Exercise 1 – the Perceptron

In the first exercise you will build an implementation of a Perceptron, test it, and extend it to implement the Perceptron learning rule.

If you finish before the break there are some challenge exercises which will help you explore the strengths and weaknesses of the Perceptron. You might also want to try them once the workshop is over.

You will find a solution provided for the main exercise . Consult the solutions if you get stuck, or ask a friend (or one of the workshop leaders).

Neurons are stateful; they *remember* their weights and biases. For this reason the solution uses an Object-oriented style, implementing the Perceptron as an APL class.

Main exercise:

1. Create a Perceptron class, with a constructor that takes a vector of weights and a scalar bias as arguments.
2. Implement its activation function which should multiply the weights by the input vector, add the bias and fire (output a 1) if the result is greater than zero. If the perceptron does not fire its output should be 0.
3. Test the Perceptron code. With two weights, each 0.6 and a bias of -1.0, your perceptron should output the AND of its (binary) inputs.
4. Extend your perceptron by implementing the Perceptron learning rule. You can initialise the weights and bias to 0.0.

Recall that the learning rule is defined like this:

The goal is to learn a set of targets. You can represent a target as (input_vector expected_output)
For a given target, you will need to

1. find the actual_output
2. calculate the error (expected_output - actual_output)
3. add the error to the bias
4. multiply the error by the input and add to the weights

See if you can train your Perceptron to learn to implement the NAND function.

The Perceptron is unlikely to learn all the targets right away, so you will need to repeat the training until nothing changes or you run out of patience.

Not every set of targets can be learned by a Perceptron, so you need a way of stopping if the Perceptron is still changing after lots of training.

Challenge Exercises

1. Try starting with non-zero weights and biases when training the Perceptron to learn the NAND function.
2. Extend your implementation so that it can calculate the outputs for a fully-connected layer containing several Perceptrons. The layer will have a matrix of weights, with one row per Perceptron, and a vector of biases. Each Perceptron will get the same inputs; each will produce an output determined by its weights and bias. You could train the layer so that one Perceptron learns the OR function, while another learns the AND function.
3. Extend your implementation so that it can calculate the outputs for a matrix containing multiple inputs. Extend the learning rule so that it learns a batch of targets at a time.
4. Try generating some random targets. See if the Perceptron can learn them.
5. See if you can train your Perceptron to implement the XOR function. If not, why not?