In *AAMAS Adaptive Learning Agents Workshop* (ALA 2012),
Valencia, Spain, June 2012.

Intrinsically Motivated Model Learning for a Developing Curious Agent

Todd Hester
University of Texas at Austin
Department of Computer Science
Austin, TX 78751
todd@cs.utexas.edu

Peter Stone
University of Texas at Austin
Department of Computer Science
Austin, TX 78751
pstone@cs.utexas.edu

ABSTRACT

Reinforcement Learning (RL) agents could benefit society by learning tasks that require learning and adaptation. However, learning these tasks efficiently typically requires a well-engineered reward function. Intrinsic motivation can be used to drive an agent to learn useful models of domains with limited or no external reward function. The agent can later plan on its learned model to perform tasks in the domain if given a reward function. This paper presents the TEXPLORE with Variance-And-Novelty-Intrinsic-Rewards algorithm (TEXPLORE-VANIR), an intrinsically motivated model-based RL algorithm. The algorithm learns models of the transition dynamics of a domain using decision trees. It calculates two different intrinsic rewards from this model: one to explore where the model is uncertain, and one to acquire novel experiences that the model has not yet been trained on. This paper presents experiments demonstrating that the combination of these two intrinsic rewards enables the algorithm to learn an accurate model of a domain with no external rewards and that the learned model can be used afterward to perform tasks in the domain. While learning the model, the agent explores the domain in a developing and curious way, progressively learning more complex skills. In addition, the experiments show that combining the agent's intrinsic rewards with external task rewards enables the agent to learn faster than using external rewards alone.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Reinforcement Learning, Artificial Curiosity, Developmental Learning

1. INTRODUCTION

Reinforcement Learning (RL) agents could be useful in society because of their ability to learn and adapt to new environments and tasks. However, they typically require a well-engineered reward function to specify the task to be learned and enable it to be learned efficiently. Intrinsic motivation can be used to make agents learn more efficiently by augmenting external rewards in a task. They can also be used in the absence of external rewards to drive an agent to learn as much as possible about the world. This learned knowledge can be used later to perform tasks in the world.

This paper presents an intrinsically motivated model-based RL algorithm, called TEXPLORE with Variance-And-Novelty-Intrinsic-Rewards (TEXPLORE-VANIR), that learns a model of a domain without external rewards. The agent is based on a typical model-based RL framework. TEXPLORE-VANIR combines model learning through the use of decision trees with two unique intrinsic rewards calculated from the tree models. TEXPLORE-VANIR uses its decision tree models to calculate two different intrinsic rewards. The first reward is based on *variance* in its models' predictions to drive the agent to explore where its model is uncertain. The second reward drives the agent to *novel* states which are the most different from what its models have been trained on so far. The combination of these two rewards enables the agent to explore in a developing curious way, learning progressively more complex skills in the domain. In addition, the agent can learn an accurate and useful model of the domain.

This paper presents two main contributions:

1. Novel methods for obtaining intrinsic rewards from a decision tree based model of the world.
2. The TEXPLORE-VANIR algorithm for intrinsically motivated model learning.

Section 4 presents experiments showing that the algorithm: 1) explores in a developing, curious way; 2) learns a more accurate model than other approaches; and 3) can use its learned model later to perform tasks specified by a reward function. In addition, it shows that the agent can use the intrinsic rewards in conjunction with external rewards to learn a task faster than if using external rewards alone.

2. BACKGROUND

This section presents background in two main areas. Section 2.1 covers background material on Reinforcement Learning (RL), which is used as the framework for TEXPLORE-VANIR. Section 2.2 describes background on other approaches to Intrinsic Motivation (IM).

2.1 Reinforcement Learning

We adopt the standard Markov Decision Process (MDP) formalism for this work [23]. An MDP is defined by a tuple $\langle S, A, R, T \rangle$, which consists of a set of states S, a set of actions A, a reward function $R(s,a)$, and a transition function $T(s,a,s') = P(s'|s,a)$. In each state $s \in S$, the agent takes an action $a \in A$. Upon taking this action, the agent receives a reward $R(s, a)$ and reaches a new state s', determined from the probability distribution $P(s'|s, a)$. Many domains utilize a factored state representation, where the state s is represented by a vector of n state variables: $s = \langle x_1, x_2, ..., x_n \rangle$. A policy π specifies for each state which action the agent will take.

The value $Q^\pi(s, a)$ of a given state-action pair (s, a) is an estimate of the expected future reward that can be obtained from (s, a) when following policy π. The goal of the agent is to find the policy π mapping states to actions that maximizes the expected discounted total reward over the agent's lifetime. The optimal value function $Q^*(s, a)$ provides maximal values in all states and is determined by solving the Bellman equation:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a'), \qquad (1)$$

where $0 < \gamma < 1$ is the discount factor. The optimal policy π is then as follows:

$$\pi(s) = \mathrm{argmax}_a Q^*(s, a). \qquad (2)$$

RL methods fall into two general classes: model-based and model-free methods. Model-based RL methods learn a model of the domain by approximating $R(s, a)$ and $P(s'|s, a)$ for each state and action. The agent can then calculate a policy (i.e. plan) using this model. Model-free methods update the values of actions only when taking them in the real task. One of the advantages of model-based methods is their ability to plan multi-step exploration trajectories. The agent can plan a policy to reach intrinsic rewards added into its model to drive exploration to interesting state-actions.

2.2 Intrinsic Motivation

This section presents background on Intrinsic Motivation (IM). Work on intrinsically motivated agents originally came from two different goals [16]. The first goal is for the intrinsic motivation to drive the agent to maximize its knowledge about the world and its ability to control it. The second goal is to enable cumulative, open-ended learning on robots.

Work towards the first goal fits nicely within the RL framework. Intrinsic rewards can be used with model-free RL agents to drive them to update their value function (and thus learn a good policy) as quickly as possible [5]. For model-based agents, rewards can be used to drive the agent to learn an accurate model as efficiently as possible [2].

These different approaches demonstrate that the correct intrinsic motivation is dependent on the type of algorithm. For example, with a Q-LEARNING agent [25], it makes sense to give intrinsic rewards for where the value backups will have the largest effect, as done in [5]. When learning with a tabular model, the agent must gain enough experiences in each state-action to learn an accurate model of it. Thus it makes sense to use intrinsic motivation to drive the agent to acquire these experiences, as done by R-MAX [2].

This work takes the approach of using a model-based RL algorithm in a domain with no external rewards. This approach can be thought of as a pure exploration problem, where the agent's goal is simply to learn as much about the world as possible. Bayesian methods such as Duff's optimal probe [7] attempt to solve this problem optimally, but are computationally intractable. TEXPLORE-VANIR extends a model-based RL algorithm called TEXPLORE to use intrinsic motivation. Its intrinsic rewards should drive it to quickly learn an accurate model in a domain with no external rewards.

3. ALGORITHM

This section presents the TEXPLORE with Variance-And-Novelty-Intrinsic-Rewards algorithm (TEXPLORE-VANIR), an algorithm for an intrinsically motivated curious agent. First we present some essential properties that the agent should have, before presenting its specific approach to model learning and intrinsic motivation.

Our goal is to develop an RL agent that uses intrinsic rewards to explore in a developing curious way and learn a useful model of the domain's transition dynamics. To this end, we have the following desiderata for such an algorithm:

1. The algorithm should be model-based, both to enable multi-step exploration trajectories and to allow the agent to use the learned model later to perform tasks.

2. It should incorporate generalization into its model learning to learn the model quickly.

3. It should not be required to visit every state-action in the task.

4. It should decide which areas of the state space are interesting to it without having to sample them first.

Now, we present TEXPLORE-VANIR, an algorithm that has all of these desired properties. It uses decision trees to learn models that generalize predictions across state-actions in the domain, similar to SPITI [6] and TEXPLORE [10]. It incorporates intrinsic rewards that drive the agent towards two types of state-actions: ones where its model is uncertain and ones where its model may have generalized incorrectly. TEXPLORE-VANIR extends the TEXPLORE model-based RL algorithm [10] by incorporating intrinsic rewards that motivate the agent to explore in a developing, curious way to learn useful models of domains with no external rewards.

TEXPLORE-VANIR follows the typical approach of a model-based RL agent. It plans a policy using its learned model (including intrinsic rewards), takes actions following that policy, acquiring new experiences which are used to improve its model, and repeats. In order to be applicable to learning on robots, TEXPLORE-VANIR uses the Real-Time Model Based Architecture presented in [9]. This architecture uses approximate planning with UCT [13] and parallelizes the model learning, planning, and acting such that the agent can take actions in real-time at a specified frequency.

The following section presents details on TEXPLORE's model learning approach, which is used by TEXPLORE-VANIR. Then, Section 3.2 describes how the learned models are used in calculating intrinsic rewards for exploration by TEXPLORE-VANIR.

3.1 Model Learning

Making the intrinsically motivated agent model-based enables it to: 1) plan multi-step exploration trajectories; 2) learn faster than model-free approaches; and 3) use the learned model to solve tasks given to it after its learning. It is desirable for the model to generalize the learned transition and reward dynamics across state-actions. This generalization enables the model to make predictions about unseen or infrequently visited state-actions, and therefore not have to visit each and every one. Thus, TEXPLORE-VANIR approaches the model learning task as a supervised learning problem, with the current state and action as the input, and the next state as the output to be predicted. In order to improve generalization, TEXPLORE-VANIR follows the approach of [15] and [11] in predicting the relative transitions $(s' - s)$ between states rather than absolute outcomes.

Like Dynamic Bayesian Network (DBN) based RL algorithms [8, 4], TEXPLORE-VANIR learns a model of the factored domain by learning a separate prediction for each of the n state features. TEXPLORE-VANIR assumes that each of the state variables transitions independently, as DBN-based methods do. Therefore, the separate feature predictions can be combined to create a prediction of the complete state vector.

TEXPLORE-VANIR follows the approach of TEXPLORE [10] by using a decision tree to predict the change in each state feature. The decision trees are learned using an implementation of Quinlan's C4.5 algorithm [18]. The C4.5 algorithm builds a tree that splits on the inputs (current state features and action), with each leaf of the tree making a different prediction. The splits are formed by choosing the feature with the highest information gain. Decision trees were chosen because they generalize broadly at first, but can be refined with training to make accurate predictions for individual state-actions.

Another desirable property for the model learning algorithm to have is a measure of uncertainty in the model's predictions. TEXPLORE-VANIR accomplishes this goal by building multiple hypotheses of the true model of the domain in the form of a random forest. The variance of the different trees' predictions can be used as a measure of the uncertainty in the model. A random forest is a collection of m decision trees, each of which differ because they are trained on a random subset of experiences and have some randomness when choosing splits at the decision nodes. The agent then plans over the average of the predictions made by each tree in the forest. Random forests have been proven to converge with less generalization error than individual tree models [3]. A complete diagram of TEXPLORE-VANIR's model learning process is shown in Figure 1.

3.2 Intrinsic Motivation

The main contribution of this paper is a method for extending the model-based TEXPLORE algorithm for learning specific RL tasks to the TEXPLORE-VANIR algorithm for exploration by a curious agent. This curiosity is operationalized via intrinsic rewards for 1) preferring to explore areas of the state space where there is a large degree of uncertainty in the model, and 2) preferring regions of the state space that are far from previously explored areas (regardless of how certain the model is).

The variance of the predictions of each of the trees in the forest can be used to motivate the agent towards the state-actions where its models disagree. These state-actions are the ones where there are still multiple hypotheses of the true model of the domain that make differing predictions. TEXPLORE-VANIR calculates a measure of the variance in the predictions of the change in each state feature for a given state-action:

$$D(s,a) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} D_{KL}(P_j(x_i^{rel}|s,a)||P_k(x_i^{rel}|s,a)), \quad (3)$$

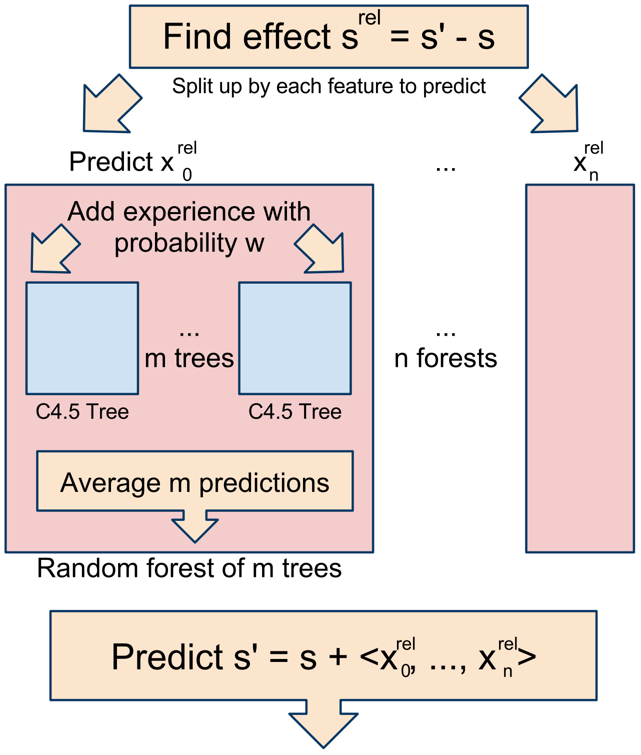

Find effect s^rel = s' - s
Split up by each feature to predict
Predict x^rel_0
...
x^rel_n
Add experience with probability w
...
m trees
C4.5 Tree
C4.5 Tree
...
n forests
Average m predictions
Random forest of m trees
Predict s' = s + <x^rel_0, ..., x^rel_n>

Figure 1: *Model Learning.* This is how the algorithm learns a model of the domain. The agent calculates the difference between s' and s as the transition effect s^{rel}. Then it splits up the state vector and learns a random forest to predict each state feature. Each random forest is made up of stochastic decision trees, which get each new experience with probability w. The random forest's predictions are made by averaging each tree's predictions, and then the predictions for each feature are combined into a complete model of the domain.

where for every pair of models (j and k) in the forest, it sums the KL-divergences between the predicted probability distributions for each feature i. This measure $D(s,a)$ tells us how much the predictions of the different models disagree. This measure is different than just measuring where the predictions are noisy, as $D(s,a)$ will be 0 if all the tree models predict the same stochastic outcome distribution. An intrinsic reward proportional to this variance measure, called the VARIANCE-REWARD, can then be incorporated into the agent's model for planning:

$$R(s,a) = vD(s,a), \quad (4)$$

where v is a coefficient determining how big this reward should be. This reward can be combined with other rewards (intrinsic or extrinsic) to drive the agent.

This reward will drive the agent to the state-actions where its models have not yet converged to a single hypothesis of the world's true dynamics. However, there will still be cases where all of the agent's models are making incorrect predictions. Therefore, TEXPLORE-VANIR also needs a measure of how likely it is for the model's predictions to be incorrect. For the decision tree model that TEXPLORE-VANIR uses, the model is more likely to be incorrect when it has to general-

State Features	ID, X, Y, KEY, LOCKED, RED-E, RED-W, RED-N, RED-S, GREEN-E, GREEN-W, GREEN-N, GREEN-S, BLUE-E, BLUE-W, BLUE-N, BLUE-S,
Actions	EAST, WEST, NORTH, SOUTH, PRESS, PICKUP

Table 1: Properties of the Light World domain.

ize its predictions farther from the experiences it is trained on. Therefore, TEXPLORE-VANIR utilizes a second intrinsic reward based on the L_1 distance in feature space from a given state-action and the nearest one that the model has been trained on. If X is a set of all seen state-actions, then the L_1 distance from a given state-action to the nearest seen state-action is:

$$\delta(s,a) = \min_{\langle s_x, a_x \rangle \in X} || \langle s, a \rangle - \langle s_x, a_x \rangle ||_1. \qquad (5)$$

A reward proportional to this distance, called the NOVELTY-REWARD, drives the agents to explore the state-actions that are the most novel compared to the state-actions it has seen before:

$$R(s,a) = n\delta(s,a), \qquad (6)$$

where n is a coefficient determining how big this reward should be. One nice property of this reward is that given enough time, it will drive the agent to explore *all* the state-actions in the domain, as any unvisited state-action is different in some feature from the visited ones. However, it will start out driving the agent to explore the state-actions that are the most different from ones it has seen.

The TEXPLORE with Variance-And-Novelty-Intrinsic-Rewards algorithm (TEXPLORE-VANIR) is completed by combining these two intrinsic rewards together. They can be combined with different weightings of their coefficients (v and n) to drive the agent to both explore novel state-actions where its model may have generalized incorrectly and state-actions where its model is uncertain. In addition, these two intrinsic rewards can be combined with an external reward defining a task. A combination of these two intrinsic rewards should drive the agent to explore in a developing and curious way: seeking out novel and interesting state-actions, while exploring increasingly complex parts of the domain.

4. EMPIRICAL RESULTS

Evaluating an agent's curiosity is not as straightforward as evaluating a standard RL agent on a specific task. Rather than attempting to accrue reward on a specific task, a curious agent's goal is better stated as preparing itself for any task. We therefore evaluate TEXPLORE-VANIR in four ways on a complex domain with no external rewards. First, we measure the accuracy of the agent's learned model in predicting the domain's transition dynamics. Second, we test whether the learned model can be used to perform tasks in the domain when given a reward function. Third, we examine the agent's exploration to see if it is exploring in a developing, curious way. Finally, we demonstrate that TEXPLORE-VANIR can combine its intrinsic rewards with external rewards to learn faster than if it was given only external rewards. These results demonstrate that the intrinsic rewards and model learning approach TEXPLORE-VANIR uses are sufficient for the agent to explore in a developing curious way and to learn a transition model that is useful for performing tasks on the domain.

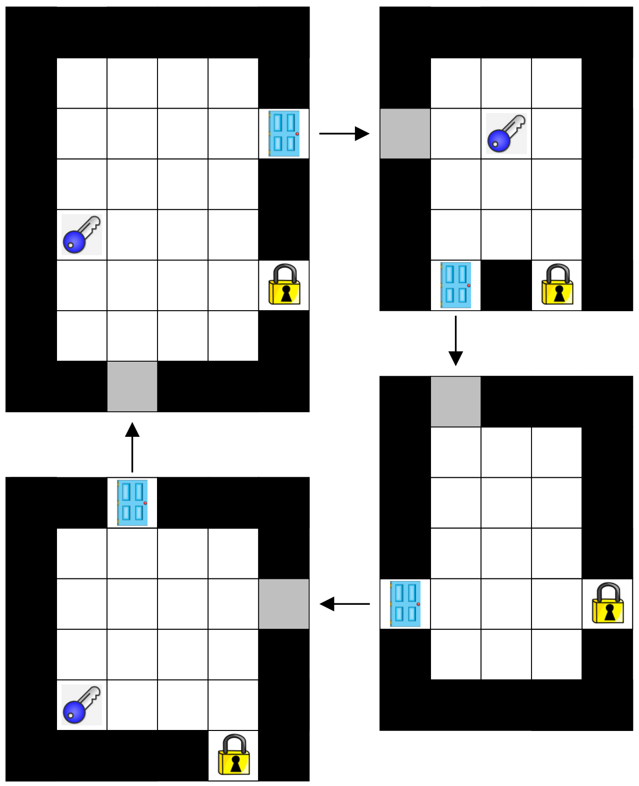


Figure 2: The Light World domain. In each room, the agent must navigate to the key, pickup they key, navigate to the lock, press it, and then navigate to and exit through the door to the next room.

The agent is tested on the Light World domain, presented in [14], and shown in Figure 2. In this domain, the agent goes through a series of rooms. Each room has a door, a lock, and possibly a key. The agent must go to the lock and press it to open the door, at which point it can then leave the room. It cannot go back through the door in the opposite direction. If a key is present, it must pickup the key before pressing the lock. Open doors, locks, and keys each emit a different color light that the agent can see. The agent has sensors that detect each color light in each cardinal direction. The sensors have a maximal value of 1 when the agent is at the light, and their values go down to 0 when the light is 20 steps away. The agent's state is made up of 17 different features: its X and Y location in the room, the ID of the room it is in, whether it has the KEY, whether the door is LOCKED, as well as the values of the 12 light sensors, which detect each of the three lights in the four cardinal directions. The agent can take six possible actions: it can move in each of the four cardinal directions, PRESS the lock, or PICKUP the key. The first four actions are stochastic; they move the agent in the intended direction with probability 0.8 and to either side with probability 0.1 each. The PRESS and PICKUP actions are only effective when the agent is on top of the lock and the key, respectively. The agent starts in a random state in the top left room in the domain, and can proceed through the rooms indefinitely. The states features and actions of the domain are listed in Table 1.

This domain is well-suited for this task because the domain has a rich feature space and complex dynamics. There

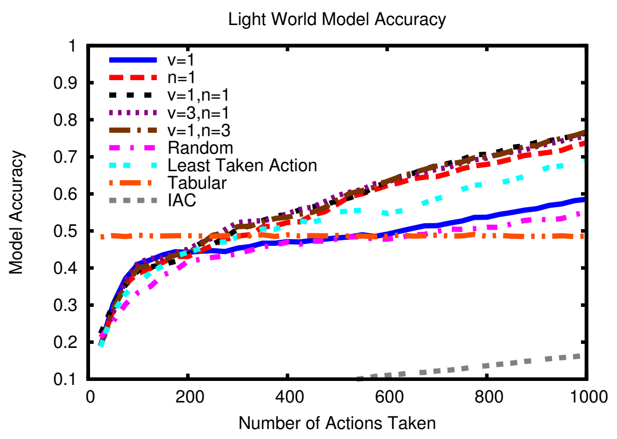


Figure 3: Model accuracy of each algorithm plotted versus number of steps the agent has taken, averaged over 30 trials. The versions of TEXPLORE-VANIR with $n >= 1$ learn the most accurate models.

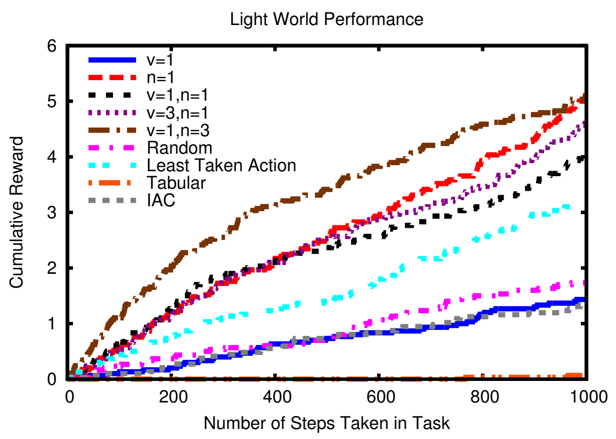


Figure 4: Cumulative rewards received by each algorithm over the 1000 steps of the task, averaged over 30 trials. TEXPLORE-VANIR with $v = 1, n = 3$ receives the most reward.

are simple actions that move the agent in different directions, as well as more complex actions (PICKUP and PRESS) that interact with objects in different ways. There is a progression of the complexity of the uses of these two actions. Picking up the key is easier than pressing the lock, as the lock requires the agent to have already picked up the key and not yet unlocked the door.

Five versions of the algorithm using the following sets of coefficients are compared:

1. v = 1, n = 0
2. v = 0, n = 1
3. v = 1, n = 1
4. v = 3, n = 1
5. v = 1, n = 3

In addition, TEXPLORE-VANIR is tested against the following agents:

1. Agent which selects actions *randomly*
2. Agent which selects the *least taken action* at each state
3. Agent which acts randomly with a *tabular* model
4. Intelligent Adaptive Curiosity (IAC) [17]

These four algorithms provide two different ways to explore using TEXPLORE-VANIR's tree model, as well as an algorithm using a tabular model, and an existing model for intrinsically motivated learning (IAC).

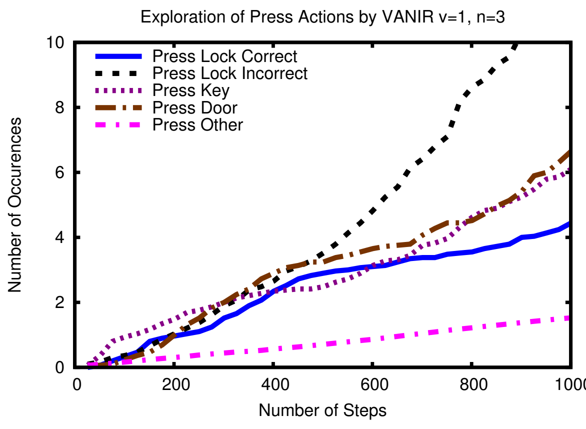
Intelligent Adaptive Curiosity (IAC) [17] is a method for providing intrinsic reward to encourage a developing agent to explore. Their approach does not adopt the RL framework, but is similar in many respects. IAC splits the state space into regions and attempts to learn a model of the transition dynamics in each region. It maintains an error curve for each region and uses the slope of this curve as the intrinsic reward for the agent, driving the agent to explore the areas where its model is improving the most. Since this approach is not using the RL framework, it selects actions only to maximize the immediate reward, rather than the discounted sum of future rewards. Another drawback of this approach is that before intrinsic rewards from any region can be calculated, the agent must take actions from that part of the domain a few times to build an error curve for its model of that region.

All the algorithms are run in the Light World domain for 1000 steps without any external reward. During this phase, the agent is free to play and explore in the domain, all the while learning a model of the dynamics of this world. For some of the experiments, a second phase of the experiment is run with external rewards to see if the agent's learned model is useful. All of the algorithms use the RTMBA parallel architecture [9] and take actions at 1 Hz.

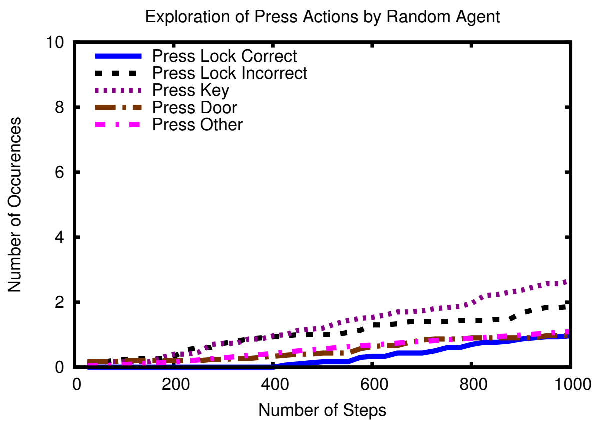
First, the accuracy of the agent's learned model is examined. After every 25 steps, 5000 state-actions from the domain are randomly sampled and the model's predicted probabilities of each next state are compared with the true next state probabilities. Figure 3 shows 1.0 minus the average error in the predicted probabilities of the model learned by each agent. This figure shows that the versions of TEXPLORE-VANIR with $n >= 1$ learn the most accurate models, with the version with $v = 1, n = 3$ having the highest accuracy. IAC, the TABULAR model, and the random agent have the poorest model accuracy.

While all the versions of TEXPLORE-VANIR with $n >= 1$ reach similar levels of model accuracy, it is more important for the algorithm to be accurate in the interesting and useful parts of the domain than for it to be accurate about every state-action. Therefore, we want to test if the learned models are useful to perform a task. After the algorithms learned models without rewards for 1000 steps, they are provided with a reward function for a task. The task is for the agent to continue moving through the rooms (requiring it to use the keys and locks). The reward function is a reward of 1 for moving from one room to the next, and a reward of 0 for all other actions. The agents are tested using their learned transition models, the given external reward function, and no intrinsic rewards for 1000 steps on the task.

Figure 4 shows the cumulative external reward received



(a) TEXPLORE-VANIR with $v = 1, n = 3$



(b) Random Agent.

Figure 5: This plot shows the number of times that TEXPLORE-VANIR with $v = 1, n = 3$ and a Random Agent select the press action in various states over 1000 steps in the task with no external rewards, averaged over 30 trials. Note that the random agent attempts the press action much less than TEXPLORE-VANIR does. TEXPLORE-VANIR starts out trying to press the key, which is the easiest object to find, and eventually does learn to press the lock, but has difficulty learning when to press the lock (it must be with the key but without the door already being open). The agent does not try calling the press action on random states very often. In contrast, the random agent calls press action on random states more often than it calls it correctly on the lock.

by each algorithm over the 1000 steps of the task. For comparison, an optimal policy would be expected to receive a cumulative reward of 60.2 over the 1000 steps. This figure shows more variation in the abilities of each agent to perform the task. Although all the versions of TEXPLORE-VANIR with $n >= 1$ have similar model accuracies, TEXPLORE-VANIR with $v = 1, n = 3$ has the greatest cumulative reward for most of the steps. At the end of the 1000 steps, however, TEXPLORE-VANIR with $v = 1, n = 3$ is equaled by TEXPLORE-VANIR with $n = 1$. In comparison, the versions exploring randomly or with $v = 1$ perform much worse than TEXPLORE-VANIR with $n >= 1$ and all the methods using tree models perform much better than the tabular model.

Next, the exploration of the TEXPLORE-VANIR agent with $v = 1, n = 3$ is examined. In addition to learning an accurate and useful model of the task, we desire the agent to exhibit a developing curiosity. Precisely, the agent should progressively learn more complex skills in the domain, rather than explore randomly or exhaustively. In this domain, the agent should first learn to pick up the key, then learn how to use the key at the lock, and then how to go through the door to the next room.

Figures 5(a) and 5(b) show the number of times that TEXPLORE-VANIR with $v = 1, n = 3$ and the random agent select the PRESS action in various states over 1000 steps in the task with no external rewards, averaged over 30 trials. Comparing the two figures clearly shows that TEXPLORE-VANIR calls the PRESS action many more times than the random agent. Figure 5(a) also shows that TEXPLORE-VANIR tries PRESS on objects more often than on random states in the domain. In contrast, Figure 5(b) shows that the random agent tries PRESS on arbitrary states in the domain more often than it uses it correctly.

Analyzing the exploration of TEXPLORE-VANIR further, Figure 5(a) shows that it initially tries PRESS on the key, which is the easiest object to access, then tries it on the lock, and then on the door (which is not visible until unlocked). The figure also shows that TEXPLORE-VANIR takes longer to learn the correct dynamics of the lock, as it continues to PRESS the lock incorrectly, either without the key or with the door already unlocked. These plots show that TEXPLORE-VANIR is acting in an intelligent, curious way, trying actions on the objects in order from the easiest to hardest to access, and going back to the lock repeatedly to learn its more complex dynamics.

Finally, not only should the agent's intrinsic rewards be useful when learning in task without external rewards, they should also make an agent in a domain with external rewards learn more efficiently. For this experiment, the algorithms are run for 1000 steps with their intrinsic rewards added to the previously used external reward function that rewards moving between rooms. Figure 6 shows the cumulative external reward received by each agent over the 1000 steps of the task. The versions of TEXPLORE-VANIR with both $n >= 1$ and $v >= 1$ receive the most reward, with TEXPLORE-VANIR with $v = 1, n = 3$ once again performing the best, receiving an average of 4.2 reward over the 1000 steps. In contrast, the agent using only external rewards (TEXPLORE-VANIR with $v = 0, n = 0$) receives only an average of 0.07 reward over the 1000 steps, and the agent using a tabular model receives 0 reward. These results demonstrate that TEXPLORE-VANIR's intrinsic rewards can also be used to speed up learning in tasks that already provide external rewards.

These results show that TEXPLORE-VANIR, in particular with $v = 1, n = 3$, out-performs all other algorithms on the four evaluations. It learns more accurate models than the other approaches when given no external reward and can use its learned model to perform better on a task than the other methods as well. It explores the domain in a curious manner progressing from state-actions with easier dynamics to those that are more difficult. Finally, TEXPLORE-VANIR

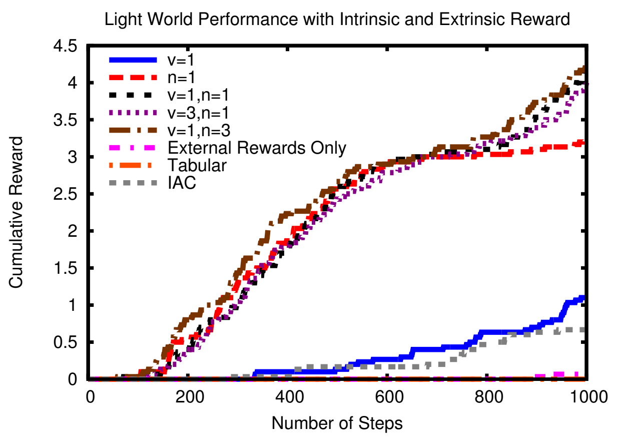


Figure 6: Cumulative rewards received by each algorithm, using intrinsic and external rewards combined, over the 1000 steps of the task, averaged over 30 trials. The versions of TEXPLORE-VANIR with both $n >= 1$ and $v >= 1$ receive the most reward, while the agent using only external rewards performs very poorly.

can use its intrinsic rewards to speed up learning in a case where it is given external rewards.

5. RELATED WORK

This section reviews related work on intrinsic motivation, with a focus on work which is built within the RL framework.

Schmidhuber [19] tries to drive the agent to where the model has been improving the most, rather than trying to estimate where the model is poorest. The author takes a traditional model-based RL method, and adds a confidence module, which is trained to predict the absolute value of the error of the model. This module could be used to create intrinsic rewards encouraging the agent to explore high-error state-action pairs, but then the agent would be attracted to noisy states in addition to poorly-modeled ones. Instead the author adds another module that is trained to predict the changes in the confidence module outputs. Using this module, the agent is driven to explore the parts of the state space that most improve the model's prediction error.

IAC [17] (described in the previous section) and later Robust-IAC [1] are based on a similar idea. They are both methods that encourage a developing agent to explore areas of the state space where the model is improving the most. These approaches do not use the RL framework and have no way of incorporating external rewards, but could be used to provide intrinsic rewards to an existing RL agent. Both of these last two algorithms require the agent to have sampled part of the state space before it can judge if intrinsic rewards should be provided in that region.

R-MAX [2] is a reinforcement learning method that explores efficiently through the use of intrinsic rewards. The algorithm learns a maximum-likelihood tabular model of the task. The model used by the algorithm provides the maximum reward in the domain, r_{max}, as an intrinsic reward to any state-actions that have been visited less than m times. This reward drives the agent to visit each state-action m times so that the agent can learn an accurate model of each state-action. The algorithm is guaranteed to learn an optimal policy in a number of time steps polynomial in the number of state-actions in the domain.

Jonsson and Barto [12] take a similar approach to TEXPLORE-VANIR, in that they also learn trees to model the domain. Their method learns conditional trees using Bayesian Information Criterion to perform splits. Since having a uniform distribution over input values will provide the best information for making splits in the tree, their method provides intrinsic motivation for actions that would increase the uniformity of the inputs to the tree. This reward only drives local exploration, but does enable the agent to quickly learn accurate models of certain tasks. This work was extended to perform more global exploration by adding options to set each state feature to any possible value [24]. The agent could then select options to set features to values where it could then take actions to better improve the uniformity of input features to its trees. However, this approach assumes that the agent can set each feature of the domain independently and learn options to do so.

Singh, Barto, and Chentanez [20] present an approach to learning a broad set of reusable skills in a playroom domain. They learn option models for a variety of skills and show that the agent progresses from learning easier to more difficult skills. However, the skills the agent is to learn are predefined, rather than being entirely intrinsically motivated.

Simsek and Barto [5] present an approach for the pure exploration problem, where there is no concern with receiving external rewards. They provide a Q-LEARNING agent [25] with intrinsic rewards for where its value function is most improving. This reward speeds up the agent's learning of the true task. However, such a reward to make the agent perform more value backups on its value function is not necessary for model-based agents, which can perform all the necessary backups using its model without having to re-visit each state-action. Stout and Barto [22] extend this work to the case where the agent is learning multiple tasks and must balance the intrinsic rewards that promote the learning of each skill. These algorithms still require an external reward, as the intrinsic reward is speeding up the learning of the task defined by the external reward function.

Singh et al. [21] present an interesting perspective on intrinsically motivated learning. They argue that in nature, intrinsic rewards come from evolution and exist to help us perform any task. Agents using intrinsic rewards combined with external rewards should be able to perform better on tasks than those using solely external rewards. For two different algorithms and tasks, they search over a broad set of possible task and agent specific intrinsic rewards and find rewards that make the agent learn faster than if it solely used external rewards.

6. DISCUSSION

This paper presents the TEXPLORE-VANIR algorithm for intrinsically motivated learning. This algorithm combines decision tree based model learning with two novel intrinsic rewards. One reward drives the agent to where the model is uncertain in its predictions, and the second drives the agent to acquire novel experiences that its model has not been trained on. Experiments show empirically that TEXPLORE-VANIR can learn accurate and useful models in a domain with no external rewards. In addition, TEXPLORE-VANIR's intrinsic rewards drive the agent to learn in a developing and curious way, progressing from learning easier to more difficult skills. TEXPLORE-VANIR can also combine its intrinsic rewards with external task rewards to learn a task faster than using external rewards alone.

One of the major advantages of TEXPLORE-VANIR in comparison to others is that it does not require the agent to sample each region of the state space first to determine if they are interesting. It can decide certain states are interesting because of the novelty of their state features without having to visit that region of the state space first. The agent takes an intelligent approach to exploration, building multiple hypotheses of the true dynamics of the domain and driving itself through the variance motivation to explore where the various hypotheses make different predictions. In addition, the agent continues seeking out novel states that may provide different dynamics from what it has seen before.

Our main interest in the pursuit of this research is to apply it to developmental learning on robots. Therefore, one area of future work is to extend the algorithm to work in large continuous state spaces, such that it can apply on a robot. Once the algorithm is extended in this way, we plan to perform experiments on humanoid robots such as the Aldebaran Nao.

7. REFERENCES

[1] A. Baranes and P. Y. Oudeyer. R-IAC: Robust Intrinsically Motivated Exploration and Active Learning. *IEEE Transactions on Autonomous Mental Development (TAMD)*, 1(3):155–169, Oct. 2009.

[2] R. Brafman and M. Tennenholtz. R-Max - a general polynomial time algorithm for near-optimal reinforcement learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 953–958, 2001.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] D. Chakraborty and P. Stone. Structure learning in ergodic factored MDPs without knowledge of the transition function's in-degree. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning (ICML)*, June 2011.

[5] O. Şimşek and A. G. Barto. An intrinsic reward mechanism for efficient exploration. In *ICML*, pages 833–840, 2006.

[6] T. Degris, O. Sigaud, and P.-H. Wuillemin. Learning the structure of factored Markov Decision Processes in reinforcement learning problems. In *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML)*, pages 257–264, 2006.

[7] M. Duff. Design for an optimal probe. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 131–138, 2003.

[8] C. Guestrin, R. Patrascu, and D. Schuurmans. Algorithm-directed exploration for model-based reinforcement learning in factored MDPs. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 235–242, 2002.

[9] T. Hester, M. Quinlan, and P. Stone. RTMBA: A real-time model-based reinforcement learning architecture for robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.

[10] T. Hester and P. Stone. Real time targeted exploration in large domains. In *Proceedings of the Ninth International Conference on Development and Learning (ICDL)*, August 2010.

[11] N. Jong and P. Stone. Model-based function approximation for reinforcement learning. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007.

[12] A. Jonsson and A. G. Barto. Active learning of dynamic bayesian networks in markov decision processes. In *SARA*, pages 273–284, 2007.

[13] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning (ECML)*, 2006.

[14] G. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 895–900, 2007.

[15] B. Leffler, M. Littman, and T. Edmunds. Efficient reinforcement learning with relocatable action models. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 572–577, 2007.

[16] M. Lopes and P.-Y. Oudeyer. Guest editorial: Active learning and intrinsically motivated exploration in robots: Advances and challenges. *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2(2):65–69, 2010.

[17] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evolutionary Computation*, 11(2):265–286, 2007.

[18] R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[19] J. Schmidhuber. Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991.

[20] S. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 17*, 2005.

[21] S. P. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2(2):70–82, 2010.

[22] A. Stout and A. Barto. Competence progress intrinsic motivation. In *Proceedings of the Ninth International Conference on Development and Learning (ICDL)*, pages 257–262, 2010.

[23] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[24] C. M. Vigorito and A. G. Barto. Intrinsically motivated hierarchical skill learning in structured environments. *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2(2), 2010.

[25] C. Watkins. *Learning From Delayed Rewards.* PhD thesis, University of Cambridge, 1989.