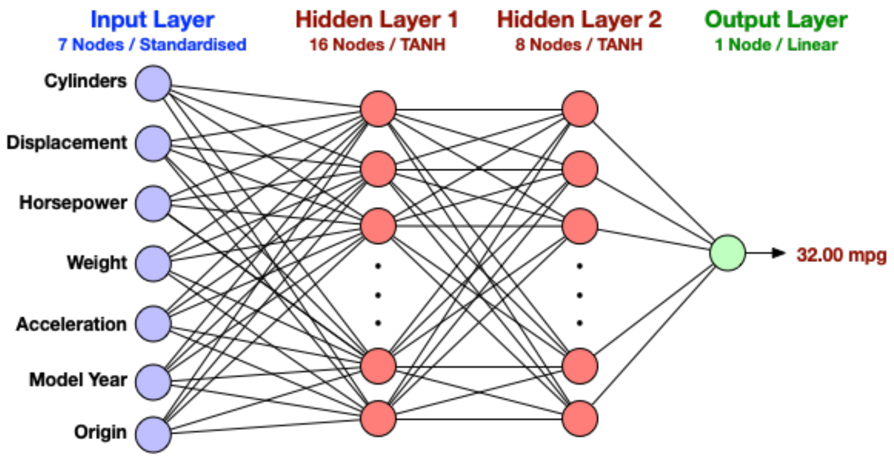
Department of Computer Science & Applied Physics

Using Regression to Predict the MPG of a Car

The *Auto MPG Dataset* is from the *StatLib* library at Carnegie Mellon University and was used in the 1983 American Statistical Association Exposition. The dataset contains 398 instances of car features that can be used to predict miles per gallon (US gallons) using regression. The dataset contains features such as number of cylinders and weight which have very different ranges. Consequently, the dataset will need to be normalised or standardised before it can be used to train a neural network.



In this practical, we will use the network topology shown above to predict the MPG of a car based on 7 features. The network has two hidden layers and, as the problem relates to regression and not classification, just one output node is needed.

Exercises

- *Create a new class called* ***MPGRunner*** and add the declarations for the 2D arrays *data* and *expected* from the file ***AutoMPG.txt***.

- Standardise the data set using the following statement:
```
Aicme4jUtils.standardise(data);
```

- Use the class *NetworkBuilder* to create a neural network with the following configuration (there are two hidden layers):
  - **Input Layer:** 7
  - **Hidden Layer 1:** 16 (TahH)
  - **Hidden Layer 2:** 8 (TahH)
  - **Output Layer:** 1 (Linear)
  - **Alpha:** 0.001
  - **Beta:** 0.95
  - **Epochs:** 1000000
  - **Min Error:** 0.000000001
  - **Loss Function:** Mean Squared Error (L2 Loss)

- Test if the network is fully trained using the following. Note that, although most of the predictions are labelled as errors, they are within range of the expected data.
```
for (int i = 0; i < data.length; i++) {
   long predicted = round(net.process(data[i], Output.NUMERIC));
   long actual = round(expected[i][0]);
   out.print(predicted + "==" + actual + "\t");
   out.println(actual == predicted ? "[OK]" : "[Error]");
}
```

- Change the network topology to a single hidden layer with a ReLU function as follows and then re-train and test the network:
```
.hiddenLayer("Hidden1", Activation.RELU, 5)
```

  What difference, if any, did the second hidden layer make to the accuracy of the network?

- Change the loss function from **MSE (Mean Squared Error)** to **MAE (Mean Absolute Error)** and then retrain and test the network. Do the same for the **SSE (Sum of Squared Errors)** loss function. What effect, if any, does the loss function have on the training time and predictive accuracy of the network?