Pulse Based Input for Echo State Neural Networks
Bachelor Project

P.G.J. Smit, s1617346@student.rug.nl

April 5, 2011

Abstract

Most papers demonstrating an Echo State Neural Network (ESNN) use a scale measure input to control the output. In this paper we describe and show the results for 6 different scenarios to find out what pulse-based amplitude modulation method is best. In these scenarios we vary the information offered on the pulse amplitude and the pulse width. Is information coded on these two independent variables processed with the same error and optimal parameters? Our results show that information on the pulse amplitude yields a lower error compared to information on the pulse width.

1 Introduction

One problem with ESNN encountered in an earlier study (van de Sanden (2010)) was that network activity fades over time. What begins as a perfect gait for a biped robot soon ends in a halt. One solution is the one tested in this paper, modulation. If the goal is to make the robot walk at a steady pace the input into the ESN would normally be a stable value corresponding with that speed. This provides no extra activity into the network as long as the desired speed remains the same. Using amplitude modulation every desired step results in a pulse on the input nodes, giving the whole network a thrust of activity. In this paper we focus on coding information on the width and height of that pulse.

1.1 Echo State Neural Networks

Echo State Neural Networks (Jaeger (2001)) are recurrent neural networks that have sparse connections and lack ordering in the hidden layer or 'Dynamic reservoir. See figure 1 for an illustration. The training of this type of network is in the adjustment of the connections to the output neuron, the rest of the connections are generated randomly and are not adjusted during training. It is difficult to find the best parameters for the dynamic reservoir as they seem different for each task. We use the Monte Carlo algorithm (Metropolis and Ulam (1949)) combined with a sweep as technique for finding the best combination of parameters.

In this experiment leaky integrator neurons are used; every time step a bit of activity from the last time step remains in the neuron. Leaky-integrator ESNs are only slightly more complicated to implement and use than standard ESNs and appear to us as quite flexible devices when timescale phenomena are involved (Jaeger, Lukosevicius, Popovici, and Siewert (2007)).

1.2 Pulse-based Input

The usage of a pulse based input is not an entirely new phenomenon. At least two earlier studies used a pulse to feed information into the network. We will discuss these before explaining the difference with our own experiment.

1.2.1 Timer experiment

An experiment with a timer is described in Jaeger (2002). The ESNN in this experiment has two input neurons: one to express a 'go' signal, one to express the length of the timer. If no input is offered the output neuron will stay close to 0. When the network receives a pulse on the first input the activity of the output neuron shoots to a high equilibrium and stays there until an amount of time has passed. The second input controls the amount of time the output neuron stays high. A higher input leads to

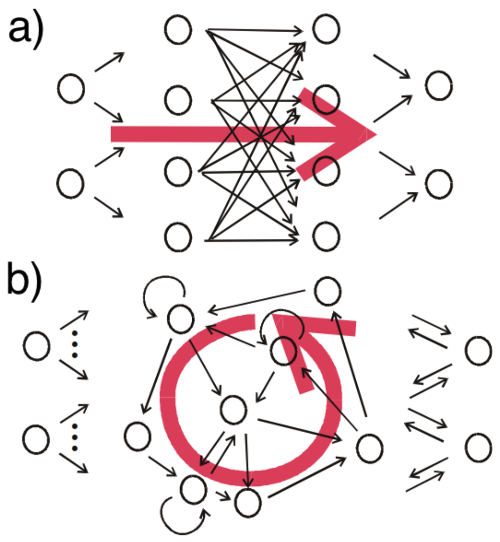

Figure 1: Two common types of neural networks. The network a) illustrates a regular multilayer perceptron. Notice the ordering of layers, in this case two layers. Network b) illustrates a typical recurrent network, like an echo state network. Image cited from Jaeger (2002)

a longer period the output stays high. Thus mimicking a timer.

1.2.2 Memory experiment

An experiment with four bits memory is described in Sussillo and Abbott (2009). The ESNN in this experiment has eight input neurons and four output neurons. Each output neuron has two input neurons paired with it through learning. If a pulse is offered at the first of the two the output neuron rises to a high equilibrium. If a pulse is offered at the second of the two the output neuron will drop to a low equilibrium. When no input is offered the output neuron will move about the equilibrium it was last set to.

1.3 This study

These two studies used the pulse only to give one type of information; either timing or a status change. We will explore how a pulse can be used for manipulating a sine output. A pulse has two independent variables: the height or amplitude and the width or length. We will first find out if they are both equally effective to convey information, then we will use them concurrently.

2 Method

In our experiment we will try to use the two variables of a pulse, height (H) and width (W) to produce a sine wave with a amplitude (A) and a period (P). We have constructed six different scenarios paired in three tests. In each test we will compare a H to A and W to P relation situation versus a H to P and W to A relation. Figure 2 shows an example for all scenarios.

HonP: Pulse height codes for the desired period of the sine wave. Pulse width and the desired amplitude of the sine wave remain constant.

AonP: Pulse width codes for the desired period of the sine wave. Pulse height and the desired amplitude of the sine wave remain constant.

AonW: Pulse height codes for the desired amplitude of the sine wave. Pulse width and the desired deriod of the sine wave remain constant.

HonW: Pulse width codes for the desired amplitude of the sine wave. Pulse height and the desired period of the sine wave remain constant.

HonP+AonW: Pulse height codes for the desired period of the sine wave. Pulse width codes for the desired amplitude of the sine wave.

AonP+HonW: Pulse width codes for the desired period of the sine wave. Pulse height codes for the desired amplitude of the sine wave.

We will compare the ten best scores with a t-test for the pairs 1 and 2, 3 and 4, 5 and 6, to see if there is a significant difference between the two scenarios. The reason to use only the best ten is because in a situation where our method is used (for example to control a locomotion in a robot) only the best parameter combinations will examined further. We consider the ten best results to be representative of this.

2.1 Network Parameters

To find the best parameters for each scenario we perform a sweep on a pre-determined interval The tested parameters are: the amount of neurons in the reservoir ($10 < n < 100$, $\Delta n = 5$), the chance of connection between neurons in the reservoir ($0.01 < c < 0.1$, $\Delta c = 0.01$) and the amount of activity carried to the next time step ($0.1 < l < 0.4$, $\Delta l = 0.05$). For each step in the sweep we perform a run of the monte carlo alogrithm with 20 child nodes to find the best solution in that neighborhood. Each combination is tested three times, only the lowest error will be logged.

The input neuron is connected to one in every ten neurons in the reservoir with a uniform random weight. The output neuron is connected to every neuron in the reservoir, weights from the reservoir to the output clamps to zero during training and computed by the training algorithm after training, weights from the output neuron to the reservoir are initialized uniformly random.

Algorithm 2.1 Compute best result

```
for $nIndex = 10$ to $100$ step $5$ do
  for $cIndex = 0.01$ to $0.1$ step $0.01$ do
    for $lIndex = 0.1$ to $0.4$ step $0.05$ do
      $n \leftarrow nIndex$
      $c \leftarrow cIndex$
      $l \leftarrow lIndex$
      $bestError \leftarrow Infinite$
      for $i = 1$ to $20$ do
        for $j = 1$ to $3$ do
          $Error \leftarrow ESN(n, c, l)$
          if $Error$ <bestError then
            $bestError \leftarrow Error$
            $nBest \leftarrow n$
            $cBest \leftarrow c$
            $lBest \leftarrow l$
          end if
        end for
        $n \leftarrow nBest + discreteNoise(-3, 3)$
        $c \leftarrow cBest + intervalNoise(-0.01, 0.01)$
        $l \leftarrow lBest + intervalNoise(-0.02, 0.02)$
      end for
      $logfile \leftarrow$ all variables
    end for
  end for
end for
```

Algorithm 2.1 shows our optimization script. The three parameters varied are n: the amount of neurons in the reservoir, c: the chance of connection between two neurons in the reservoir and l the percentage of activation in a neuron that has not been retained from the previous time step. Because of the random initialization of the network every run yields a different result. Each parameter combination is tested three times to ensure we do not miss a good combination. Twenty times we will look for a better combination right next to the starting point. The amount of noise is about one step of the sweep, this way the script will not stray away too far from the beginning.

To measure the error we use the mean squared error function described in equation (2.1). Where d is the desired output signal, y is the output from the network and L is the amount of time steps in the signal.

$$MSE = 1/L \cdot \sum_{i=1}^{L} (d_i - y_i)^2 \tag{2.1}$$

The formula below is the update rule for the activation of the neurons in the reservoir. $x[n]$ is the vector with the activation of the neurons on time step n, $t[n]$ the the activation of the input neuron on time step n, $d[n]$ the activation the output neuron on time step n.

$$\begin{aligned} x[n] &= (1 - l) \cdot x[n-1] + l \cdot tanh(W_{in} \cdot t[n] + \\ &\quad W_{dr} \cdot x[n-1] + W_{back} \cdot d[n-1]) \end{aligned}$$

2.2 Pulse creation

The input and desired output are constructed in blocks, these blocks are merged until a signal with more than 2000 time steps has been created. Then 25 empty time steps are added to the beginning of the desired output and added to the end of the input signal to create a delay slightly longer than the longest possible pulse. The formula used for creating the output is (2.2). Where A is the amplitude of the sine wave ($0.2 < A < 0.8$ uniform randomly chosen or $A = 0.8$ in the fixed amplitude scenarios) and P the amount of time steps the wave is long ($P = 120 \vee P = 180 \vee P = 240$ uniform randomly chosen or $P = 120$ when in the fixed period scenarios).

$$Output = A \cdot sin(2 \cdot \pi \cdot P) \qquad (2.2)$$

The input signal is created using the following steps:

1. Create a empty vector as long as the corresponding output.
2. Determine the desired height H. For the P to H scenarios this is $H = P/240$. For the A to H scenarios the is $H = A/0.8$.
3. Determine the desired width W. For the P to H scenarios is is $W = 0.1 \cdot P$ $W = 30 \cdot A$. Then round W to the nearest integer.
4. Set the first W values for the vector to the value H, the rest to zero.

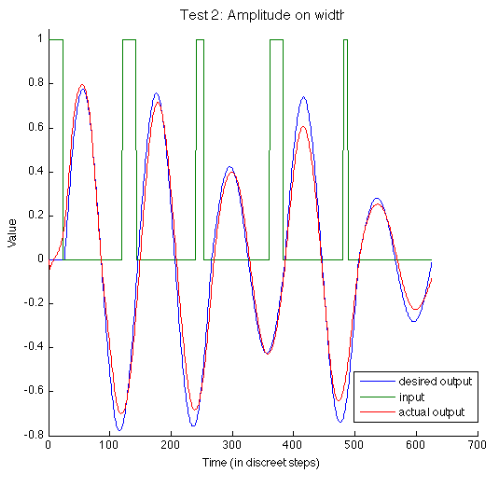

Figure 3: An example output (red) with the corresponding input used in this experiment (green) and the desired output (blue) used in other experiments. The mse for this run was 0.005. For illustrative purposes the length of this example is only 625 time steps long, while testing we used approximately 2000 time steps

3 Results

3.1 HonP versus AonP

The t-test for the best 10 results show that there is a significant difference in MSE between the two scenarios ($p = 5.0742 \cdot 10^{-4}$). There is no significance between the amount of neurons in the reservoir ($p = 0.18$). Period on height having the lowest error on average ($MSE = 0.03$ on the best run). All (P on H) or most (P on W, 8) of the ten lowest errors are encountered with equal or less than 30 neurons in the reservoir. Out of the ten best only the ones with less than 15 neurons in the reservoir have a chance of connection with another neuron higher than 0.02 (3 out of 10 for both scenarios).

3.2 AonW versus HonW

The t-test for the best 10 results show that there is a significant difference in MSE between the two scenarios ($p = 0.0033$). There is no significance between the amount of neurons in the reservoir ($p = 0.12$). Amplitude on height having the lowest error ($MSE = 0.0003$ on the best run). In this case, all of the ten lowest errors are encountered with equal or more than 30 neurons in the reservoir. The amount of activation carried over to the next time step is almost always under 20% (10 out of 10 for A on H, 8 out of 10 for A on W).

3.3 HonP+AonW versus AonP+HonW

The t-test for the best 10 results show that there is a significant difference in MSE between the two scenarios ($p = 5.2785 \cdot 10^{-9}$). Period on width, amplitude on height sporting the lowest error ($MSE = 0.0004$ on the best run). Most (9 out of 10) it's solutions have more than 50 neurons in the reservoir. While amplitude on width, period on height all have less than 50 neurons. This difference is significant ($p = 0.0014$).

4 Discussion

We expect that the scenarios with modulation on the pulse width will have a higher error or will have

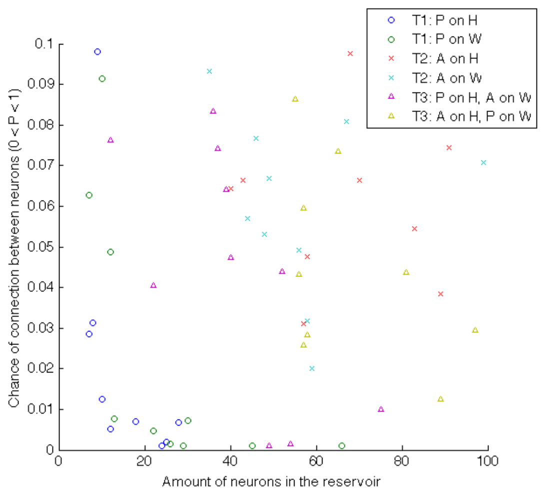

Figure 4: Showing the ten best parameter combinations for both scenarios of each test. The amount of neurons in the dynamic reservoir on the x-axis, the chance of connection between neurons in the reservoir on the y-axis.

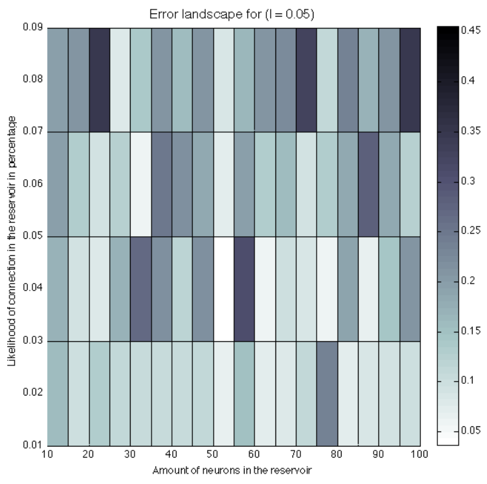

Figure 6: The error landscape for scenario 5 with $l = 0.05$. Lighter areas have lower error than darker areas.

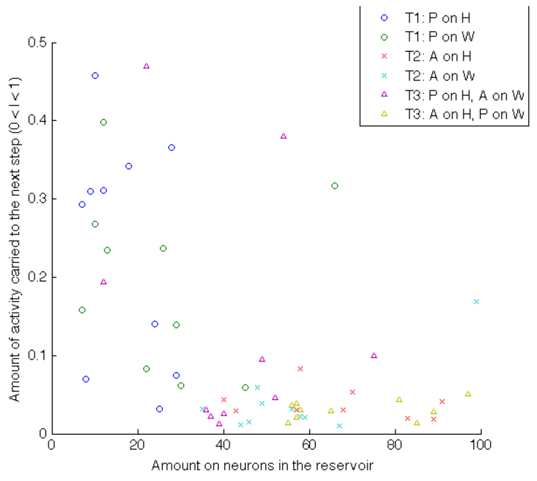

Figure 5: Showing the ten best parameter combinations for both scenarios of each test. The amount of neurons in the dynamic reservoir on the x-axis, the percentage of activation in the neuron carried over to the next time step on the y-axis.

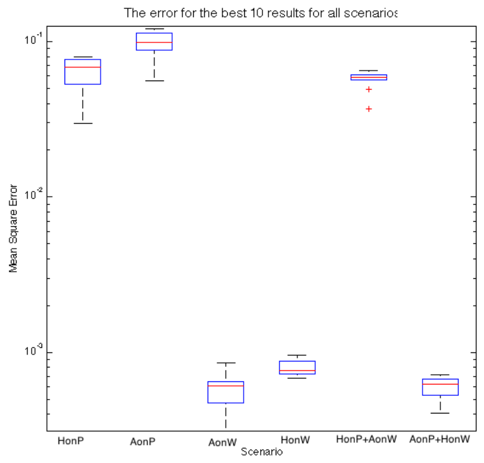

Figure 7: The error for the best 10 results for each scenario. Note that the y-axis (Mean Square Error) is logarthimic.

solutions with more neurons compared to the scenarios with pulse height modulation. Extracting information from the width of a pulse is a more difficult task than from the height of a pulse. Pulse height offers it's value for an number of time steps. For extracting pulse width information the network has to count the number of time steps the input signal is active. So not only is more recollection involved, the information is also offered only once.

In line with our expectations is that if information is offered to the system, information on the pulse height yields lower errors compared to information on pulse width (tests 1 and 2). Combining that with the results of test 3 it is clear that period information is harder to process than amplitude information of the sine wave.

In test 1 the best solutions have a reservoir smaller than 30 neurons. One explanation for this could be that more neurons lead to overfitting. In test 2 and 3 the best solutions use more neurons in the reservoir. Test 3 utilizes more information so it would be logical if the best solutions use more neurons. For test 2 we do not have an explanation. It could also be that there is a complex relationship between the other parameters that we have not found out about yet.

The idea for this experiment came from van de Sanden (2010). A lot of time was spend on his code but it sadly it failed to run. In the experiment van de Sanden used a ESNN to control biped locomotion. This would be an excellent way to test our own tests: error values only give so much information, actually see how a robot would respond would give them much more meaning. An error resulting in a slightly faster gait is no serious problem but an error destabilizing the gait would be disastrous.

In the article described in 1.2.2 (Sussillo and Abbott (2009)) a new way of training a ESNN is explained, called FORCE learning. In our experiments we calculate the weights from the neurons in the reservoir to the output neuron using the pseudo inverse of the activity during training, all other weights are initialized random and stay the same. FORCE learning has a few scenarios, in the most extreme version it alters all weights in the reservoir and to the output during training. It does not directly use the desired output signal but the the error between the desired and the produced output. This method is closer to being biological plausible. However, we could not produce a working implementation of our experiments using this training algorithm.

Due to time constrains we did not compare the pulse method to the regular input method. The regular input would probably yield lower errors. The height and width of a pulse must be stored in the reservoir somehow, with the normal input this would not be necessary. Our third test would be impossible to do without a second input neuron; basically eliminating the difficulty of the test.

Further research is also needed to find out the effects of unexpected input signals. Consider the situation where two pulses are offered right after each other: would the network add the resulting sine waves on the output neuron?

This research shows that pulse modulation is a possible way to offer input to an ESNN. We use only one modulation technique: one pulse per sine wave. Situations with multiple pulses per wave are another modulation technique that should be explored further. This would give the power to modify a wave mid-cycle.

References

Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks. GMD report 148, German National Research Center for Information Technology, 2001.

Herbert Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology, 2002.

Herbert Jaeger, Mantas Lukosevicius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335 – 352, 2007.

N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 247(44):335–341, 1949.

David Sussillo and L.F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63:544–557, august 2009.

Mart van de Sanden. Biped locomotion. In review, December 2010.

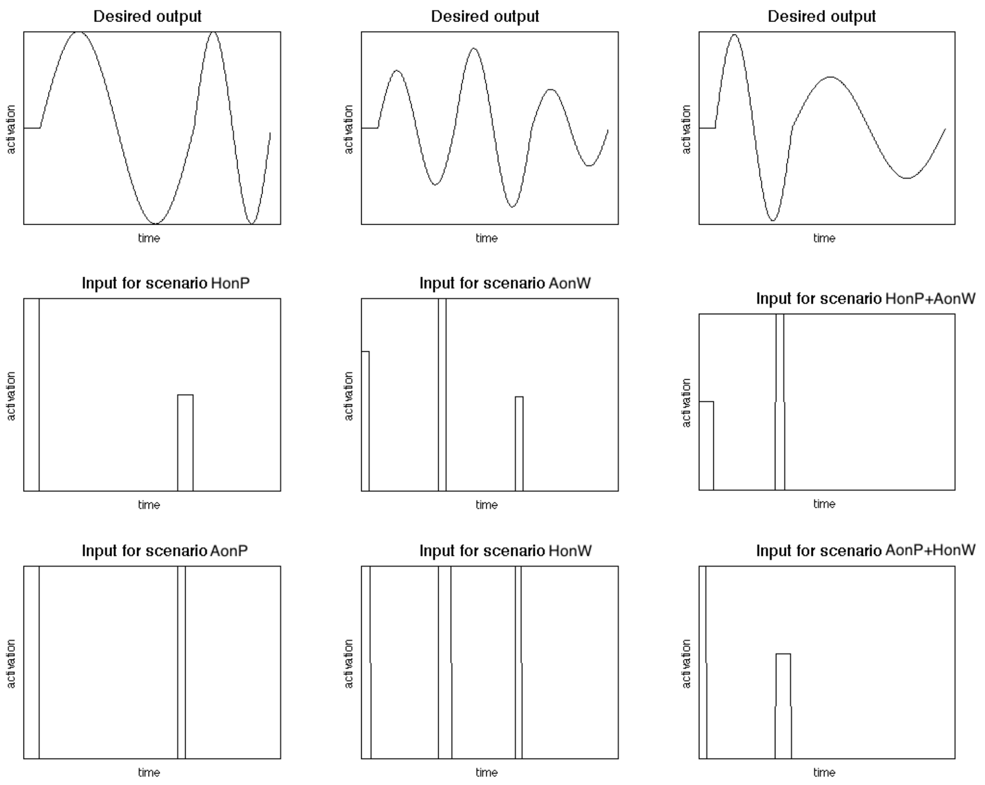

Figure 2: Example output (top row) with the corresponding input used in this experiment (second and third rows). In the first column the amplitude is fixed (corresponding with scenarios HonP and AonP), in the second column the frequency is fixed (corresponding with scenarios AonW and HonW), in the third column both are varied (corresponding with scenarios HonP+AonW and AonP+HonW). For illustrative purposes the length of this example is approximately 250 time steps long, while testing we used approximately 2000 time steps