming group 1
k = 1 # assuming split 1
data = read_csv('ds1-%02d-%d-nn-tr.csv' % (g,k))
X_train_1 = data.iloc[:,:-1].to_numpy()
y_train_1 = data.iloc[:,-1].to_numpy()
data = read_csv('ds1-%02d-%d-nn-te.csv' % (g,k))
X_test_1 = data.iloc[:,:-1].to_numpy()
y_test_1 = data.iloc[:,-1].to_numpy()
```

Listing 1: Loading of the dataset.

These datasets comprise **electrocardiogram signals** (ECG) of heartbeats for the normal case (class label 0) and four more cases affected by different arrhythmia and myocardial infarction (class labels 1 to 4). These signals have been preprocessed and segmented, so that each segment is a heartbeat, and hence one sample of the dataset; each segment consists of signal values for 125 time instants. Figure 1 shows one example for each class.

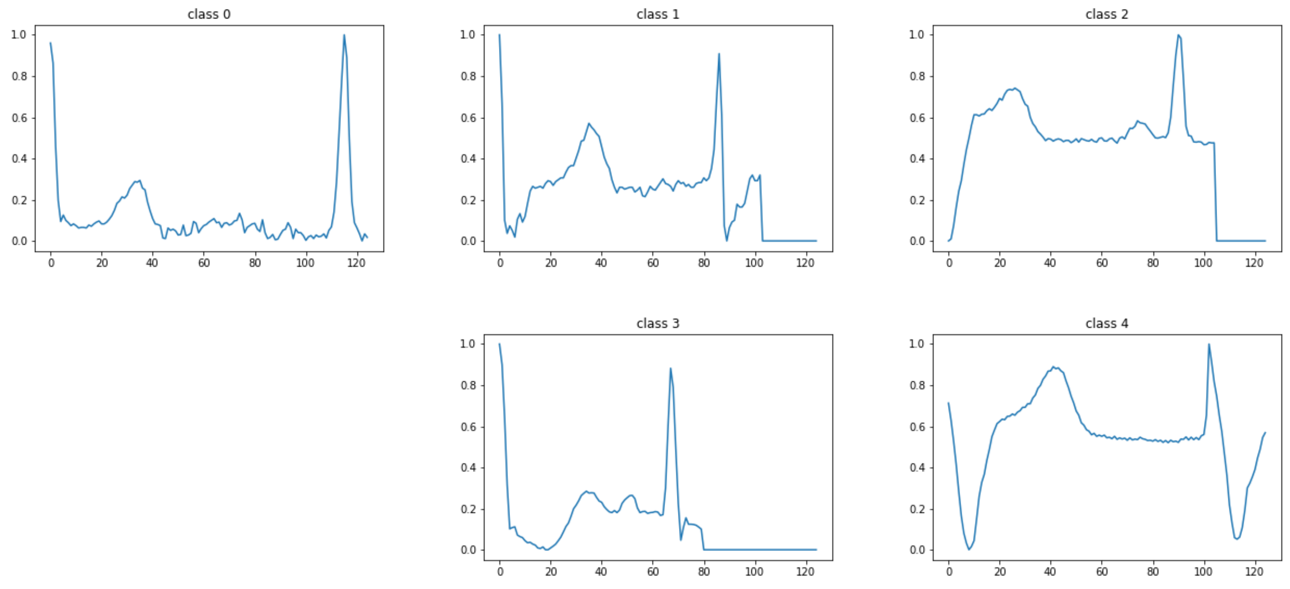

Figure 1: Samples for each class of the dataset.

As can be observed, signal values are already normalized between 0 and 1, so that no extra normalization is needed *a priori*.

Now, to address the multi-class classification problem described above and contained in datasets *ds1-...*, design and evaluate an FFNN following the next methodology using the *training* and *test sets* of the *first split* of your dataset:

T1. Start with a basic configuration: a single hidden layer network and the most basic activation function. Try with three different amounts of neurons, and keep the best configuration for the next task.

For training, set a fixed learning rate, the most basic optimizer, a reasonable batch size, the most direct loss function and choose enough epochs to let the training converge. Tune the learning rate until you achieve convergence and a reasonable performance. To measure performance, use the *accuracy* for the *test set.*

T2. Next, check whether a change in the activation function of the hidden layer neurons improves the classification performance. If that is the case, continue with the alternative activation function.

T3. Try with a dynamic learning rate and use it from now on if the performance does not get worse.

T4. Change to an alternative optimizer and keep it if the performance gets better.

T5. Switch to another loss function to check whether the performance level increases.

T6. Add a second hidden layer with a reasonable number of neurons and check whether a performance gain is obtained. If that is the case, keep the second layer.

T7. Try with larger and smaller batch sizes (one of each). You should observe that the training time also changes. Adopt the size leading to highest performance.

For each task T1-T7 above, report on the performance attained by each configuration.

T8. For the best configuration found, determine the performance of the network using the *accuracy*, *precision*, *recall* and f_1 metrics ([2]. To this purpose, you have to:

a) Train the network for every one of the splits (using the corresponding *training set*). Show that the training has achieved convergence in each case by means of an appropriate plot.

b) Evaluate the network for every split (using the corresponding *test set*), reporting the *confusion matrix* and the aforementioned metrics.

c) Calculate and provide the average value for each metric.

DELIVERY INSTRUCTIONS:

- To implement the solutions to tasks T1 – T8, you can either use a notebook file (`.ipynb`) or separate python files (`.py`). In the latter case, use a python file for each task and include inside all the source code that is needed to run the solution to the task.

  The name of the python files has to be `alltasks.ipynb`, or `task1.py`, `task2.py`, etc.

- Brief/suitable comments are expected in the source code.

- A report of the work done has to be delivered by/on January 8, 2024 in PDF form. The report can be generated by exporting the notebook file (after full execution) or using a separate text editor; you can find a template in `.docx` format in the course web page that you can adapt for the `.ipynb` case.

  Upload a Zip container to package the report (with name `report.pdf`) and the source code files (.ipynb or .py file(s)).

- This work can be done in groups of 2 students (contact me with the name of the members of the group for getting a group number; the members of each group and their group number will be included in a list available in the course web page).

- IMPORTANT NOTICE: An excessive similarity between the reports/source code released can be considered a kind of plagiarism.

[2] `https://scikit-learn.org/stable/modules/model_evaluation.html`