Pulse-width Modulated Artificial Neural Networks for Engineering Systems

MacLeod, Christopher

Publication date:
2019

Pulse-width Modulated Artificial Neural Networks for Engineering Systems

Pulse Modulated Neurons have been suggested as alternatives to non-time-varying units such as Perceptrons and to complex biologically inspired spiking models. In particular, Pulse Width Modulated (PWM), Pulse Frequency Modulated (PFM) and Pulse Position Modulated (PPM) units have been investigated in a variety of problem domains [1 – 3]. They were shown to be especially useful in legged robotic control.

This report restricts itself to a discussion of the Pulse Width Modulated Unit (PWMU). This is for two primary reasons: Firstly, it can perform many of the tasks of the other two types. Secondly, it is particularly useful in the control of motors and similar actuators. These devices are often specifically designed to accept PWM inputs – and many microcontrollers and processing units are similarly equipped.

The general idea of the PWMU is that it produces an output with a low mark to space ratio at low activations and a high mark to space ratio at high activations. This is shown in figure 1.

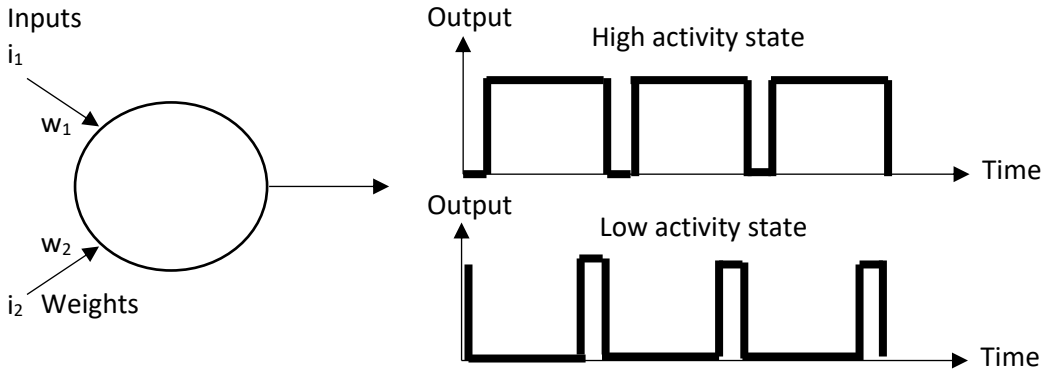

Figure 1, The general principle of the PWMU.

There are several advantages to this approach. As mentioned above, this pulse-width signal is widely used in DC motor, servo and actuator control. Of course it is possible to use a Perceptron type neuron and an amplitude to pulse-width converter. However, using a network of pulsed neurons (because their interaction allows finer control of the system) affords an increased flexibility and more sensitivity of the output dynamics.

Another advantage of this type of network is that it contains a timing aspect that allows good control over the time-domain aspect of the output signal. This is particularly important in robots and other systems that depend on timing critical tasks. It may also prove important in recognising time-dependent signals, although this has not been explored completely.

In a similar way to spiking neurons, the time-changing aspect of their performance may also prove critical in very advanced systems which seek to mimic biological behaviour and ultimately consciousness [4]. These considerations have led to PWMUs being used as a universal unit in advanced modular neural networks [5].

OPERATION AND THEORY

Looking again at the basic unit as shown in figure 1. The inputs may be PWM signals or Perceptron-like amplitude signals (the activation function discussed below can process both types). The weights are simply positive or negative floating-point numbers, as in a Perceptron unit.

The amplitude of the output is fixed at one unit or zero. There are three output timing parameters as indicated by the labels A, B and C in figure 2.

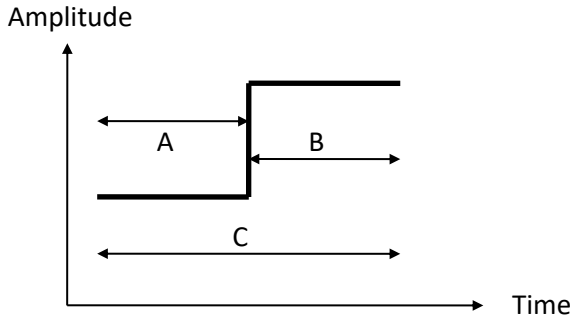

Figure 2, Output from a PWMU showing the three waveform timing parameters.

In the diagram, time-period C is fixed, period B increases with the unit activation and A is complementary to B (as B increases, A decreases). The unit hard-limits when $A = C$ or $B = C$. Mathematically:

$For\ 0 \leq B\ \leq C$

$$B = kJ_t$$

$$A = C - B$$

Where C is the maximum time period of a PWM cycle and is set by the user. k is a constant which relates B to the activity J_t of the neuron (at time t).

And:

$For\ B > C$

$$B = C, A = 0$$

$For\ B < 0$

$$B = 0, A = C$$

Once the unit has been triggered, it cannot be re-triggered until time C (it is effectively switched off when undergoing its cycle). Note that in figures 1 and 3 the unit does not completely saturate – this is an alternative and is shown in these diagrams in order to make the pulse dynamics clearer.

One simple way to calculate the neural activation is to use a Weighted Sum and Leaky Integrator as shown in the equation below:

$$J_t = \left(\sum_{n=1}^{m} i_n w_n \right) + \alpha J_{t-1}$$

Where J_t is the activation at the present time (t), i_n is the n^{th} input to the node, w_n is the n^{th} weight of m total inputs and weights (as in a normal ANN). J_{t-1} is the activation at the previous time step and α is a constant between 0 and 1. The Leaky Integrator [6] term (αJ_{t-1}), is widely used in pulsing networks, to compensate for the absence of input for time periods during the waveform cycle (since it makes the activation depend on previous as well as current inputs).

If it is necessary to convert the PWM signalling into an amplitude modulated output for any reason, this can be done using a simple sigmoidal function as shown in figure 3.

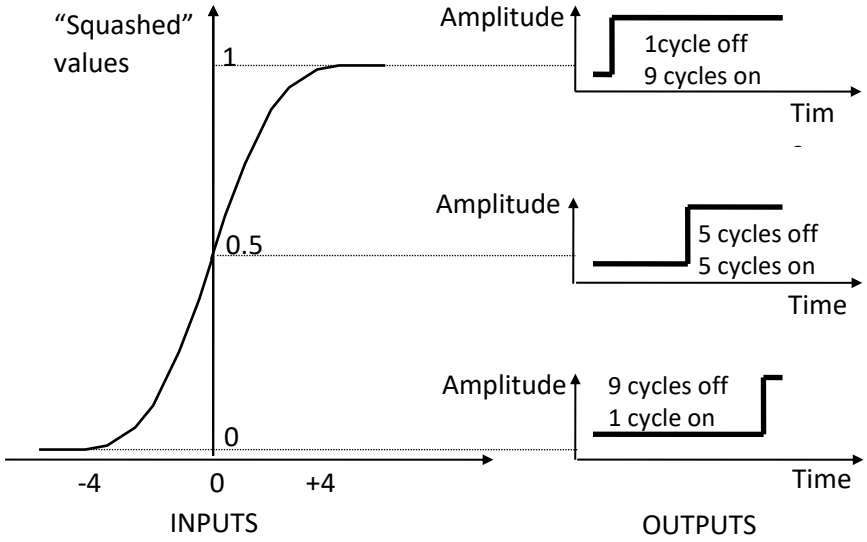

Figure 3, PWM to AM conversion using a sigmoid function.

One disadvantage of using time-dependant units is the slightly more complex coding involved. In particular as each unit goes through its cycle it is necessary to keep track of its temporal position and update the other units concurrently or nearly so. There are several ways of doing this, but the easiest is probably to put all the units in a simple loop, one iteration of which corresponds to a master clock, each unit then updates its own state once per iteration. The idea is shown in figure 4.

```
Main loop:
        Neuron 1:
                If neuron 1 is triggered or in operation then
                        Start (or continue with current) action of neuron 1
                        Increment neuron 1 clock
                If neuron 1 is not triggered or cycle is over then
                        Reset neuron 1 clock
        Neuron 2:
                If neuron 2 is triggered or in operation then
                        Start (or continue with current) action of neuron 2
                        Increment neuron 2 clock
                If neuron 1 is not triggered or cycle is over then
                        Reset neuron 2 clock
        .
        .
        .
        .

        Neuron n:
                If neuron n is triggered or in operation then
                        Start (or continue with current) action of neuron n
                        Increment neuron n clock
                If neuron n is not triggered or cycle is over then
                        Reset neuron n clock
End loop
```

Figure 4, simple operation of a temporally varying network.

There are two simple ways of calculating the error from this type of network for use in a supervised training algorithm. The first is to use the graph in figure 3 to convert the current network output into a number and then subtract this from the target value. This is crude however and rather defeats the purpose of having fine control over the network dynamics as it only matches total mark-to-space ratio.

The second, more useful method, is to allocate a “1” error to each system clock cycle where the output waveform does not match the target and a “0” where it does. The error for all the cycles under consideration may then be added together to give the total error or divided by the total number of cycles under consideration to give a normalised error. Figure 5 shows an example of this idea graphically.

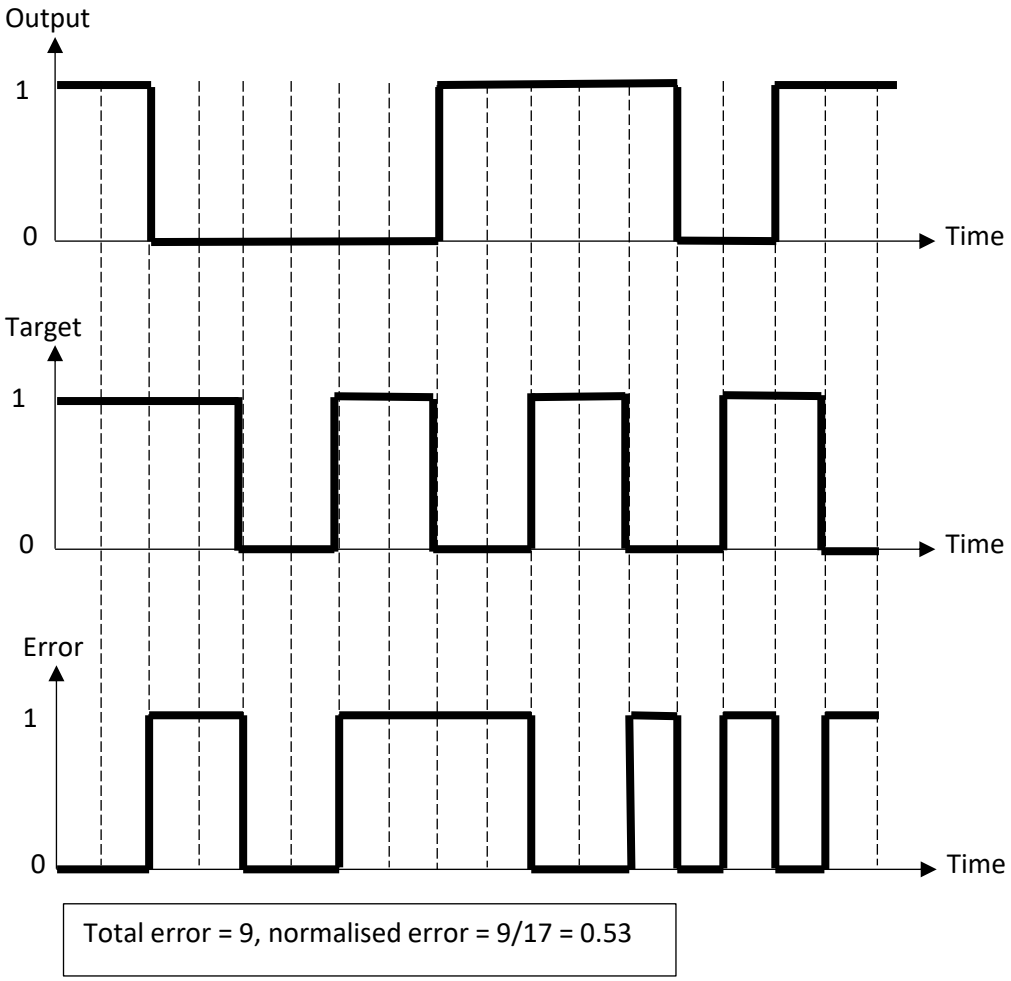

Figure 5, calculating error by clock cycle.

In previous work [1 – 3] it was shown that PWMUs could be used as pattern recognisers with the same performance (and capabilities) as standard perceptron units (allowing for the discretization of the output into time-steps) and trained for this application using both Backpropagation (BP) and Evolutionary Algorithms (EAs). It was also shown that, unlike a standard perceptron network, they could be trained to produce complex timing and sequencing signals.

In the case of Back Propagation networks the parameters C, k and α where manually set for the network. In the Evolutionary Algorithms these are set by the algorithm (they were part of the string or chromosome).

References

1. Niccolo Capanni, Christopher MacLeod, Grant Maxwell, William Clayton, Artificial Biochemical Networks, CIMCA'2005, Vienna, Austria. 2005.
2. Niccolo Capanni, The functionality of spatial and time domain artificial neural models , PhD Thesis, The Robert Gordon University, Aberdeen, 2006.
3. Niccolo Capanni, Christopher MacLeod "Artificial biochemical networks: a different connectionist paradigm", Artificial Intelligence Review, 33 (1/2), 2010. pp. 123 – 134.
4. Hopkin, K., 1996. dot dot dot dash dash dash, New Scientist, vol 150, 2030. pp 40 – 43.
5. C. MacLeod, G. M. Maxwell and S. Muthuraman, Incremental Growth in Modular Neural Networks, Engineering Applications of Artificial Intelligence, Vol 22, Issue 4/5, 2009. pp 600 – 666, doi:10.1016/j.engappai.2008.11.002.
6. Kevin Gurney, An introduction to neural networks, University College London Press, 1997. pp. 20-24.