REDUCING ENVIRONMENTAL SWINGS WITH A RECURRENT NEURAL NETWORK CONTROL SYSTEM

Sam Skrivan, Dr. Jianna Zhang, Dr. Debra Jusak

Western Washington University, Department of Computer Science
516 High Street | Bellingham, WA 98225
U.S.A.
samsk@microsoft.com
Jianna.Zhang@wwu.edu
Deb.Jusak@wwu.edu

ABSTRACT
Maintaining environmental stability in a dynamic system is often a very important task for a control system. Swings in environmental conditions can be harmful to equipment and cause inefficiencies in the system. These swings are present throughout our lives, in your living room for example, when you set the thermostat to 68 degrees the actual temperature cycles above and below 68 degrees. This project uses a Recurrent Neural Network (RNN) in an Aquarium Control System to reduce the swings in temperature, pH level, and conductivity. The results of testing the online system and simulations show that the RNN does reduce these swings.

KEY WORDS
Intelligent Control Systems, Artificial Intelligence, Recurrent Neural Networks, Programmable Logic Controller, CORBA, and Aquarium Control System

1. INTRODUCTION
There has been a substantial amount of work using Artificial Neural Networks (ANNs) in control systems, both in research [4][8] and industrial applications [9][10]. The demands of control systems, to perform well with complex and changing problems, is well matched with the ability of ANNs to learn and operate over a dynamic non linear problem space.

Early work, before 1990, using ANNs with control systems is surveyed by Antsaklis [4]. Antsaklis puts forward the idea that control system development has always been driven by the need to deal with increasingly complex systems, and to be able to do this with less upfront knowledge of the conditions that the system will actually encounter. Neural Networks operate very well under these requirements. The research of Nguyen and Widrow was discussed which uses neural network to steer a trailer truck as it is backing up to a loading dock. Nguyen and Widrow used a feed-forward multilayered Neural Network which was trained with the Backpropagation algorithm. Kraft and Compagna looked at comparing contemporary control techniques designed using control theory against neural network controllers.

Some more recent work [9] demonstrates the benefit of using Neural Networks to enable a reconfigurable flight control system, by exploiting their ability to identify patterns in a highly non linear environment. Shin and Kim contend that most control systems use a linear model, which performs very well when operating within normal variations. However when operation is outside of these bounds, conventional control systems are not always effective. They show that by assuming a non linear model, neural networks perform well when there are unexpected conditions, such as system failures.

In [10] Neural Networks are used in an elevator supervisory control system. A three layer neural network is used with over 30 inputs. The Neural Network control system is compared against a system using fuzzy rules. The Neural Network system reduces waiting times and avoids the “bunching problem” encountered with other techniques. The authors additionally highlight the ability of neural networks to continuously learn allowing them to adapt to changes in behaviors of the elevator users. The learning of the network enables good short and long term predictions of behavior.

Tools have been developed to enable a wider adoption of neural networks in control systems. In [11] the authors describe a Matlab toolset that assists the user to design and simulate control system with neural networks. The user can choose among several strategies such as direct inverse control, internal model control, nonlinear feed forward, feedback linearization, optimal control, gain scheduling based on instantaneous linearization of neural network models and nonlinear model predictive control. Tools such as these have reduced the technical barriers to entry of using Neural Networks in control systems.

This evolving use of Neural Networks, from early dabbling to more complex research and then to a wider industrial audience, drove the need to evaluate the
effectiveness of these control systems. Chen and Mills [8] developed a useful framework for analyzing the ability of a neural network control system to generalize to previously unseen conditions. They claim that the traditional approach of using an error calculation within the network is not sufficient. Their approach evaluates the success of the system by looking at the outcome of the control system rather than the network error. This is largely because in dynamic systems there is often a feedback loop and the desired outcome of the network is not known. Using this methodology the authors show that a successfully trained network always generalizes “well”.

2. PROBLEM ADDRESSED
The Aquarium Control System (ACS) seeks to maintain an aquarium at a consistent state with regard to temperature, conductivity and pH. The ACS is a Supervisory Control and Data Acquisition (SCADA) system, using a Programmable Logic Controller (PLC) to read the instruments and control the actuators. The instruments that are used are a thermometer, a conductivity analyzer, and a pH analyzer. The actuators of interest are a heater, a fresh water pump and a CO2 doser. The PLC is programmed with logic that polls the instruments and based on the readings and a set of thresholds sets the actuators on or off.

When for example, the thermostat returns a reading of 65 degrees and the threshold is set at 70 degrees, the heater is turned on. The heater begins to heat the water and when the thermostat returns a reading greater than 70 the heater is turned off. Because of the time it takes for the heat to distribute through the water and to register with the thermometer the actual temperature in the aquarium swings above and below the threshold see figure 1. This situation occurs with the conductivity and pH level in the tank as well.

![Figure 1 - Environmental Swings](image_url)

This project attempts to come up with a general solution to reduce these environmental swings. A Recurrent Neural Network is used to control the PLC. The inputs to the network are the readings from the three instruments: temperature, conductivity, and PH. The outputs of the network are the settings for the actuators: heater, fresh water pump, and CO2 doser. The ability of the RNN to adapt to changes in input and to learn the complex interactions of the temperature, conductivity and pH, make it a compelling choice.

One of the primary challenges of the problem is that the thermostat reading alone is not enough information to decide whether the heater should be turned on or off. For example if the input was 79 degrees and the threshold was 80 degrees, the desired output for the heater control is not known. If the temperature is climbing quickly then the best decision may be to turn off the heater, on the other hand if the temperature is falling then one may want to turn the heater on. Two levels of recurrence are added to our neural network to enable “memory” of the recent past. The recurrence in the network provides us with this context by taking the outputs of the hidden layers at time t and using them as inputs to the hidden layers at t+1. In addition the previous inputs of the instruments are used for the inputs of the three other input nodes.

3. PROJECT DESIGN
The Recurrent Neural Network has six inputs. Three are from the instruments: temperature, conductivity and pH level readings. The other three are previous outputs of the network, providing the first aspect of the networks recurrence. There are three output nodes for each of the actuators: heater, fresh water pump, and CO2 injector. The sigmoid activation function is used and an output of greater then .5 signals the system to turn on the activator, and less then .5 turns off the actuator. There is one hidden layer of varying size, and a feedback loop from each of the hidden nodes back to itself, providing the second level of recurrence. The network uses the Backpropagation algorithm in the online phase of the project. Figure 2 shows the architecture of the RNN.

![Figure 2 - RNN Design](image_url)

Looking at the problem one can observe that there is a point on the upward slope of the temperature graph where the system should turn off the heater, for example, before the temperature reaches the threshold (figure 1), thus dampening the curve above the threshold line. The goal of the system is to minimize the swings above and below the threshold which could also be described as minimizing the rate of change of the measured input. Intuitively it makes sense that if the rate of change was increasing; the system should take action to slow it down.
Based on these observations the desired output and the network error are calculated proportional to the slope of the input lines, the higher the rate of change, the more correction that is applied to the network. First the inputs are classified into four quadrants according whether the reading is above or below the threshold, and if it has a positive or negative slope, see figure 3.

In the 1st and 3rd quadrant the desired output is known; either the temperature is hot and getting hotter and the heater should be shut off, or cold and getting colder and the heater should be turned on. However in the 2nd and 4th quadrants, it is apparent that the system should take the opposite action before the threshold is crossed, but when to take that action is not known. In other words it is known that the system should turn off the heater when the temperature is rising but before the reading reaches the threshold. In these cases the network is trained to take the opposite action proportional to the slope. For example, in the 2nd quadrant, where the temperature is falling, the instance error is calculated with the following formula:

$$e_k = O_k - (O_k - O_{0})M$$

where $O_k$ is the k output, M is the slope of the input line, and the expected output is 0 (heater off).

As the rate of change increases, the correction on the network to drive the output to 0 increases. The feedback in the RNN allows the network to encode knowledge of previous states and it starts to exhibit the desired behavior in the 2nd and 4th quadrants.

Using this error calculation, the networks “learns” to shut off the actuator before the threshold is crossed and slows down the rate of change as the reading approaches the threshold. At the transition from quadrant 3 to quadrant 4 the slope is very close to 0 so the error to the network is small and the correction will not cause a premature change to the actuator.

Thus the network is trained to not only look at the input of the instruments but also the context of that reading. Given a reading, the decision to turn on or off the actuator is based on if that reading is rising or falling. This context is encoded in the recurrence of the network.

Eventually, these error calculations will cause the system to creep back down the line and as soon as the readings cross into the fourth quadrant the network turn off the actuator. The second part of the algorithm deals with this, by looking in the past and evaluating the system performance of the recent past with the more distant past. A running total of the estimation of the area above (and below) the threshold is stored, for the previous 30 minutes and the 30 minutes before that (figure 4).

The area under the threshold is compared against the previous area during the last swing, and if it is increasing the expected result for the second and fourth quadrant is switched from off to on or vice versa. This has the effect of moving the system back closer to the threshold before action is taken.

The ACS system has a CORBA server that has methods to access the PLC to read the instruments and control the actuators. This allows for a distributed system that does all of the traditional functions of a SCADA system including, alarms, trends, reports and monitoring. This allows for a more robust control environment with much greater flexibility, enabling modifying or adding capabilities to the system.

The Control System implements a CORBA client to plug into the aquarium system and access the PLC. When it is online it turns off the onboard logic of the PLC. The client makes calls to read the instruments at a parameterized interval and runs these through the network. The outputs are then sent to the plc to control the actuators.

4. APPROACH

 Initialization Phase. When the system starts, the network will be trained on a training set, which uses the thresholds programmed into the PLC. By the end of the initialization the network has been trained to send an off signal to the heater when the temperature crosses above the threshold and an on signal when it drops below the threshold. The same will be true for the fresh water pump, and nitrogen output. Because the actions taken by the control system aren’t affecting the readings the system does not use recurrence. Back propagation is used to train the network. In figure 5 the UI is shown for the initialization phase. The parameters for the system are the number of nodes in the hidden layer and the training weight. The network error is shown for the three inputs and the graphs show the thresholds and the points where the network is taking the opposite action from the threshold logic.
Figure 5 - Initialization Phase

The training data was created based on the sine function to simulate the expected behavior of the system.

**Online Phase.** Once the error of the system on the training set reaches an acceptable level, it will shift to the online phase. During this phase the application will poll the PLC for the actual readings in 30 second intervals. Those readings will be the inputs to the network and the outputs of the network will be sent to the PLC to control the actuators. The error of the system will be calculated based on the difference of the reading with the n-1 reading, for all of the sensors. The correction of the system will be proportional to the difference in the two readings. The system will be continuously learning to reduce the variation in the readings; the rate of change.

In addition a rolling metric is kept to be used in the network training. The metric is an approximation of the area above and below the threshold for the three inputs. The difference of the immediate previous 30 minutes and the 30 minutes preceding that will be calculated (figure 4). If the network performance is declining, interpreted as larger area calculation, the training algorithm will flip the desired outcome in quadrants 2 and 4 (see Figure 3), and when improvement is seen the expected outcome in the 2nd and 4th quadrant will flip back.

The application shows the current reading that was obtained from the instruments, and the current state of the actuators, with green signaling on and red for off.

5. **TESTING**

To test the system data was collected from the aquarium being controlled by the PLC’s internal programming as well as from the aquarium being controlled by the recurrent neural network control system. These datasets were then compared. The goal for the system was to reduce the environmental swings, so the absolute value of the distance above and below the thresholds was used as the system performance.

Data was collected on the system using the PLC’s internal programming for different periods of the day. The data for the system using the threshold logic was captured, turning the actuators on or off when the reading crossed the threshold. The learning system was then turned on and it controlled the aquarium for different periods throughout the day.

6. **RESULTS**

After analyzing the data of the running system, a few observations were made. The instruments were set to capture data with the maximum resolution. This had the effect that in a 5 gallon aquarium the temperature of the water stayed very close to the threshold, within one or two degrees. Additionally since data was being captured at such a high resolution, the readings jumped around quite a bit. This had the effect of obscuring the trend of the readings. To filter out some of this noise, the system took three readings and used the average for the input of the system. This modification was made for the tests without learning as well.

Another challenge for testing was that the learning component was run on a remote machine and the latency between a reading request and a result varied widely. To add some consistency to the results both the control of the system with learning and without learning were run from a remote machine. This had the effect of increasing the latency between consecutive readings to approximately 30 seconds to 1 minute.

Another observation was the pH level stayed very constant under both systems. Additionally controlling the fresh water pump required more than an on or off signal, so it was left out of the testing of the system. Therefore the analysis of the results focused on the temperature.

With any real world application there is a lot of noise that comes into the system, and this was true with the aquarium. To feel more comfortable with our findings additional tests will be run and results will continue to be accumulated.

The sum of the distances between the threshold and the reading was used as the main indicator of system performance. For the same number of data points there was a 21% improvement when the system was using the learning component. The maximum reading was higher without the learning component, while minimum reading was lower with the learning component (see figure 6).

<table>
<thead>
<tr>
<th>Sum of Distance from Threshold</th>
<th>Without Learning</th>
<th>With Learning</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>135.66</td>
<td>107</td>
<td>-21.10%</td>
<td></td>
</tr>
<tr>
<td>Maximum Distance from Threshold</td>
<td>9</td>
<td>10.667</td>
<td>-18.50%</td>
</tr>
<tr>
<td>Maximum Reading</td>
<td>1589</td>
<td>1584</td>
<td>-0.33%</td>
</tr>
<tr>
<td>Minimum Reading</td>
<td>1572</td>
<td>1569.33</td>
<td>-0.18%</td>
</tr>
</tbody>
</table>

Figure 6 – Test Results

Figure 7, is the graph of the details of one of the tests, for the system being controlled without learning. The readings cycle above and below the threshold for the first have of the test and then hover above the threshold for the second half.
The learning component test (figure 8) hovers much closer to the threshold, for the first half of the test and once it crosses over the threshold starts to show bigger variations in the readings. The slope of the line before each reading is important because this was used in the training, so the steeper the slope the more likely the system will take action.

Another observation is that all but three of the readings for the learning component are below the threshold, while the test without the learning has the majority of its readings above the threshold. This aligns with expectations because; as the readings are rising towards the threshold the learning system turns the heater off before it actually crosses the threshold. Without the learning component the reading must cross through the threshold before the actuator is changed. Because there is such a long time gap in the readings it causes the system to show more fluctuation.

The next two graphs (figure 9 and 10) show the distance from the thresholds for both the tests of the learning and non learning systems. The graph of the system without the learning component, bounces around quite a bit from one data point to the next. The simulation only models the heating of the aquarium and left the other inputs constant. The first graph (figure 11) is of the initial heating period, where the temperature starts at 70 degrees and attempts to get to 80 degrees. There is a dramatic improvement with the learning algorithm. Without the learning, the system has a large over correction, but with the learning the heater is turned off far earlier and does not over shoot the threshold by much. The line with learning is much more ragged showing that it is turning on and off the heater much more frequently.

The next graph (figure 12) is a zoom in on the readings of the two simulations once the initial heating has smoothed out. The series from the learning system is hovering at a higher temperature than that without learning, but it is
difficult to see if one is performing better then the other. What is interesting is that the learning graph cycles for the most part above the threshold, and does not cross below 80 degrees very often.

![Figure 11 - Simulation of Learning vs. Non Learning](image)

The next graph (figure 13) shows the percentage improvement of the learning simulation versus the non learning simulation. Specifically it is the percent improvement of the running sum of the distance of the two simulations to the threshold. So at time $t = 400$ the performance since $t = 0$ is 40% better then the non learning algorithm. So there is a very large initial improvement that tapers off over the entire simulation.

![Figure 12 - No Learning Delta vs. Learning Delta](image)

7. CONCLUSION AND FUTURE WORK

Tests continue to be run on the system to get a better understanding of the affects of the learning component. The noise and the latencies of the aquarium make it difficult to get a precise view of the effects of our learning module, however the data shows that using the learning component reduced the temperature swing by approximately 20%. Tests run on a simulator back up those results and give a better insight into much longer tests. The simulation provides the ability to look at the system running for long periods of time, and without the noise of actual instruments. Our technique had a dramatic improvement when there is a change in temperature and the system reacts to that change. Over long period of time there was still improvement but much smaller. We would like to try to move to a more volatile environment where the natural swings are more dramatic.

Backpropagation algorithm was used for training the neural network because it is relatively performant and simple to implement. It works well for an online control system. The “memory” in such a network degrades quickly with temporal data. We would like to try some other networks and training algorithms such as Backpropagation through Time (BPTT) and Real Time Recurrent Learning (RTRL). These algorithms have shown promise when context from more distant temporal data is needed.

Research is continuing on the Aquarium System, including longer running test. New instruments, actuators and PLC’s are being added. We would like to continue to expand our system to control more environmental factors. One of the strong benefits of a neural network solution is the ability to generalize. We would like to test out the generalization of our technique.

REFERENCES