Continual Learning through Evolvable
Neural Turing Machines

Benno Lüders, Mikkel Schläger, Sebastian Risi
IT University of Copenhagen
Copenhagen, Denmark
{blde, mihs, sebr}@itu.dk

Abstract

Continual learning, i.e. the ability to sequentially learn tasks without catastrophic forgetting of previously learned ones, is an important open challenge in machine learning. In this paper we take a step in this direction by showing that the recently proposed Evolving Neural Turing Machine (ENTM) approach is able to perform one-shot learning in a reinforcement learning task without catastrophic forgetting of previously stored associations.

1 Introduction

A hallmark ability of humans is to continually learn from experience. However, creating artificial agents that can continuously adapt to novel situations and learn new skills within their lifetime without catastrophic forgetting of previous learned skills remains an unsolved challenge in machine learning [1]. The recently introduced progressive neural network approach [2], which creates a new neural network for each task, takes a step towards continual learning, but requires the manual identification of tasks. Another example is the cascade-correlation algorithm [3] that can incrementally learn new feature detectors while avoiding forgetting. Ellefsen et al. [4] showed that by encouraging the evolution of modular neural networks, catastrophic forgetting in networks can be minimized, allowing them to learn new skills while retaining the ones already learned. In contrast to their approach, in which associations are learned over multiple network presentations, our method can learn new associations in one-shot.

The presented approach builds on the recently developed Neural Turing Machine (NTM) [5] that enables agents to store long-term memories by augmenting neural networks with an external memory component. The differentiable architecture of the original NTM can be trained through gradient descent and is able to learn simple algorithms such as copying, sorting and recall from example data. In the work presented here we build on an evolvable version of the NTM (ENTM) that allows the approach to be directly applied to reinforcement learning domains [6, 7].

While gradient-based methods have shown to scale to highly complex problems, a neuroevolutionary approach (i.e. training neural networks through evolution; [8]) that does not rely on differentiability, offers some unique advantages. First, in addition to the network’s weights, the optimal neural architecture can be learned at the same time. Second, a hard memory attention mechanism is directly supported and the complete memory does not need to be accessed each time step. Third, a growing and theoretically infinite memory is now possible. Together these mechanisms are especially important for a continual learning system, in which the optimal network architecture is not known a priori, memory is ideally unbounded, and a hard attention mechanism only changes specific memory locations at a time, thereby reducing the chances of overriding previously learned information.

In this paper, we demonstrate that by evolving the network’s topology and weights starting from a minimal structure, the optimal structure for a continually learning network can be found automatically. The evolved NTM solutions are able to perform one-shot learning of new associations and
Figure 1: Evolvable Neural Turing Machine. In addition to the NTM specific inputs and outputs, the ANN has domain dependent actuators and sensors. The structure of the network’s hidden layers is automatically determined by the NEAT algorithm.

learn to avoid catastrophic forgetting by using their external memory component. Additionally, we present a new ENTM default jump technique that facilitates the evolution of high-performing solutions. With increasing computational resources it might now be feasible to combine the strength of gradient-based approaches with the ability of evolutionary approaches to optimize mechanisms that are difficult to differentiate.

2 Neuroevolution

The weights and topologies of the networks in this paper are trained by the Neuroevolution of Augmenting Topologies (NEAT) approach [9]. The main challenge with evolving neural networks is that some of the primary principles of Evolutionary Algorithms (EAs), mainly crossover, are difficult to apply to networks with arbitrary topologies. NEAT solves this problem by using historical markers for all genes in the chromosome; every time a new mutation is performed it receives a historical marking, which is a unique identifying number. This way networks with arbitrary topologies can be combined in a meaningful way. Additionally, NEAT begins with simple networks (i.e. networks with no hidden nodes and inputs directly connected to the network’s outputs), and augments them during the evolutionary process through mutations (adding nodes and connections) and crossover. The NEAT implementation in in this paper (SharpNEAT) uses a complexity regulation strategy for the evolutionary process: A threshold defines how complex the evolving networks can be, before the algorithm switches to a simplifying phase, in which it gradually reduces complexity.

In order to enable these evolving ANNs to learn during their lifetime, efforts have been made to allow them to adapt online and learn from past experience [10, 11, 12, 13, 14]. However, a common problem for these techniques is their inability to sustain long-term memory. The weights that encode a particular task are often overridden when a new task is learned, resulting in catastrophic forgetting [15]. The ENTM approach aims to overcome this challenge.

3 Evolvable Neural Turing Machines (ENTMs)

Based on the principles behind the NTM, the recently introduced Evolvable Neural Turing Machine (ENTM) uses NEAT to learn the topology and weights of the neural controller [6, 7]. That way the topology of the network does not have to be defined a priori (as is the case in the original NTM setup) and the network can grow in response to the complexity of the task. The ENTM often finds compact network topologies to solve a particular task, thereby avoiding searching through unnecessarily high-dimensional spaces [6]. Because the network does not have to be differentiable, it can use hard attention and shift mechanisms, allowing it to generalize perfectly to longer sequences in a copy task. Additionally, a dynamic, theoretically unlimited tape size is now possible.
Figure 1 shows the ENTM setup in more detail. The ENTM has a single combined read/write head. The network emits a write vector \( w \) of size \( M \), a write interpolation control input \( i \), a content jump control input \( j \), and three shift control inputs \( s_l, s_0, \) and \( s_r \) (left shift, no shift, right shift). The size of the write vector \( M \) determines the size of each memory location on the tape. The write interpolation component allows blending between the write vector and the current tape values at the write position, where \( M_h(t) \) is the content of the tape at the current head location \( h \), at time step \( t \), \( i_t \) is the write interpolation, and \( w_t \) is the write vector, all at time step \( t \):

\[
M_h(t) = M_h(t-1) \cdot (1 - i_t) + w_t \cdot i_t.
\]

The content jump determines if the head should be moved to the location in memory that most closely resembles the write vector. A content jump is performed if the value of the control input exceeds 0.5. The similarity between write vector \( w \) and memory vector \( m \) is determined by:

\[
s(w, m) = \frac{\sum_{i=1}^{M} |w_i - m_i|}{M}.
\]

At each time step \( t \), the following actions are performed in order: (1) Record the write vector \( w_t \) to the current head position \( h \), interpolated with the existing content according to the write interpolation \( i_t \). (2) If the content jump control input \( j_t \) is greater than 0.5, move the head to location on the tape most similar to the write vector \( w_t \). (3) Shift the head one position left or right on the tape, or stay at the current location, according to the shift control inputs \( s_l, s_0, \) and \( s_r \). (4) Read and return the tape values at the new head position.

In the original ENTM setup \[6, 7\], the tape has an initial size of one and its size is increased if the head shifts to a previously unvisited location at either end. This is the only mechanism to increase the tape size in the original setup. Thus to create a new memory, the ENTM write head has to first jump to one end of the tape, and then perform a shift towards the correct direction to reach a new, untouched memory location. This procedure also requires marking the end of the tape, to identify it as the target for a subsequent content jump. In this paper we introduce a new default jump mechanism that should facilitate creating new memories without catastrophic overriding learned associates. The default jump utilizes a single pre-initialised memory location (initialized to values of 0.5), which is always accessible at one end of the tape. It can provide an efficient, consistent opportunity for the write head to find an unused memory location during a content jump, in which a new memory must be recorded. Once the default memory location is written to, a new default memory is created at the end of the tape. Additionally, we introduce a minimum similarity threshold for the content jumps; if a content jump is performed, and no location on the tape meets the minimum similarity threshold, the head will instead jump to the default location.

Season Task: Agents in this paper are evaluated on the season task \[4\], which tests an agent’s ability to withstand catastrophic forgetting (Figure 2a). During its lifetime, the agent is presented with a number of different food items. Some are nutritious, some are poisonous. The goal of the agent is to learn and remember which food items can be eaten. The task is split into days, seasons and years. Each day, every food item will be presented in a random order. After a certain number of days, the season will change from summer to winter. During winter, a different set of food items will be presented. When the seasons switch the agent now has to remember what it learned during the previous season to achieve the highest possible score.

Each lifetime has three years, and each year has two seasons. Each season has four food items (presented in a random order), of which two are poisonous. We use a random number of days per season.
Figure 3: Evolved Solution Example. (a) The best evolved network quickly learns the correct food associations and reaches a perfect score of 1.0, displaying one-shot learning abilities. Additionally, the associates learned in earlier seasons are still remembered in later seasons. Memory usage during learning is shown in (b). Days 3 and 4 of season 1, and everything past day 2 in season 2 are not shown but solved perfectly. Legend: E-I: ANN output that determines if the food item should be eaten. E-O: ANN inputs from the environment; summer item (1–4), winter item (5–8), reward (9), punishment (10). E-S: Score indicator. TM-W: Write vector. TM-I: Write interpolation. TM-C: Content of the tape at the current head position after write. E-J: Content jump input. TM-S: The three shift values, in descending order: left, none, right. TM-R: Read vector. TM-H: Current head position (after control operations).

(b)
item is saved into its own memory location, summing up to the four locations used. Memorisation is accomplished by connecting the punishment input to the write interpolation output.

5 Conclusion

We showed that the ENTM is able to overcome catastrophic forgetting in a reinforcement learning task, taking a step towards continual learning agents. The evolved ENTM solutions allow successful one-shot-learning during the first day and can perfectly distinguish each food item afterwards. Moreover we augmented the ENTM with a default jump mechanism that facilitates the evolution of learning agents. An important next step is to combine the advantages of this evolutionary-based approach (e.g. automatic optimization of network topologies, hard attention mechanism, variable memory size) with the recent success of gradient-based learning methods. Given the increase in computational resources, this combination starts to be a promising research direction [16].

References