**GALWAY-MAYO INSTITUTE OF TECHNOLOGY**

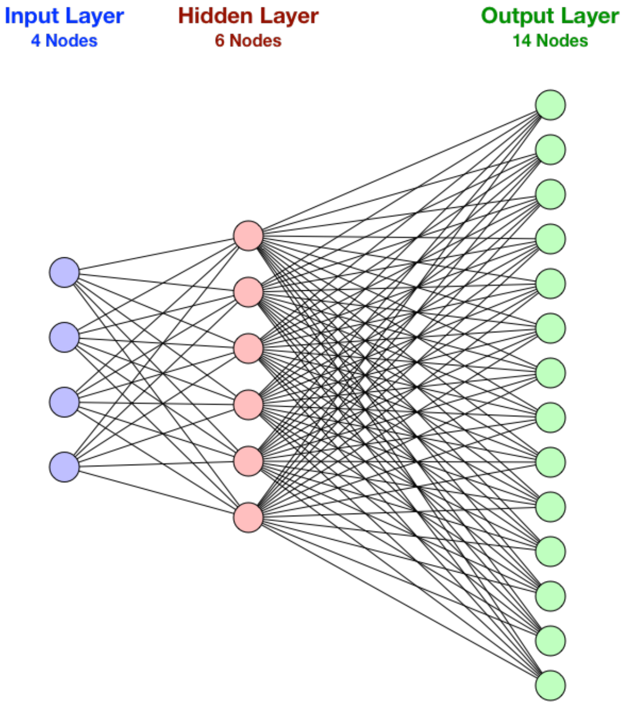
***Department of Computer Science & Applied Physics***

# ***A Pattern Matching Neural Network: Identifying a Binary Signal***

## Overview

In this example, we will create a neural network to classify a stream of 4 bits into one of 14 different categories. For example, the input vector {1, 1, 1, 0}, representing a binary signal should map to "Category 1". The binary signal {1, 1, 1, 0} could be an instruction to close a door or start and alarm. In order to achieve this goal, a total of 4 nodes (one for each bit) are required in the input layer and 14 nodes (one for each category) in the output layer. The number of nodes in the hidden layer is computed as *#hidden-layer < 2 * #input-layer*.

The training data consists of a mapping of inputs, e.g. {1, 1, 1, 0} to outputs, e.g. {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}. Note that, as the data in question represents a binary stream of inputs and expected outputs, it does not need to be normalized.




## Exercises

- ***Create a new class called SignalRunner*** and add a ***main()*** method. Inside ***main()***, add the declarations for the 2D arrays ***data*** and ***expected*** from the file **signals.txt** in the Zip archive **aiNeuralNetData.zip**.

- ***Create an instance of the class NeuralNetwork*** with 4 input nodes, 6 hidden nodes, 14 output node and a sigmoidal activation function:

```
NeuralNetwork nn = new NeuralNetwork(Activator.ActivationFunction.Sigmoid, 4, 6, 14);
```

- ***Instantiate the back-propagation training algorithm*** and ask it to train the network with the training data, a learning rate of 0.2 and a maximum of 500 epochs.

```
BackpropagationTrainer trainer = new BackpropagationTrainer(nn);
trainer.train(data, expected, 0.2, 500);
```

- ***Create the following data set*** required to test if the network is fully trained:

```
double[] test = {1, 1, 0, 1};
double[] result = nn.process(test);
System.out.println(Utils.getMaxIndex(result) + 1);
```

- ***Add some "noise" to one of the signals*** by changing one of the bits in the test array. If the network is stable, it should be able to accommodate a certain level or noise without error.

- ***Experiment with changing the number of nodes in the hidden*** layer and examine the impact that this has on the robustness and stability of the neural network. The following formulae are all valid mechanisms for computing the size of the hidden layer. Note that the value ***alpha*** is a scaling factor in the range [2..10]:

  #hidden-layer = #input-layer + #output-layer
  #hidden-layer = (#input-layer * 0.66) + #output-layer
  #hidden-layer = sqrt(#input-layer * #output-layer)
  #hidden-layer = (number of trainin samples) / (**alpha** * (#input-layer + #output-layer))