The MaRz Algorithm: Towards an Artificial General Episodic Learner

Christian Rodriguez, Giselle Marston, William Goolkasian, Ashley Rosenberg, and Andrew Nuxoll(✉)

University of Portland, Portland, OR 97203, USA
nuxoll@up.edu

Abstract. An artificial general intelligence must be able to record and leverage its experiences to improve its behavior. In this paper, we present a novel, general, episodic learning algorithm that can operate effectively in an environment where its episodic memories are the only resource it has available for learning.

1 Introduction

Episodic memory is one of three types of long term memory generally recognized in humans [1]:

Procedural Memory memory of how to act (e.g., how to walk, ride a bicycle, juggle)
Semantic Memory memory for facts (e.g., trees have trunks, the earth orbits the sun, ripe bananas are yellow)
Episodic Memory memory for events (e.g., what time you arrived at your hotel, what color shirt you wore yesterday).

More broadly, episodic memories have a temporal component. They consist of a sequence of episodes that an agent experiences as it moves through time [23]. An agent has the ability to retrieve past episodes from a memory queue and also recognize when its current situation is similar to a past episode. Episodes consist not only of what the agent is sensing but also what it might be thinking about at the time.

An artificial general intelligence must be able to not only record but leverage its experiences to improve its behavior. Furthermore, these experiences are at their most general in their original, unprocessed, unfiltered form. Despite this, most artificially intelligent agents created to date lack an episodic memory. The agent can record information which the programmer has specifically instructed it to record but it lacks a general ability to record all its experiences all the time. Nor can it retrieve a memory based upon an open-ended cue.

In humans, an impaired episodic memory is called amnesia. Evidence from psychology indicates that an amnesiac's ability to learn new semantic memories is impaired by amnesia. The reasons for this are not entirely clear but it seems

T. Everitt et al. (Eds.): AGI 2017, LNAI 10414, pp. 154–163, 2017.
DOI: 10.1007/978-3-319-63703-7_15

likely that semantic learning relies upon the ability to perceive cause and effect and, thus, relies upon the ability to perceive an ordering to events. Therefore, it is reasonable to conclude that a general purpose episodic memory may be an essential component of an effective artifical general intelligence.

In this paper, we present our work towards understanding how episodic memory is used for learning by exploring algorithms an agent can use to learn in situations where its episodic memories are the only resource it has available for learning. Specifically, the agent is given no knowledge about the environment it occupies or the task is must perform and we compare a reinforcement learning agent to a novel episodic learning algorithm of our design.

Furthermore, our goal is also to create an effective, general episodic memory. That is, an episodic memory that can operate in any environment without need to be configured for that environment.

2 Blind FSM Environment

The environment we have selected for this research is a deterministic finite state machine [7] with the following properties:

1. a single goal state
2. each state has a transition for each letter in the alphabet
3. there is a path from each state to the goal state.

The agent has only the following resources:

1. a single goal state sensor so that it knows when it has reached the goal state. It has no other sensors.
2. knowledge of the state machine's alphabet and, thus, what actions are available to it at any given time

Notably the agent does NOT know:

1. how many states there are
2. how many of those states are goal states
3. what state it is currently in
4. the transition function

As soon as the agent senses the goal state, it is immediately moved to a randomly-selected non-goal state. Thus the simulation can be run indefinitely with the agent repeatedly discovering the goal state.

Given the simplicity of this environment, it may seem trivial to create an agent with optimal behavior. However, the agent's lack of sensors means that it can not distinguish a non-goal state from any other non-goal state. There also is no consistent starting state. In other words, the agent faces maximal degree of perceptual aliasing [26].

In this situation, an agent using a traditional machine learning algorithm is mainly ineffective as it relies upon associating a best action with each state

that it can distinguish. To learn in this environment the agent must leverage sequences of episodes (see Previous Work below).

Ideal behavior, given the agent's lack of perception, is to determine what we refer to in this research as an optimal universal sequence of actions. A universal sequence of actions is one that will always take the agent to the goal state regardless of its starting state. It may reach the goal state before it completes the sequence but it will always reach the goal. An optimal universal sequence is a universal sequence that reaches the goal in the least number of steps, on average, over all possible starting states.

This environment was selected for its simplicity and flexibility while still meeting the requirement of an environment that requires a successful agent to have an episodic memory. Furthermore, it provides a clearly defined range of behavior. Specifically, an agent can not perform better than an optimal universal sequence and an agent with a random policy provides a non-arbitrary upper limit for bad behavior. Notably, an agent that always takes random steps will always eventually reach the goal.

3 Previous Work

3.1 Solving Finite Automata

The process of determining the transition function for a given state machine (i.e., machine identification) is well established [9,14]. However, all algorithms we are aware of rely upon being in a given starting state. These algorithms often rely upon testing a range of different input sequences on the machine and thus provide a foundation for our approach.

3.2 Perceptual Aliasing

Perceptual aliasing or hidden state are terms used in machine learning for the situation where the agent can perceive multiple states identically either due to insufficient or noisy sensors [26]. Notably, human behavior in environments with perceptual aliasing has also been studied [5].

A common technique to address perceptual aliasing in machine learning is to create a memory for the agent's sensing and to make decisions based upon sequences of episodes rather than individual ones [3,11,12]. We use this same approach in this research.

3.3 Artificial Episodic Memory

While unusual, artificial episodic memories have been created in the past for various purposes including empathic robots [6], non-player characters in multiplayer games [2] and solving physics problems [22].

Nuxoll and Laird [17] describe a set of cognitive capabilities granted or facilitated by an episodic memory. They demonstrate a few of these using a general

episodic memory system that is part of the Soar cognitive architecture [8]. More recently, a general implementation has been created for the Icarus cognitive architecture [13]

We believe that creating an artificial, general episodic memory presents three main challenges to the creator:

1. An ever-growing data store. The size of the episodic store is presumed to be finite yet new episodes are constantly being created. Thus, the agent and the agent must find a way to forget previous episodes and/or compress the overall data store to keep it below a given maximum size [15,18,24]
2. Retrieval from a given cue. A good system should be able to quickly return a "good" or "best" match for a given cue. The definition of a good match in a general context is not entirely clear. Most systems rely upon a simple cardinality of match or a system based upon term frequency-inverse document frequency (tf-idf) [20]
3. Learning. The agent should be able to leverage its experiences to improve its behavior. It's not clear how a general episodic memory system should tie in to the agent's ability to learn.

This research is currently focused on the third challenge listed above. Specifically, the episodic memories we are using allow the episodic memory to grow indefinitely and rely upon exact matches for retrieval. However, we are making this decision with an eye toward addressing all three challenges in the medium term.

A fundamental approach to using episodes in learning is to retrieve one or more past episodes that are most similar to the current situation. The agent can then make decisions by examining the outcomes of its actions in those past situations. All episodic memory facilitated learning we are aware of – including the work we are presenting here – is based on this approach. See [16] for a representative example.

In most cases the retrieval is deliberate, but there is some effort to make retrieval spontaneous when it is relevant [10].

3.4 Episodic Learning

Finally, this work builds upon previous research to build a general-purpose episodic learner in an environment that requires an episodic memory. Walker et al. [25] demonstrated building an successful sequence from the last action backward was valuable in such an environment. This insight formed the basis for the use of the suffix in the search nodes that MaRz uses. Faltersack et al. [4] took initial steps towards a general purpose episodic memory learner using a different learning approach than MaRz.

4 Nearest Sequence Memory

The Nearest Sequence Memory (NSM) algorithm was introduced by McCallum [12] as a way to address environments with perceptual aliasing. NSM proved to be

capable of finding optimal behavior in our Blind FSM environment and thus was useful to us as a basis for "good" behavior to compare our episodic memory to.

NSM maintains a simple episodic memory for the agent wherein each episode consists of the action, percepts and reward. (Note: Since the only percept that the agent has in the Blind FSM environment is its goal sensor, its percepts and reward are the same in that environment.) Furthermore, each state has a Q-value associated with it Q-Learning [21].

As a small example, consider a Blind FSM environment where the alphabet is a, b. The sequences below depict the agent's memory of its first 55 episodes.

```
aabbaaaabbaabababbbabbabaabaaabbaabababbabababbbabbabaaaba
00000000001111111111222222222233333333334444444444555555
01234567890123456789012345678901234567890123456789012345
```

The letters read from left to right indicate each action that the agent took over the course of its entire past. If the letter is underlined, that indicates that the agent reached the goal in that episode. The pair of digits (read vertically) under each letter are index values to facilitate referencing parts of the memory in the subsequent text.

At each time step, the agent considers each possible future action. Presuming it were to select that action for its current episode, it then searches its episodic memory for the k sequences from its past that provide the best (longest) match to its current situation given the potential action (i.e., a form of k-nearest neighbor learning). The agent selects the action for which the overall Q-value of its k matching neighbors is the highest.

If the action is taken, the associated k neighbor states are updated using the Q-Learning rule.

To operate effectively, NSM requires that certain values be pre-configured for the environment:

1. k (for kNN)
2. a learning rate
3. a discount factor
4. a chance of random action
5. how the previous value should change over time

Each time we made a substantial change to the size of the Blind FSM (i.e., number of states or alphabet size) we were compelled to adjust these values in order to return the algorithm to its most effective behavior.

5 MaRz Algorithm

For this research, we introduce the MaRz algorithm. MaRz maintains an episodic memory similar to NSM's that consists of the action selected and percept (goal) at each state (see the example in the previous section).

Unlike NSM, MaRz selects sequences of actions, rather than individual actions. Nominally, MaRz *considers* all sequences in order from shortest to

longest. For example, if the alphabet is a, b MaRz would consider trying the following sequences in a strict order: a, b, aa, ab, ba, bb, aaa, aab, aba, etc. If the alphabet was a, b, c, d the ordering would be: a, b, c, d, aa, ab, ac, ad, ba, bb, bc, etc. Generally, this is the same order you would list increasing numbers in base b, where b is the size of the alphabet and the letters of the alphabets are the digits.

It is important to note that the agent may choose to skip a sequence it is considering and it may even choose to jump back to a skipped sequence and continue forward considering all untried sequences from that point forward. Thus, the sequences are not tried in order they are considered. Nonetheless, this strict ordering of sequences is important to the algorithm.

A simplified version of MaRz could simply try every sequence it considers and eventually find a universal sequence. (A reminder to the reader: the concept of universal sequence was defined above in the Blind FSM Environment section). The key insight in this case is that such an agent is performing a breadth-first search through the space of possible sequences. MaRz instead uses an approach comparable to a memory-bounded A*-Search [19] through the space of these sequences.

A search node used by MaRz has the following attributes which are outlined here and will be explained more further on:

a suffix this is a particular sequence of letters. A sequence that ends with this suffix "matches" this search node. Also, notably, the length of this suffix is the "g" value in the context of A* search.

a queued-sequence the shortest sequence matching this node's suffix that has been considered but has not yet been tried by the agent. This may be unset (null).

failure list a list of indexes into the agent's episodic memory where a sequence ending with this suffix was tried and failed.

success list as above, but a list of successes. These lists are used to calculate the failure rate of the suffix which acts as the "h" value in the context of A* search.

The agent maintains several values as the algorithm executes:

1. a list of all nodes on the frontier of the search space. This list is kept to a certain maximum size. If a new node needs to be added to the list but the list is already at maximize size, then the node with the smallest overall value (suffix length + inverse failure rate) is evicted. Initially this list contains only a single node whose suffix is empty (zero letters) and, thus, matches any sequence.
2. a reference to the node in the node list that is the active node. Initially this is a reference to the only node in the list.
3. a value called NST (next sequence to try) indicating what sequence is to be considered next in the strict ordering defined above. The initial value of NST will always be a sequence of length 1 containing the first letter in the alphabet.

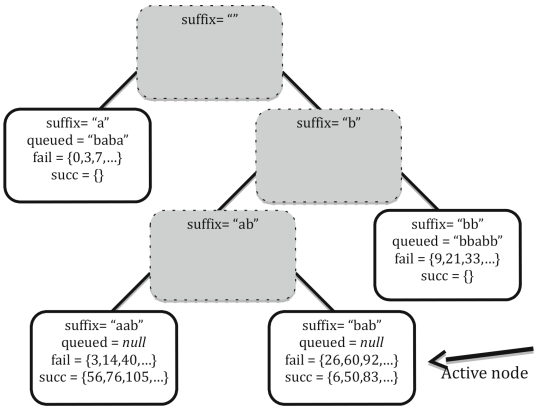

Fig. 1. An example search tree illustrating the MaRz algorithm for a finite state machine with a two letter alphabet a, b. Shaded nodes are in the frontier of the search.

The algorithm proceeds by repeating the following steps indefinitely (Fig. 1):

1. If the value of NST does not match the active node's suffix then it will not be tried. Locate the non-active node in the frontier list that it does match and set that node's queued sequence to NST if its value has not already been set. Then, advance the NST to the next value in order and repeat this step until a match is found.
2. Enact the NST. This will have one of three results:
   failure the agent does not find the goal. Update the active node's failure list and return to the previous step to select a new sequence to try.
   early success the agent reaches the goal partway through enacting the sequence. In this case, stop when the goal is reached and locate the non-active node whose suffix matches the partial sequence that was tried. Update that node's success list. Then, repeat this step by enacting the NST again.
   success the agent reaches the goal. Update the active node's success list.
3. Expand the active node. Specifically, the active node is removed from the frontier list. A new node is added to the frontier list for each letter of that alphabet. The suffix used by each new node is created by prepending that letter to the active node's suffix. The queued value is unset and the parent's successes and failures are divied up among the matching children.
4. The node in the frontier list with the highest overall value (e.g., the smallest sum of suffix length + overall failure rate) is selected as the new active node. If that node has a value for its queued sequence, then NST is reset to the queued sequence value. Otherwise, update the NST to the next value in order.

6 Results

To test the efficacy of NSM and MaRz we placed each agent in a randomly generated blind FSM with a prescribed number of states and prescribed alphabet

(number of actions). We then allowed it to run for a set number of successive goals starting with an empty episodic memory. Each time the agent reached the goal state, the number of steps taken (since the last goal) was recorded.

The results of this experiment were generated with randomly-generated blind FSMs with 30 states and alphabet size of 3. This experiment was then repeated 1000 times (each time with a different blind FSM of the given size) and the results were averaged. The result is shown in Fig. 2 below. The x-axis counts successful trips to the goal state. The y-axis is the average amount of steps the agent took to reach the goal that time. The horizontal line at the bottom is an approximation of the average length of the optimal universal sequence for blind FSM.

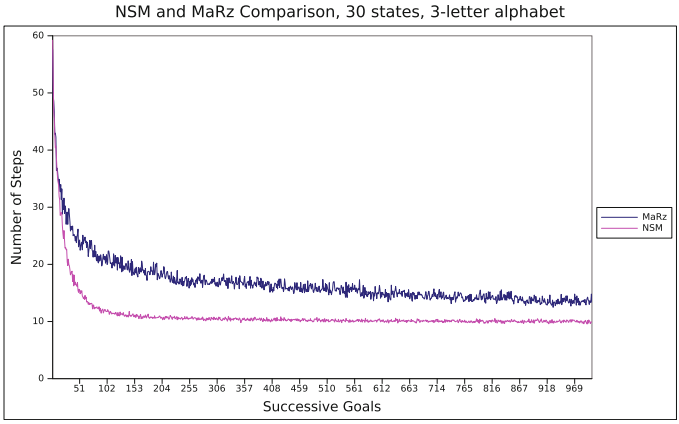

Fig. 2. Sample results from our comparison.

As can clearly be seen, both agents are demonstrating a classic learning curve and both are asymptotically approaching optimal behavior over time. The NSM agent learns somewhat faster than MaRz. However, MaRz requires essentially very little configuration. It does not use a learning rate, a discount factor, or a chance of taking a random action.

We ran this experiment on a variety of sizes of blind FSMs and saw similar results each time. The configuration used for Fig. 2 is representative.

We did find that we could get the agent to learn faster by tuning the relative importance of the failure rate vs. the suffix length, but the absence of tuning did not prevent the agent from finding a solution. This can be thought of as tuning the agent along the continuum of fully breadth-first or best-first search.

7 Discussion

This work demonstrates our progress towards a general episodic learning algorithm. Our results show that we have created an agent that can be successful

an environment where effective behavior requires a long term memory. Furthermore, this is a general learning algorithm. No environmental-specific tuning was required.

It is notable that an automated search through the space of tuning parameters for NSM could yield an equally effective result. This was not explored in our work and we believe it merits investigation.

Much can be done to expand upon these results and both NSM and MaRz. From our perspective, the most pressing issue is that MaRz has only been tested in a single simple environment. The Blind FSM environment we used was crafted to be unsolvable without an episodic memory. As such, it stands at one end of a continuum of perceptual aliasing. At the other end are environments in which the agent perceives each state uniquely and no state is ever repeated twice. It is also not clear how MaRz would perform in a non-deterministic environment. Overall, testing MaRz in a variety of environments seems like a logical next step.

References

1. Anderson, J.R.: Cognitive Psychology and Its Implications. Worth Publishers, New York (2000)
2. Brom, C., Lukavský, J., Kadlec, R.: Episodic memory for human-like agents and human-like agents for episodic memory. Int. J. Mach. Conscious. **2**(2) (2010)
3. Crook, P., Hayes, G.: Learning in a state of confusion: perceptual aliasing in grid world navigation. Towards Intel. Mob. Robots **4** (2003)
4. Faltersack, Z., Burns, B., Nuxoll, A., Crenshaw, T.L.: Ziggurat: steps toward a general episodic memory. In: AAAI Fall Symposium: Advances in Cognitive Systems (2011)
5. Gureckis, T.M., Love, B.C.: Short-term gains, long-term pains: how cues about state aid learning in dynamic environments. Cognition **113**(3), 293–313 (2009)
6. Ho, W.C., Dautenhahn, K., Nehaniv, C.L.: Computational memory architectures for autobiographic agents interacting in a complex virtual environment: a working model. Connection Sci. **20**(1), 21–65 (2008)
7. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Automata Theory, Languages, and Computation. Pearson, Boston (2006)
8. Laird, J.E.: The Soar Cognitive Architecture. MIT Press, Cambridge (2012)
9. Lee, D., Yannakakis, M.: Testing finite-state machines: state identification and verification. IEEE Trans. Comput. **43**(3), 306–320 (1994)
10. Li, J., Laird, J.E.: Spontaneous retrieval from long-term memory for a cognitive architecture. AAAI **2015**, 544–550 (2015)
11. Loch, J., Singh, S.P.: Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In: ICML 1998, pp. 323–331 (1998)
12. McCallum, R.A., Tesauro, G., Touretzky, D., Leen, T.: Instance-based state identification for reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 377–384 (1995)
13. Menager, D., Choi, D.: A robust implementation of episodic memory for a cognitive architecture. In: Proceedings of Annual Meeting of the Cognitive Science Society (2016)
14. Moore, E.F.: Gedanken-experiments on sequential machines. Automata Stud. **34**, 129–153 (1956)

15. Nuxoll, A., Tecuci, D., Ho, W.C., Wang, N.: Comparing forgetting algorithms for artificial episodic memory systems. In: Proceedings of the Symposium on Human Memory for Artificial Agents, AISB 2010, pp. 14–20 (2010)
16. Nuxoll, A.M., Laird, J.E.: Extending cognitive architecture with episodic memory. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence. AAAI Press, Vancouver (2007)
17. Nuxoll, A.M., Laird, J.E.: Enhancing intelligent agents with episodic memory. Cogn. Syst. Res. **17**, 34–48 (2012)
18. Ram, A., Santamaria, J.C.: Continuous case-based reasoning. Artif. Intel. **90**(1), 25–77 (1997)
19. Russell, S.J.: Efficient memory-bounded search methods. In: ECAI 1992, vol. 92, pp. 1–5 (1992)
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)
21. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, vol. 1. MIT press, Cambridge (1998)
22. Tecuci, D., Porter, B.: A generic memory module for events. In: Proceedings of the 20th Florida Artificial Intelligence Research Society Conference (FLAIRS), Key West, FL (2007)
23. Tulving, E.: Elements of Episodic Memory. Clarendon Press, Oxford (1983)
24. Vanderwerf, E., Stiles, R., Warlen, A., Seibert, A., Bastien, K., Meyer, A., Nuxoll, A., Wallace, S.: Hash Functions for Episodic Recognition and Retrieval. In: Proceedings of the 29th Florida Artificial Intelligence Research Society Conference (FLAIRS), Key West, FL (2016)
25. Walker, B., Dalen, D., Faltersack, Z., Nuxoll, A.: Extracting episodic memory feature relevance without domain knowledge. In: Biologically Inspired Cognitive Architectures (BICA), pp. 431–437 (2011)
26. Whitehead, S.D., Ballard, D.H.: Learning to perceive and act by trial and error. Mach. Learn. **7**(1), 45–83 (1991)