Faculty of Sciences - Tetouan
Mathematics Department
Pr. Y. El Yazidi

In this tutorial, you will build a neural network with a single hidden layer for a classification task using the Iris dataset, where we distinguish three roses, Iris Setosa, Versicolour and Iris Virginica. This means the output will have three features:

- The probability of the sample being Iris Setosa (Class 0),
- The probability of the sample being Iris Versicolour (Class 1),
- The probability of the sample being Iris Virginica (Class 2).

Our goal is to experiment different activation, loss functions and analyze the results. First, upload the **Iris dataset** from `sklearn.datasets`. Split your dataset into two part, 70% for the training phase and the rest for testing your model.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

data = load_iris()    # Load Iris dataset
X, y = data.data, data.target

# One-hot encode the target labels
encoder = OneHotEncoder(sparse=False) # Converts categorical labels into a format
  that is compatible with neural networks.
Y = encoder.fit_transform(y.reshape(-1, 1))

# Split the dataset into training and test sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
  random_state=42)
# random_state is useful for debugging, sharing code, and ensures that the
  comparison is fair because the data splits are identical.
```

1. In this part, we set the loss function to the Mean Square Error, gradient descent for the back-propagation process, we choose the accuracy metric and we will run our model for 100 iterations.

   - In the hidden layer we can consider `sigmoid`, `tanh` and `relu`.
   - In the output layer we can consider `sigmoid` and `Softmax`.

   We have 6 cases to investigate. Plot the loss decay for each case in a single figure, and print the accuracy. Discuss the obtained results.

2. Now, consider the best choices for the activation function for the hidden layer and for the output layer. We will test two different cost functions, `Cross-Entropy` and `Hinge` function. Train and test you model for this two loss functions and then plot the loss decay for each case in a single figure, and print the accuracy. Discuss the obtained results.

3. Again, we consider the best configuration concluded in last questions and we seek the best optimizer. We will test RMSProp and Adam optimizers to train your model. Similar to last question, Turn you model for this two optimizers, plot the loss decay for each case in a single figure, and print the accuracy. Discuss the obtained results.

4. Now, based on concluded configuration to our model, choose a good metric for evaluating the performance.

5. To enhance our learning, we split our data into three parts, 70% for the training phase, 15% for the validation and the rest for the test phase. Perform this operation and make some conclusion.

```
X_train, X_val_test, Y_train, Y_val_test = train_test_split(X, Y, test_size
  =0.3, random_state=42)
X_val, X_test, Y_val, Y_test = train_test_split(X_val_test, Y_val_test,
  test_size=0.5, random_state=42)
```

6. Now, we import the following

```
from sklearn.model_selection import cross_val_score
```

   it will allow us to use the Cross-Validation techniques and compare the obtained scores. Perform this task for different number of folds from your choice.

What you should deliver: a single Python notebook contain all the parts and it must be well commented. The deadline for this tutorial is **Jan. 31th 2025**.

PS. To create your model you are advised to use from the tensorflow library. Auto-generated notebooks with AI are detectable.