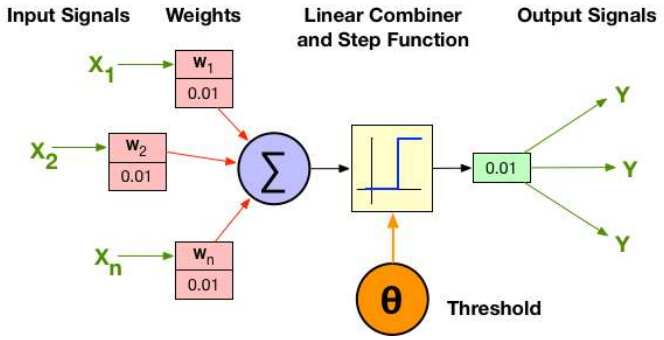
Department of Computer Science & Applied Physics

Training a Perceptron the Logical AND and OR Operations

Overview

Recall that the aim of a Rosenblatt perceptron is to classify inputs and that the model consists of a linear combiner and a hard limiter. The weighted sum of the inputs is applied to the hard limiter to compute an activation. If the activation is at or above a threshold (**θ**), the perceptron fires.



The weighted sum is computed (e.g. with a simple ***for*** loop), and a step activation function is then used as a hard limiter. In the context of Boolean operations, a single-layer perceptron can be trained in a small number of epochs compared to other activation functions, such as the sigmoidal and hyperbolic tangent functions. However, a single-layer perceptron can only classify inputs that are linearly separable. While hidden layers can be added to create a multi-layer perceptron that can potentially classify multi-dimensional and highly complex data, in practice, a sigmoidal function is used instead of a step function to create a **multi-layer neural network**.

Exercises

1. The permutations of the two operands for the logical OR and AND operators can be represented as a 2D array. Add the following declaration to the ***main()*** method:

```
float[][] data = {
   {0.00f, 0.00f},
   {1.00f, 0.00f},
   {0.00f, 1.00f},
   {1.00f, 1.00f}
};
```

2. The following array of floats corresponds to the expected truth values for the **logical AND** operations for each row of operands in the 2D array.

```
float[] expected = {0.00f, 0.00f, 0.00f, 1.00f};
```

   Add this declaration to the ***main()*** method as well.

3. Instantiate a Rosenblatt perceptron with two input variables, including initializing the weights randomly, and then train it with the training data in 10,000 epochs.

```
Perceptron p = new Perceptron(2);
p.train(data, expected, 10000);
```

   Suggested values to adopt:
   - $\theta = 0.2$ (threshold);
   - $\alpha = 0.1$ (learning rate);
   - $(w_1, w_2) \in [-1.0, 1.0]$ (weights).

4. Test the perceptron by iterating through the values in the 2D array of the training data as follows and then visually inspect the results for accuracy.

```
for (int row = 0; row < data.length; row++){
   int result = p.activate(data[row]);
   System.out.println("Result " + row + ": " + result);
}
```

5. Your programme should exhibit the following console output, including the weights' values before and after the training:

```
Perceptron [weights=[-0.39551866, 0.025439024]]
Training complete in 7 epochs.
Perceptron [weights=[0.10448133, 0.12543902]]
Result 0: 0
Result 1: 0
Result 2: 0
Result 3: 1
```

6. For the logical OR operation, change the expected truth values to the following array of data and then train and test the perceptron.

```
float[] expected = {0.00f, 1.00f, 1.00f, 1.00f};
```