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

Handout # 1: Feed-Forward Neural Networks (FFNN)

NOTE 1: Problem P1 requires *training* and *test datasets*. They are, respectively, stored in `dsxx1tr.txt` and `dsxx1te.txt` files:

```
import numpy as np
group = '01' # assuming group 1
ds = 1       # assuming problem 1
data = np.loadtxt('ds'+group+str(ds)+'tr.txt')
X_train = data[:, 0:2]
y_train = data[:, 2]
data = np.loadtxt('ds'+group+str(ds)+'te.txt')
X_test = data[:, 0:2]
y_test = data[:, 2]
```

Class labels are 1 for ω_1 and 0 for ω_2.

NOTE 2: Problem P1 also requires the use of `tensorflow-keras` (`https://keras.io/`), `scikit-learn` (`https://scikit-learn.org`) and `matplotlib` (`https://matplotlib.org/`).

P1. **Given datasets `dsxx1tr.txt` and `dsxx1te.txt`**, find a suitable FFNN-based classifier. You have to define and train a network with two hidden layers with, respectively, **nh** and **nh - 1** hidden neurons.

a) Normalize the training data to ensure zero mean and unit variance. (Consider the pre-processing functions of `https://scikit-learn.org/stable/modules/preprocessing.html`, in particular the **StandardScaler**.)

b) Using the *training set*, find a classifier of adequate performance, i.e. *accuracy* $\geq$ 95% for the *test set*, for **nh** = 3, 5 and 7. For each case, run the training several times, e.g. 3, and keep the best training. Define a validation set comprising 20% of the training set in each case. In case it is not possible to achieve 95% of accuracy, keep the best training you have been able to obtain.

c) Provide the following plots and performance measurement results for the model found:

1. the evolution of the *loss* function and the *accuracy* for the training and validation sets;
2. the *classification map*, i.e. a plot with the evaluation of the network for a 'regular' subset (grid) of points (remember to superimpose the *training samples*);
3. the *confusion matrices* for both the training and the test sets, and
4. the *test accuracy*, *test precision*, *test recall* and *test f1-score*.

The **performance assessment module** of scikit-learn (`https://scikit-learn.org/stable/modules/model_evaluation.html`) may be useful for the requests above.

Use different markers and/or colours for each class in each plot.

- A report of the work done has to be released by March 22, 2022 in electronic form as a notebook file (.ipynb).
- Provide the requested data and plots/figures at each point above. For figures, use appropriate titles, axis labels and legends to clarify the results reported.
- Suitable comments are expected in the source code.
- This work has to be done individually (see the number of group in *Aula Digital*).

The following source code may help to obtain the plots requested above:

```
def plot_results(model, X, y):

   w1i = np.array(np.where(y == 0))
   w2i = np.array(np.where(y == 1))

   plt.figure()

   # plot samples
   plt.plot(X[w1i,0],X[w1i,1],'+r')
   plt.plot(X[w2i,0],X[w2i,1],'+g')
   plt.axis('equal')
   plt.title('samples and decision boundary')

   # plot the decision boundary
   ax = plt.gca()
   xlim = ax.get_xlim()
   ylim = ax.get_ylim()

   # create grid to evaluate model
   xx = np.linspace(xlim[0], xlim[1], 30)
   yy = np.linspace(ylim[0], ylim[1], 30)
   YY, XX = np.meshgrid(yy, xx)
   xy = np.vstack([XX.ravel(), YY.ravel()]).T
   ZZ = model.predict(xy)

   # plot the boundary
   ax.contour(XX, YY, ZZ.reshape(XX.shape), colors='k', levels=[0.5], alpha=0.5, linestyles=['--'])

   plt.show(block=False) # to force visualization

   # plot the classification map
   plt.figure()
   plt.imshow(ZZ.reshape(XX.shape).T, origin='lower', extent=(xlim[0], xlim[1], ylim[0], ylim[1]),
       cmap='RdYlGn')
   plt.colorbar()
   plt.plot(X[w1i,0],X[w1i,1],'+k') # r
   plt.plot(X[w2i,0],X[w2i,1],'+w') # g
   plt.xlabel('x1')
   plt.ylabel('x2')
   plt.axis('equal')
   plt.title('classification map')
   plt.show(block=False) # to force visualization
```