# Biologically Inspired Task Abstraction and Generalization Models of Working Memory

By

Mike Jovanovich

A thesis submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Middle Tennessee State University

December 2017

Thesis Committee:

Dr. Joshua L. Phillips, Chair

Dr. Sal Barbosa

Dr. Cen Li

## ABSTRACT

We first present a model of working memory that affords generalization. By separating stimuli in such a way that filler representations may flow through the model based on the state of gates, which are opened or closed in response to role signals, an action selection network is afforded the ability to learn a response to fillers that is independent of the roles in which they were encountered. Next, we present n-task learning, an extension of temporal difference learning that allows for the formation of multiple policies based around a common set of sensory inputs. In order to allow for state inputs to take on multiple values, they are joined with an arbitrary input called an abstract task representation. Task performance is shown to converge to optimal for a dynamic categorization problem in which input features are identical across all tasks.

## TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER I.

## INTRODUCTION

Working memory, which plays a fundamental role in learning, language comprehension, goal planning, and fluid general intelligence, continues to be the subject of extensive study, and the progress made in the last several decades has greatly increased our understanding of how it can play such a key role in those behaviors we typically think of as distinctly human. Much of today's research attempts to define the structure of mental representations, the mechanisms by which these mental representations are manipulated within working memory, and the limits such a neural architecture might impose.

In this paper we present two computational models, each addressing a different function that working memory is known to subserve. The ability to generalize to new scenarios based on previously learned information remains a difficult challenge for machine learning models, with relatively few solutions. Our first work (Chapter II) provides a framework to accomplish role-filler binding that is inspired by the biological mechanisms of working memory gating. Additionally, temporal difference learning models 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. In our second work (Chapter III), we present n-task learning, an algorithm that utilizes abstract representations to form multiple policies based around a common set of sensory inputs.

# CHAPTER II.

# GENERALIZATION AND SYMBOL-BINDING USING WORKING MEMORY GATES

## Introduction

Humans display a remarkable ability to act appropriately when encountering novel situations. The ability to understand a familiar concept when that concept is encountered in a new context allows us to make sense of a world which would otherwise present an overwhelming array of feature combinations. We use these faculties of generalization, for example, when learning a new card game, finding onions at an unfamiliar grocery store, or understanding a strange grammatical phrasing when reading a book. The creative scientific insight and the artistic vision that become manifest in the mind come from generativity, the manipulation of familiar mental structures to create new ones.

Because role-filler bindings serve as the first step to building full fledged structured representations [5], a neural architecture that is able to generalize must have a way to bind roles to fillers. In this paper we examine the implications of two working memory architectures that use gates to control the flow of representations into and out of the model. Background information on the biological mechanisms that underlie working memory gating is provided in the following section. We repeat the sentence presentation task from Kriete et al. (2013), contrasting our model and results with this study. Finally, we derive conclusions about the limitations and affordances of the various model architectures, and consider the biological implications of each.

## Working Memory Gates

Study of the brain components involved in working memory, specifically the interactions between the prefrontal cortex (PFC) and basal ganglia (BG), offers many insights into how we might model temporally extended reinforcement learning problems. Models in which the BG act as a critic to control updating of PFC contents (input or maintenance

gating) have been shown to successfully mimic human performance [7, 21, 22]. More recent models show that the BG can be used to control output gating from the PFC, driving action selection and solving the problems of temporal and structural credit assignment for complex tasks [13, 14, 17, 18]. Chatham, Frank, and Badre offer additional human evidence for both kinds of gating, using PET and fMRI data to support this theory [3, 4].

Anatomical studies of the PFC show isolated stripes of neurons with strong inter-stripe neural connectivity but weak intra-stripe connectivity [13, 14, 17, 18]. These PFC stripes are connected to distinct regions of the BG in a circuital pattern. When no interference is present, working memory neurons retain their activation states through patterns of recurrent neural firing. In response to changes in dopamine levels, the BG may open pathways that cause the PFC stripes to be influenced by external stimuli, updating the contents of working memory. Likewise, the contents of PFC stripes can be selectively released to affect activity elsewhere in the brain. This selective updating and releasing of representations is critical to the performance of complex tasks, as shown in the works of O'Reilly et al., cited above.

The work presented in this paper parallels the work of Kriete et al. (2013) both in its aim to reinforce theories of generalization based on output gating and to explore the limitations of various gate-based architectures when applied to different generalization scenarios. While Kriete et al. attempt to implement a neural architecture that is biologically accurate, however, we have greatly abstracted and simplified the model. In their work, for example, each brain component of the working memory system is implemented with its own set of neurons, and neural activity within the system is used to control gates. Representations are also in the form of distributed patterns of neural activity, and learning is accomplished through the PVLV [9] algorithm. In this work the PFC stripes act as containers that hold an exact copy of external stimuli representations, and temporal difference learning is used to train the various gates. Biologically inspired functional components of working memory are preserved, while unneeded biological implementation details have been removed. The utility

of such an approach has been shown in prior work where complex task scenarios can be handled well by models without intricate knowledge of the underlying neurobiology, and particular interest has been demonstrated within the field of robotics [2, 6, 8, 21, 22, 29].

## Methods

### Sentence Presentation Task

Experiment design reflects the sentence presentation and query task from [13]. A three word sentence was presented one word at a time (designated as "store" trials), then queried with a sentential role (the "query" trial). Each episode, therefore, consisted of four timesteps: three stores and a query. Table 1 shows the inputs for one example sentence. The query trial was considered correct if the chosen word (filler) for this trial matched its assigned role in the sentence. To form the sentences, words were chosen for each of the three roles at random from a set of ten with replacement. From the set of possible sentences (as constrained by the testing protocols below), 200 unique sentences were chosen for training and 100 different sentences for testing. In each epoch the model was first trained on the training set, then the percent accuracy was measured against the test set. Epochs were repeated as many times as needed until the model achieved an accuracy of 95% on the test set. Four protocols, the first three of which were taken from Kriete et al. (2013), were used to determine generalization capability:

- **Standard Generalization** For this test, each word was used in each sentential role during training. This test measures the ability of the model to interpret novel combinations of previously experienced role-filler pairs.

- **Spurious Anticorrelation** Anticorrelations, in which a pair of words never appear in the same sentence, were introduced to the training set. Each word from the first half of the set was paired with a word from the second half to make five anticorrelations from the ten words in the set (e.g. "boy" was never paired with "ball"). The test set was then constructed to include a pair of these anticorrelated words in each sentence.

Traditional machine learning techniques, such as neural networks, can sometimes fail when anticorrelations are introduced because the network uses previously learned correlations to predict an output.

- **Full Combinatorial** To test the performance of the model when encountering novel role-filler pairs, two words were never used in the "agent" sentential role during training. The test set always included one of these two words in the agent role, and this role was always queried.

- **Novel Filler** Humans are faced with many scenarios in which the role can be identified, but the filler of this role is unfamiliar. When reading a paper, for example, we have no difficulty pointing out the author, even if we have never before seen the name. In this test, we withhold two of the fillers from the testing set. These two fillers are then presented and queried at each trial of testing.

## Models

All tests were run using three models, shown in Figure 1: one with only an input gating mechanism (IG model) and two with both input and output gating (IGOG and IGOG-AA models). All models have a working memory maintenance layer, $wm_m$. The IGOG models have an additional working memory output layer, $wm_o$. Both IG and IGOG make use of a single layer multiclass perceptron neural network with a linear activation function and a softmax output to map working memory input to an action choice. The temperature for the softmax function was set to 0.125. These networks were trained at each timestep using the delta rule for learning. For "store" trials, the network was trained to select the action corresponding to the filler that was presented.

The contents of the working memory layers are used by the various models to arrive at an action selection decision. In the IGOG models, working memory is represented by the disjunction of all stripes. In order for the action selection layer to determine a mapping of

stripe contents to a particular stripe in IG, each stripe's contents is conjunctively joined with a unique stripe identifier, denoted below by *sid*:

The decision whether to open or close is made by each gate independently based on internal input, as shown in Table 1. The SARSA form of temporal-difference learning was used to choose a gating action, $a$. $Q$ values were approximated using a single layer perceptron neural network:

$$wm_o^{igog} = s_1^o \vee s_2^o \vee ... \vee s_n^o \tag{1}$$

$$wm_m^{ig} = (s_1^m \wedge sid_1) \vee (s_2^m \wedge sid_2) \vee ... \vee (s_n^m \wedge sid_n) \tag{2}$$

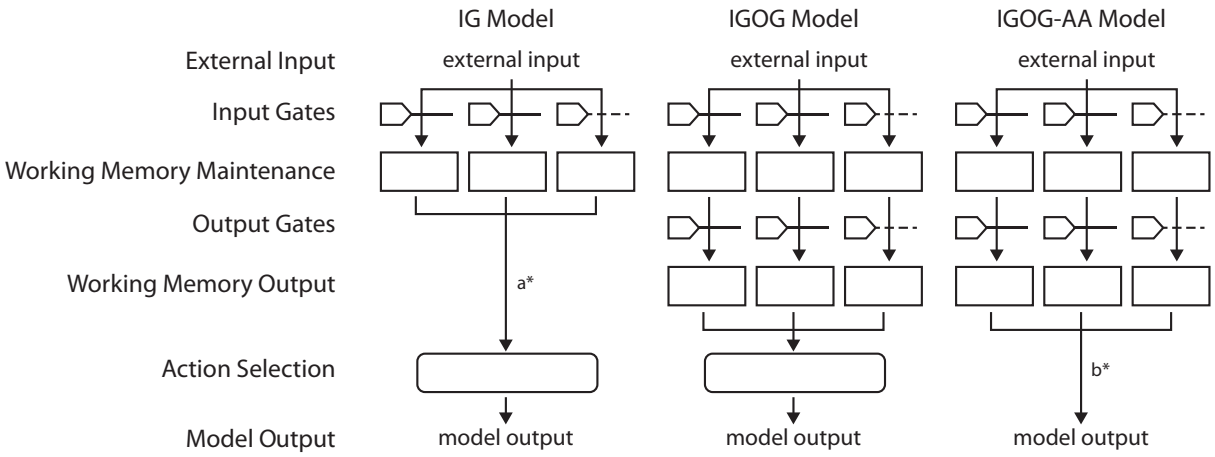


Figure 1: This diagram shows the architectures for the three working memory models. Rectangles represent working memory stripes, which can either be empty or contain an external (PFC) input representation. Flow of external inputs through the model is controlled by the gating layers. Input and output gates can be in either an open or closed state, as indicated by solid or broken lines respectively. The action selection layer (rounded rectangle) is a neural network that maps input to an action choice. $a^*$: In the IG model, the contents of each stripe is conjunctively joined with a unique stripe identifier when used as input to the action selection layer. $b^*$: The IGOG-AA model does not explicitly model action selection. In this case the success reward is given in the case that working memory output contains only the external input filler that correctly answers the given role query.

In the above equations, *wm* represents contents of a working memory layer, either *m* for maintenance or *o* for output. The model is indicated by *ig* for IG or *igog* for both IGOG and IGOG-AA. Contents for a working memory stripe is indicated by *s*, with a subscript for the stripe index and superscript to indicate working memory layer. All experiments in this work use three stripes.

Since the action selection network learns to map fillers to actions during training, at testing time the network will have no experience with any fillers that were not present in the training set. To accommodate scenarios in which a novel filler is associated with a known role we have included the IGOG-AA (Assumed Action) model. Here it is assumed that the mapping of a filler in working memory to the selection of the action corresponding to that filler is already learned. In this model the "query" trial was considered correct if one of the stripes contained a representation for the appropriate filler and all other stripes were empty.

$$a = \underset{c \in C}{argmax}((op \wedge role \wedge c) \cdot w + b) \tag{3}$$

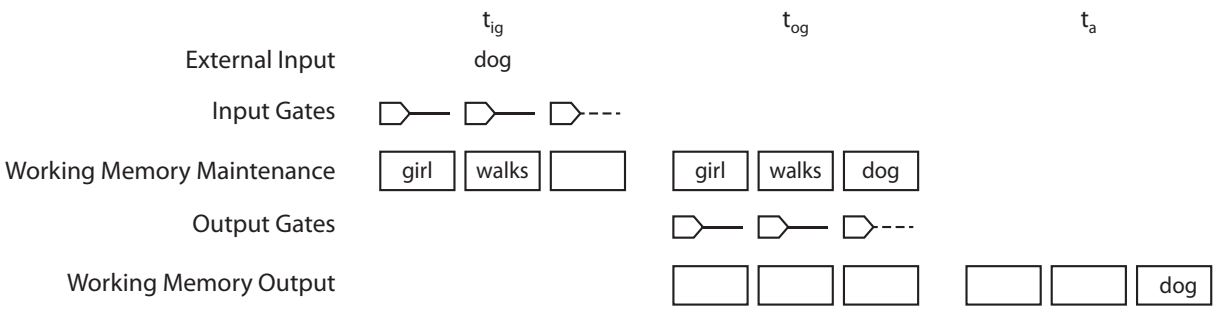


Figure 2: This figure shows how external representations move through the model for a single timestep of the IGOG model. At $t_{ig}$ the filler "dog" is presented, and the third input gate opens in response to the internal input. The first two stripes in working memory maintenance retain their current contents, but the contents of the third stripe are replaced with the external input representation. At $t_{og}$ the third output gate opens in response to the same internal input, allowing "dog" to flow through to the working memory output layer. At $t_a$ the action selection network selects a response based on the contents of the working memory output stripes.

In the above equation, the $op$ (store or query) and $role$ (sentential role) are joined with an action candidate $c$ from the set $C = \{open, close\}$. Because of the network architecture, values are a simple dot product of this representation and a weight vector $w$ that is specific to one particular gate, plus a bias term. Additionally, an $\varepsilon$-greedy approach ensured that non greedy actions were chosen a small proportion of the time ($\varepsilon = .025$).

Temporal credit for input and output gating actions (opening and closing of the gates) was assigned using eligibility traces. A correct "query" trial resulted in a global reward of 1.0. For incorrect query responses, and for all "store" trials, the reward was zero. We have set the $\gamma$ parameter to 1.0 so that future rewards are not discounted, and a relatively high $\lambda$ value of 0.9 ensures that reward for the final trial is distributed almost equally to all prior timesteps. Each sentence presentation is considered an episode. The bias term, $b$, for all neural networks used in this work (both for the gating $Q$ function and for action selection) was set to 1.0 and held constant throughout training, and all networks used a learning rate of 0.1.

The flow of filler representations from one layer to the next is governed by the input and output gates. Each timestep can be broken down into three substeps, as shown in Figure 2. Working memory stripes in all layers start off empty at the beginning of each sentence presentation. At substep $t_{ig}$, any stripe in $wm_m$ with an input gate in the open state will have its current representation replaced with the external representation. If the external representation is null, as in the case of the "query" trial, the stripe would then be emptied. If the state of the input gate is closed then the stripe retains its current representation. These representations may likewise flow from $wm_m$ to $wm_o$ at substep $t_{og}$ depending on the state of the output gates. Each stripe in the output layer is connected only to its corresponding stripe in the maintenance layer, so there can never be contamination from representations in other stripes. The IG model simply omits the $t_{og}$ substep, and input to action selection comes from $wm_m$ rather than $wm_o$. Finally, action selection occurs at substep $t_a$.

Table 1: Model Inputs

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| **External / Working Memory** | girl | walks | dog | |
| **Internal / Gate** | store $\wedge$ agent | store $\wedge$ verb | store $\wedge$ patient | query $\wedge$ agent |

Unitary holographic reduced representations (HRRs) [23] of length 1024 were used to encode all inputs. Conjunction was accomplished by circular convolution, and disjunction by normalized addition of vectors.

The separation of inputs into two groups, internal and external, plays a crucial part in the ability of the model to perform the given task. In our experiments the words are considered external inputs and the sentential roles are considered internal inputs. Although both the fillers and the roles with which they are associated are needed to solve the tasks, only the representations for the fillers are stored in working memory stripes. The stripes themselves come to correspond to the roles. This is possible because the input and output gates for the stripes utilize role information when determining whether an open or closed state is most beneficial.

## Results

Generalization results for each test are shown in Figure 3. For the standard generalization and spurious anticorrelation result sets, all models were able to achieve an accuracy on the test set that was close to the the 95% training accuracy. In the full combinatorial set, however, the IG model only reached a level of accuracy that would be obtained by guessing a response at random. Only the model that did not use an action selection network, IGOG-AA, was able to perform near training level accuracy for the novel filler test.

In order to respond appropriately to the query following each three word sentence presentation, the models must be able to encode six pieces of information - the fillers along with their roles - using only three working memory stripes. Because external representations

are not used to make gating decisions, each stripe comes to represent a single role. In other words, each of the neural networks that control the gating actions is able to learn a policy in which the "open" action is selected only when the role with which it identifies is presented to it. This role identification is accomplished entirely through reinforcement learning. The performance limitations of each model can by understood by examining the contents of the final working memory output layer of each model.

In the IG model, an action is selected using the contents of $wm_m^{ig}$, shown in Equation 2, as input. By associating the contents of each stripe with a *sid*, the contents are effectively mapped to a particular stripe. This allows for external representations to take on different meanings based on spacial location: a filler in the first stripe can mean something different from a filler in the third stripe. As long as all role/filler combinations are experienced at any point in training, as in the first two testing protocols, these combinations will be understood in the test set. Since all of $wm_m$ is presented to the action selection layer, the policy learned here involves choosing a correct action in the presence of two distracting representations, one from each of the other stripes. This model fails, however, when it encounters a novel role/filler combination. Although the external representation is gated into the working

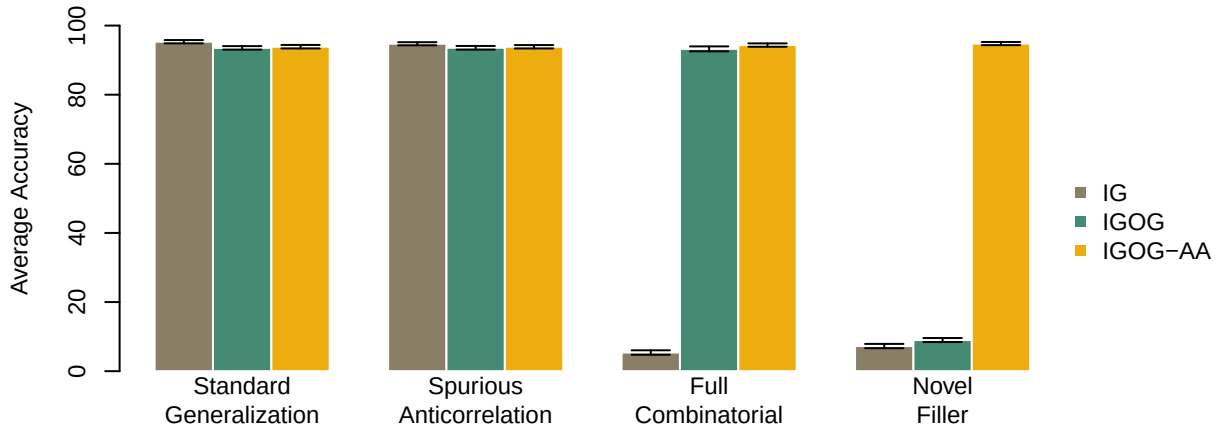


Figure 3: Results of the generalization tests for each model, with error bars to show a 95% confidence interval. One-hundred tests were run for each testing protocol.

memory stripe that is identified with the appropriate role, the action selection network has no experience with the resulting *sid*/filler representation.

The IGOG model allows for the contents of $wm_m$ to be filtered by another gating layer before action selection takes place. This eliminates the need for the *sid*s. A policy is learned whereby only a single output gate (and no input gates) is opened on query trials. The action network then learns to map each filler to its corresponding action. As with the IG model, poor performance on the novel filler test is due not to a bad gating policy, but to the action selection network's inexperience with novel representations.

To overcome issues resulting from an inexperienced action selection mechanism, we have used the assumed action scheme, IGOG-AA, described in the methods section. From the results of all four tests, we see that a role-based gating policy can be formed when reward is provided in response to correct representations in $wm_o$.

## Discussion

This work provides evidence for an account of generalization ability that is based on working memory gating. Each gate's decision to open or close does not depend on the state of other gates, and gates within each layer open or close in parallel. All gates are trained in response to a global reward signal. In both the IG and IGOG models, each stripe becomes tightly coupled with a role. This coupling is possible because role information is used to control gates, and filler information, which later drives action selection, moves separately through the model based on the state of the gates.

Alternative neural network architectures that attempt to model sequence processing do not provide an explanation for the types of behavior found in humans. Simple recurrent networks (SRNs) have been used in cases where data are structured and componential. While we do not compare directly with a SRN, it is shown in Kriete et al. (2013), that a SRN performs worse than the model with indirection on sentence presentation task.

We propose that the working memory representation in IG, which uses *sid*s to associate

stripe contents with a stripe, aligns with theories of neural representation in which fillers for a particular role are encoded by varying patterns of activity over a fixed population of neurons. In this view, learning of role/filler combinations is constrained in such a way that nothing can be learned about a particular filler outside the context of a role. We see from the results that this model performed poorly when a familiar filler was encountered in an unfamiliar role.

In this work, as in the indirection model from Kriete et al. (2013), learned gating policies are used as part of the role-filler binding mechanism. A critical difference between IGOG and the indirection model is that IGOG does not make use of pointers. Output working memory stripes in IGOG are not bound to any type of role identifier, and because the $wm_o$ representation does not communicate spacial information of component stripe contents, there is nothing to identify which stripe the fillers come from (as with the *sid*s in IG). It is for this reason that we do not need a model with explicit pointers to achieve good generalization results. In IGOG, the output representation for the word "dog" is the same regardless of which stripe it is stored in, whereas in the indirection model the distributed representation of $wm_o$ would change depending on the stripe in which it was stored. We see this not as evidence counter to a theory of generalization based on indirection, but as an abstraction consistent with the encodings and mechanisms used in our model. The underlying idea is that we can gain experience with a filler in a way that is independent of a role context, so that when we encounter this filler in a novel role the neural representation for the filler does not look completely new.

The IGOG-AA model shows that the learning of role bindings can be accomplished through only the use of a reward signal which is achieved in response to isolated output of a single working memory representation. Although we have described the fillers as novel in that they represent an unexperienced combination of stimuli, as with a name that we have never encountered, we believe that the neural representation for such a filler would likely be

represented as a composite of the underlying familiar stimuli, as with the letters or phonemes that comprise the unfamiliar name. In other words, the failure of IGOG to generalize well in the novel filler test does not discredit the viability of a dual-gating working memory architecture. The findings presented in Kriete et al. (2013) and reiterated in this work offer one account of how learned representations can be combined in novel ways to accommodate "novel filler" scenarios.

Hummel et al. present a model of role-binding in which the recall of roles is accomplished via comparison of working memory with long term memory representations [5, 10, 11]. This type of scheme can be used to extend or offer further explanations for the work presented in this paper. For example, we have explicitly provided role information to the working memory gating system. Contextual cues in working memory could provide a basis for recognition and recall of these roles.

Because the contents of working memory were not required to solve the tasks in this work, these contents were not provided to the gates. This lack of working memory input to inform gating decisions would, however, prevent the current IGOG model from efficiently solving hierarchical problems, in which a remembered context cue must be used to solve a task (e.g. AX-CPT). In future research we seek to study the means by which neural representations flow into the BG, in order to more accurately model the type of information that is used to control gating.

In these experiments, each sentence presentation constituted one temporal difference learning episode, and forgetting of stripe contents was only possible on the "query" trial. Future work may explore the effects of a continuous sentence presentation task (where there may be more than one agent, for example, from different sentences), and may include a forgetting mechanism in order to more accurately model human performance.

Finally, while the approach outlined in this paper does allow for binding of fillers to a role, it does not allow for rebinding of these fillers. Often times our thinking about role-filler

bindings may change with the addition of new context. Further research into role-filler rebinding based on pointer swapping could provide insight into the neural mechanisms underlying this capability.

The models presented here offer a simplified and abstracted model of working memory input and output gating functionality. Because of the ease with which these types of models can be built and scaled, they can potentially save both time and computing resources as compared to models that use architectures aligning with strict biological constraints. Abstract models like these are ideal for working memory protyping and conceptual development.

Working memory forms the basis for a biological agent's ability to solve complex problems. Abstracted models of working memory have been shown to aid robotic agents in overcoming many of the issues associated with temporally extended tasks. A model of working memory that allows for generalization can be used to further increase the capabilities of machine learning agents by enabling them to understand known fillers when encountered in unfamiliar roles. In the near future, it is not hard to imagine that generativity, the mental (or virtual in this case) binding of roles and fillers that have never before been experienced together, may also be used to influence an agent's action choices.

# CHAPTER III.

# N-TASK LEARNING: SOLVING MULTIPLE OR UNKNOWN NUMBERS OF REINFORCEMENT LEARNING PROBLEMS

## Introduction

Functionality typically associated with human and animal working memory has been emulated successfully in several computational models [7, 13, 14, 16–19, 21, 22, 25], which make use of temporal difference (TD) learning algorithms. 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) [27]. Even when the state signal meets this criterion there is frequently information surrounding the task that cannot be encoded in state. Frequent changes to the optimal policy that are driven by hidden information can confound learning in TD models [1].

These types of problems frequently occur in the daily activities of humans. 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 building 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, the reward you desire, you would then open the other door. In your mind 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 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. present evidence that humans engage in representation learning, homing in on task relevant features in an attempt to reduce complexity of a state representation [16]. 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, $W(f)$:

$$V(S) = \sum_{f \in S} W(f) \tag{4}$$

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 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. give support for the argument that working memory is responsible for flexibly updating goals [19]. 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" [25]. 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 [3, 4, 7, 13, 14, 25]. It has been shown that working memory is updated when levels of the neurotransmitter dopamine are phasically elevated [3, 7, 14, 16, 24, 25]. When a reward is expected but not delivered, the resulting negative error signal prompts an update to working memory contents [19]. 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 [3, 17], as can be modeled by the gain parameter to a sigmoidal neural activation function [7, 18].

## 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. 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 [28]. In this case an option is selected and influences action choices for all timesteps until a goal state is reached or the option expires after a predifined number of timesteps. 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 predifined 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 [30] 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.

## Methods

### Experimental Protocol

Our experiment is similar to the dimension selection task described in [25]. 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 Wisconsin Card Sorting Task). The rule is selected at random, and always differs between two consecutive rounds. Figure 4 shows an example round, and all parameter values are given in Table 2.

It is important to note that the distinguishing feature of this problem is that is composed

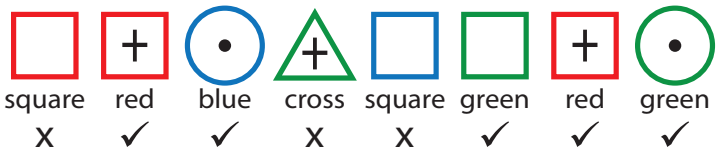


Figure 4: Shown here is an example round of the experiment with $l$ set to three using stimuli composed of three features drawn from three dimensions: color, shape, and fill. The round consists of eight trials. The stimuli are shown in the top row, and the selected action is shown in the middle row. The bottom row designates a correct trial with a checkmark, and an incorrect trial with an "x". The rule for this round was set to "color".

Table 2: 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) |
| $\varepsilon$ | 0.005 | Probability that the SARSA algorithm will make a non-greedy action choice |
| $\alpha_q$ | 0.05 | Learning rate for the $Q$ function update |
| $\alpha_a$ | 0.0075 | Learning rate for the $A$ function update |
| $\alpha_t$ | 0.002 | Learning rate for the $t$ threshold update |
| $a$ | 0.5 | Task add threshold for dynamic nTL (5 tasks) |

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 never repeated then the problem becomes a non-stationary bandit problem. Without a way to associate learning with a task representation, the agent sees the problem as a non-stationary armed 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:

- We define a *task* to be any policy that consistently leads to reward under some set of conditions. Our experiment, therefore, may have up to five tasks, as each dimension of the stimuli may serve as the categorization rule that determines reward.

- In a particular experiment some dimensions may never be used for the rule, and serve only as distractors. We refer to the number of distinct rules that are used throughout the experiment as the number of *external 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 over the course of a single rule, from the time a new 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*.
- When we wish to denote a rule change, the completing of one round and beginning of another, we use the term *external task switch*.
- Incorrect trials are also referred to as *sub-optimal moves*.

## Holographic Reduced Representations

Unitary holographic reduced representations (HRRs) [23] 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 a direct result of this choice, and we see no reason why alternate encodings such as vectors [15], tensors [20], or spatio-temporal encodings [10] could not produce similar results.

## Working Memory Computational Model

In this section we 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”), the state and action sets 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 7. 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

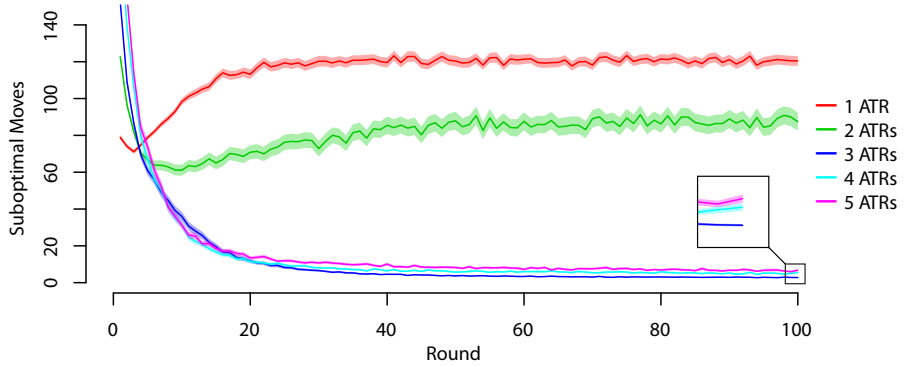


Figure 5: Static nTL - Shown here are the average number of sub-optimal moves taken at each round. Mean values for 1000 runs are shown by the solid lines, with shading to show a 95% confidence interval. The number of external tasks, $N$, is three. Optimal performance will converge to $(\sum_{j}^{N-1} j)/(N-1)$ sub-optimal moves per round.

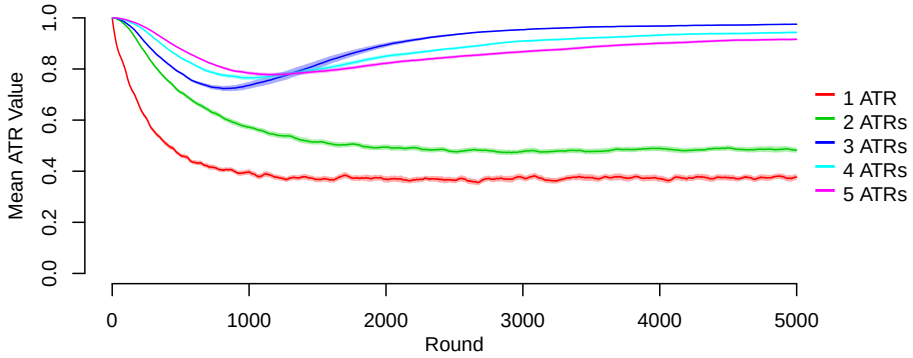


Figure 6: Static nTL - Mean ATR values for 100 runs are shown by the solid lines, with shading to show a 95% confidence interval. The number of external tasks for this experiment was three.

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 $\varepsilon$-soft policy that makes non-greedy decisions a small fraction ($\varepsilon$) of the time.

An action is selected by forming the conjunct representation of state with each candidate action representation 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:

$$m = \underset{c \in C}{argmax}((s \wedge c) \cdot W_q + b) \tag{5}$$

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 [12] 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):

$$\Delta w_i = \alpha_q[sgn(\delta) * log(|\delta| + 1) * (s \wedge m)_i] \tag{6}$$

In the above equation, $w_i$ indicates the value of the weight vector at index $i$, $\alpha_q$ is the learning rate, $\delta$ 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 $\lambda$ and $\gamma$ 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 will 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 current ATR representation, $atr$. In this way the same input, when bound to two separate ATRs, is seen as two distinct concepts by the base algorithm, and consequently produces two distinct values. No alterations to the base algorithm are required in order for this approach to work. Our action selection equation now becomes:

$$m = \underset{c \in C}{argmax}((s \wedge c \wedge atr) \cdot W_q + b) \tag{7}$$

The weight update is modified in a similar manner; where before in Equation 6 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 $s \wedge m \wedge atr$.

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, $\alpha_a$ is the learning rate for ATRs, and $\delta$ 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:

$$A(atr) \leftarrow A(atr) + \alpha_a[sgn(\delta) * log(|\delta| + 1)] \tag{8}$$

Trials that result in a task switch do not incur an update to the $A$ function. 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 the external tasks.

The model determines when to switch to a new ATR via a threshold, $t$. When the TD error ($\delta = r - Q(s,m)$) 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 $\alpha_t$ is the learning rate for $t$ , and $\delta$ is the TD error:

$$t \leftarrow t - \alpha_t[sgn(\delta) * log(|\delta| + 1)] \tag{9}$$

**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 [16, 26].

### Additional Details for Experiment Setup

A list of all parameter values is provided in Table 2. All experiments were conducted using R version 3.4.0. The source code used for these experiments can be downloaded freely from: `urlredactedforanonymity`

## 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

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. We would expect such an agent to perseverate on previously learned features when a task

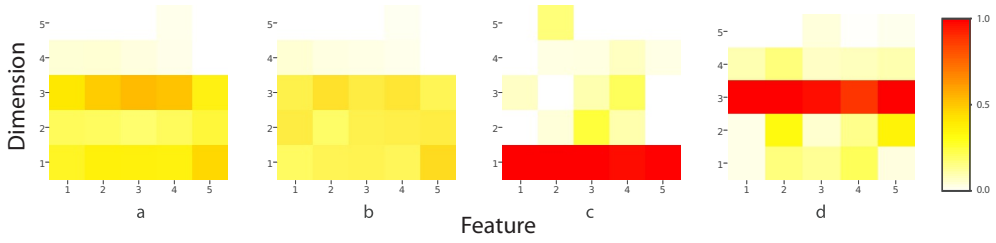


Figure 7: Shown here are values for feature selection among features present in the entire stimuli set. *a* and *b* show results from trials that were taken from the static nTL experiment using a single ATR to learn three external tasks. In *a*, which corresponds to a trial at the beginning of a round, we see that dimensional representations from a previous round have begun to form along dimension 3. By *b*, a trial later in the same round, no dimension appears to be more valuable than any other. *c* and *d* show feature selection values after 5000 trials for two of three ATRs that are used to complete the same task. *c* has come to represent dimension 1 and *d* has come to represent dimension 3.

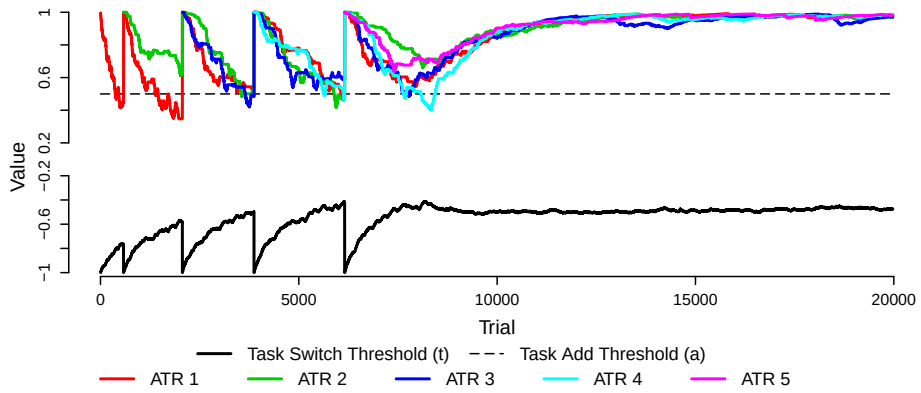


Figure 8: Dynamic nTL - ATR values for a scenario in which the number of task representations adapts to achieve optimal performance. The number of external tasks in this experiment was five.

switch occurs, and the results confirm that perseveration occurs. When only one ATR is available, many more sub-optimal moves are taken in each round (see Figure 5). When an external task switch occurs, action choices that were valuable in the previous round are tried first (see Figure 7a). Only after a period of unlearning the previous external task (see Figure 7b) does the model begin to select new actions.

Figure 5 shows how task performance changes as a function of the number of ATRs being used. Although a statistically significant difference is shown, 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 external tasks (3 ATRs), having one too few (2 ATRs) resulted in 3043% more sub-optimal moves per round after 100 rounds, whereas having one too many (4 ATRs) resulted in only 97% more sub-optimal moves per round.

From Figure 6 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 7 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 7c and 7d 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 external tasks (see Figure 8). 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 groups 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 [16] 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 [19]. 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 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 an external task 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 [19].

When the number of ATRs matches the number of external tasks, the mean of the agent's estimated values for the ATRs (the $A$ function) converges to the goal reward value of the external 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 external 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 external 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 categorization 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 external task distribution is uniformly stochastic, the number of external tasks does not change, and 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 an external 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 external 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.

# CHAPTER IV.

## CONCLUSION

The significance of the brain's ability to selectively update and sustain mental representations is difficult to overstate. Conscious targeted attention to a feature or idea provides the foundation for executive decision making, and very likely for the learning of structured representations. In the pursuit of a complete and functionally accurate model of working memory, we have explored ways in which gate-based architectures can be used to afford the types of generalization capabilities found in humans. We have also implemented an algorithm that accommodates scenarios in which multiple tasks must be performed under conditions where an external task cue is not provided.

This thesis work has combined two lines of study that are tied together by a common goal, to extend functionality for abstracted computational models of working memory. We have followed in the interdisciplinary tradition whereby advances in the understanding of human biology lead to questions that can be explored by abstract computational models. These models, in turn, offer new hypotheses for biologists. Although the models in this work implement mechanisms that are tightly coupled with current evidence from the field of cognitive neuroscience, they are by nature a working memory for an artificial agent. Working memory models have been used with robotic agents to great effect, and we suspect that these models or a derivation thereof can be used to improve upon existing functionality.

## BIBLIOGRAPHY

[1] Brunskill, E. and Li, L. The Online Coupon-Collector Problem and Its Application to Lifelong Reinforcement Learning. *CoRR*, abs/1506.03379, 2015.

[2] Busch, M. A., Skubic, M., Keller, J. M., and Stone, K. E. A Robot in a Water Maze: Learning a Spatial Memory Task. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1727–1732. IEEE, 2007.

[3] Chatham, C. H. and Badre, D. Multiple Gates on Working Memory. *Current Opinion in Behavioral Sciences*, 1:23–31, February 2015.

[4] Chatham, C., Frank, M., and Badre, D. Corticostriatal Output Gating During Selection from Working Memory. *Neuron*, 81(4):930–942, February 2014.

[5] Doumas, L. A. A., Hummel, J. E., and Sandhofer, C. M. A Theory of the Discovery and Predication of Relational Concepts. *Psychological Review*, 115(1):1–43, 2008.

[6] Erdemir, E., Frankel, C. B., Kawamura, K., Gordon, S. M., Thornton, S., and Ulutas, B. Towards a Cognitive Robot That Uses Internal Rehearsal to Learn Affordance Relations. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2016–2021. IEEE, 2008.

[7] Frank, M. J., Loughry, B., and O'Reilly, R. C. Interactions Between Frontal Cortex and Basal Ganglia in Working Memory: A Computational Model. *Cognitive, Affective, & Behavioral Neuroscience*, 1(2):137–160, 2001.

[8] Gordon, S. M., Kawamura, K., and Wilkes, D. M. Neuromorphically Inspired Appraisal-Based Decision Making in a Cognitive Robot. *IEEE Transactions on Autonomous Mental Development*, 2(1):17–39, March 2010.

[9] Hazy, T. E., Frank, M. J., and O'Reilly, R. C. Neural Mechanisms of Acquired Phasic Dopamine Responses in Learning. *Neuroscience & Biobehavioral Reviews*, 34(5):701–720, April 2010.

[10] Hummel, J. E. and Holyoak, K. J. Distributed Representations of Structure: A Theory of Analogical Access and Mapping. *Psychological Review*, 104(3):427, 1997.

[11] Hummel, J. E. and Holyoak, K. J. Relational Reasoning in a Neurally Plausible Cognitive Architecture: An Overview of the Lisa Project. *Current Directions in Psychological Science*, 14(3):153, 2005.

[12] John, J. A. and Draper, N. R. An Alternative Family of Transformations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(2):190–197, 1980.

[13] Kriete, T., Noelle, D. C., Cohen, J. D., and O'Reilly, R. C. Indirection and Symbol-Like Processing in the Prefrontal Cortex and Basal Ganglia. *Proceedings of the National Academy of Sciences*, 110(41):16390–16395, October 2013.

[14] Kriete, T. and Noelle, D. C. Generalisation Benefits of Output Gating in a Model of Prefrontal Cortex. *Connection Science*, 23(2):119–129, June 2011.

[15] Mitchell, J. and Lapata, M. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429, November 2010.

[16] Niv, Y., Daniel, R., Geana, A., Gershman, S. J., Leong, Y. C., Radulescu, A., and Wilson, R. C. Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms. *Journal of Neuroscience*, 35(21):8145–8157, May 2015.

[17] O'Reilly, R. C. Biologically Based Computational Models of High-Level Cognition. *Science*, 314(5796):91–94, 2006.

[18] O'Reilly, R. C. and Frank, M. J. Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia. *Neural Computation*, 18(2):283–328, 2006.

[19] O'Reilly, R. C., Noelle, D. C., Braver, T. S., and Cohen, J. D. Prefrontal Cortex and Dynamic Categorization Tasks: Representational Organization and Neuromodulatory Control. *Cerebral Cortex*, 12(3):246–257, 2002.

[20] Papalexakis, E. E., Faloutsos, C., and Sidiropoulos, N. D. Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms. *ACM Trans. Intell. Syst. Technol.*, 8(2):16:1–16:44, October 2016.

[21] Phillips, J. and Noelle, D. Working Memory for Robots: Inspirations from Computational Neuroscience. In *Proc. from 5th Intl Conf on Development and Learning*, 2006.

[22] Phillips, J. L. and Noelle, D. C. A Biologically Inspired Working Memory Framework for Robots. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pages 599–604. IEEE, 2005.

[23] Plate, T. A. Holographic Reduced Representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

[24] Rafati, J. and Noelle, D. C. Lateral Inhibition Overcomes Limits of Temporal Difference Learning. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, pages 1931–1937, 2015.

[25] Rougier, N. P., Noelle, D. C., Braver, T. S., Cohen, J. D., and O'Reilly, R. C. 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, 2005.

[26] Shepard, R. N., Hovland, C. I., and Jenkins, H. M. Learning and Memorization of Classifications. *Psychological Monographs: General and Applied*, 75(13):1, 1961.

[27] Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

[28] Sutton, R. S., Precup, D., and Singh, S. Between Mdps and Semi-Mdps: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[29] Tugcu, M., Wang, X., Hunter, J. E., Phillips, J., Noelle, D., and Wilkes, D. M. A Computational Neuroscience Model of Working Memory with Application to Robot Perceptual Learning. In *Third IASTED International Conference on Computational Intelligence (CI)*, 2007.

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