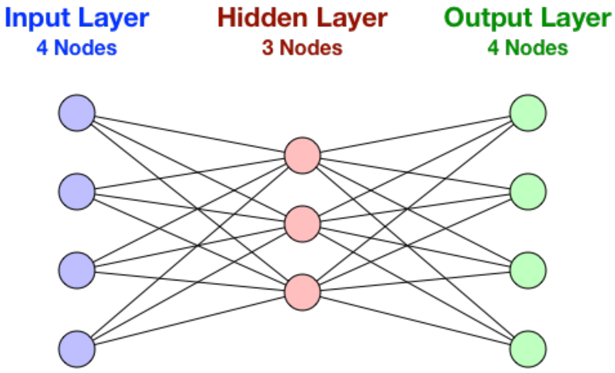
ATU Department of Computer Science & Applied Physics

Controlling a Game Character Action Using a Neural Network

Overview

In this practical, we will use a neural network to control a game character's actions given a set of different input attributes.



The input layer contains 4 nodes that map an input vector to the following attributes.

1. **Health** (2 = Healthy, 1 = Minor Injuries, 0 = Serious Injuries)
2. Has a **Sword** (1 = Yes, 0 = No)
3. Has a **Gun** (1 = Yes, 0 = No)
4. Number of **Enemies**

For example, the input vector {2, 0, 1, 2} means "healthy, no sword, has a gun and two enemies". The "number of enemies" input value will have to be normalized if its range is large relative to the other input values. In such a case, some nodes in a neural network will have a large impact on the final result, i.e. some parts of a network can be more biased than others. The neural network should map the set of inputs to the output vector {0, 0, 1, 0} that relates to one of the following categories of action, i.e. "Hide".

1. Panic
2. Attack
3. Hide
4. Run

Each action is represented in the neural network by one of the four nodes in the output layer.

Exercises

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

- Use the class NetworkBuilder to create a neural network with the following configuration:
    - **Input Layer:** 4
    - **Hidden Layer:** 2 (TANH)
    - **Output Layer:** 4 (TANH)
    - **Alpha:** 0.01
    - **Beta:** 0.95
    - **Epochs:** 100000
    - **Min Error:** 0.00001
    - **Loss Function:** Sum of Squares

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

```
double[] test1 = {0, 1, 0, 1}; //Panic
out.println("2,0,1, 1=>" + net.process(test1, Output.LABEL_INDEX));
```

- ***Progressively increase the "number of enemies"*** in the test data by an order of magnitude from {1, 10, 100, 1000, 10000} and examine the output and stability of the network.

- Normalise the data in both the training and test data and examine the result. Use the following syntax to normalise between -1 and 1:

```
Aicme4jUtils.normalise(data, -1, 1);
```

- Progressively change the min and max normalisation values to more extreme values and examine the effect that this has on the ability of the network to learn.

- Standardise the training and testing data using the following syntax and examine the result:

```
Aicme4jUtils.standardise(data);
```

- When and why would you use standardisation instead of normalization?