A MAZE LEARNING COMPARISON OF ELMAN, LONG SHORT-TERM MEMORY, AND MONA NEURAL NETWORKS

THOMAS E. PORTEGYS
School of Information Technology, Illinois State University, Campus Box 5150
Normal, Illinois, 61790, USA
Email: portegys@gmail.com

This study compares the maze learning performance of three artificial neural network architectures: an Elman recurrent neural network, a Long Short-Term Memory (LSTM) network, and Mona, a goal-seeking neural network. The mazes are networks of distinctly marked rooms randomly interconnected by doors that open probabilistically. The mazes are used to examine two important problems related to artificial neural networks: (1) the retention of long-term state information and (2) the modular use of learned information. For the former, mazes impose a context learning demand: at the beginning of the maze, an initial door choice forms a context that must be remembered until the end of the maze, where the same numbered door must be chosen again in order to reach the goal. For the latter, the effect of modular and non-modular training is examined. In modular training, the door-associations are trained in separate trials from the intervening maze paths, and only presented together in testing trials. All networks performed well on mazes without the context learning requirement. Mona and LSTM performed well on context learning with non-modular training; the Elman performance degraded as the task length increased. Mona also performed well for modular training; both LSTM and Elman performed poorly with modular training.

Keywords: Maze learning; context learning; modular learning; Elman recurrent neural network; Long Short-Term Memory network; Mona goal-seeking neural network.

1. Introduction

For decades researchers have used mazes as a means to measure the learning capabilities of animals [1]. For instance, the effects of a proposed memory-enhancing drug may be tested on rats by observing how well they learn mazes in comparison to a control group. More recently mazes have been employed as a means to investigate learning in artificial neural networks [2,3,4]. These studies typically use variations of a T-maze as a test environment in which learning is accomplished through exploration. The mazes used here are too complex to be learned by exploration; they are exploited through training, which also poses interesting challenges to neural network learning.

This study compares the maze learning performance of three artificial neural network architectures: an Elman recurrent neural network, a Long Short-Term Memory (LSTM) network, and Mona, a goal-seeking neural network. In this experiment, a set of related mazes is generated and presented to the neural network for training and testing. The related mazes are instances generated from a common probabilistic maze called a *metamaze* such that any two instances might contain common and divergent path segments. The aim is to challenge a neural network with a task that demands both sequence learning and the ability to discriminate diverging paths.

An important function of many organisms is the ability to use contextual information in order to increase the probability of achieving goals [5,6,7]. For example, a street address has a particular meaning only in the context of the city it is in. Context learning demands the retention of state information for an extended duration, a challenge for most artificial neural networks. To test context learning, at the beginning of the maze an initial door choice forms a context that must be remembered until the end of the maze, where the same numbered door must be chosen again in order to reach the goal. Thus the learner must retain this door association while navigating the intervening path through the maze.

Two types of training are done for context learning: modular and non-modular. In modular training, the context door-associations are trained in separate trials

from the intervening maze paths, and only presented together in end-to-end testing trials. The maze entry and exit rooms form interfaces connecting the modular training sets. In non-modular training, the door-associations and maze instances are presented together as they are in the testing trials.

Testing success with modular training suggests modular internal representations. The issue of modularity in neural networks is an important one: monolithic (non-modular) neural networks such as the classic feed-forward perceptron often store representations of independent input spaces in a non-modular fashion, making them difficult to scale and re-train. Moreover, the biological neural networks of many animals feature a great deal of modularity both in function and structure. This allows them to re-train dynamically, a significant survival advantage in the face of a changing environment.

Azam's [8] survey of modular neural networks cites the Hierarchical Mixture of Experts (HME) as a prominent architecture. In this and several other schemes, an *a priori* partitioning of the input space is done in order to create a tree structure of neural networks. Tan and Nolfi [9] devised a means of autonomously partitioning the input space of a maze learning task by recognizing a dynamical phase change as an indicator of a transition to a different partition. Optimally, each partition is then managed by a dedicated recurrent neural network. This system was able to learn a two-room maze but instabilities requiring further study were also observed.

One of the aims of this project is to determine how well the Mona network can modularly self-organize in an autonomous manner without relying on adjunct networks. Additionally, the issue of modularity in learning systems may be thought of in relation to the well-known AI frame problem [10]. The frame problem is about things that change and things that remain the same when an action is taken in the world. For learning, the question is how knowledge is represented such that when the world changes only the affected portions of the representation are changed.

Elman networks are a type of Recurrent Neural Network (RNN) having a state-retention capacity that allows them to classify temporal input sequences, which for the maze learning task consist of goal paths. Elman networks have been widely studied and have proven to be successful in a variety of tasks, such as grammar [11,12] learning and text classification [13]. Another reason for selecting an Elman network for the metamaze task is to observe how state information degrades over time [14]. The context learning task is used for this purpose. The Elman network used for the maze task is described in a subsequent section.

The Long Short-Term Memory (LSTM) network [14] is designed to allow state information to be retained over extended periods of time through the use of special activation units called memory blocks. LSTM networks have been shown to solve tasks that require long-term retention of state information [15]. One such task, the Embedded Reber Grammar, bears some similarity to the context learning task. The LSTM network used for the maze task is described in a subsequent section.

Although a connectionist architecture, Mona also has elements of a state-based planning system. While planners in artificial intelligence [16] are typically symbolic and not connectionist systems, it seems clear that neural networks must also be able to perform planning if they are to function as biological networks do. Mona has been successfully used on a number of tasks, including cooperative nest-building [17] and learning a 3D grid environment for a simulated foraging robot [18]. Mona models the homeostatic need-reduction mechanism that animals possess as an integrated motivation mechanism designed to produce responses to reach goals that reduce needs. It is described more fully in a later section.

Q-Learning [19] was also compared as a base level on the metamaze task since it keeps only the current room mark as state information, underscoring the importance of sequence learning.

2. Description

2.1. *Metamazes*

A metamaze is a network of uniquely marked rooms interconnected by doors that open probabilistically. One room serves as the maze entry and another as the exit, which is the goal. The task is to move from the entry to the exit room. An instance of a metamaze is generated by resolving the probabilities that determine the open-ability of the doors. Thus a metamaze may generate a set of maze instances that are similar yet vary in some details. For example, a path having a high probability may appear in most instances, but not in all. The neural

networks are trained to follow optimal paths for a specific set of instances.

Each room contains a fixed number of doors (with one exception for context learning). A connection is a randomly determined directional link from a door in a source room to a target room. Doors without connections are visible but cannot open. Each connected door in a maze instance is constructed using a probability that it will open. A door that will open allows a learner to transition to the target room. If a door does not open the learner remains in the source room. A door may be thought of as a teleport control that might not be connected to anything and that might not work. If it does work the learner will observe that the room marking is different in the target room.

A maze instance is determined by a metamaze and a random number generator seed. Using the seeded random number generator a resolution is made for each connected door probability to determine its openability. So a door with a 10% probability in the metamaze has a 10% chance of being openable in the instance, for example. A path through a maze instance is a sequence of rooms and door connections from the entry room to the exit room.

In order to discover "interesting" maze instances for the project, a program was written to randomly generate and score metamazes of varying (5, 10, and 15) numbers of rooms, with varying (3 and 5) numbers of doors per room. Metamazes were scored primarily on the number of paths and secondarily on the average path length. The number of random connections per metamaze grew with the number of rooms and doors. It was also found that higher scoring (see below) mazes were generated if connections were forced to be bi-directional, that is, if room A had a connection to B, then B also had a connection to A, although not necessarily with the same numbered door. This allows backtracking. For each metamaze, 10 instances were generated and searched for goal paths. A set of instances was then selected with the requirement that it must be possible to train a learner to traverse a goal path regardless of which instance is presented. In other words, a properly trained learner can always solve a maze without error, defined as trying a wrong door, even though it may have to try doors that do not open and loop to gain information about which instance it is in. For each of the 6 possible maze dimension values (rooms × doors), 1000 metamazes were generated and the best 50 retained as suitable training mazes. Table 1 shows the average number of paths for the various room and door combinations of the retained mazes. Table 2 shows the average path lengths.

Table 1. Average metamaze goal paths.

		ROOMS		
		5	10	15
DOORS	3	1.46	1.62	2.36
	5	1.76	3.34	3.62

Table 2. Average metamaze goal path length.

		ROOMS		
		5	10	15
DOORS	3	3.96	6.62	6.28
	5	3.93	5.65	6.16

As the maze is navigated, knowledge of the specific instance is revealed and can be utilized to find the goal. To illustrate, consider the metamaze and its 3 instances (A, B, and C) shown in Figure 1.

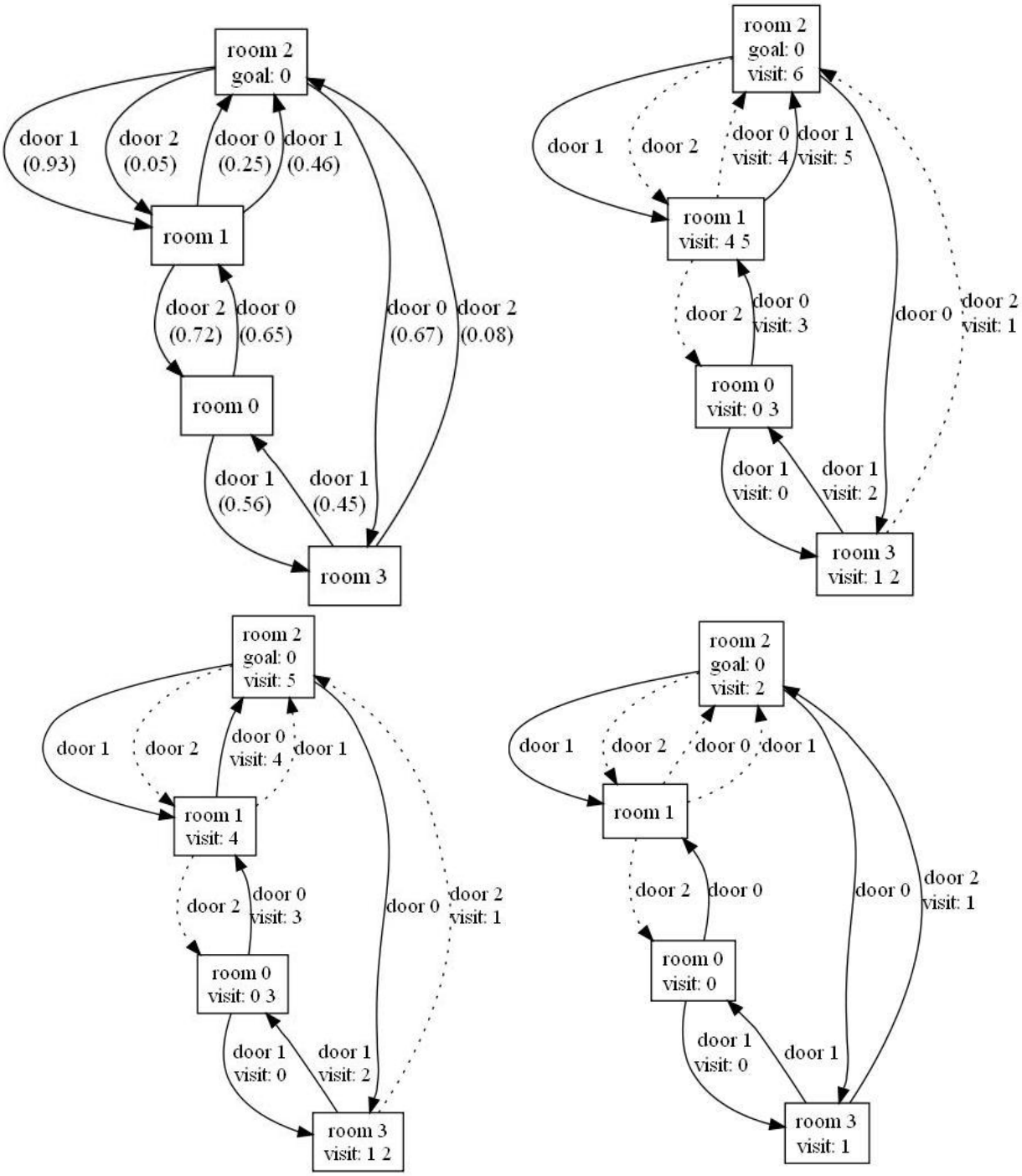

Fig. 1. Example metamaze (top left) with instances, A (top right), B (bottom left), and C (bottom right).

Each vertex represents a room. Room 0 is the start room and room 2 is the goal. In the metamaze, each edge is a door annotated with the probability of it being openable. Although each room actually has the same number of doors, only doors that are connected to rooms are shown for clarity. In the maze instances, the open or closed resolution of the probabilities is denoted by solid and dashed edges, respectively. The sequence of rooms and connections visited for each instance is also marked, so in C the path taken to the goal is rooms 0, 3, 2 via doors 1 and 2 respectively. Note that for mazes A and B the learner also visits room 3 even though it is blocked by the closed door 2. The learner then returns to room 0 and proceeds to the goal via room 1 (although using different doors in A and B). The reason the learner is thusly trained is this: for maze instance C, if the learner proceeds directly to room 1 it will find itself blocked from the goal and also unable to retract to room 0. So the successful scheme is to train the learner to try room 3 first.

The maze search algorithm finds the shortest goal path "knowing" only what the learner can know at every step; initially the learner knows only that the maze is one of a set of possible instances. The aim is to find paths that achieve the goal. This is an interesting problem in its own right and as it turns out is not a simple algorithm, although outside the scope of this paper. It is included in the download noted in the conclusion. As mentioned previously, the maze discovery program invokes the search algorithm for a set of instances and selects a subset of instances that always allow a learner to find a path to the goal.

2.1.1. Context learning mazes

A portion of this study is devoted to the learning and application of context information, which is represented as an additional maze learning task. At the beginning of the maze, an initial door choice forms a context that must be remembered until the end of the maze, where the same numbered door must be chosen again in order to reach a goal. The learner must learn both the door associations and the intervening path through the maze in order to complete a trial successfully. An example of such a maze is shown in Figure 2.

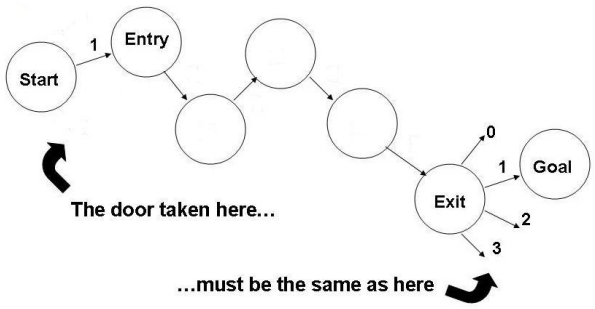

Fig. 2. A context learning maze.

In the start room only one of the possible doors is visible and is open to the maze entry room. The entry-to-exit room portion of the maze is an instance of a metamaze as described above, where the exit room now has door choices. Here the learner must choose the same numbered door as that leading from the start room to the entry room in order to reach the goal room. It can be seen that the context in this problem is to retain information about which door led from the start room in order to repeat this choice at the exit room. A single wrong door try is counted as a trial failure. In order to ensure that the context information varies, for each trial the door leading from the start room is randomly determined.

2.2. *Neural networks*

2.2.1. Elman recurrent neural network

An Elman network, also known as a Simple Recurrent Network, contains feedback units that allow it to retain temporal state information useful in classifying sequential input patterns. These feedback units reside in a context layer as shown in Figure 3. Each hidden layer unit has a connection to a corresponding context unit with a fixed weight of 1. All other weights are trainable. A processing step is then:

1. Set input unit activations to input pattern.
2. Compute hidden unit activations using input from input and context layers.
3. Compute output unit activations.
4. Copy new hidden unit activations to context layer.

The Elman network was created with the Stuttgart Neural Network Simulator (SNNS) version 4.2 (www-ra.informatik.uni-tuebingen.de/SNNS/). For maze learning the network was configured with 25 hidden

and 25 context units, which proved to perform well for the basic task. For a maze with 5 doors there were 10 input units, 5 to encode the room mark and 5 for the doors, and 5 output units for the door selection. The network was initialized with the SNNS "JE_Weights" initialization function, standard for Elman networks. The initial connection weights were randomly set to values between -1 and 1, and the initial activation of the context units was 0. The learning function was "JE_BP" with a default learning rate of 0.2, and the update function was "JE_Order", both typical for Elman networks.

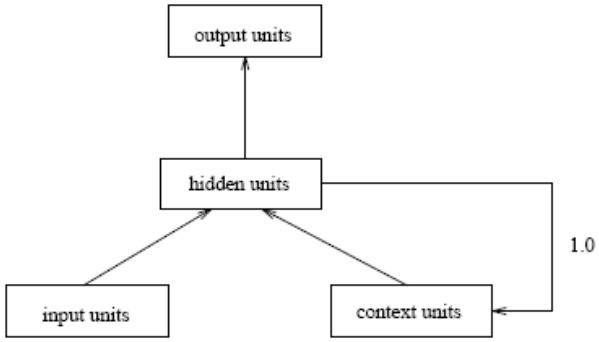

Fig. 3. Elman recurrent network.

The following training and testing procedure was used for each of the 50 selected metamazes.

For non-context metamaze training, a training pattern file was generated for each metamaze instance, each containing a unique goal path consisting of a sequence of input/output pairs, where the room mark and visible doors comprise the input and the correct door choice the output. The network was trained in 1000 epochs, where in each epoch all the training patterns were presented in a random order. The activation states of the neurons were reset before each sequence presentation. For testing, the connection weights were frozen and the patterns were presented in order. A pattern was scored as correct if the network produced the correct sequence of outputs; otherwise it was scored as incorrect.

For non-modular context learning, pattern files were created for all possible metamaze instance and context door combinations. So if there were 3 maze instances and 5 doors per room, there were 15 pattern files, each containing an "end-to-end" sequence from start room to goal. These were used for both training and testing. Again, there were 1000 training epochs, with all the patterns randomly presented each epoch. Testing consisted of presenting all the patterns in the set and scoring them based on producing the correct output sequence.

For modular context training, there were 2 sets of files. One set trained the maze instance paths, and was thus identical to the non-context set, and the other trained the context door associations. Each context door association file contained 2 patterns: a start room pattern and an exit room pattern. So for example the door 0 training file contained a start room pattern with only door 0 visible as input and the output being the choice of door 0; the second pattern was the exit room with all doors visible as input, and the output again being door 0. Modular context testing used the same set of end-to-end patterns used for the non-modular context testing. In each of the 1000 training epochs, all the patterns in the 2 training sets were randomly presented. Testing followed the training epochs.

2.2.2. Long Short-Term Memory (LSTM) neural network

A problem in training a simple recurrent neural network such as the Elman is that the gradient of the error quickly vanishes as the time lag between the output and the relevant input increases, leading to the inability to train the retention of long-term state information.

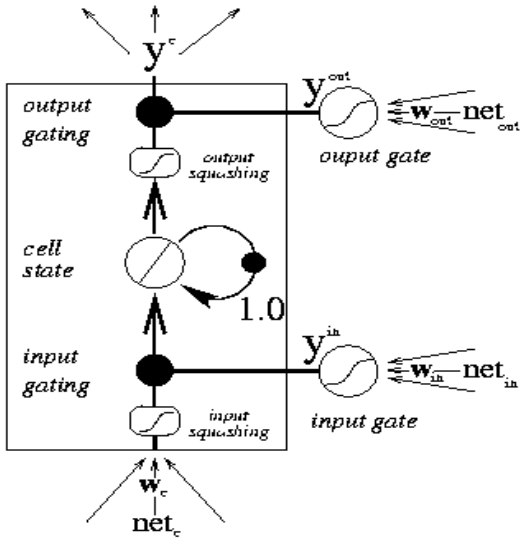

Fig. 4. LSTM single-celled memory block.

In the LSTM network, the hidden units of a conventional recurrent neural network can be replaced by memory blocks, each of which contains one or more memory cells. A memory block containing a single cell is shown in Figure 4. The cell is a simple linear unit with a single self-recurrent connection with weight set to 1. In the absence of any other input, this connection serves to preserve the cell's current state indefinitely. Deciding what information to store, and when to output that information lies with a multiplicative input and output gating unit, respectively.

The LSTM code was obtained from the Institute of Bioinformatics at the Johannes Kepler University (www.bioinf.jku.at/software/lstm). After some trial-and-error tuning, the network was configured with 8 single-celled memory blocks and 10 hidden units, and the learning rate set at 0.1. The inputs and outputs were encoded as for the Elman network. Training and testing were also done as for the Elman network, except that 2500 training epochs were performed.

2.2.3. Mona goal-seeking neural network

2.2.3.1. Description. Mona is a model based on the rationale that brains are goal-seeking neural networks. It has a simple interface with the environment: all knowledge of the state of the environment is absorbed through senses. Responses are expressed to the environment with the goal of eliciting sensory inputs which are internally associated with the reduction of needs.

Events can be drawn from sensors, responses, or the firing states of component neurons, calling for three types of neurons: those attuned to sensors are receptors, those associated with responses are motors, and those mediating other neurons are mediators. A mediator presides over a sequence of neuron firing events, retaining state information and driving need-based *motive* through the network to activate motor responses that will move the system toward goals that reduce needs. Mediators can be structured in hierarchies representing environmental contexts. The state of the environment is represented by the configuration and *enablement* of mediator neurons. An enabled mediator can be thought of as representing a sequence of stimuli and responses that will reliably occur in the environment.

A processing step in Mona is as follows:

1. **Sense**. Environmental stimuli are sensed, firing appropriate receptors. These firings cascade upward into overarching mediators, creating an updated network firing state.
2. **Enable**. Using the firing state of the network, the enablements of component neurons are updated. For example, if a mediator's cause event fires, it can enable its effect event indicating that the environmental situation corresponding to the effect event is pending.
3. **Learn**. Firing neurons are matched with previous firing events to hypothesize causal sequences that are captured by the creation of new mediators.
4. **Drive**. The reduction of needs is associated with the firing of associated goal neurons. It is through these neurons that motive propagates into the network. Based on needs, goals, and the updated enablement state of the network, motive accumulates in appropriate motor neurons. Multiple needs can either compete or cooperate in driving the network. A need whose level is low contributes less drive than one whose level is high.
5. **Respond**. A relatively highly motivated motor neuron activates a response.

As an example, what follows in Figures 5-7 is a sequence that illustrates several processing steps in an actual assembly of neurons that control a door association in the context learning portion of the maze task. There are 3 mediators: one controlling the events at the start room, one for the maze exit room, and a mediator connecting the other mediators. This assembly interacts with, but is modularly independent of, other neurons that govern the inner maze navigation. The assembly shown is for door 0; other door associations will have their own assemblies.

Several notational definitions must first be given. The triangles are receptor neurons, the inverted triangles are motor neurons, and the ovals are mediators. Mediators are linked to a single cause neuron, a single effect neuron, and 0 or more intervening intermediate neurons. The '*' tag on a link signifies which event neuron is expected to fire next. A firing neuron is shown by a double outline. Mediator neurons also display their enablement state as a value from 0 to 1. Neurons having a goal value are annotated. Finally, motive is shown in block arrows driving through the network. The labels on the neurons were placed manually for descriptive reasons.

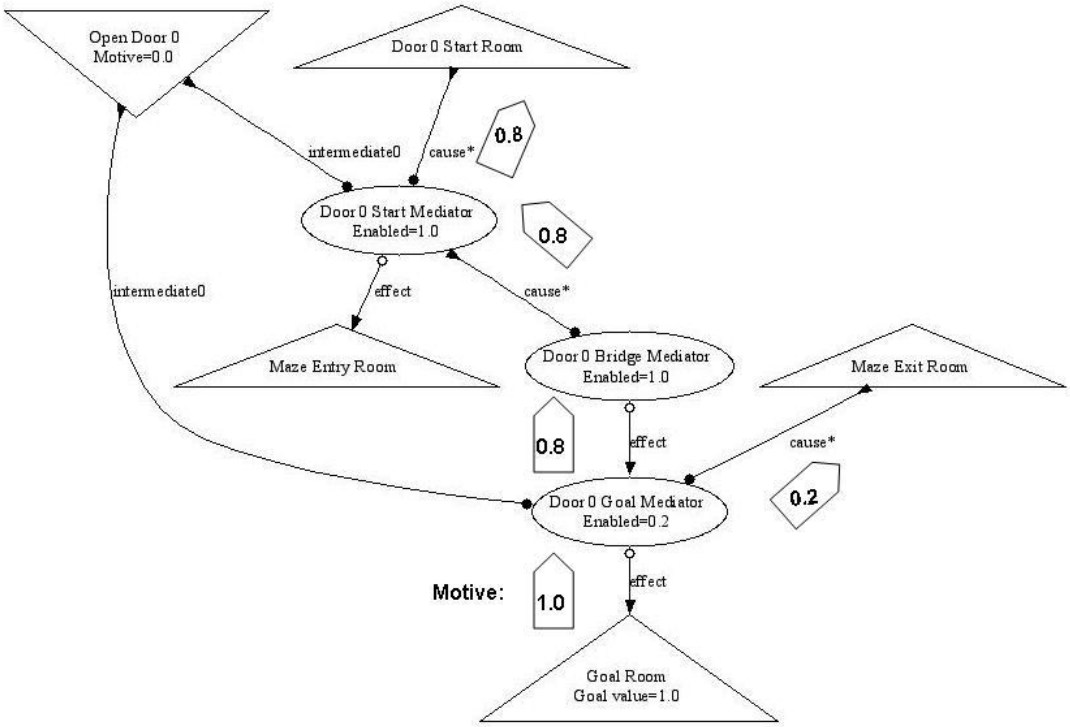

Fig. 5. The initial state of the *Door 0 Correspondence* neuron assembly in a maze with 5 doors per room. Since the *Door 0 Goal Mediator* is only partially enabled (20% a priori chance that door 0 is correct), motive propagated from *Goal Room* neuron splits accordingly, with 80% going to the *Door 0 Bridge Mediator* to further enable the *Door 0 Goal Mediator* neuron. Within the *Door 0 Bridge Mediator*, motive is shunted into the expected component event, the *Door 0 Start Room* receptor neuron.

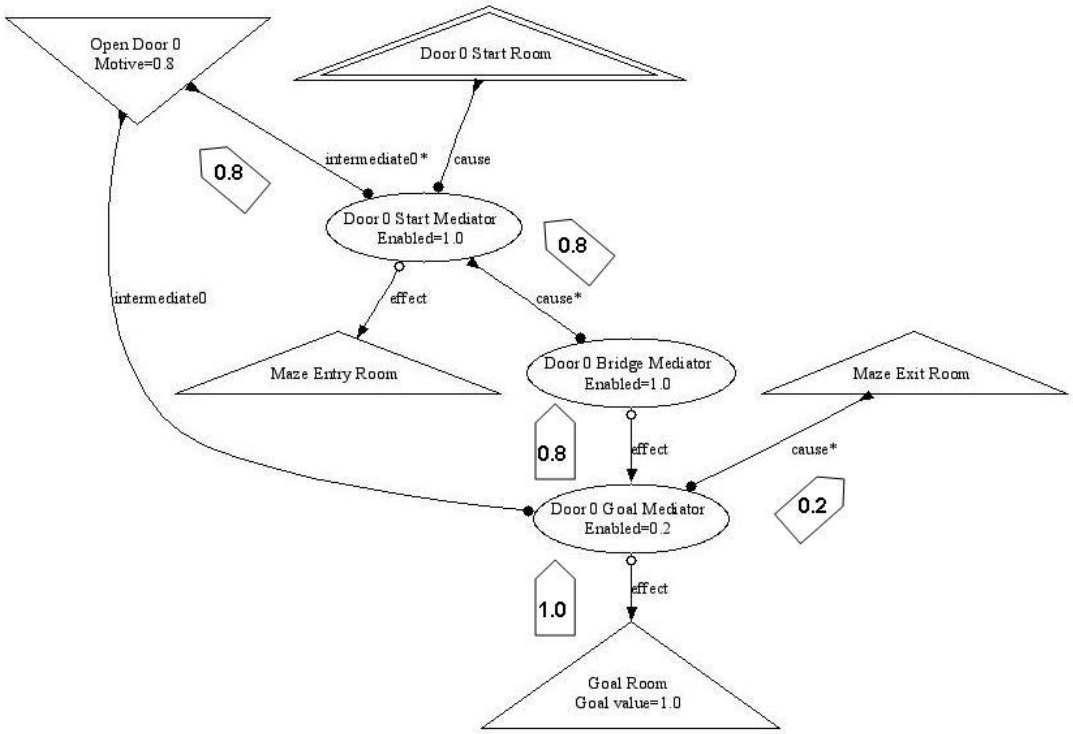

Fig. 6. The *Door 0 Start Room* receptor neuron fires when the learner senses the start room having door 0 as the only possible choice. The next expected event is the firing of the *Open Door 0* motor neuron.

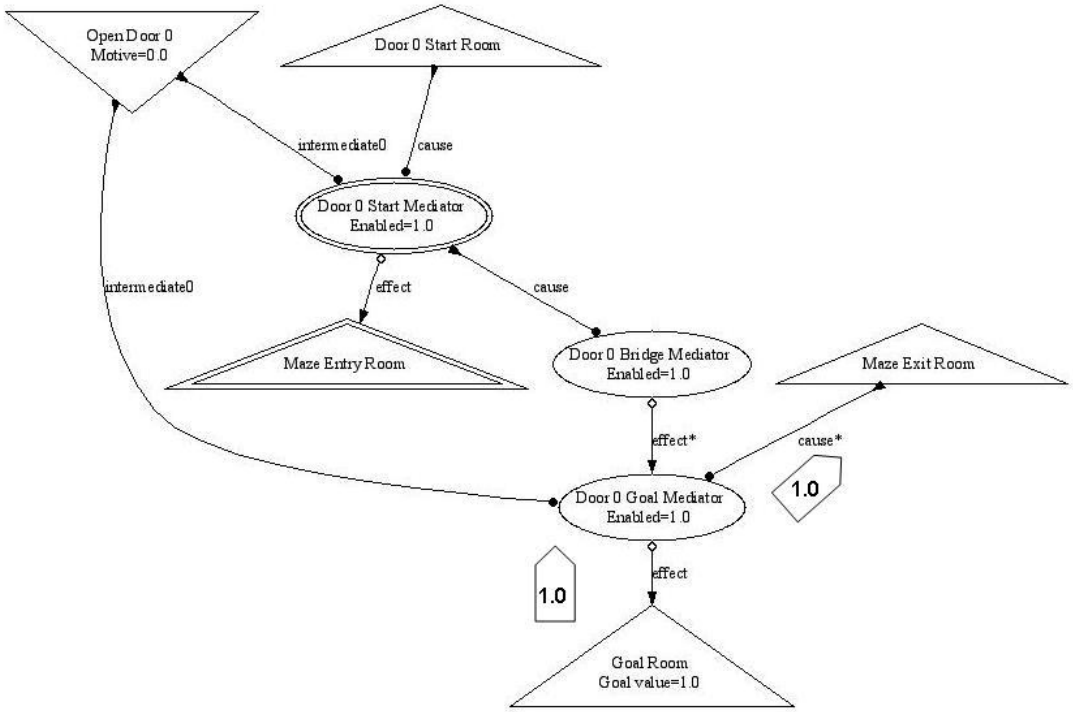

Fig. 7. The *Maze Entry Room* receptor fires, completing the sequence and thus also firing the *Door 0 Start Mediator*. Since door 0 is now known to be the correct door, the *Door 0 Bridge Mediator* enables its expected effect event, the *Door 0 Goal Mediator*. The *Door 0 Goal Mediator* now redirects motive entirely into the *Maze Exit Room* receptor. From there, motive will propagate into the maze navigation neurons (not shown) driving responses to move to the maze exit room.

The neuron assembly is quiescent while the other neurons are at work navigating the maze, motivated to reach the maze exit room. Note that information reflecting the correct door 0 choice is stored in the enablement state of the *Door 0 Goal Mediator*, awaiting the firing of its cause, the *Maze Exit Room* receptor. Once *Maze Exit Room* fires, motive propagates fully into *Open Door 0* to express its associated response, resulting in the achievement of the goal.

A mediator consists of:

- Level:
  $\begin{cases} 0: \text{contains receptor and motor events.} \\ > 0: \text{contains lower level mediator events.} \end{cases}$
- Neuron event sequence: for level 0 mediators, this is a {{receptor, motor}*, receptor} sequence.
- Current event.
- Enablement: firing "probability" given cause event firing.
- Motive: driven through network from goal neurons when associated needs arise.
- Notification list: mediators for which this mediator is an event.

As each event neuron fires, the current event is advanced. When the effect (final) event fires, the mediator itself fires which notifies all its parent mediators and resets the current event to its cause (initial) event. If an event does not fire within a specified length of time, the current event is also reset to its cause event.

The enablement of a mediator roughly represents a conditional firing probability based on the opportunity

given by the firing of its cause event. Enablement reflects how it performs when given opportunities:

$$enablement = \sum motive_{firing} \,/\, \sum motive_{firing\text{-}opportunity} \quad (1)$$

Motive is used to weight firing opportunities, thus highly motivated firings and misfiring (time-outs) impact enablement more strongly. Conversely, for a low motivation firing opportunity, the enablement is less impacted.

As previously noted, motive is injected into the network through neurons that are designated as goals, and as such are associated with needs. Fluctuations in need values are reflected in changing motive values. Motive may be driven into a mediator in two ways:

- From a parent mediator for which it is an event. In this case the firing of the event is necessary for the firing of the parent.
- From its effect event. In this case the mediator's effect event is the cause event of another mediator which is driving its cause, producing a "chaining" interaction between the mediators.

In either case, motive is an influence to fire the mediator by firing motor neurons that produce a desired stimulus/response stream. If the mediator has a high enablement, meaning that its firing is reliably independent of context, it will drive input motive "down" into its current event to fire it. If it has a low enablement, it will drive input motive "up" into parent mediators for which it is an effect event in order to establish a context in which firing is more favorable. This is determined as follows:

$$down\text{-}motive = input\text{-}motive \times enablement \,/\, (enablement + \sum enablement_{eligible\text{-}parent}) \quad (2)$$

$$up\text{-}motive_{eligible\text{-}parent} = (input\text{-}motive - down\text{-}motive) \times (enablement_{eligible\text{-}parent} \,/\, \sum enablement_{eligible\text{-}parent}) \quad (3)$$

Eligible parent mediators are those which do not already index the mediator as a current event, since context has already been established by them. Motive is also attenuated as it propagates in order to avoid excessive propagation of motive.

During the drive phase, motive is recorded in each neuron. For a mediator, this is used to update its enablement. For a receptor neuron, input motive is driven into parents in a manner similar to that used by mediators. For a motor neuron, motive is simply stored. Response selection is then a choice weighted by the motive stored in the motor neurons associated with each response.

For learning new mediators, neuron firings which occur sequentially in a specified window of time are included in a mediator with the following probability:

$$creation\text{-}probability = p_{cause\text{-}event} \times p_{effect\text{-}event} \quad (4)$$

$$p_{event} = (motive \,/\, MAX\_MOTIVE)^{MOTIVE\_DAMPER} \times (enablement \,/\, MAX\_ENABLEMENT)^{ENABLEMENT\_DAMPER} \quad (5)$$

The probability is dependent on the motive and enablement of the firing events relative to maximum values. Receptors and motors use a constant for enablement. The damper parameters are a means to control the mediator creation rate. A new mediator is given an initial enablement which could be high enough to replace a low enablement mediator from the network. Its survival then depends on increasing its enablement through the update formula (1).

2.2.3.2. Procedure. For the metamaze task, a maximum of 200 learned mediators was allowed; of these, approximately 50 were found to be useful in performing the task. A constant need was associated with the goal room receptor, motivating Mona to seek the goal room. Mona's responses were overridden with the correct response in order to produce the desired sequence of stimuli/responses that would allow Mona to learn the goal paths. This proved quite effective: it was observed that Mona could learn some metamaze instance goal paths in a single trial. However, quick learning can be a double-edged sword: many neurons were also created that represented piece-parts of the desired paths. Some of these were useful for redundancy reasons, but others were "parasitic" in nature: they thrived and grew stronger along with the correct neurons, yet once strong enough to challenge for control they matched paths inappropriately and produced the wrong responses. A solution for this was to suppress the enablement update of a mediator that is "subsumed" by a temporally longer mediator.

Training and testing was done in a similar fashion to the Elman and LSTM networks, except 200 training epochs were run. For testing, 100 trials of each test maze were run: since this was the first time that Mona was able to operate without forced responses, the extra trials were to observe any degradation in performance –

no degradation was noted. The score was calculated based on the percentage of trials that the goal room was reached without error.

2.3. *Q-Learning*

In addition to the Elman and Mona runs, Q-Learning was compared as a base level on the metamaze task since it keeps only the current room mark as state information, highlighting the importance of sequence learning. Although each room in a maze is uniquely marked, looping and unsuccessful door tries cause this to be a sequential task. The Q-Learning learning rate and discount parameters were set to 0.9, and the minimum Q-value set to 0.001.

3. Results

The results for non-context metamazes having 3 doors are shown in Figure 8, and the results for 5 doors are shown in Figure 9. These are averages for the 50 selected metamazes. The Elman, LSTM, and Mona networks perform nearly perfectly for the 3 door mazes, and quite well (>90%) for the 5 door mazes. Q-Learning was most successful in mazes with fewer goal paths, but did poorly overall, testifying to the sequential nature of the task.

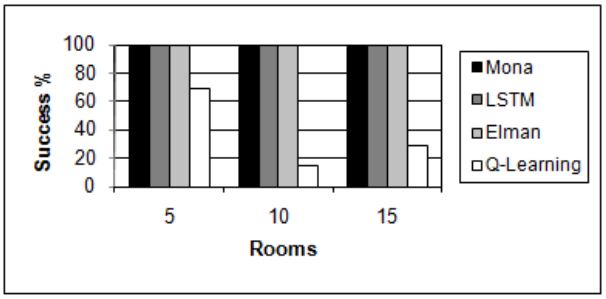

Fig. 8. Non-context 3 door performance.

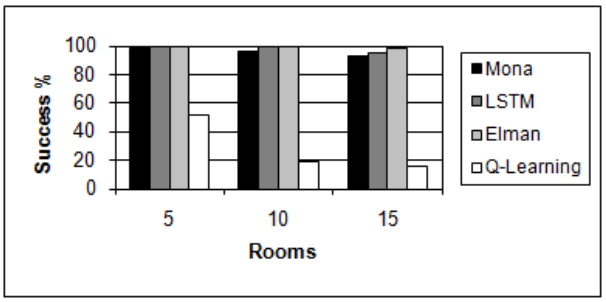

Fig. 9. Non-context 5 door performance.

The results for non-modular context metamazes having 3 doors are shown in Figure 10, and the results for 5 doors are shown in Figure 11. These show a different story for the Elman network, becoming more pronounced as the number of rooms and the path lengths (as shown in Table 2) grows. As expected, the Elman network loses context state information as the amount of intervening processing increases. LSTM and Mona perform respectably (>75%). Q-Learning, having no context state information to draw on, performs poorly.

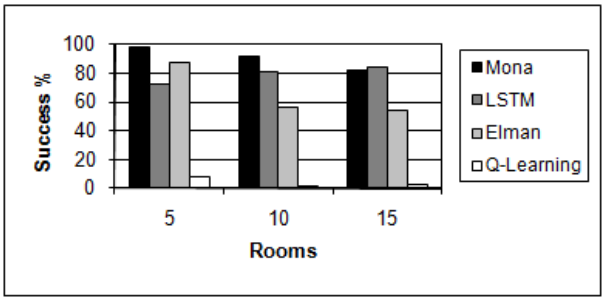

Fig. 10. Non-modular context 3 door performance.

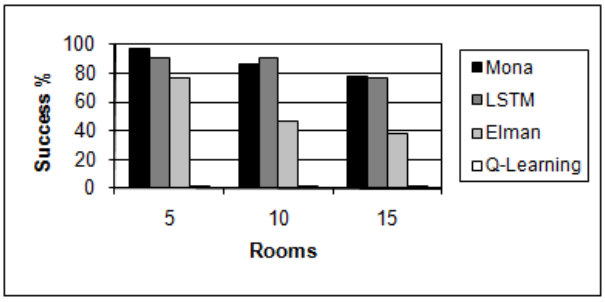

Fig. 11. Non-modular context 5 door performance.

In Figures 12 and 13, the modularly trained Elman and LSTM networks show striking performance degradations. These poor performances indicate that the two types of training do not positively interact at testing time, a result expected for a monolithic architecture. This is a significant drawback, since the two parts of the task are essentially independent. Interestingly, Mona achieves superior performance with modular training than with non-modular (>90%), which is an encouraging result for two reasons: (1) modular training is simpler, involving fewer training sequences, and (2) one would plausibly expect to have better results

training an animal using a modular approach that places events that are to be associated closer together in time. Figures 5-7 shows one of the learned modular structures that allow Mona to succeed. Q-Learning was not attempted with modular learning since it relies on a single state-space to operate.

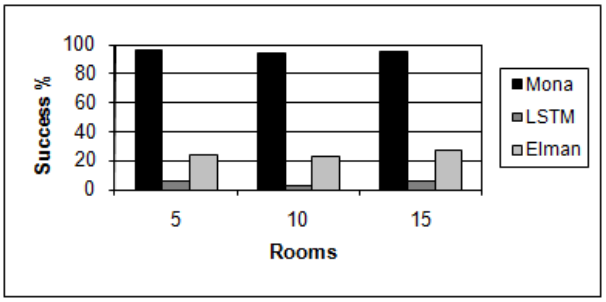

Fig. 12. Modular context 3 door performance.

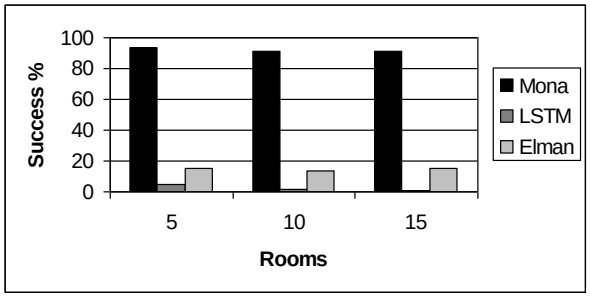

Fig. 13. Modular context 5 door performance.

4. Conclusions

Results show good performance for the Elman, LSTM, and Mona networks on metamazes that do not impose the context learning requirement. The LSTM and Mona networks are also capable of learning the context maze task with non-modular training. Modular context learning tests indicate continued good performance for Mona; examination reveals independent yet cooperating neural structures formed by different training modules. The observation is that modular training leads to modular internal structures.

The poor performance of the Elman and LSTM networks with modular training highlights the problem of using these types of neural networks on tasks having a dynamic nature that may cause components to change, necessitating retraining. This is a serious issue since so many real-world environments pose challenges that fall into this category. For these tasks it would be beneficial to be able to retrain only the portions that change and have interactions with the rest of the system remain functional.

One of the original inspirations for Mona was to exhibit some properties, such as the rapid learning of novel and changing environments, which animals possess but conventional computer systems largely do not. Furthering this goal, Mona's learning and goal-seeking capabilities have been utilized as the nervous system of simple creatures in a simulated world [18]. Using a combination of instinct evolution and experiential learning, the creatures are able to acquire foraging skills and knowledge to explore and exploit their environment. Knowledge gained in simulated environments has also laid the groundwork for work with learning robots.

5. Resources

The C++ source code for Mona and the metamaze project are available at:
www.itk.ilstu.edu/faculty/portegys/research/metamaze/metamaze.zip
It can be compiled with either gcc/make or Microsoft Visual Studio.
Developments and other projects using Mona are on CodePlex (www.codeplex.com/mona).
The author is currently with Microsoft and can be contacted at either tom.portegys@microsoft.com or portegys@gmail.com.

References

1. H. A. Carr 1913, "Maze studies with the white rat". I. Normal animals. J. Anim. Behav. 7: 259-275.
2. Y. Yamauchi and R. Beer 1995, "Sequential behavior and learning in evolved dynamical neural networks". Adaptive Behavior, 2(3):219-246.
3. J. Blynel and D. Floreano 2003, "Exploring the T-Maze: Evolving Learning-Like Robot Behaviors using CTRNNs". In Raidl, G. et al. (Eds.) Applications of Evolutionary Computing, Heidelberg: Springer Verlag.
4. C. Johansson and A. Lansner 2002, "A neural reinforcement learning system". Tech. Rep. TRITA-NA-P0215 Dept. of Numerical Analysis and Computing Science Royal Institute of Technology, Stockholm, Sweden.
5. R. Turner 1998, "Context-Mediated Behavior for Intelligent Agents", International Journal of Human-Computer Studies special issue on "Using Context in Applications", 48(3), pp. 307-330.
6. R. Sun and C.L. Giles 2001, "Sequence Learning: From Recognition and Prediction to Sequential Decision Making", IEEE Intelligent Systems, 16(4).

7. T. Portegys 2005, “Learning Environmental Contexts in a Goal-Seeking Neural Network”, Journal of Intelligent Systems, Vol. 16, No. 2.
8. F. Azam 2000, “Biologically Inspired Modular Neural Networks”, Ph.D. Dissertation submitted to Virginia Polytechnic Institute and State University.
9. J. Tani and S. Nolfi 1999, “Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems”. Neural Networks, 12(7–8):1131–1141.
10. D. Dennett 1984, “Cognitive Wheels: The Frame Problem in AI”. In Minds, Machines, and Evolution. C. Hookway, ed. pp. 128-151. Cambridge University Press.
11. P. Rodriguez, J. Wiles and J. Elman 1999, “A recurrent neural network that learns to count”. Connection Science, 11, 5-40.
12. A. Smith 2003, “Grammar Inference Using Recurrent Neural Networks”, Department of Computer Science, University of San Diego, California, www.cse.ucsd.edu/~atsmith/
13. J. Farkas 1995, “Document classification and recurrent neural networks”, Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research, p.21, November 07-09, Toronto, Ontario, Canada.
14. S. Hochreiter and J. Schmidhuber 1997, “Long short-term memory”, Neural Computation, 9(8), 1735-1780.
15. F. Gers, J. Schmidhuber and F. Cummins 2000, “Learning to forget: Continual prediction with LSTM”, Neural Computation, 12(10), 2451-2471.
16. S. Benson and N. Nilsson 1995, “Reacting, Planning and Learning in an Autonomous Agent”, Machine Intelligence 14, Edited by K.Furukawa, D. Michie, and S. Muggleton. Oxford: Clarendon Press.
17. T. Portegys 2001, “Goal-Seeking Behavior in a Connectionist Model”, Artificial Intelligence Review, 16 (3):225-253.
18. T. Portegys 2007, “Instinct and Learning Synergy in Simulated Foraging Using a Neural Network”, The 2007 International Conference on Artificial Intelligence and Pattern Recognition (AIPR-07), Orlando, Florida, USA.
19. C. Watkins 1989, “Learning from Delayed Rewards”, Thesis, University of Cambridge, England.