Motivated Learning from Interesting Events: Adaptive, Multitask Learning Agents for Complex Environments

Corresponding author:
Kathryn Merrick
University of Sydney, School of Information Technologies
and National ICT Australia,
kkas0686@it.usyd.edu.au
School of Information Technologies, J12
University of Sydney,
NSW 2006,
AUSTRALIA.

and

Mary Lou Maher
University of Sydney,
Key Centre for Design Computing and Cognition
Abstract

This paper presents a model of motivation in learning agents to achieve adaptive, multi-task learning in complex, dynamic environments. Previously, computational models of motivation have been considered as speed-up or attention focus mechanisms for planning and reinforcement learning systems, however these different models do not provide a unified approach to the development or evaluation of computational models of motivation in different learning settings. This paper models motivation for machine learning as a process that reasons about the states and changes encountered by an agent to produce a learning stimulus that focuses learning and action. Context free grammars and events are introduced as adaptable representations of states and learning tasks. This extends existing learning algorithms to complex, dynamic environments in which tasks cannot be completely predicted prior to learning. Two agent models are presented for motivated reinforcement learning and motivated supervised learning, which incorporate this model of motivation. The formalisms used to define motivated reinforcement learning agents and motivated supervised learning agents further allow the definition of consistent techniques for evaluating motivated learning agent models. Three new metrics are introduced for evaluating learning efficiency, characterizing computational models of motivation and visualizing the emergent behavior of motivated learning agents. The paper concludes with a demonstration of the motivated reinforcement learning agent model that uses novelty and interest as the motivation function. The model is evaluated using the new metrics, showing that motivated reinforcement learning agents using general, task-independent concepts such as novelty and interest can learn multiple task-oriented behaviors.

Keywords

Agent, motivation, interest, machine learning.
Motivated Learning from Interesting Events:  
Adaptive, Multitask Learning Agents for Complex Environments

Imbuing artificial systems with the ability to learn allows them to change their structure and improve their performance at a task. However, popular learning algorithms such as reinforcement learning (Sutton & Barto, 2000) and supervised learning (Russel & Norvig, 1995) require some form of learning stimulus – such as a reward signal or examples of correct behavior – in order for learning to occur. As a result, learning algorithms have typically been applied to single, isolated tasks for which it is possible to prepare a task-oriented reward signal or a set of task-oriented examples (Singh, Barto, & Chentanex, 2005). While the focus on machine learning for solving single tasks has been critical for the development of computational models of learning, many real world problems do not conform to the assumptions made by existing algorithms. In particular, the physical world and many large-scale, persistent virtual worlds are characterized by the ability of humans or human controlled avatars to modify the content of the environment. As a result, these environments contain many potential learning tasks and these tasks may change over time. For example, in persistent virtual game worlds, where a learning system might control the behavior of a support character, support characters that cannot evolve and change in response to changes in their environment impact negatively on the experience of gamers (Laird & van Lent, 2000). Likewise, in developmental robotics domains it is useful for robots to evolve and modify sets of behaviors that enable them to solve a variety of relevant problems (Weng et al., 2001).

In reinforcement learning, the use of task-oriented learning stimuli assumes that the tasks to be learned are understood well enough by system designers for them to provide an adequate reward signal to facilitate learning. Weng et al (2001) point out that this assumption cannot be justified for systems required to exhibit lifelong learning. In supervised learning, the use of task-oriented learning stimuli assumes that the tasks to be learned can be observed by system designers to produce a set of examples of correct behavior for completing the task. In complex environments where multiple humans or human controlled avatars act simultaneously, observations of their behavior tend to produce examples of multiple simultaneous tasks. In such situations, a learning process is required that can focus attention on individual tasks to learn coherent behaviors for completing them.

To extend learning systems to environments where task-oriented learning stimuli are not available, more flexible motivation mechanisms are required. Psychologists have proposed a number of methods by which this is achieved in natural systems. These range from biological theories such as drives (Woodworth, 1918) – which work within the biological systems of a behaving organism – to cognitive and social theories – which encapsulate abstract concepts such as curiosity and interest (Berlyne, 1966). More recently, the notion of self-motivation has also been adopted as a means by which artificial systems may achieve greater autonomy though selecting their own focus of attention. A number of algorithms have been developed to model various forms of motivation for use in both planning (Luck & d'Inverno, 1998) and reinforcement learning systems (Kaplan & Oudeyer, 2003; Schmidhuber, 1991; Singh et al., 2005). While these algorithms have demonstrated potential to display lifelong, adaptable, multitask planning or learning, they do not provide a consistent approach to the development or evaluation of different computational models of motivation in different learning settings.

This paper models motivation for machine learning as a process that reasons about the states and changes encountered by an agent to produce a learning stimulus that focuses learning and action. Context free grammars and events are introduced as adaptable representations of states and learning tasks to extend existing learning algorithms to complex, dynamic environments in which tasks cannot be predicted prior to learning. Two agent models are
presented for motivated reinforcement learning and motivated supervised learning that incorporate this model of motivation. The formalisms used to define motivated reinforcement learning agents and motivated supervised learning agents further allow the definition of techniques for evaluating motivated learning agent models. Three new metrics are introduced for evaluating learning efficiency, characterizing computational models of motivation and visualizing the emergent behavior of motivated learning agents. The paper concludes with a demonstration of the motivated reinforcement learning agent model that uses novelty and interest as the motivation function. The model is evaluated using the new metrics, showing that motivated reinforcement learning agents using general, task-independent concepts such as novelty and interest can learn multiple task-oriented behaviors.

1. Current Approaches to Learning Agents

The field of machine learning is concerned with building systems that can change their structure to improve their performance with respect to some task (Nilsson, 1996). This task may involve identifying important structures or features in data — called unsupervised learning — or it may involve the development of a mapping from input data to output data, as is the case in supervised or reinforcement learning. When supervised or reinforcement learning techniques are used in situations where input data represents the state of an environment and output data represents actions that can be taken to change the state of that environment, the machine learning algorithm may be thought of as a learning agent. Agents are systems that sense their environments using sensors, reason about their sensory input using some characteristic reasoning process and act in their environment using effectors. Learning agents generally have two broad types of sensors as shown in Figure 1: those allowing them to sense the state of their environment and those allowing them to sense some stimulus that directs their learning. Supervised learning agents, for example, have a training phase in which they can sense both the current state of their environment and an example of the correct action to perform when in that state. Similarly, reinforcement learning agents are able to sense the current state of their environment and a reward signal indicating the value of that state to the agent. The characteristic reasoning process of a learning agent is a learning algorithm such as a supervised neural network, decision tree, policy iterator or temporal difference learner. Effectors connect the learning agent to its environment by enabling actions to be performed.

Frequently, learning algorithms are represented as a single process, as is the case in Figure 1. In the remainder of Section 1, we present existing reinforcement learning and supervised learning within agent frameworks. We decompose these algorithms into three key processes: sensation, learning and action. This allows us to better consider the role of motivation in Section 2.
1.1. Reinforcement Learning Agents

Reinforcement learning (RL) (Sutton & Barto, 2000) uses rewards to guide agents to learn a function that represents the value of taking a given action in a given state with respect to some task. In the RL framework, a learning agent interacts with an environment over a series of discrete time steps. At each time step t, the agent observes some state $O(t)$ of its environment and a reward $R(t)$. The agent updates its behavioral policy $\pi$ using an update rule such as Q-learning (Watkins, 1989), then chooses an action $A(t)$ using an action selection function combined with an exploration rule. Typical exploration functions such as $\epsilon$-greedy or Boltzman exploration choose the optimal action most of the time and non-optimal actions with some small probability. The action causes the environment to transition to a new state in which the agent can observe $O(t+1)$ and a reward signal $R(t+1)$. The objective of the agent is to use systematic trial and error to learn a behavior that tends to increase the long-term sum of values of the reward.

Typically, RL is thought of as a single process that takes a state and reward value as input and outputs an action. In this paper, we decompose RL into three sub-processes: sensation, learning and action as shown in Figure 2(a). In this model, $W(t)$ represents the state of the agent’s environment at time t. $S(t)$ represents the state of the environment as sensed by the agent at time t. The sensation process $S$ is responsible for disassembling $S(t)$ into the observed state of the environment $O(t)$ and the reward $R(t)$. The sensation process is sometimes referred to as the input function (Kaelbling, Littman, & Moore, 1996). The input function is often assumed to be the identity function, that is, agents can sense the complete state of their environment, however partially observable environments have also been considered. $O(t)$ is commonly an attribute based representation of the environment such as a fixed length vector $(o_1(t), o_2(t), o_3(t), \ldots, o_{|O|}(t))$. The learning process $L$ performs an update to incorporate the observed state into the behavior $\pi(t-1)$ in memory $M$. Finally, the activation process $A$ uses the exploration function and action selection rule to select an action $A(t)$ to perform from the updated behavior $\pi(t)$. The chosen action $A(t)$ triggers a corresponding effector $F(t)$ that makes a change to the agent’s environment.

![Figure 2. Agent models for (a) reinforcement learning and (b) supervised learning.](image)

1.2. Supervised Learning Agents
Supervised learning (SL) (Russel & Norvig, 1995) uses examples of the correct action to take given a particular observation to assist agents to learn a function that represents the action to take in a given state with respect to some task. We also represent SL in terms of three processes: sensation, learning and activation. However in the SL framework, a learning agent may either learn or act at each time step. If at time step t, the agent observes some state $O(t)$ of its environment and an example action $X(t)$ then it learns by updating its behavioral policy $\pi$. This is called a training step. If at time step t, the agent observes just a state $O(t)$ then it acts by choosing an action $A(t)$ from the behavioral policy $\pi$. This is called a testing step. $A(t)$ causes the environment to transition to a new state. The objective of the agent is to use observation to learn a behavior that tends to maximize the percentage of testing steps that choose actions that agree with example actions provided in training steps for the same observed states.

Various approaches have been taken to represent the learned behavioral policy $\pi$ in SL, including decision tree representations and supervised neural network representations (Russel & Norvig, 1995). An agent model that could incorporate one of these SL strategies is depicted in Figure 2(b). In this model, $W(t)$ represents the state of the agent’s environment at time t. $S(t)$ represents the state of the environment as sensed by the agent at time t. The sensation process $S$ is responsible for disassembling $S(t)$ into the observed state of the environment $O(t)$ and the example action $X(t)$ if it is provided. If $X(t)$ is provided, the learning process $L$ incorporates the observed state and example into the behavior $\pi(t-1)$ in memory $M$ by performing a supervised learning update such as the decision tree update if a decision tree representation is being used for $\pi$, or a weight update if a supervised neural network representation is being used. If $X(t)$ is not provided, the activation process $A$ uses the behavior $\pi(t-1)$ to select an action $A(t)$ to perform. The chosen action $A(t)$ triggers a corresponding effector $F(t)$ that makes a change to the agent’s environment.

1.3. Motivated Learning Agents

RL offers a way for agents to learn by direct interaction with their environment. As such, it has been a popular starting point for motivated learning research while the role of motivation in SL has had little attention. Early motivated reinforcement learning (MRL) research focused on the development of motivation processes to direct learning. Schmidhuber (1991), for example, focused on the development of a motivation model in which an agent desires to improve its knowledge of its environment. Saunders and Gero (2001) developed a novelty based model of motivation to explore solution and performance spaces in design domains. Kaplan and Oudeyer (2003) focused on developing a general motivation function designed to drive the mastery of any sensory-motor device.

In an effort to formalize the growing body of literature combining motivation with RL, Singh et al (2005) introduced a multi-option RL framework that can learn policies for multiple tasks in the presence of both a motivation signal and an extrinsic reward signal. Extrinsic reward is assumed to be task-oriented while the motivation signal is expected to model more general biological or cognitive phenomena such as pleasure or interest. In practice, however, Singh et al (2005) used a second task-specific reward function to represent motivation.

While motivation has played several roles in RL systems, including those of speed-up and attention focus mechanisms, existing models have not addressed the issues associated with extending RL and SL agents to multi-task learning in complex, dynamic environments for which the tasks cannot be completely known at design time. New techniques are required that offer more flexible representations of such environments. Furthermore, new metrics are required to analyze the emergent multi-task learning of motivated reinforcement learning and motivated supervised learning (MSL) agents.
2. Focusing Learning Using Motivation

Motivated learning can be thought of as a meta-learning technique in which a motivation process provides a learning algorithm with a motivation signal that focuses learning. Within this setting, however, motivation plays a different role in RL and SL. In existing MRL models, the motivation signal has been used either in addition to or instead of an extrinsic reward signal. In this paper, we focus on the latter model because it can operate in environments where there is insufficient design time knowledge to provide a task-oriented, extrinsic reward signal. The role of the motivation process in this model is to use general, task-independent concepts to generate a motivation signal to stimulate the learning of task-oriented behaviors. Thus, our motivation process extends RL to complex dynamic environments by creating a motivation signal to replace the extrinsic reward. In SL, however, the issue that arises in complex, dynamic environments is not the lack of a learning stimulus, but rather the over supply of such stimuli. In complex environments agents may be exposed to examples of many different tasks simultaneously. In order for task-oriented behaviors to be learned, input examples must be filtered to focus attention on individual tasks. Thus, the role of the motivation process in this model is to use general, task-oriented concepts to filter examples to stimulate the learning of task-oriented behaviors.

In the remainder of this section, we introduce context free grammars as a flexible representation of environments about which there is only limited design time knowledge. We expand the role of the sensation process from the production of an observed state of the environment to include the production of events to represent potential learning tasks as changes in the environment. We define motivation as a process that takes observed states and events as input and produces a motivation signal that will direct learning in MRL or arbitrates between learning and action in MSL. Finally, we present models for MRL and MSL agents that incorporate this motivation framework.

2.1 Context Free Grammars as an Adaptable Representation of the Environment

In this section, we use persistent virtual worlds such as those in MMORPGs as an example of a complex, dynamic environment. These virtual environments are composed of objects that can be manipulated by human controlled player characters to design houses, businesses or other artifacts such as weapons or furniture. Because human controlled player characters can add content to the virtual environments, the environments change over time. However the exact nature of those changes is not predictable at design time. In existing MRL and RL models, the state of the environment observed by an agent is represented as a fixed length vector of attributes. For example, a room in a virtual world might be described in terms of the location of its walls as follows:

\[
O(t) = (\text{wall1X, wall1Y, wall2X, wall2Y, wall3X, wall3Y, wall4X, wall4Y})
\]  

However, a fixed length vector representation becomes inadequate in dynamic virtual environments when new objects can be introduced or old objects removed as it is not possible to know how long the length of the vector should be. One approach uses placeholder variables that can take the values of new objects. However, such representations place a hard limit on the number of new objects that can be introduced and hold large amounts of redundant data when old objects are remove. As an alternative to fixed length vectors, an observed state \(O(t)\) may be represented as a string from a context-free grammar (Merceron, 2001; Merrick, 2007)(CFG). A CFG for the room scenario might look like:

\[
\begin{align*}
O(t) & \rightarrow <\text{walls}> \\
<\text{walls}> & \rightarrow <\text{wall}><\text{walls}> \mid \varepsilon \\
<\text{wall}> & \rightarrow <\text{wallID}><\text{wallX}><\text{wallY}>
\end{align*}
\]
The CFG in Equation 2 allows for a variable number of walls. Thus, the state representation is flexible enough to describe rooms that are partially built and rooms of different shapes bounded by different numbers of walls. This is important in virtual environments where building is a gradual and creative process. For the remainder of this paper, we will consider each observed state \( O(t) \) to be a string from the context-free language \( O \) defined by a CFG \((V, T, P, O(t))\) where:

- \( V \) is a set of variables or syntactic categories,
- \( T \) is a finite set of terminals such that \( V \cap T = \emptyset \),
- \( P \) is a set of productions of the form \( V \rightarrow v \) where \( V \) is a variable and \( v \) is a string of terminals and variables,
- \( O(t) \) is the start symbol.

Agents initially receive information about their environment from their sensors. Agents may have a set of several different sensors. We refer to the data provided by sensors as sensations. In worlds represented by CFGs, each sensor may return a variable number of sensations. For example, an agent that observed the room environment in Equation 2 might have used an object sensor that can sense data of the form:

In this case the object sensor may return any number of sensations greater than or equal to zero depending on the number of walls sensed. Each sensor assigns a label \( L \) to each sensation, such that that two sensed states can be compared by comparing the values of sensations that have the same label where such sensations exist. Labels can be constructed using combinations of the grammar terminals in \( T \) and variables in \( V \). For example, sensations from the sensor in Equation 3 might be labeled using a combination of the \( \text{objectID} \) terminals and other property name variables as follows:

Other sensors that might be required by agents in game environments include inventory sensors, location sensors and avatar sensors.

2.2 Representing Tasks Using Events as Changes in the Environment

There are two distinct kinds of learning task: maintenance tasks and achievement tasks. In maintenance tasks, agents attempt to maintain themselves in states that have certain desirable properties. In achievement tasks, agents attempt to repeat some change. To create dynamic, adaptable learning agents, this paper focuses on achievement tasks. We introduce events as a general means of representing potential achievement tasks without explicitly describing the task in advance.

2.2.1 Events
We introduce events to represent potential achievement tasks as changes that occur in an agent’s environment. Events allow agents to represent and reason about changes in their environment in addition to information about the current observed state. We represent events in terms of the difference between two observed states. The difference between two observed states, \( O(t') = (o_1(t'), o_2(t'), \ldots, o_L(t')) \) and \( O(t) = (o_1(t), o_2(t), \ldots, o_L(t)) \) where \( t' < t \) as a vector of difference variables is calculated using a difference function \( \Delta \) as shown in Equation 5.

\[
O(t) - O(t') = (\Delta(o_1(t), o_1(t')), \Delta(o_2(t), o_2(t')), \ldots, \Delta(o_L(t), o_L(t'))) \tag{5}
\]

An event is a combination of difference variables. A combination is an unordered selection of difference variables made without repetition. There are \( 2^{|O(t) - O(t')|} - 1 \) possible combinations of difference variables. In complex environments where \( |O(t) - O(t')| \) is large, agents focus on only a subset of these combinations as events. This subset is defined by an event function mapping the difference \( O(t) - O(t') \) to a set of events \( E_{O(t)} \).

### 2.2.2 Difference Functions

A difference function \( \Delta \) assigns a value to the difference between two observations \( o_{L(t)} \) and \( o_{L(t')} \) in the observed states \( O(t) \) and \( O(t') \). A number of example difference functions are presented in Table 1. They differ in the range of output values they produce, which in turn affects the amount of information they make available for reasoning. Difference Function 1 offers the most information about the change between successive observations as it calculates the magnitude and direction of the change. In contrast, Difference Function 2 only describes the direction of the change. Difference Function 3 offers the least information, showing only whether or not a change has occurred. Such a difference function might be useful in situations where a numerical difference is not meaningful. For example, when comparing the difference between two observations of attributes with string values rather than numeric values.

<table>
<thead>
<tr>
<th>Difference Function</th>
<th>Output Range</th>
<th>What the agent can understand</th>
</tr>
</thead>
</table>
| 1 \( \Delta(o_{L(t)}, o_{L(t')}) = \begin{cases} o_{L(t)} & \text{if } -\exists o_{L(t')} \\
 & \text{if } -\exists o_{L(t)} \\
 & o_{L(t)} - o_{L(t')} & \text{if } o_{L(t)} - o_{L(t')} \neq 0 \\
 & \text{null otherwise} & \end{cases} \) |
| \((-\infty, \infty)\) | The size of change in an observation. |
| 2 \( \Delta(o_{L(t)}, o_{L(t')}) = \begin{cases} 1 & \text{if } o_{L(t)} > o_{L(t')} \\
 & -1 & \text{if } o_{L(t)} < o_{L(t')} \\
 & \text{null otherwise} & \end{cases} \) |
| \([-1, 1]\) | Whether an observation has increased or decreased. |
| 3 \( \Delta(o_{L(t)}, o_{L(t')}) = \begin{cases} 1 & \text{if } o_{L(t)} = o_{L(t')} \\
 & \text{null otherwise} & \end{cases} \) |
| \([1]\) | Whether an observation has changed or not. |

### 2.2.3 Event Functions

Event functions define which combinations of difference variables an agent recognizes as events. By focusing attention on selected difference variables, agents can focus attention on a subset of events and thus a subset of achievement tasks. Two sample event functions are shown in Table 2. The first function provides a narrow focus of attention, with each event containing only one non-zero difference variable. In contrast, using the second function each event focuses attention on every difference variable.

<table>
<thead>
<tr>
<th>Event Function</th>
<th>Number of Events</th>
<th>Events Recognized</th>
</tr>
</thead>
</table>
| 1 \( E_{O(t)} \) = \( E_{L(t)} = (e_{1(t)}, e_{2(t)}, \ldots, e_{L(t)}) \) | \( |O(t) - O(t')| \) | \( E_{1(t')} = (\Delta(o_{1(t)}, o_{L(t')}), 0, 0, 0, \ldots) \)
| \( E_{2(t')} = (0, \Delta(o_{2(t)}, o_{L(t')}), 0, 0, \ldots) \) | \( \ldots \) |
Like the observed states from which they are computed, events may be of varying length or even empty, depending on the number of observations to change. Events may also be normalized for use as input to algorithms used in the motivation process.

Figure 3 summaries our framework for modeling the sensation process for MRL and MSL agents in environments about which there is limited design time knowledge. The sensation process transforms raw sensor data from the environment into two structures to facilitate further reasoning. The observed state $O(t)$ represents the current state of the environment and events $E_{O(t)}$ represent potential learning tasks as changes in the environment. Events are computed using a difference function to compute changes in individual observations and an event function to combine difference variables into events. We use strings from a CFG as a variable length representation of observed states and events. Observed states and events are then passed to the motivation process.

**Figure 3.** The sensation process for MRL and MSL agents in environments about which there is only limited design time knowledge.

### 2.3 Modeling Motivation for Reinforcement and Supervised Learning

The role of the motivation process in RL is to use general, task-independent concepts to generate an intrinsic reward signal that will stimulate the learning of task-oriented behaviors. We represent potential learning tasks using events. Thus, in RL, the motivation process takes at least observed states and events as inputs and outputs at least an intrinsic reward signal to trigger the learning process. Such a motivation process can be thought of as an extrospective motivation process as it computes the motivation signal based only on data from the agent’s environment. In contrast, introspective motivation processes also take into consideration information from other reasoning processes. This may include data output by either the learning or action processes. The motivation process may also maintain memory of the agent’s experiences in its environments. This includes memory of observed states or events. Our model of motivation for RL agents is shown in Figure 4(a).

In SL, agents in complex environments may be exposed to examples of many different tasks simultaneously. The role of the motivation process in this model is to use general, task-oriented concepts to filter examples and stimulate the learning of task-oriented behaviors. This filter functions by directing only subsets of all possible examples to the learning or action processes. Other examples that are not currently motivating do not trigger any
response. Thus, in SL, the motivation process takes at least observed states, examples and events as inputs and optionally outputs a trigger to the learning or action processes. Introspective motivation processes also take into consideration information from other reasoning processes. This may include data output by either the learning or action processes. The motivation process may also maintain memory of the agent’s experiences in its environments. This includes memory of observed states, examples or events. Our model of motivation for SL agents is shown in Figure 4(b).

![Figure 4. Motivation processes for (a) MRL agents and (b) MSL agents.](image)

Our model of motivation for MRL and MSL agents does not define the specific functions used in this process, rather we provide a formal framework which allows us to experiment with and develop metrics for different computational models of motivation. In Section 3 we present three metrics for evaluating MRL and MSL agents using this framework. In Section 4 we show how an existing model of interest can be modified for use in our MRL framework and evaluate this model using our metrics.

### 2.3.1 Motivated Reinforcement Learning Agents

A MRL agent that incorporates our motivation framework is depicted in Figure 5(a). The agent model has four processes: sensation, motivation, learning and activation. \( W(t) \) represents the state of the agent’s environment at time \( t \) and \( S(t) \) represents the state sensed by the agent at time \( t \). The sensation process \( S \) produces an observed state \( O(t) \) and computes events \( E_{O(t)} \) as described in the previous section. The motivation process \( M \) uses our motivation framework to produce a motivation signal \( R_m(t) \) which is then forwarded to the learning process \( L \). The learning process \( L \) performs a RL update such as Q-learning to incorporate the observed state and motivation signal into the behavioral policy \( \pi(t-1) \). This produces an updated behavior \( \pi(t) \), which is stored in memory \( M \). The activation process \( A \) uses an exploration function and action selection rule to select an action \( A(t) \) to perform from the updated behavioral policy \( \pi(t) \). The chosen action \( A(t) \) triggers a corresponding effector \( F(t) \), which makes a change to the agent’s environment.

In order to apply techniques such as Q-learning to observed states represented by strings from a CFG, the state-action table is initialized to be empty. Each time a new state is encountered the agent adds the state to the table, along with entries for each available action, initialized at zero.

### 2.3.2 Motivated Supervised Learning Agents

A motivated supervised learning (MSL) agent model that could incorporate a SL strategy such as decision tree learning or supervised neural network learning is depicted in Figure
5(b). In this model, \( W(t) \) represents the state of the agent’s environment at time \( t \). \( S(t) \) represents the state of the environment as sensed by the agent at time \( t \). The sensation process \( S \) is responsible for decomposing the sensed state into an observed state \( O(t) \), an example action \( X(t) \), if it was provided, and producing a set of events \( E_{o}(t) \). The motivation process \( M \) reasons about the current set of events \( E_{o}(t) \) and the agent’s prior experiences \( E_{p}(t-1) \) and learned behavior \( \pi_{p}(t-1) \) to decide whether the agent should learn about, act on or ignore the sensed data. If learning is chosen, the learning process \( L \) incorporates the observed state and example into the behavior \( \pi_{p}(t-1) \) in memory \( M \) by performing a supervised learning update such as the decision tree update if a decision tree representation is being used for \( \pi \), or a weight update if a supervised neural network representation is being used. If acting is chosen, the activation process \( A \) uses the behavior \( \pi_{p}(t-1) \) to select an action \( A(t) \) to perform. The chosen action \( A(t) \) triggers a corresponding effector \( F(t) \) which makes a change to the agent’s environment.

![Figure 5. Agent models for (a) MRL (b) MSL.](image)

3. Metrics for Motivated Learning Agents

Motivated learning agents are meta-learners in which a motivation function provides a learning algorithm with a motivation signal that focuses learning. The broad role of the motivation function is to use general concepts to motivate adaptable, multi-task learning. A number of existing techniques have been used to measure the emergent learning ability of MRL agents. These techniques can be categorized as those that measure learning efficiency, those that characterize the output of the motivation function and those that visualize the emergent behavior of the agent. Existing techniques have tended to be designed to evaluate specific models of motivation or for use in specific application domains. As a result, they do not provide a good means of comparing the performance of different types of motivated learning agents.

In this section we present three new metrics for evaluating motivated learning agents. The behavioral variety metric evaluates learning efficiency, the focus of attention metric characterizes the output of the motivation function and the scatter plot visualization charts the emergent behavior of the learning agent. The purpose of these new metrics is to provide a consistent approach to each of the three categories of evaluation method so that different types of motivated learning agents can be analyzed and compared. Our metrics extend
existing metrics beyond MRL to include MSL agents and can be used to evaluate different models of motivation.

3.1 Evaluating Learning Efficiency Using Behavioral Variety

The two most common approaches to learning efficiency metrics for RL chart either the reward gained or the number of actions to complete a task against time or learning episodes. In contrast, the learning efficiency of SL agents is commonly measured using the percentage of testing steps which choose actions that agree with example actions provided in training steps for the same observed states. To create an approach that can be applied in both MRL and MSL settings, we present a new metric, behavioral variety, which summarizes learning efficiency based on the tasks (events) learned by an agent. While some existing models do not use the event structure, it is a simple matter to compute events for evaluation purposes by comparing successive states encountered by the learning agent.

In the traditional chart based approach to visualizing the learning efficiency of RL or SL agents, the flat part of the learning curve is generally identified as the period of time that a task has been learned. Flat portions of the learning curve represent periods in which the learned policy is relatively stable, resulting in the learned task being repeated with approximately the same number of actions. To develop metrics to summaries learning efficiency in multi-task learning settings, we first define mathematically what it means for a task to be learned. We model this as the standard deviation $\sigma_{E(t)}$ of the number of actions used to solve a task $E$ during the last $h$ learning episodes (occurrence) for $E$:

$$
\sigma_E = \sqrt{\frac{1}{h-1} \sum_{i=1}^{h} (a_i - \bar{a}_E)^2}
$$

(6)

where $\bar{a}_E$ is the mean number of actions required to repeat $E$ during the last $h$ learning episodes for $E$. A learning episode for $E$ comprises all the actions performed between successive occurrences of the event $E$.

Behavioral variety summarizes learning efficiency by measuring the number of tasks learned. The behavioral variety $V$ of the agent increases, the first time the standard deviation of the number of steps required to repeat a task approaches within some error $r$ of zero. The error term accommodates the random component of exploration strategies such as $\varepsilon$-greedy exploration.

$$
V(t+1) = \begin{cases} 
V(t) + 1 & \text{if } \sigma_{E(t)} < r \text{ for the first time} \\
V(t) & \text{otherwise}
\end{cases}
$$

(7)

Thus, rather than producing multiple curves to represent multiple learned tasks as is required for visualizing existing metrics, we can produce charts that summaries information about $\sigma_E$. The emergent behavioral variety of different MRL algorithms over a set period of time can be compared using simple line charts to indicate the time at which behaviors emerge. Agents that learn policies for more tasks in the same period of time have a greater learning efficiency using this metric.

3.2 Characterizing the Output of the Motivation Function as Focus of Attention

A number of existing techniques have been used to characterize the output of the motivation function. For example, Kaplan and Oudeyer (2003) use line charts to show the evolution of the three motivational variables, predictability, familiarity and stability while Saunders and Gero (2001) use bar charts to characterize the evolution of novelty within their design agents.
To create a more general approach, not tied to the specific motivation function, we again use events as a starting point. Focus of attention characterizes the output of the motivation process as the amount of time an agent dedicates to learning a particular task. This may be measured as an average over multiple lifetimes of a particular agent to gain insight into the types of tasks preferred by agents with a particular motivation function. Alternatively, this metric may be used to measure the focus of attention of a single agent to characterize the type of ‘personality’ it has developed.

In MRL, an agent’s focus of attention on an event $E$ can be calculated as the number of times it assigns the maximum reward $R_{max}$ to that event. In MSL an agent’s focus of attention on an event $E$ can be calculated as the number of times the motivation process triggers the learning process for that event.

### 3.3 Visualizing Emergent Behavior

Existing techniques for visualizing emergent behavior such as the visualization of the headpan position by Kaplan and Oudeyer (2003) have tended to be specific to the application being studied. To create a more general approach which is not tied to the specific application being studied, we introduce a visualization that is based on the actions performed by the agent. The visualization consists of a scatter plot which charts actions performed against time. Behaviors are visible as repeated patterns in the plot. This can be applied to different agents, including MRL and MSL agents, regardless of the types of actions they use.

### 4. Demonstration: Motivated Reinforcement Learning Agents using Interesting Events to Control Support Characters in Virtual Game Environments

This section uses the metrics presented in Section 3 to demonstrate the performance of a computational model of interest in MRL. We first show how an existing model of interest can be modified for MRL and then evaluate the performance of the model in a simulated game environment in which agents control the non-player characters.

#### 4.1 The Environment

The environment used for this demonstration is a simple role-playing game scenario. In role-playing games, support characters fill the game world with interesting people for players to interact with such as merchants who sell equipment, tradesmen who teach skills, guards or innkeepers. However, in contrast to enemies or competitors, support characters are among the least sophisticated artificially intelligent characters in current computer games, limited in both their interactions with players and in their behavior (Laird & van Lent, 2000). The behavior of support characters is generally limited to a looping animation with a few scripted action sequences triggered by a player’s actions. Motivated learning agents offer an alternative to this type of repetitive behavior as they are able to continually change their behavior by learning new sequences of actions in response to their experiences in their environment. In addition, they simplify the character development process for game designers as individual tasks do not have to be pre-programmed. In our game scenario, shown in Figure 7, world states are strings from the grammar in Figure 6.

$$W = \begin{align*}
W_a & \rightarrow \text{<agent><environment>} \\
\text{<agent>} & \rightarrow \text{<location><inventory>} \\
\text{<location>} & \rightarrow \text{<mine> | <smithy> | <forest> | <carpenter>} \\
\text{<mine>} & \rightarrow 1 \\
\text{<smithy>} & \rightarrow 2 \\
\text{<forest>} & \rightarrow 3 \\
\text{<carpenter>} & \rightarrow 4
\end{align*}$$
Thus, in this scenario, a state describes an agent and its environment. An agent has a location and an inventory. Possible locations are enumerated with values between one and four. An agent’s inventory describes the objects it is currently carrying. The agent’s environment is described in terms of the objects the agent can currently sense or “see”. Agents can only see objects at their current location. Thus, depending on their location, agents may be able to see different numbers of objects. While in this simple scenario it is of course possible to predict the exact objects in advance, in complex, dynamic game worlds where players can add content during the course of the game this is not possible. Objects have a value of 1 when they are seen. Some example sensed states in this environment are shown in label-sensation format (L:s) format in Equation 8.

\[
S_1 = \{(location:2)(seePick:1)(seeForge:1)\} \\
S_2 = \{(location:2)(inventoryPick:1)(seeForge:1)\} \\
S_3 = \{(location:4)(inventoryPick:1)(seeAxe:1)(seeLathe:1)\}
\]

(8)

In order for agents to interact with their environment, the following actions are available:

\[
\begin{align*}
A_{a} & \rightarrow \text{pick-up <object>} | \text{move <direction>} | \\
<\text{direction}> & \rightarrow \text{north} | \text{south} | \text{east} | \text{west} \\
<\text{object}> & \rightarrow \text{pick} | \text{forge} | \text{iron} | \text{weapons} | \text{axe} | \text{lathe} | \text{timber} | \text{furniture}
\end{align*}
\]

(9)

In total, there are 20 actions available to an agent. These are enumerated in Table 3 and include actions for moving around in the world, picking objects up, and using objects. Possible tasks therefore include traveling from place to place, cutting timber, mining iron, forging weapons and crafting furniture. Not all actions are available in all world states. Use actions are only available for a particular object if that object is in the current <inventory> or <objects> list of the agent as appropriate for that object. For example, it is appropriate to pick-up an axe and have it in inventory for later use, but the forge, which is too heavy to be picked up, can be used whenever the agent can sense it. Move actions are available in any world state. A use action produces the desired result, such as using the pick to mine iron, 90% of the time and no result 10% of the time. Pick-up actions are only available for a particular object if that object is in the current <objects> list of the agent, that is, if the agent can sense it.

<table>
<thead>
<tr>
<th>Table 3 – Enumeration of agent actions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_i(move, north)</td>
</tr>
<tr>
<td>A_i(move, south)</td>
</tr>
<tr>
<td>A_i(move, east)</td>
</tr>
<tr>
<td>A_i(move, west)</td>
</tr>
<tr>
<td>A_i(pick-up, pick)</td>
</tr>
<tr>
<td>A_i(use, pick)</td>
</tr>
<tr>
<td>A_i(pick-up, iron)</td>
</tr>
<tr>
<td>A_i(use, iron)</td>
</tr>
</tbody>
</table>
The actions which would produce the state sequence in Equation 8 are Action 5 and Action 3: picking up the pick then moving east. Using Difference Function 1 from Table 1 and Event Function 2 from Table 2, the events produced by these actions are:

\[ E_{10} (\{(\text{inventoryPick}:1)(\text{seePick}:-1)\}) \]
\[ E_{11} (\{(\text{location}:2)(\text{seeForge}:-1)(\text{seeAxe}:1)(\text{seeLathe}:1)\}) \]

\[ E_{10} \] describes the addition of one pick to the agent’s inventory and the removal of one pick from the agent’s line of site. Thus, this event represents a successful execution of the pick-up pick action. Likewise \[ E_{11} \] describes a change in location of two units, the removal of the forge from the agent’s line of sight and the addition of the axe and lathe to the agent’s line of site. Thus, this event represents a successful execution of the move east action between the smithy and the carpenter’s shop. Successful execution of a use action will produce an event which either adds or removes one unit of a raw material to the agent’s inventory. For example, the event \[ E(\text{inventoryIron}:1) \] is a result of successful use of the pick while \[ E(\text{inventoryIron}:-1) \] represents successful use of the forge. We do not represent the addition of weapons or furniture to the agent’s line of sight as this would cause a potentially infinite state space which cannot be addressed by standard RL algorithms.

While the environment used in this demonstration is relatively simple, it is sufficient to demonstrate the ability of motivated learning agents to learn solutions to multiple tasks. For the experiments in this paper, we implemented this game environment as a Java simulation. The agent model can also be connected to the Second Life virtual environment using XML-RPC to communicate sensations and actions between the agent program and the virtual world (Merrick & Maher, 2006). In Second Life, there is potential for human interaction with the agents and their environment, making the environment dynamic and unpredictable.

4.2. The Agent

In a game environment, support characters must develop a set of interesting behaviors to define their personality. As there are no examples from which they can learn, MRL agents are the most appropriate choice, in preference to MSL agents. In addition to the
computational models of motivation developed with reinforcement learning in mind, a number of other computational models have been developed for use in other types of agents. These include models of biological theories of motivation such as drive theory (Canamero, Avila-Garcia, & Hafner, 2002) and cognitive theories such as curiosity and interest (Saunders & Gero, 2002). Cognitive theories about phenomena such as curiosity and interest explain this search in terms of constant adjustments and adaptations to a baseline level of stimulation from the environment which in turn defines some moderate, optimal stimulation level. As we are interested in building agents that can adjust and adapt their behavior to learn multiple tasks in response to their environment, these cognitive theories make an ideal starting point for motivation functions.

Saunders and Gero implemented a computational model of interest for social force agents (Saunders & Gero, 2002) by first detecting the novelty of environmental stimuli then using this novelty value to calculate interest. The novelty of an environmental stimulus is a measure of the difference between expectations and observations of the environment where expectations are formed as a result of an agent’s experiences in its environment. Saunders and Gero (2002) model these expectations or experiences using an Habituated Self-Organizing Map (HSOM) (Marsland, Nehmzow, & Shapiro, 2000). Interest in a situation is aroused when its novelty is at a moderate level, meaning that the most interesting experiences are those that are similar-yet-different to previously encountered experiences. The relationship between the intensity of a stimulus and its pleasantness or interest is modeled using the Wundt curve (Berlyne, 1971). The Saunders and Gero (2002) model can be modified for use in MRL with variable length events as input (Merrick & Maher, 2006).

In order to compute interest, we use Difference Function 1 from Table 1 to assign a value to changes in observations. This difference function provides the most information about changes in observations. These differences are combined using an Event Function 2 from Table 2 to produce a single event. This event, which provides complete information about the change that occurred at a particular time step, is passed to the motivation process. Empty events representing no change to the environment are assigned an interest value of zero. Events that are not empty are passed to an HSOM to compute novelty and then through the Wundt curve to compute interest. We modify the basic SOM algorithm to accept variable length inputs by initializing each neuron U as a zero length vector. Each time a stimulus event \( E(t) \) is presented to the SOM, each neuron \( U \) is lengthened by adding randomly initialized variables \( u_L \) with any labels \( L \) that occur in \( E(t) \) but not in \( U \). The winning neuron is chosen by selecting the neuron \( U(t) \) with the minimum distance \( d \) to the stimulus event where \( d \) is calculated using the SOM distance function modified to accept variable length events by incorporating the chosen difference function as follows:

\[
d = \sqrt{\sum L \Delta(u_{L(t)} \cdot e_{L(t)})^2}
\]  

(11)

All neurons in the winning neighborhood are then moved closer to the input stimulus by adjusting their weights using the SOM update equation, again modified to incorporate the difference function:

\[
u_{L(t+1)} = u_{L(t)} + \eta \Delta(e_{L(t)}, u_{L(t)})
\]  

(12)

\( 0 \leq \eta \leq 1 \) is the learning rate of the SOM. The activities of each neuron are then propagated up the synapse to the habituating layer. The synaptic efficacy \( N_{(j)} \), which represents the novelty of the stimulus event \( E_{(j)} \), is then calculated using Stanley’s model of habituation.
\[
\frac{dN(t)}{dt} = \alpha [N(0) - N(t)] - \sigma(t) 
\]

(13)

\(\alpha\) is a constant governing the rate of habituation while \(\tau\) governs the rate of recovery\(^1\).

After the novelty of a given stimulus has been generated, interest \(I\) is calculated using the Wundt curve in Equation 14. The Wundt curve provides positive feedback \(F^+\) for the discovery of novel stimuli and negative feedback \(F^-\) for highly novel stimuli. It peaks at a maximum value for a moderate degree of stimulation as shown in Figure 8, meaning that the most interesting events are those that are similar-yet-different to previously encountered experiences. \(F^\max\) is the maximum positive feedback, \(F^-\max\) is the maximum negative feedback, \(\rho^+\) and \(\rho^-\) are the slopes of the positive and negative feedback sigmoid functions, \(F^\min^+\) is the minimum novelty to receive positive feedback and \(F^-\min\) is the minimum novelty to receive negative feedback.

\[
I(2N(t)) = F^+(2N(t)) - F^-(2N(t)) = \frac{F^-\max}{1 + e^{-\rho^+ (2N(t) - F^\max)}} - \frac{F^+\max}{1 + e^{-\rho^- (2N(t) - F^-\max)}} 
\]

(14)

As this model uses only a single event at each time step and a single motivation function, interest, the interest value for the event \(E(t)\) at time \(t\), is used directly as the motivation signal \(R_m(t)\). This value is passed from the motivation process \(M\) to the learning process \(L\).

![Figure 8. The Wundt curve is the difference between positive and negative feedback functions. Image from (Merrick & Maher, 2006).](image)

Table 4 shows the parameter values used in these agents as well as the parameters used in the metrics. Initially, all our agents were located at the mine and carrying nothing. Each agent was allowed to run for 20,000 iterations, meaning that it would perform exactly 20,000 actions in its lifetime. Where appropriate, results are averaged over 20 runs of each agent and show the 95% confidence interval.

<table>
<thead>
<tr>
<th>Table 4 – Agent parameters and their values.</th>
</tr>
</thead>
</table>

\(^1\) Splitting the habituation constant \(\tau\) into \(\tau_1\) and \(\tau_2\) where \(\tau_1\) governs the rate of habituation in neurons in the winning neighbourhood and \(\tau_2\) governs the rate of habituation in losing neurons allows the HSOM to learn more quickly than it forgets if \(\tau_2 > \tau_1\).
### Reinforcement Learning Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma )</td>
<td>0.9</td>
</tr>
<tr>
<td>( \beta )</td>
<td>0.9</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Motivation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \eta )</td>
<td>0.1</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>1.05</td>
</tr>
<tr>
<td>( \tau_1 )</td>
<td>3.3</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>14.3</td>
</tr>
<tr>
<td>( F_{\max}^+ ) and ( F_{\max}^- )</td>
<td>1</td>
</tr>
<tr>
<td>( \rho^+ ) and ( \rho^- )</td>
<td>10</td>
</tr>
<tr>
<td>( F_{\min}^+ )</td>
<td>0.5</td>
</tr>
<tr>
<td>( F_{\min}^- )</td>
<td>1.5</td>
</tr>
</tbody>
</table>

### Metric Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h )</td>
<td>20</td>
</tr>
<tr>
<td>( r )</td>
<td>1</td>
</tr>
<tr>
<td>( R_{\max} )</td>
<td>0.8</td>
</tr>
</tbody>
</table>

### 4.3. Results

Figure 9 shows the evolution of behavioral variety for MRL agents in the game environment. The motivation function in the MRL agent rewarded between 14 and 17 interesting tasks, including a mixture of traveling between different locations \( E_1((\text{location}:1)) \ldots \), cutting timber \( E_{10}((\text{inventoryTimber}:1)) \), crafting furniture \( E_9((\text{inventoryTimber}:1)) \), mining iron \( E_{12}((\text{inventoryIron}:1)) \) and forging weapons \( E_{11}((\text{inventoryIron}:1)) \). The rate of emergence of behaviors is initially relatively fast, but slows over time. This is because the interest based motivation function tends to focus attention on simpler behaviors earlier in the agent’s life and more complex behaviors later. More complex behaviors take more time to learn and thus the rate of emergence of new behaviors slows. Figure 9 shows that new behaviors cease to emerge after about 8000 time steps, despite there being a total of approximately 45 potential learning tasks in the environment. Rather than learning every task, each agent tends to focus its attention on a particular subset of tasks in the environment. This means that, after the first 8000 steps, agents tend to specialize in a subset of tasks in their environment. This is also captured by the focus of attention metric.

![Figure 9. Evolution of behavioral variety by MRL agents.](image)

Figure 10 shows the relative time spent learning different tasks over 20 runs of the MRL agent. In general, the MRL agent tends to devote more time to traveling tasks involving a change in location than it does to tasks such as mining iron, forging weapons, cutting timber or crafting furniture. This is an artifact of the HSOM based motivation function, which tends to favor more frequent events such as the location changes. In general, the more frequent
events tend to be those which are easier to learn about, because they occur more readily, thus providing more potential learning episodes for the agent.

While Figure 10 shows that the general trend among 20 MRL agents is to focus attention on more frequent events, there is substantial variation between individual MRL agents. Figure 11 shows the focus of attention by two specific MRL agents. The key focuses of the first agent are traveling between the carpenter shop and the forest, the forest and the mine and between the smithy and the carpenter shop. That is, the first agent represents a support character that is a traveler. The agent controlling this character finds traveling events interesting and rewards them, resulting in a tendency to learn and perform traveling behaviors. In contrast, the key focuses of the second agent are cutting timber, traveling between the forest and the carpenter shop, crafting furniture and traveling from the carpenter shop to the forest. That is, the second agent represents a support character that is a cabinet maker. In summary, each agent, while initially identical, focuses its attention on a slightly different set of tasks during its lifetime. This is useful for game developers as it means that support characters with different behaviors can be achieved without having to individually program the behavior of each character. Figure 11 also shows that, while each agent has a number of key tasks on which their attention is primarily focused, they also spend significant amounts of time on other tasks. For example, the traveler agent also spends some time making weapons and furniture. This variation avoids the monotonous, repetitive behavior that is found in existing support characters.

Figure 10. Average focus of attention by 20 MRL agents.

Figure 11. Focus of attention by two specific MRL agents displaying emergent traveler behavior (top) and emergent cabinet making behavior (bottom).
Finally, Figure 12 shows the actual actions performed by an MRL agent against time and identifies tasks learned. The key for the actions is shown in Table 3. At T1, the action scatter plot shows a preference by the agent for actions 18, 10, 1 and 2: using the lathe and the axe and moving north and south between the forest and the carpenter shop. Noise on the scatter plot is a result of the random action selection in the ε-greedy exploration strategy. In contrast, in the period denoted T2, the action scatter plot shows a preference by the agent for actions 14, 6, 1 and 2: using the forge and pick and traveling north and south between the mine and the smithy. In other periods, the action scatter plot shows a preference for actions 1, 2, 3 and 4: the move north, south, east and west actions. In these periods the agent is performing a traveling behavior between some subset of the locations in the environment. Figure 12 shows visually how an MRL agent develops and adapts clear behavioral patterns in response to shifting interest. The action scatter plot shows periods in which different subsets of actions are used and it shows several different subsets of actions being performed at different times in the agent’s life.

![Figure 12. Actions performed by an MRL agent and some of the tasks learned.](image)

In addition to demonstrating that task-independent models of motivation can motivate task-oriented emergent learning, these results have demonstrated the new metrics introduced in Section 3. They show how the evolution of behavioral variety, focus of attention and emergent behavior of motivated learning agents can be evaluated in a manner that is independent of the particular model of motivation or problem domain being studied. These metrics extend existing work by providing a consistent approach to the evaluation of motivated learning agents using different motivation or learning models. In particular, our metrics have provided insight into the effects of motivation on learning that have not previously been possible. The rate of emergence of different behaviors revealed by the behavioral variety metric and the focus of attention metrics characterize the emergent behavior as focusing first on simpler tasks and later, more briefly, on more complex tasks. This insight has the potential to inform the development of new models of motivation which are able to exhibit more efficient learning by modifying the focus of attention to provide more time for more complex tasks.

### 5. Discussion and Future Work

While this paper has explored two models for motivated learning and demonstrated the potential for task independent models such as interest to motivate task-oriented reinforcement learning, our experiments also raise a number of questions about the scalability and extensibility of motivated learning systems. In the next sections we discuss our results so far with reference to the future research directions they suggest.
5.1. Scalability and Dynamic Environments

The results in this paper have shown that it is possible to use a general, task-independent motivation process to motivate the learning of multiple tasks. This has favorable implications for developing systems capable of life-long learning in complex environments. However, while we have demonstrated the ability of motivated learning agents to identify and learn about multiple tasks in a simple, static environment we do not yet fully understand the scalability issues that may be associated with motivated learning agents in complex, dynamic environments. Three further sets of experiments that analyze scalability would contribute to this understanding.

First, to understand the behavior of motivated learning agents as the complexity of potential learning tasks increases, the test environment used in this paper could be scaled so that two, three or more units of raw material are required to create a finished product. This increases the number of actions required by the agent to complete a task. Our hypothesis is that the interest based motivation process will fail at some point to motivate learning of more complex tasks. This is because the nature of the novelty based Wundt curve such that, there is no guarantee that a given event will remain interesting for long enough for a behavior to be learned. One possible solution to this problem lies in the development of motivation functions that take into account the agent’s current competency, represented in terms of the policy \( \pi \) output by the learning process.

A second way in which the complexity of the environment using in this paper might be increased is through the addition of further raw materials and tools. Our hypothesis is that this would not affect the performance the interest based model. Agents using this model should continue to identify a similarly sized subset of tasks to be learned.

Finally, the demonstration in this paper does not consider environments that change while the agent is learning. In such environments, tasks identified as interesting may become impossible to achieve or new tasks may become available. To understand scalability of motivated learning agents in such environments, our environment could be modified by the random appearance of a monster which blocks access to one of the raw materials. This would render a number of tasks obsolete but also introduce new tasks in the form of fighting the monster. Our hypothesis is that, given enough time, motivated learning agents should be able to adapt to such changes.

5.2. Hierarchical Learning for Recall and Abstraction

In this paper, we have proposed motivated learning agent models incorporating standard RL and SL algorithms. These algorithms maintain only a single policy representing a solution for a single task at any particular time. While this is appropriate for applications in which adaptability is more important than memory, such as NPCs in computer games where it is the emergent learning that contributes to the appeal of the characters, in other applications it may be an advantage for agents to be able to recall and reuse learned behaviors. In addition to the roles identified in this paper, the motivation process would also play a role in motivating the creation or removal of learned behavioral abstractions.

5.3. Motivated Supervised Learning

While MRL was an appropriate choice for character control in gaming environments, in other environments learning by trial and error is inappropriate. For example, one of the key goals of intelligent environment research is to build environments such as meeting rooms that can unobtrusively adapt to the changing needs of their inhabitants. In such scenarios, learning by trial and error is inappropriate, however the actions of human occupants of the meeting room...
form a potential source of examples for SL. Intelligent environments must necessarily learn multiple tasks from examples produced from many, often simultaneous users, making motivated supervised learning a possible approach. While we now understand the architecture of a motivated supervised learning agent, further research is required to develop models of motivation for such agents.

5.4. Societies of Motivated Agents

In this paper we have focused on the emergent of task oriented behaviors in a single agent. However in many situations agents must interact with their environment as part of a society. The game scenario in this paper is just one example of a situation in which one would expect to find multiple agents working towards their own goals. Our experimental domain could be extended to provide a greater understanding of the behavior of motivated learning agents in multi-agent settings by placing multiple agents in the environment simultaneously.

6. Conclusion

This paper has introduced motivated learning within an agent framework by presenting models for motivated reinforcement learning agents and motivated supervised learning agents. In our models, a motivation process provides a learning stimulus to reinforcement learning algorithms to direct learning and acts as a filter in supervised learning to focus learning. We introduced context-free grammars as a flexible representation of environments about which there is only limited design time knowledge, and events to represent tasks as changes in the environment. We then presented agent models for motivated reinforcement learning and motivated supervised learning which use this framework. These models use a consistent approach to the description of motivation which allow us to extend existing work beyond motivated reinforcement learning to consider the role of motivation in supervised learning. The formalisms used to describe motivation in reinforcement learning and supervised learning enable the definition of new metrics for evaluating the ability of these new types of learning agents to achieve adaptive, multi-task learning. These metrics provide a consistent approach to the evaluation of motivated learning agents using different models of motivation or different learning algorithms. We used these metrics to evaluate a computational model of interest as a motivation process for motivated reinforcement learning agents controlling NPCs in a simulated computer game environment. This showed a number of key results. Firstly, we showed that it is possible to use a general, task-independent motivation process to motivate the learning of multiple tasks. The presence of a task-oriented extrinsic reward signal is not necessary for learning to occur. Because motivated learning agents focus their attention based on their experiences in their environment, agents which have different experiences will learn about different tasks, making it possible to create a number of agents with different competencies using a single agent architecture and a common motivation function. This work offers a starting point for a range of future work directions towards achieving learning in complex, dynamic and open ended environments where content and tasks cannot be predicted in advance.
Acknowledgements

This research was supported by a National ICT Australia PhD scholarship. National ICT Australia is funded by the Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council.
References