

# A Neural Network-based Approach to Modeling the Allocation of Behaviors in Concurrent Schedule, Variable Interval Learning

Erica J. Newland, Songhua Xu and Willard L. Miranker

Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA

erica.newland@aya.yale.edu, songhua.xu@yale.edu, miranker@cs.yale.edu

## Abstract

*In this paper we present a neural network-based model of the acquisition of choice behaviors. We employ a multi-layer perceptron, trained using back-propagation with a modified desired-output vector, to model behavior in concurrent-schedule, variable-interval, reinforcing learning situations. We show that our model can be used to describe and predict steady-state behavior and learning patterns at the molar level.*

## 1. Introduction

Operant conditioning trains voluntary behavior under the expectation of reinforcement; reinforcement occurs only if the operant response occurs [1]. When reinforcement ceases or becomes independent of response, responses become less frequent. This is also known as the assignment of credit, and understanding the mechanism by which this occurs is known as the “assignment of credit problem” [2]. In this paper, we develop a computational model of the assignment of credit using a neural network-based approach.

General principles of learning have been well established through experimental evidence, and a number of mathematical models have been developed to try to explain the process by which the assignment of credit occurs [3][4][5]. The models generally postulate that the better predicted a reinforcer is, the less efficient that reinforcer is at altering behavior. This concept is also the guiding philosophy for the back-propagation algorithm on the multi-layer perceptron. In this paper we exploit these similarities and suggest a neural network-based model of learning that employs the back-propagation algorithm under an operant conditioning paradigm. We do not, however, purport to model how reinforcement learning is carried out at the neuronal level in the brain.

## 2. Motivation

We sought to investigate whether the outputs of a simple multilayer perceptron trained using a modified back-propagation algorithm could model the decisions made by a laboratory subject trained using a variable-interval, concurrent schedule paradigm. The model presented was developed as a simulation of a concurrent-schedule laboratory setup used by Banna and Newland to obtain an understanding of choice acquisition in rats [6][7].

In concurrent schedule learning, two choices, for example two levers, are available to the subject. A response on each manipulandum (for example, a lever press) is reinforced with a pre-determined probability. This probability is determined by a variable interval (VI) schedule of reinforcement. On a VI-30 schedule on the left lever, for example, the first response on the left lever to occur at least  $X$  seconds after the trial begins – where  $X$  is a uniformly distributed random variable with a mean value of 30 seconds – is reinforced with a sucrose pellet. The other lever, meanwhile uses a distinct schedule of reinforcement which may also be VI-30 or which may have some a different expected time of reinforcement, for example 90 seconds. The procedure described is known as the two-key procedure. Concurrent schedules can be arranged to be dependent or independent of each other. Under a dependent schedule, a simulation of which is presented in this paper, when one reinforcer becomes active (on), the timer for the other schedule is put on hold until the first reinforcer is actually delivered [6]. Under this particular experimental setup, subjects are trained in a single session on a number of different learning-schedules.

Changeover delays are used in concurrent schedule designs to avoid reinforcing a steady alternation between manipulanda. A changeover delay is a forced delay that occurs when the subject switches from one reinforcement schedule to the other. That is, for  $z$

seconds after the switch, there is no chance of reinforcement [6][7].

### 3. System design

A multi-layer perceptron trained on a modified back-propagation algorithm was used to model the acquisition of choice. The code was adapted from a simple back-propagation algorithm provided by [8]. Back-propagation is usually associated with the learning-with-a-teacher paradigm, in which a set of expected outcomes is compared to the neural network's output, and the network's weights are adjusted according to the calculated error. Put otherwise, in traditional back-propagation learning, the entirety of information about the environment is available to the network [9]. Models of the acquisition of choice in a two-key procedure, however, should reflect the fact that the subject only has minimal information about the environment available. Thus we modified the back-propagation algorithm to reflect the lack of available information about the environment. We were not interested in a particular set of convergent weights for our model but instead on the model's outputs throughout the entirety of the learning process.

We let each neuron's activation function be  $\tanh(x)$ . There were  $n$  steps in the learning process ( $n$  chances for weight updates). We let  $x(i)$  denote an input to network at time step  $i$  and  $y(i)$  denote an output of the network at time step  $i$ . We let  $d(i)$  denote the "desired-output" at time step  $i$ . The concept, and re-definition of, desired-output is discussed below. The error term represented the difference between  $y(i)$  and  $d(i)$ . The training occurred sequentially (the network will be updated after the presentation of each pair  $(x(i), d(i))$ ). The seed input was a 0; each subsequent  $x(i)$  equaled  $y(i-1)$ .

#### 3.1 Decisions as Expectations

Conditioning models are often built upon a hypothesis that the learning process involves the generation of expectations about future events. This hypothesis is well summarized by Gallistel in his book *The Organization of Learning*: "When confronted with a choice between alternatives that have different expected rates for the occurrence of some to-be-anticipated outcome, animals, humans, and otherwise, proportion their choices in accord with the relative expected rates" [10].

We assume that decisions are made on the basis of expectations of reinforcement. From this assumption, we construct the following notation. We let

expectation of reinforcement on lever 1 be given on a scale of  $[0,1]$ , where 1 indicates 100% confidence in reinforcement. Expectation of reinforcement on lever 2 is given on a scale of  $[-1, 0]$ , where -1 indicates 100% confidence in reinforcement. If, at time step  $i$ , the expected reinforcement on lever 1 is 0.6, then this can be interpreted as 60% confidence that there will be reinforcement on lever 1. In this case, we say that if lever 1 is pressed, a reward is expected. If, however, the expected reinforcement on lever 1 is 0.2, then this indicates that it is considered more likely for there not to be reinforcement on either lever than for there to be reinforcement on one of the levers. In this case, we say a reward is not expected. We work under the reasonable assumption that with no expected reward, a subject would prefer not to exert the energy to press a lever and thus neither lever should be chosen. When a reward is expected, the subject will press the lever from which it expects a reward.

So we designed the output of our neural network to be in the form of the subject's expectation of reinforcement. An output  $y(i)$  equals 1 was associated with 100% confidence in choosing lever 1, while an output  $y(i)$  equals -1 was associated with 100% confidence in choosing lever 2. If the absolute value of  $y(i)$  was less than or equal to 0.5, then this indicated that no reinforcement was expected, so the simulated subject did not "press a lever" at time-step  $i$ .

#### 3.2 Memory

At any time step, a subject's memory, which is denoted here as  $M_i$ , informs its estimation of the state of the environment. Experimental data suggests that a feasible model for weighing these past experiences involves the use of the leaky integrator, which was introduced by Bush and Mosteller in 1955 [11]. The leaky integrator is a linear operator commonly used in dynamic models of short-term memory in operant conditioning [2]:

$$M_i = wM_{i-1} + (1-w)C(i-1)$$

$C$  is a vector of values  $C(i)$ , each of which represents knowledge of some characteristic of the state of the environment precisely, and only, at time-step  $i$ . A small constant  $w$  places higher weight on new experiences, while a large constant places higher weight on old experiences.

#### 3.3 Design and Adaptation of the Desired-Output Vector

Traditional training by back-propagation ultimately hinges on a comparison of the neural network's output

at each time step with a predetermined desired output for that time step. However, the desired output in the experimental paradigm being modeled is generally probabilistic in nature and at specific times even requires modification in response to the neuron's output. That is to say, in our model the  $d(i)$  that existed at time  $t$  equals 1, was often very different from the  $d(i)$  that existed at time  $t$  equals  $i$ .

We defined the length of a session as the number of time steps (for example, seconds) in the learning process. A single activation on lever  $j$  lasted until the subject pressed lever  $j$ . We let  $d(i) = 1$  correspond to a time-step for which lever 1's reinforcement was activated. Similarly, we let each  $d(i) = -1$  correspond to a time-step when lever 2's reinforcement was activated. Initially,  $(100/x)$  percent of the entries in  $d$  had a value of 1 and  $(100/y)$  percent of the entries in  $d$  had a value of -1. The rest of the entries in  $d$  were temporarily filled with 0's. The schedules are referred to here as VI-x, VI-y, where  $x$  and  $y$  represent the expected number of seconds until the reinforcer for lever 1 or lever 2, respectively, is activated.

Below we list and justify the rules used for creating and updating the desired-output vector during training. The vector was updated in the order that the rules are listed; a single  $d(i)$  value was often changed more than once. We were careful to consider ways that  $d(i)$  must be changed so that the simulated subject did not receive more information about the environment than a laboratory subject would receive.

- If  $y(i) > 0.5$  and  $d(i) = 1$ , then  $d(i+1) = 1$ . If  $y(i) < -0.5$  and  $d(i) = -1$ , then  $d(i+1) = -1$ . If lever  $i$  was pressed and reinforcement was received, then the decision to press lever  $i$  was strengthened by the reinforcer. Because reinforcement was received, assigning the value  $d(i) = 1$  did not give the simulated subject information about the environment that it should not have had. If the expected reinforcement was 1, then the subject's expectations were met and the reinforcer had no impact on future behavior.
- If  $y(i) \leq 0.5$  AND  $d(i) = 1$ ,  $d$  was lengthened by inserting into  $d$  an element  $d(i+1)$  with value 1. If  $y(i) \geq -0.5$  AND  $d(i) = -1$ ,  $d$  was lengthened by inserting into  $d$  an element  $d(i+1)$  with value -1. If a lever's reinforcer was activated but no lever was pressed, then the reinforcer remained activated. New entries for  $d$  were added in order to simulate the dependent schedule. The value of this new  $d(i)$  was changed again in a subsequent step.
- If  $y(i) > 0.5$  AND  $d(i)$  was neither -1 nor 1, then  $d(i+k) = y(i+k)$ . If  $y(i) < -0.5$  AND  $d(i)$  was neither -1 nor 1, then  $d(i+k) = y(i+k)$ . In this

situation, the subject pressed no lever and therefore received no new information about the environment. That is, no weights in the neural network were updated.

- If  $y(i) > 0.5$ , but  $d(i)$  did not equal 1, then  $d(i) = wMi-1+(1-w)C(i-1)$ , where  $C(i-1)$  was given by the number of reinforcements on lever 2 at time  $i-1$ . If  $y(i) < -0.5$  but  $d(i)$  was not -1,  $d(i) = wMi-1+(1-w)C(i-1)$  where  $C(i-1)$  was given by the number of reinforcements on level 1 at time  $i-1$ . If the simulated subject pressed lever  $i$  but received no response, the simulated subject next chose between lever  $j$  and not pressing a lever. This choice was made based on its memory (determined by the leaky integrator) of lever  $j$ 's reinforcement schedule.
- If  $|y(i)| \leq 0.5$ , set  $d(i) = y(i)$ . Nothing was learned in this time-step.
- If  $y(i) > 0.5$ , then at the occurrence of the first  $k$  for which  $y(i+k) < -0.5$ , new entries  $d(i+k)$  and  $d(i+k+1)$  were inserted into  $d$  such that the values of  $d(i+k) = d(i+k+1) = 0$ . If  $y(i) < -0.5$ , then at the occurrence of the first  $k$  for which  $y(i+k) > 0.5$ , new entries  $d(i+k)$  and  $d(i+k+1)$  were inserted into  $d$  such that  $d(i+k) = d(i+k+1) = 0$ . This represented a changeover delay of 2 time steps (or seconds). We chose a replacement value of 0 in order to compensate for the lack of time information conveyed by the back-propagation algorithm. That is, we investigated if setting the new values to zero could adjust for the fact that back-propagation algorithm makes the association between switching levers and the lack of reinforcement essentially impossible to make.

## 4 Results and analysis

The figures that follow illustrate simulations using a neural net with one input node, two hidden layers of two nodes each, one output node, a training rate of 0.3, and a momentum of 0.3. A large momentum term was found to produce stays as long as 1,000 seconds on the rich lever under some conditions. Each time-step represented one second.

We designed our sessions to model those used by Banna and Newland in [6] and [7]. Banna and Newland conducted sessions that lasted 120 minutes each. During the first 30 minutes, which we refer to as Stage 1, two VI-30 schedules were run; this yielded an expectation of 2 reinforcers per minute. For the next 90 minutes, which we refer to as Stage 2, the original schedules were replaced with new schedules. These schedules featured a "rich" lever and a "lean" lever. In

each session, the ratio of the expected number of reinforcements on the lean lever to the expected number of reinforcements on the rich lever was either 1, 1/4, 1/8, 1/16, or 1/32. In both Stage 1 and Stage 2 the expected number of reinforcers per minute was held constant at 2.

Experimentally, steady-state behavior in concurrent scheduling is described by the generalized matching relation [12]:

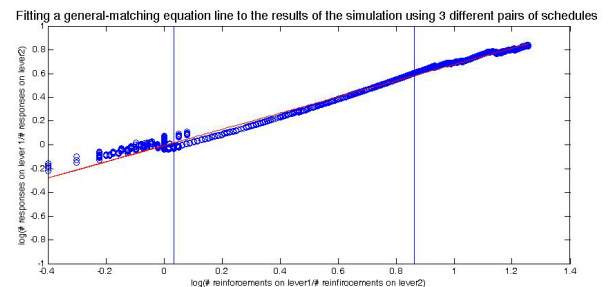
$$\log \frac{B_1}{B_2} = \log c + a \log \left( \frac{R_1}{R_2} \right)$$

where  $B_1/B_2$  represents the ratio of responses on lever 1 to responses on lever 2 and  $R_1/R_2$  represents the ratio of reinforcers on lever 1 to reinforcers on lever 2. We tracked the learning rate as calculated at each time-step. We show our results in Figure 1. Each data point represents the cumulative values measured at each time-step over a 3-stage experiment. The slope of the line is 0.6827 and the intercept is -0.0066. The intercept of -0.0066 indicates no bias for either lever and the coefficient of 0.6827 suggests underfitting of the general matching equation. This is consistent with results of [6] and [7]. In these works, the average slope value for a single rat's behavior in one session was 0.63. Similar results were found when we modeled the other VI schedules. These results suggest that in the model, just as in the laboratory, the general learning – or “success” – trend, measured in terms of the ratio of number of responses to number of reinforcers, is consistent no matter the reinforcement schedule. However, the percent of time spent not responding on a lever is higher in our simulation than in [6] and [7].

A better understanding of the response pattern can be acquired by studying Figure 2, which shows responses throughout an entire session. The horizontal lines indicate the lever-press cutoffs of 0.5 and -0.5. The data points on the extremes of each of these horizontal lines indicate lever presses. As is expected from a back-propagation algorithm, output values change gradually. We see a consistent swapping of levers during the baseline, when the levers' schedules are identical; following the switch to a 32:1 reinforcement schedule, however, we see that lever presses are highly concentrated on the rich lever. Responses on the lean lever are not concentrated together: even when the lean lever is chosen, the simulated subject quickly changes back to the rich lever.

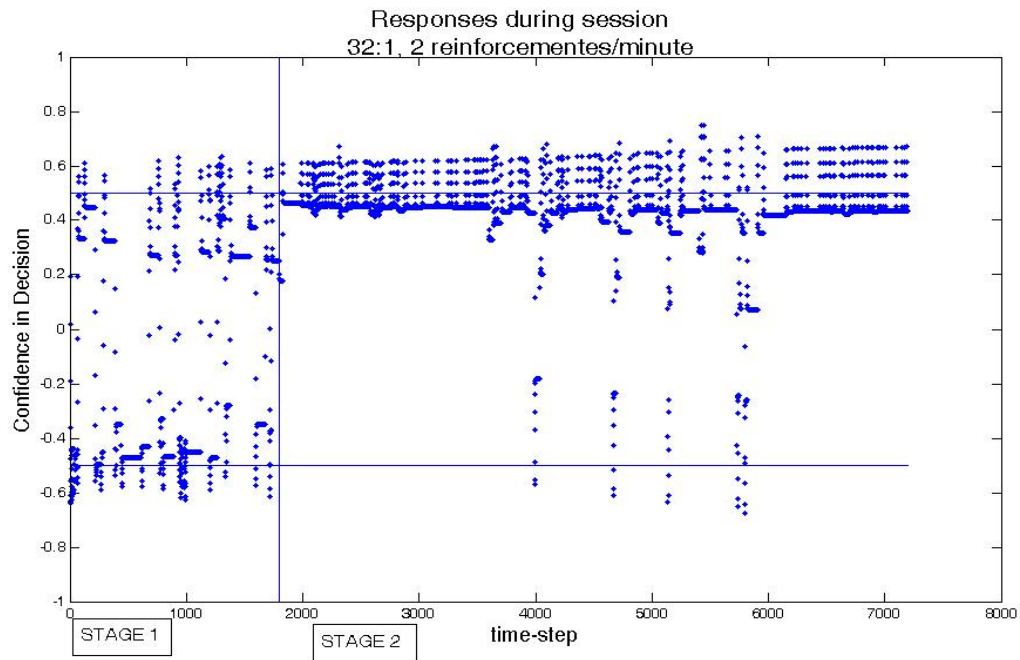
We also analyzed the more specific behavior of the

neural network within each steady state. We first analyzed the number of “responses per visit” on each lever during different stages of the simulation. We defined a visit as follows: a visit on lever  $i$  ends when lever  $j$  is pressed for the first time following the pressing of lever  $i$ ; a visit does not end if there is no lever press during a time step. This first choice of lever  $j$  represents the beginning of a visit on lever  $j$ . During stage 1, the range of durations of the visits and the distribution of durations were comparable in our model and the laboratory results, although the absolute number of visits was much higher in the laboratory results. For stage 2 trials, there were not a sufficient number of lever presses, particularly on the lean lever, to determine if the distributions of responses per visit were similar.



**Figure 1. Response ratios calculated at each time step in one simulated experiment. The vertical lines represent the transition between a 1:1 schedule ratio and a 32:1 ratio, and the transition between a 32:1 ratio and a 16:1 ratio, respectively.**

Finally, a 2-multi-layer perceptron system was also implemented. In this design, each multi-layer perceptron corresponded to one of the levers. The output of each perceptron corresponded to the expectation of a reinforcer on the lever it represented. The simulated subject chose the lever represented by the perceptron whose output had the largest absolute value. If neither perceptron outputted a value greater than 0.5 then no lever was chosen. The rules implemented were nearly identical to those described earlier for the single multi-layer perceptron system. Only the “winning” perceptron, if there was a perceptron with output of magnitude greater than 0.5, was updated at each time-step. The results were not consistent nor did they resemble experimental results. This design was not pursued further.



**Figure 2. Simulation Responses.**

## 5. Conclusion

The consistent applicability of the generalized matching equation to the simulation data lends credence to the use of a modified back-propagation algorithm on a single multi-layer perceptron to produce molar models of behavior. This is remarkable in light of the simplicity of the model. We plan to further investigate the molecular behavior of our simulated subjects within each state as well as during transitions between the states and to compare the results of these simulations to laboratory experiments, such as those in [6] and [7].

## 6. Acknowledgements

We would like to thank Kelly Banna and Chris Newland for suggesting this investigation and for sharing their data.

## 7. References

[1] B. F. Skinner, *The Behavior of Organisms; an Experimental Analysis*. New York, London: D. Appleton-Century Company, incorporated, 1938.  
 [2] V. Dragoi and J. E. R. Staddon, "The dynamics of operant conditioning", *Psychol. Rev.*, vol. 106, pp. 20-61, January 1999.

[3] A. Dickinson, *Contemporary Animal Learning Theory*. Cambridge Eng.; New York: Cambridge University Press, 1980.  
 [4] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins, "Learning and Sequential Decision Making", in *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, M. Gabriel and J. Moore, Ed. Cambridge, Mass: MIT Press, 1990, pp. 539-602.  
 [5] R. H. Bauer and J. M. Fuster, "Delayed-matching and delayed-response deficit from cooling dorsolateral prefrontal cortex in monkeys", *J. Comp. Physiol. Psychol.*, vol. 90, pp. 293-302, Mar. 1976.  
 [6] K.M. Banna, "Drug Effects on Behavior in Transition: Does Context Matter?" doctoral Dissertation, Dept. of Psychology, Auburn University, Auburn, Alabama, 2007.  
 [7] K.M. Banna, and M.C. Newland (under review), "The Acquisition of Choice."  
 [8] D. Patterson, "Per-period Backpropagation", May 2006, <http://www.csee.umbc.edu/~dpatte3/nn/res/backprop.m>  
 [9] S. S. Haykin, *Neural Networks: A Comprehensive Foundation.*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1999.  
 [10] C. R. Gallistel, *The Organization of Learning*. Cambridge, Mass: MIT Press, 1990.  
 [11] R. R. Bush and F. Mosteller, *Stochastic Models of Learning*. New York: Wiley, 1964.  
 [12] W. M. Baum, "On two types of deviation from the matching law: bias and undermatching", *J. Exp. Anal. Behav.*, vol. 22, pp. 231-242, Jul. 1974.