n-task Learning: Solving Multiple or Unknown Numbers of Reinforcement Learning Problems

Author1 (MAG@Macc.Wisc.Edu)
Department of Psychology, 1202 W. Johnson Street
Madison, WI 53706 USA

Author2 (SDJ@Macc.Wisc.Edu)
Department of Educational Psychology, 1025 W. Johnson Street
Madison, WI 53706 USA

Abstract

Temporal difference learning models can perform poorly when optimal policy cannot be determined solely by sensory input. Converging evidence from studies of working memory suggest that humans form abstract mental representations that align with significant features of a task, allowing such conditions to be overcome. We here present n-task learning, an algorithm that utilizes abstract representations to form multiple policies based around a common set of sensory inputs. Inputs to a temporal difference learning algorithm are combined conjunctively with an abstract input that comes to represent attention to a task. The agent completes a dynamic categorization problem that is marked by frequently recurring tasks. The algorithm learns the correct number of tasks as well as when to switch from one task representation to another, even when the input is identical across all tasks. Task performance is shown to be optimal only when an appropriate number of abstract representations is used.

Keywords: reinforcement learning; temporal-difference learning; task switching; input abstraction

Introduction

Functionality typically associated with human and animal working memory has been emulated successfully in computational models that make use of temporal difference (TD) learning algorithms (O'Reilly, Noelle, Braver, & Cohen, 2002; O'Reilly & Frank, 2006; O'Reilly, 2006; Frank, Loughry, & O'Reilly, 2001; Kriete & Noelle, 2011; Kriete, Noelle, Cohen, & O'Reilly, 2013; Niv et al., 2015; Rougier, Noelle, Braver, Cohen, & O'Reilly, 2005; J. L. Phillips & Noelle, 2005; J. Phillips & Noelle, 2006). While these algorithms can perform well in both static and dynamic environments, their success is contingent upon the ability of the state signal to contain all relevant information for assessing the value of future states (the Markov property) (Sutton & Barto, 1998). Frequent changes to the optimal policy that are driven by hidden information can confound learning in TD models (Brunskill & Li, 2015).

While humans may take for granted the ability to solve these types of problems, they frequently occur even in our mundane daily activities. To take a simple example, imagine that you are searching for a coworker, and you know that during work hours this coworker is usually either in her office or in the meeting room next door. As you stand in the hallway looking at the two closed doors, nothing that you perceive from the environment gives you any information that would help you to know her current whereabouts. You must simply try one of the doors. If you fail to find your coworker in the first room, you would then open the other door. In this case you have switched between two policies that have lead to this reward in the past - one which involves opening the office door, and the other opening the door to the meeting room.

In this paper we describe n-task learning (nTL), a biologically inspired algorithm that serves as an extension to any member of the TD family of learning algorithms. nTL allows the base algorithm to better handle scenarios in which the agent is required to switch between several tasks with different optimal policies. We show how the model uses abstract task representations (ATRs) to identify and separate the tasks, increasing the efficacy of the base TD learning model. We also demonstrate how the algorithm is able to learn the number of tasks using only the feedback from the critic.

Background

Biological Precedents for the Model

We draw much of the inspiration for our model from work that explores the trade-offs and affordances of both activation based memories and weight based memories. Having multiple mechanisms for storage affords great flexibility in meeting memory demands, and we believe that many of the current computational learning models could benefit from these dual roles.

The nTL algorithm utilizes ATRs to inform policies. These ATRs can be thought of as a kind of filter through which the agent currently perceives the environment. Throughout this section we provide evidence for a biological analog to these ATRs, along with other details relevant to the model.

Representation Learning In multidimensional environments, dimensional attention helps to reduce complexity by facilitating the formation of more simplified representations based on a subset of the original dimensions. Niv et al. (2015) present evidence that humans engage in representation learning, homing in on task relevant features in an attempt to reduce complexity of a state representation. Of the models tested against human performance, they found the closest match to be a type of "feature reinforcement learning" (fRL), in which the value of a stimulus, S, is calculated as the sum of the weights of its component features, $W(f)$:

$$V(S) = \sum_{f \in S} W(f) \tag{1}$$

Weights for the component features are then updated at each step according to the standard temporal difference update, with weights for irrelevant features eventually decaying to zero. fMRI images showed that the frontoparietal attention control network of the brain was active toward the beginning of a task, when subjects were entertaining different alternatives for the relevant dimension. Later in learning, when subjects concentrated on a single feature, this region showed less activity. We will circle back to this result later in our discussion.

We find the nTL model to be similar to the fRL model from Niv et al. in that the stimulus values that make up the state signal are combined disjunctively before being passed into the value function (in our case the Q function for SARSA). In this way each component feature contributes to the estimated value, and the error for the trial is distributed among the state features. Trials that share one or more state features with a previous trial will have an estimated value that has been adjusted by the outcome of this previous trial.

Top-down Support in Working Memory O'Reilly et al. (2002) give support for the argument that working memory is responsible for flexibly updating goals. The authors argue that perceptual processing and action selection are influenced by representations that are held in working memory, providing what they describe as "top-down support" or "biasing". The inability to rapidly switch actively maintained representations results in perseveration on previous learning, as learning must then be accomplished through slower weight based updates. In the same study the authors attempt to show that actively maintained representations in the prefrontal cortex (PFC) are organized by level of abstraction. They cite as evidence behavior exhibited by human patients with frontal damage and experiments performed on monkeys with lesions in this region in the brain.

A later work by Rougier et al. expands on this idea by attempting to provide a model based on PFC-midbrain interaction that develops "abstract rule-like PFC representations" (Rougier et al., 2005). This model was trained on multiple tasks where stimuli comprised of features from several dimensions were presented, and reward for each task was determined by a single dimension. The model was contrasted with others lacking various anatomical structures and mechanisms. The authors showed that units in the "full PFC" network came to represent all of the features that made up a particular dimension, while other networks tended to form a representation for each stimulus.

Dynamic Gating We have previously discussed the benefits of being able to actively maintain memories; just as critical is the ability to update concepts that are currently held in memory. It is theorized that this is accomplished in the brain via a system in which the PFC exhibits active maintenance of memory representations and the basal ganglia act as a gate, allowing representations in the PFC to be updated (Frank et al., 2001; C. H. Chatham & Badre, 2015; C. Chatham, Frank, & Badre, 2014; Rougier et al., 2005; Kriete et al., 2013; Kriete & Noelle, 2011). It has been shown that working memory is updated when levels of the neurotransmitter dopamine are phasically elevated (Frank et al., 2001; C. H. Chatham & Badre, 2015; Kriete & Noelle, 2011; Rougier et al., 2005; Niv et al., 2015; Rafati & Noelle, 2015). When a reward is expected but not delivered, the resulting negative error signal prompts an update to working memory contents (O'Reilly et al., 2002). In addition to flushing retained memory representations and allowing working memory to store new external stimuli, learning is believed to be limited in the presence of a large negative error signal (C. H. Chatham & Badre, 2015; O'Reilly, 2006), as can be modeled by the gain parameter to a sigmoidal neural activation function (Frank et al., 2001; O'Reilly & Frank, 2006).

Relation to Machine Learning Models

Although the model in this work has some aspects in common with models from other machine learning domains, nTL has a unique property that sets apart from these - the ability to self-monitor and react appropriately based only on reward feedback. The problem that nTL intends to address is one in which contextual cues offer no information that can be used to determine an appropriate action selection policy, such as the Wisconsin Card Sorting Test (WCST). While we see this type of experiment used frequently in psychological literature, we have seen no machine learning models that attempt to solve this problem. In this section we contrast our algorithm with related machine learning works.

Hierarchical reinforcement learning (HRL) bares the most similarity to nTL. As an example we use Sutton et al. (1999), in which "options" are used as a form of temporal abstraction (Sutton, Precup, & Singh, 1999). In this case an option is selected and influences action choices for all time steps until a goal state is reached or the option expires after a predefined number of time steps. An ATR is an outer abstraction that effects the policy over actions. However, in nTL there is no explicit definition of this outer abstraction within the algorithm; initiation and termination of an ATR are contingent upon task performance rather than predefined states. The ability to learn tasks without the use of defined goals is the main contribution of this work.

Option-based HRL models have performed well on complex tasks, but these models are not suited for scenarios in which state representations offer no cues for choosing among options, i.e. when the function for selecting an option is independent of state input. HRL is typically used to improve performance on temporally extended tasks with sparse rewards by dividing them into sub-tasks. nTL, in contrast, is appropriate when multiple episodic tasks are periodically "shuffled", and nothing in the environment is indicative of the task type or duration.

Another related reinforcement learning model is multi-objective reinforcement learning (MORL) (see Van Moffaert & Now 2014 for example). MORL algorithms work best when there are multiple rewards for competing objectives. Because there is only one objective at any given time in our task, these algorithms would not be a good fit. However, nTL could be used to remap the objective weighting in a dynamic way, allowing MORL to be applied to the task studied here.

Long short-term memory (Hochreiter & Schmidhuber, 1997), or LSTM, has demonstrated success in capturing long-term dependencies and has recently grown in popularity across domains. It is for these reasons that we have chosen it as a state-of-the-art competitor for nTL. Because the response that satisfies the categorization rule is never revealed in the WCST, we have trained the LSTM model with an altered version that minimizes error against the correct response at each trial rather than learning with a reward signal. Although this is not true to WCST form, it does allow a basis for comparison against supervised learning algorithms.

Methods

Experimental Protocol

Our experiment is similar to the dimension selection task described in (Rougier et al., 2005). The agent is presented with a stimulus consisting of f features, selected at random from d dimensions, and is prompted to select one of the features. Feedback is given after each action selection based on whether or not the selected feature matches a categorization rule. The rule corresponds to one of the five dimensions from which the features are drawn, and no cues are given to indicate what this rule is. Correct answers are rewarded by a constant amount, $r_g = 1$. Incorrect answers incur no penalty, $r_d = 0$. After a predetermined number of consecutive correct responses, l, the round is considered learned, and the rule changes (in the style of a WCST). The rule is selected at random, and always differs between two consecutive rounds. Figure 1 shows an example round.

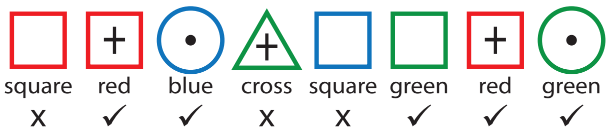

Figure 1: Shown here is an example round of the experiment with l set to three using stimuli composed of three features drawn from three dimensions: color, shape, and fill. The round consists of eight trials. The stimuli are shown in the top row, and the selected action is shown in the middle row. The bottom row designates a correct trial with a check mark, and an incorrect trial with an "x". The rule for this round was set to "color".

Table 1: Parameter Descriptions and Values

Name	Value	Description
l	8	Number of consecutive correct trials needed to complete a round
d	5	Number of dimensions for the task stimuli
f	5	Number of features in each dimension
n	1024	Size of HRR vectors
b	1	Bias value for Q and A function networks
r_g	1	Reward for a goal state (correct action)
r_d	0	Reward for a non-goal state (incorrect action)
ε	0.005	Probability that the SARSA algorithm will make a non-greedy action choice
α_q	0.05	Learning rate for the Q function update
α_a	0.0075	Learning rate for the A function update
α_t	0.002	Learning rate for the t threshold update
a	0.5	Task add threshold for dynamic nTL (5 tasks)

It is important to note that the distinguishing feature of this problem is that is composed of several tasks which each must be learned without the use of environmental cues and then remembered while completing the other tasks for success. If something in the environment is present to help the agent determine which rule is currently in effect then the problem becomes a contextual bandit problem. If the tasks are not repeated then the problem becomes a non-stationary bandit problem. Without a way to associate learning with a task representation the agent effectively treats the WCST as a non-stationary bandit problem, and performance suffers due to perseveration after rule changes (as is later shown).

The decision to directly align categorization rules with the dimensions of the stimuli was made in order to more easily draw conclusions from existing cognitive science research. Categorization rules represent the set of action choices that lead to reward, and may be chosen arbitrarily.

Key Terms

In order to remove any ambiguity concerning the testing protocol, here we define some key terms:

- In a particular experiment some dimensions may never be used for the categorization rule, and serve only as distractors. We refer to the number of distinct rules that are used throughout the experiment as the number of *tasks*.

- A *trial* consists of one presentation of a stimulus to the agent, the choosing of an action by the agent, and a reward value given to the agent by the critic. This is equivalent to a single time step in the reinforcement learning framework.

- A *round* consists of all of the trials completed by the subject during one completion of a task, from the time a new categorization rule is put into effect to the time the subject has completed l consecutive correct trials.

- When the model substitutes an ATR that is in working memory with one that is not in working memory we call this a *task switch*.

Holographic Reduced Representations

Unitary holographic reduced representations (HRRs) (Plate, 1995) are used in our experiments to encode state and action inputs. With HRRs, features are distributed over the width of an entire vector of n elements rather than tied to a particular position or index in a vector. Conjunction and disjunction of input features is accomplished mathematically by circular convolution and addition of vectors respectively. As a result, relationships between concepts are represented without increase in dimensionality.

Although we use an HRR framework for encoding in this work we do not believe the results obtained are contingent upon this choice, and we see no reason why alternate encodings such as vectors (Mitchell & Lapata, 2010), tensors (Papalexakis, Faloutsos, & Sidiropoulos, 2016), or spatio-temporal encodings (Hummel & Holyoak, 1997) could not produce similar results.

Working Memory Computational Model

In order to see the consequences of the ATR mechanism, we will first describe the working memory model as it functioned prior to the addition of ATRs, which are described in the following subsection. The state (the feature set that comprises the current trial) is in the form of a single HRR, which is the result of disjunctively combining the feature vectors for components of the stimuli. In our experiments the action choices correspond directly to the features in the environment. To differentiate between a particular feature in the state role (e.g. "seeing red") and the same feature in the action role ("selecting red"), these corresponding states / actions are comprised of different HRRs.

At each trial the model updates the assessed value of choosing a particular action while in a particular state through use of the SARSA algorithm. The Q function for SARSA is approximated using a single-layer perceptron neural network with a linear activation function. Because of this architecture, the output for the network is simply the dot product of the input HRR and a weight vector plus a scalar bias term, b, as shown in Equation 2. The weight vector is initialized in the same way as HRRs, and a bias term is set to the reward level that will be received upon reaching the goal state. Setting the bias in this way has the effect of encouraging exploration via optimistic initial values ("optimistic critic"). To counter the eventual decrease in exploration that comes from overcoming the initial optimism we also implement an ε-soft policy that makes non-greedy decisions a small fraction (ε) of the time.

An action is selected by forming the conjunct representation of state with each candidate action for the trial, and using the resultant representations as inputs into the Q function. The action that yields the greatest value, m, is then chosen. This can be formulated as:

$$m = \underset{c \in C}{argmax}((s \wedge c) \cdot w_q + b) \tag{2}$$

where s is the current state representation, C is the set of all candidate action choices for the current trial, and w_q is the weight vector for the Q function neural network. At each trial the reward is used to update weights for the Q function. In order to make learning more stable a log-modulus transformation (John & Draper, 1980) is applied to the error during updates to the Q function, A function (introduced in the next section), and t threshold (introduced in next section). This transformation mitigates learning instability due to relatively large errors (data not shown):

$$\Delta w_i = \alpha_q [sgn(\delta) * log(|\delta| + 1) * (s \wedge m)_i] \tag{3}$$

In the above equation, w_i indicates the value of the weight vector at index i, α_q is the learning rate, δ is the error, and $(s \wedge m)_i$ is the value of the HRR input vector (the eligibility trace) at index i. Although we are using SARSA to learn a policy for action selection, the experiment does not model a temporally extended task. Since all feedback is relevant to only a single trial, we have set the λ and γ TD parameters to zero, and all updates are treated as goal state updates.

n-task Learning Algorithm

As mentioned earlier, nTL can be viewed as an extension to a standard TD learning algorithm, which we call the base algorithm. We have used SARSA as the base algorithm for our experiments, but we believe other TD learning algorithms could be substituted. At the heart of this approach is the idea that any input can be bound to a task by forming the conjunct of the original input and another input that uniquely identifies an ATR. The ATR inputs are created arbitrarily, and encoded in the same way as the input features. The algorithm requires a function, A, to keep keep track of the value of each ATR. For this we simply maintain and update a vector of values that are mapped to the ATRs. If desired, a neural network could also be used to model the ATR values.

Before any input is fed into the base algorithm the input is conjunctively joined with the ATR representation that is currently in memory, atr. In this way a single input can take on multiple values by being bound to different ATRs and presented to the base algorithm. No alterations to the base algorithm are required in order for this approach to work. Our action selection equation now becomes:

$$m = \underset{c \in C}{argmax}((s \wedge c \wedge atr) \cdot w_q + b) \tag{4}$$

The weight update is modified in a similar manner; where before in Equation 3 we had only $s \wedge m$ to represent the state / action input to SARSA, now we must include the selected ATR. The new input is $s \wedge m \wedge atr$. In short, $Q(s,m)$ becomes $Q(s,m,atr)$.

Reward feedback from each trial is used to update the value of the current ATR. The error used for this update is simply the TD error for the ATR value function. In the below equation, A is the function determining the ATR values, α_a is the learning rate for ATRs, and δ is $r - A(atr)$. Note that in our experiments, the A function also uses optimistic initial values, so all ATRs start with a value equal to the goal state reward:

$$A(atr) \leftarrow A(atr) + \alpha_a [sgn(\delta) * log(|\delta| + 1)] \tag{5}$$

Trials that result in a task switch do not incur an update to the A or Q functions. In this way the current ATR is not penalized for a task change that is external to the agent. Although we do not give data here, this adaptation leads to A values that are more stable (show less fluctuation) and that converge to the average reward values for trials within the tasks.

The model determines when to switch to a new ATR via a threshold, t. When the TD error ($\delta = r - Q(s, m, atr)$) is less then t, this signals to the model that the current ATR is not well suited to handle current input, and the next sequential ATR is subbed in for the current one. This t value is first initialized to negative one times the reward for the goal state, and is updated at each trial using the TD error from the Q function, where α_t is the learning rate for t:

$$t \leftarrow t - \alpha_t[sgn(\delta) * log(|\delta| + 1)] \quad (6)$$

Static vs Dynamic n-task Learning In the case that the number of tasks is known ahead of time, the number of ATRs can be set explicitly; we refer to this as static nTL. In many cases the number of tasks is unknown, or changes with time. In dynamic nTL the number of ATRs is set automatically based on task performance. To accomplish this an additional task add threshold, a, is needed to determine the number of ATRs to maintain. Whenever a task switch occurs A values for all ATRs are averaged. If this mean value falls below a then a new ATR is added, and both A and t are reinitialized. Unlike t, the a threshold is constant. Although we find it likely that any biological analog for this threshold would be dynamic, we have not yet found a satisfactory way to model this behavior.

The model always starts with a single ATR in dynamic nTL. The decision to start with one ATR and grow toward the optimal number aligns with research showing that humans and animals assume a simple task structure in dynamic categorization tasks in which reward is determined by a single dimension, and move on to more complex hypotheses only after exhausting all simple ones (Shepard, Hovland, & Jenkins, 1961).

Additional Details for Experiment Setup

A list of parameter values is provided in Table 1. Parameters for all models were chosen by minimizing the mean number of incorrect trials over 100 tasks for 100 random runs of the experiment. All experiments using the nTL models were conducted using R version 3.4.0. The source code used for these experiments can be downloaded freely from: `url redacted for anonymity`

The LSTM model was built using TensorFlow version XX. This model performed best with a learning rate of 7.0, and a

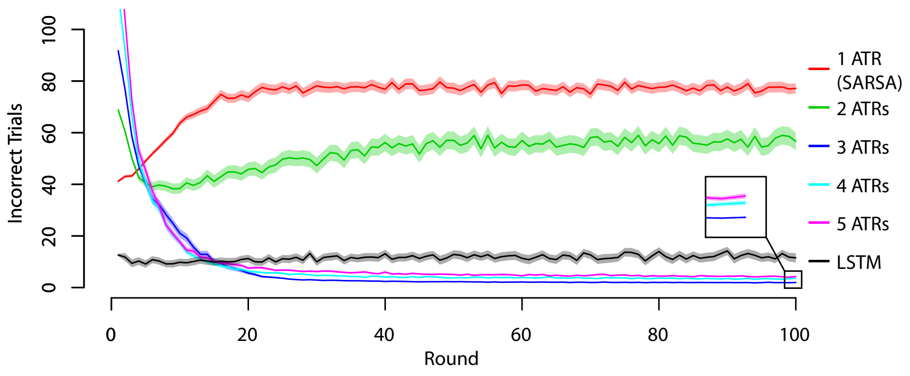

Figure 2: Static nTL - Shown here are the average number of incorrect trials taken at each round. Mean values for 1000 runs are shown by the solid lines, with shading to show a 95% confidence interval. The number of tasks, N, is three. Performance will converge to a theoretically optimal $N/2$ incorrect trials per round on average when N ATRs are used.

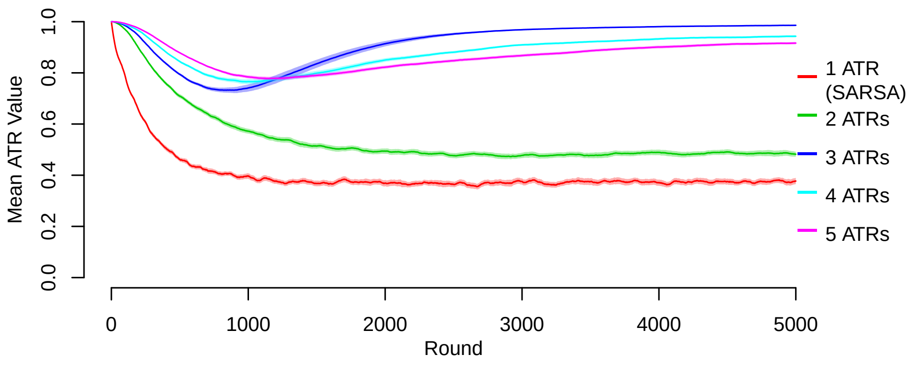

Figure 3: Static nTL - Mean ATR (A function) values for 100 runs are shown by the solid lines, with shading to show a 95% confidence interval. The number of tasks for this experiment was three.

state size of 1. We found that results remained the same for all state sizes in the tuning range. The model was trained after each trial with a sliding window of the most recent ten trials. Tuning ranges were as follows, with a step size of 0.05: learning rate (0,10], state size [1,20], sliding window size [1,40].

Results

Two experiments were conducted, both using the previously described protocol. In the first experiment the appropriate number of ATRs is known a priori, so we use static nTL. In the second, the number of ATRs is learned using dynamic nTL.

Experiment 1 - Static n-task Learning

Figure 2 shows how task performance changes as a function of the number of ATRs being used. Three tasks were present in this experiment. Although a statistically significant difference is shown between all results, we see that having too few ATRs is much more detrimental to performance than having too many. When compared to the case when the number of ATRs was equal to the number of tasks (3 ATRs), having one too few (2) ATRs resulted in 2782% more incorrect trials per round after 100 rounds, whereas having one too many (4) ATRs resulted in only 81% more incorrect trials per round. The LSTM model resulted in a 487% increase in incorrect trials, and traditional SARSA gave a 3827% increase.

By setting the number of ATRs to one, we simulate the behavior of an agent that is not capable of learning dimensional representations. This is equivalent to standard TD learning, in this case SARSA. We would expect such an agent to perseverate on previously learned features when a task switch occurs, and the results confirm that perseveration occurs. When only one ATR is available, many incorrect trials are taken in each round (see Figure 2). When a rule change occurs, action choices that were valuable in the previous round are tried first (see Figure 4a). Only after a period of unlearning the previous external task (see Figure 4b) does the model begin to select new actions.

From Figure 3 we see that values for the A function converge to the goal state reward for all ATRs only when the number of ATRs is equal to the number of external tasks. Again we see that having too many ATRs has less impact on mean ATR value than having too few.

To obtain the results shown in Figure 4 a neural network was trained using the same error value that was used to update the A function. Input consisted of the selected action for the trial conjunctively joined with the current ATR. While this network was not used directly in the nTL algorithm, it allows us to see how the ATRs influence the perceived value of selecting each feature in the state space. We see from Figure 4c and 4d that when the number of ATRs is equal to the number of external tasks, each ATR comes to represent the dimension for one external task. This is because after the initial learning and exploration period each ATR is used only for the subset of trials that correspond to an external task. If an ATR is used on a trial that results in a task switch, no weight updates take place.

Experiment 2 - Dynamic n-task Learning

For this experiment we illustrate dynamic nTL functionality with an example using five tasks (see Figure 5). The trials where t drops to the starting value of negative one indicate that the mean ATR value exceeded a, and an ATR was added. When five ATRs were present, the mean ATR value remained above a during initial exploration, and eventually converged to the value of the goal state reward. Because the first experiment provided much of the data needed to compare against models lacking a mechanism for the formation of ATRs, no additional models were used for comparison in this experiment.

Discussion

Discussing the nTL model in terms of a human actor allows us to more easily connect to previously discussed biological models. When the agent expects to receive a reward and none is given, the resulting negative error signal cues the agent to try a new strategy (switch to a new ATR), and no learning takes place. There is initially a period of rapid task switching as the agent gives many incorrect responses due to exploration and lack of learning. After a time the estimated feature selection values that are associated with each ATR stabilize, and internal task switching occurs only in response to a true external task switch. This period of rapid task switching could correspond to frontoparietal attentional control in the brain, which was shown in (Niv et al., 2015) to occur when task representations were not fully formed.

The ATRs influence the agent's thinking about the current trial through a mechanism of top-down support (O'Reilly et al., 2002). If we take a single trial (state and action) and associate it with two different ATRs, the agent will have a different assessment of value for each. Let us suppose that the agent is presented with a small red circle as stimulus, and the action candidate "select red". When the first ATR is present in working memory this action may have a high value, but when the second ATR is in working memory the value is low. We can then conclude that the first ATR has come to represent a task in which selecting red when seeing these stimuli leads to reward, and the second to represent a task in which seeing red does not lead to reward (this second ATR may have been used for a shape or size categorization task). In this way the switching of ATRs effectively becomes a filter for possible actions. The agent attends to a different subset of actions with each successive trial, iterating through hypotheses that have lead to prior success as it attempts to find one that fits the task.

We have shown that an agent with only a single ATR will perseverate when a rule change switch occurs. In this case there is a period during which the previously learned task is unlearned, followed by a period in which the new task is learned. This scenario simulates learning without activation based memories, where all learning must be accomplished through weight updates (O'Reilly et al., 2002).

When the number of ATRs matches the number of tasks, the mean of the agent's estimated values for the ATRs (the A function) converges to the goal reward value of the tasks. When too few ATRs are present this mean declines to a value that is below the goal reward. Using these two observations, we can set a threshold (a) that acts as a cutoff point for overall ATR performance. When the mean value falls below this threshold the agent increases the complexity of its thinking by adding another ATR, until the number of ATRs is equal to the number of tasks.

One notable deviation from human-like thinking in the dynamic nTL model is that all previous learning is discarded when an ATR is added. The reason for "resetting" when we reach this point is to keep the task switch threshold, t, in a range that will cause ATRs to be switched appropriately. If previous Q learning is retained (by leaving intact the Q neural network) and t is reset, then t will remain too low, resulting in too few task switches for ATRs to effectively represent the tasks. If previous Q learning is retained and t is not reset, then t will climb too high, resulting in a task switch for every trial. It is for this reason that we both reinitialize the Q weight vector and reset the task switch threshold when tasks are added.

While we believe nTL can be used to great benefit for dynamic tasks such as the one used in this experiment, we recognize that it is not appropriate for all reinforcement learning scenarios. Specifically, we have only tested the algorithm in the case where reward is constant and able to be achieved at each time step, the task distribution is uniformly stochastic, the number of tasks does not change, and all features are used for a single categorization rule at most. A model that is able to accommodate variable/probabilistic rewards, temporally extended tasks, adversarial task distributions, and the introduction and removal of tasks could be extremely useful. We would be interested to see more research on the means by which humans come to learn the number of tasks present in a given scenario, both to provide biological inspiration for further work and to assess plausibility of the current model.

Future work could also leverage environmental information for task switching decisions. Task switching currently proceeds serially, which was appropriate in this work because state information offers no information that could help the agent decide which task it is trying to solve. We can imagine many scenarios where environmental features could be indicative of a particular task, as in associative search tasks, and where previous learning may help to speed up learning on new but similar tasks, as in transfer learning.

We noticed that greater task complexity (features per task) and a higher number of tasks both lead to more instability in the value of ATRs. Another line of work could attempt to define limits to the task complexity and number of tasks that this model is able to accommodate.

References

Brunskill, E., & Li, L. (2015). The online coupon-collector problem and its application to lifelong reinforcement learning. *CoRR*, *abs/1506.03379*.

Chatham, C., Frank, M., & Badre, D. (2014). Corticostriatal output gating during selection from working memory. *Neuron*, *81*(4), 930–942.

Chatham, C. H., & Badre, D. (2015). Multiple gates on working memory. *Current Opinion in Behavioral Sciences*, *1*, 23–31.

Frank, M. J., Loughry, B., & O'Reilly, R. C. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model. *Cognitive, Affective, & Behavioral Neuroscience*, *1*(2), 137–160.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735.

Hummel, J. E., & Holyoak, K. J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, *104*(3), 427.

John, J. A., & Draper, N. R. (1980). An alternative family of transformations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *29*(2), 190–197.

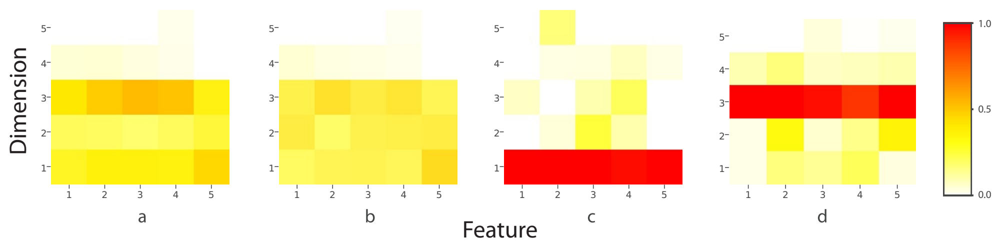

Figure 4: Shown here are feature selection values for all features in the state set. a and b show results from trials that take place after 100 rounds of training using traditional SARSA (the single ATR case) to learn three tasks. a corresponds to a trial at the beginning of the round. We see that action values along dimension three are estimated to be most valuable, indicating that this dimension was used as the previous rule. By b, a trial later in the same round, no dimension appears to be more valuable than any other. By contrast, when three ATRs are used we see the formation of clear dimensional representations. c and d show feature selection values using the first and second ATRs respectively for an arbitrarily chosen trial after 100 rounds of training.

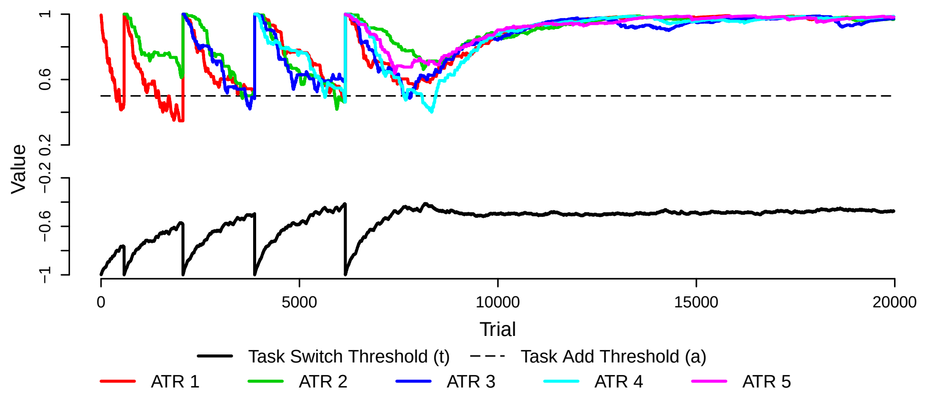

Figure 5: Dynamic nTL - ATR values for a scenario in which the number of task representations adapts to achieve optimal performance. The number of external tasks in this experiment was five.

Kriete, T., & Noelle, D. C. (2011). Generalisation benefits of output gating in a model of prefrontal cortex. *Connection Science*, *23*(2), 119–129.

Kriete, T., Noelle, D. C., Cohen, J. D., & O'Reilly, R. C. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, *110*(41), 16390–16395.

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, *34*(8), 1388–1429.

Niv, Y., Daniel, R., Geana, A., Gershman, S. J., Leong, Y. C., Radulescu, A., & Wilson, R. C. (2015). Reinforcement learning in multidimensional environments relies on attention mechanisms. *Journal of Neuroscience*, *35*(21), 8145–8157.

O'Reilly, R. C. (2006). Biologically based computational models of high-level cognition. *Science*, *314*(5796), 91–94.

O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, *18*(2), 283–328.

O'Reilly, R. C., Noelle, D. C., Braver, T. S., & Cohen, J. D. (2002). Prefrontal cortex and dynamic categorization tasks: Representational organization and neuromodulatory control. *Cerebral Cortex*, *12*(3), 246–257.

Papalexakis, E. E., Faloutsos, C., & Sidiropoulos, N. D. (2016). Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Trans. Intell. Syst. Technol.*, *8*(2), 16:1–16:44.

Phillips, J., & Noelle, D. (2006). Working memory for robots: Inspirations from computational neuroscience. In *Proc. from 5th intl conf on development and learning.*

Phillips, J. L., & Noelle, D. C. (2005). A biologically inspired working memory framework for robots. In *Robot and human interactive communication, 2005. roman 2005. ieee international workshop on* (pp. 599–604). IEEE.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641.

Rafati, J., & Noelle, D. C. (2015). Lateral inhibition overcomes limits of temporal difference learning. In *Proceedings of the 37th annual meeting of the cognitive science society* (pp. 1931–1937).

Rougier, N. P., Noelle, D. C., Braver, T. S., Cohen, J. D., & O'Reilly, R. C. (2005). Prefrontal cortex and flexible cognitive control: Rules without symbols. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(20), 7338–7343.

Shepard, R. N., Hovland, C. I., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs: General and Applied*, *75*(13), 1.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1). MIT press Cambridge.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*(1-2), 181–211.

Van Moffaert, K., & Now, A. (2014). Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, *15*(1), 3483–3512.