Artificial Intelligence ••• (••••) •••–•••

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Intrinsically motivated model learning for developing curious robots

Todd Hester *,[1], Peter Stone

Department of Computer Science, The University of Texas at Austin, United States

ARTICLE INFO

Article history:
Received in revised form 28 January 2015
Accepted 11 May 2015
Available online xxxx

Keywords:
Reinforcement learning
Exploration
Intrinsic motivation
Developmental learning
Robots

ABSTRACT

Reinforcement Learning (RL) agents are typically deployed to learn a specific, concrete task based on a pre-defined reward function. However, in some cases an agent may be able to gain experience in the domain prior to being given a task. In such cases, intrinsic motivation can be used to enable the agent to learn a useful model of the environment that is likely to help it learn its eventual tasks more efficiently. This paradigm fits robots particularly well, as they need to learn about their own dynamics and affordances which can be applied to many different tasks. This article 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 random forests. It calculates two different intrinsic motivations 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 article 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. We also present results demonstrating the applicability of this approach to learning on robots.

1. Introduction

Reinforcement Learning (RL) agents could be useful in society because of their ability to learn and adapt to new environments and tasks. Traditionally, RL agents learn to accomplish a specific, concrete task based on a pre-defined reward function. However, in some cases an agent may be able to gain experience in the domain prior to being given this task. Learning in this way is particularly useful for robots, as they must learn about their own dynamics and their environment before learning how to perform specific tasks. For example, a future domestic robot may be placed in a home and only later given various tasks to accomplish. In such cases, intrinsic motivation can be used to enable the agent to learn a useful model of its dynamics and the environment that can help it learn its eventual tasks more efficiently.

* Corresponding author.
E-mail address: todd.hester@gmail.com (T. Hester).
[1] Todd Hester is currently at Nest Labs in Palo Alto, CA.

http://dx.doi.org/10.1016/j.artint.2015.05.002
0004-3702/

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

Past work on intrinsically motivated agents arises from two different goals [1]. The first goal comes from the active learning community, which uses intrinsic motivation to improve the sample efficiency of RL. Their goal is to help the agent to maximize its knowledge about the world and its ability to control it. The second goal comes from the developmental learning community, and is to enable cumulative, open-ended learning on robots. Our goal is to use intrinsic motivation towards both goals: 1) to improve the sample efficiency of learning, particularly in tasks with little or no external rewards; and 2) to enable the agent to perform open-ended learning without external rewards.

This article presents an intrinsically motivated model-based RL algorithm, called TEXPLORE with Variance-And-Novelty-Intrinsic-Rewards (TEXPLORE-VANIR), that uses intrinsic motivation both for improved sample efficiency and to give the agent a curiosity drive. The agent is based on a model-based RL framework and is motivated to learn models of domains without external rewards as efficiently as possible. TEXPLORE-VANIR combines model learning through the use of random forests with two unique intrinsic rewards calculated from this model. 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. The combination of these two rewards enables the agent to explore in a developing curious way, learn progressively more complex skills, and learn a useful model of the domain very efficiently.

This article presents three main contributions:

1. Novel methods for obtaining intrinsic rewards from a random-forest-based model of the world.
2. The TEXPLORE-VANIR algorithm for intrinsically motivated model learning, which has been released open-source as an ROS package: http://www.ros.org/wiki/rl-texplore-ros-pkg.
3. Empirical evaluations of the algorithm, both in a simulated domain and on a physical robot.

Section 2 presents background on reinforcement learning and Markov Decision Processes. Section 3 presents work related to TEXPLORE-VANIR in the areas of reinforcement learning and intrinsic motivation. Section 4 presents the TEXPLORE-VANIR algorithm, including its approach to model learning and how its intrinsic rewards are calculated. Section 5 presents experiments showing that TEXPLORE-VANIR: 1) learns a model more efficiently than other methods; 2) explores in a developing, curious way; 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. Section 6 presents details on the code release of TEXPLORE-VANIR. Finally, Section 7 concludes the paper.

2. Background

This section presents background on Reinforcement Learning (RL). We adopt the standard Markov Decision Process (MDP) formalism for this work [2]. 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: $s = \langle x_1, x_2, \ldots, x_n\rangle$. A policy π specifies for each state which action the agent will take.

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 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 optimal value function $Q^*(s,a)$ provides maximal values in all states and is determined by solving the Bellman equation:

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

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

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

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.

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. TEXPLORE-VANIR extends a model-based RL algorithm called TEXPLORE [3] to use intrinsic motivation to quickly learn an accurate model in domains with no external rewards.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

3. Related work

Many model-based RL algorithms use "exploration bonus" intrinsic rewards to drive the agent to explore more efficiently. As one example, R-MAX [4] uses intrinsic rewards to guarantee that it will learn the optimal policy within a bounded number of steps. The algorithm learns a maximum-likelihood tabular model of the task and provides intrinsic rewards to state-actions that have been visited less than m times. These rewards drive the agent to visit each state-action enough times to learn an accurate model.

One of the more well-known intrinsic motivation algorithms is Robust Intelligent Adaptive Curiosity (R-IAC) [5]. R-IAC does not adopt the RL framework, but is similar in many respects. R-IAC splits the state space into regions and learns 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 (rewarding *competence progress*). This approach is intended for very large multi-dimensional continuous domains where learning may take many thousands of steps. An alternative to R-IAC is to learn a separate predictor of the change in model error and use its predicted values as the intrinsic reward to drive exploration [6]. Another similar algorithm is ZETA-R-MAX, which extends R-MAX [4] to classify states as known based on the empirical measure of progress in model learning and provides similar convergence guarantees [7].

Jonsson and Barto [8] 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 [9]. The agent selected 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 [10] 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 pre-defined, rather than being entirely intrinsically motivated.

Şimşek and Barto [11] present an approach for the pure exploration problem, where there is no concern with receiving external rewards. They provide a Q-LEARNING agent [12] 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 is not necessary for model-based agents, which perform value function updates by planning on their model. Stout and Barto [13] 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. This algorithm requires an external reward, as the intrinsic reward is speeding up the learning of the task defined by the external reward function.

Singh et al. [14] 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 perform better 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.

These different approaches demonstrate that the correct intrinsic motivation is dependent on the type of algorithm. For example, with a Q-LEARNING agent [12], it makes sense to give intrinsic rewards for where the value backups will have the largest effect, as done in [11]. 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 [4]. With a model learning approach that generalizes as TEXPLORE-VANIR's does, the best intrinsic rewards are different again.

The Policy Gradient Reward Design algorithm (PGRD) learns the best intrinsic rewards on-line for cases where the true reward function is given and the agent is *limited* in some way [15–17]. PGRD uses its knowledge of the true reward function to calculate the gradient of intrinsic rewards to agent return. Using this gradient, intrinsic rewards are found that enable the best agent performance given its limitations. For example, if the agent has a limited planning depth, then even with the true reward function, it cannot perform well. However, good intrinsic rewards can make up for this deficiency. This work does not apply to agents without limitations, as providing the agent with the reward function effectively solves the problem. Similarly, in tasks where the reward function is not given, then the gradient cannot be calculated and this method does not work.

Sequeira et al. [18,19] propose a set of emotion-based intrinsic rewards that present some similarities to TEXPLORE-VANIR. They present four intrinsic rewards: novelty, motivation, control, and valence. The novelty and control rewards are similar to the novelty and variance rewards in TEXPLORE-VANIR, however they are count-based with no generalization across the state-space.

Reward shaping algorithms are another set of approaches that use intrinsic rewards. These methods provide the agent with intrinsic rewards for improving performance rather than only providing the agent external rewards when the goal has been achieved. These shaping rewards are intended to improve the learning speed of the agent. Shaping rewards have been used to enable RL agents to learn to ride a bicycle [20] and speed up learning on gridworld tasks [21]. Typically, the shaping rewards are created heuristically by the user based on their knowledge of the domain [22]. These methods affect the agent's

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

exploration in the domain to speed up learning, but they typically require the user to have specific knowledge about what constitutes improvement in the task. In addition, if shaping rewards are used that are not *potential-based*, they can cause the agent to learn sub-optimal policies, diverging into cycles that receive lots of shaping reward without accruing any external reward [21].

4. TEXPLORE-VANIR

Our goal is to develop an intrinsically motivated curious agent using RL. This agent should use intrinsic rewards to 1) efficiently learn a useful model of the domain's transition dynamics; and 2) explore in a developing curious way. 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 so as to learn the model quickly.
3. It should not be required to visit every state-action, because doing so is intractable in large domains.

This article presents the TEXPLORE-VANIR algorithm [23], which has all of these properties. 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 robots, TEXPLORE-VANIR uses the Real-Time Model Based Architecture [24]. This architecture uses approximate planning with UCT [25] and parallelizes the model learning, planning, and acting such that the agent can take actions in real-time at a specified frequency.

4.1. 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 the agent does not have to visit every state-action. 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.

TEXPLORE-VANIR is built upon the TEXPLORE algorithm [3], which uses random forests to learn separate predictions of each of the n state features in the domain. In order to learn effectively, the algorithm must learn an accurate model of the domain quickly to learn the task with high sample efficiency. Although tabular models are a common approach, they require the agent to take every action from each state once (or multiple times in stochastic domains), since they learn a prediction for each state-action separately. Instead, TEXPLORE uses supervised learning techniques to *generalize* the effects of actions across states, as has been done by some previous algorithms [26,27]. Since the *relative* transition effects of actions are similar across states in many domains, TEXPLORE follows the approach of Leffler et al. [28] and Jong and Stone [27] in predicting relative transitions rather than absolute outcomes. In this way, model learning becomes a supervised learning problem with (s, a) as the input and $s' - s$ and r as the outputs to be predicted. Model learning is sped up by the ability of the supervised learner to make predictions for unseen or infrequently visited states.

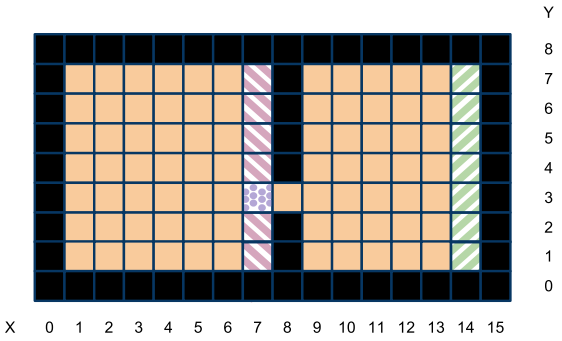
Like Dynamic Bayesian Network (DBN) based RL algorithms [29–31], the algorithm learns a model of the factored domain by learning a separate prediction for each of the n state features and the reward. The MDP model is made up of n models to predict each feature ($featModel_1$ to $featModel_n$) and a model to predict reward ($rewardModel$). Each model can be queried for a prediction for a particular state-action ($featModel \Rightarrow \text{QUERY}(\langle s, a \rangle)$) or updated with a new training experience ($featModel \Rightarrow \text{UPDATE}(\langle s, a, out \rangle)$). In TEXPLORE, each of these models is a random forest.

TEXPLORE's model learning algorithm starts by calculating the relative change in the state (s^{rel}), then it updates the model for each feature with the new transition and updates the reward model. Like DBN-based algorithms, TEXPLORE assumes that each of the state variables transitions independently. Therefore, the separate feature predictions can be combined to create a prediction of the complete state vector. The agent gets the prediction of the value of the change in each feature and adds this vector, s^{rel}, to s to get the prediction of s'.

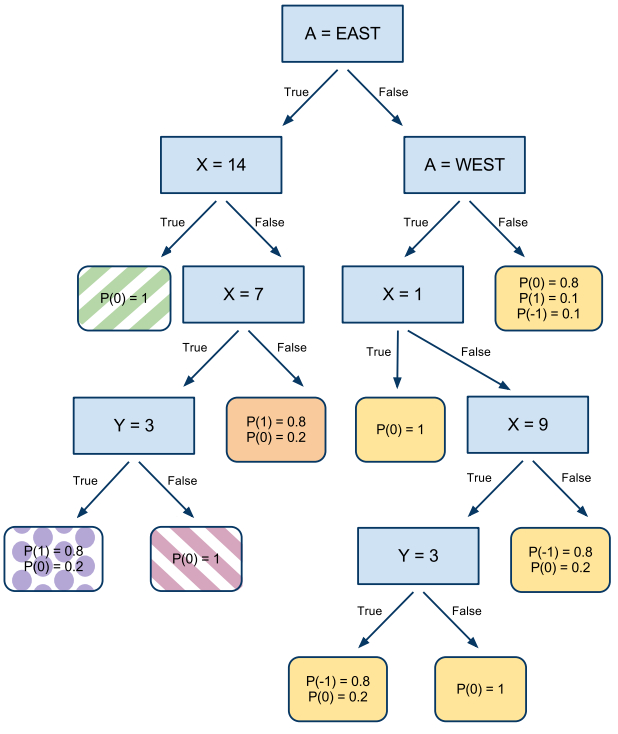
We tested the applicability of several different supervised learning methods to the task of learning an MDP model in previous work [32]. Decision trees, committees of trees, random forests, support vector machines, neural networks, nearest neighbor, and tabular models were compared on their ability to predict the transition and reward models across three toy domains after being given a random sample of experiences in the domain. Decision tree based models (single decision trees, committees of trees, and random forests) consistently provided the best results. Decision trees generalize broadly and can be refined to make accurate predictions at all states. Another reason decision trees perform well is that in many domains, the state space can be split into regions with similar dynamics. For example, on a vehicle, the dynamics can be split into different regions corresponding to which gear the car is in.

Based on these results, TEXPLORE uses decision trees to learn models of the transition and reward functions. The decision trees are learned using an implementation of Quinlan's C4.5 algorithm [33]. The inputs to the decision trees are treated both

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002



(a) Two room gridworld domain.



(b) Decision tree model predicting the change in the x feature (Δx) based on the current state and action.

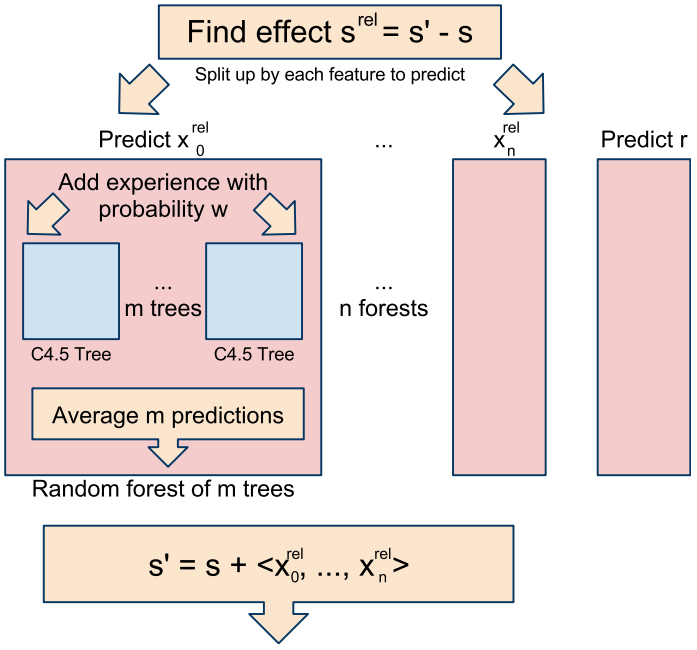
Fig. 1. This figure shows the decision tree model learned to predict the change in the x feature (or Δx). The two room gridworld is shaded to match the corresponding leaves of the left side of the tree where the agent has taken the EAST action. Each rectangle represents a split in the tree and each rounded rectangle represents a leaf of the tree, showing the probabilities of a given value for Δx. For example, if the action is EAST and $x = 14$, the agent is hitting the right wall. This input falls into the leaf on the top left, where the probability of $\Delta x = 0$ is 1.

as numerical and categorical inputs, meaning both splits of the type **if** $\mathrm{x} = 3$ and **if** $\mathrm{x} > 3$ are allowed. The C4.5 algorithm chooses the split at each node of the tree based on information gain. TEXPLORE's implementation includes a modification to make the algorithm incremental. Each tree is updated incrementally by checking at each node whether the new experience changes the optimal split in the tree. If it does, the tree is re-built from that node down.

The decision trees are the supervised learner that is called to predict each feature and reward. Each tree makes predictions for the particular feature or reward it is given based on a vector containing the n features of the state s along with the action a: $\langle s_1, s_2, \ldots, s_n, a \rangle$. This same vector is used when querying the trees for the change in each feature and for reward.

Fig. 1 shows an example decision tree predicting the relative change in the x variable of the agent in the given gridworld domain. The decision tree can split on both the actions and the state of the agent, allowing it to split the state space up into regions where the transition dynamics are the same. Each leaf of the tree can make probabilistic predictions based on the ratio of experienced outcomes in that leaf. The grid is shaded to match the leaves on the left side of the tree, making predictions for when the agent takes the EAST action. The tree is built on-line while the agent is acting in the MDP. At the start, the tree will be empty, and then it will generalize broadly, making predictions about large parts of the state space, such as what the EAST or WEST actions do. For unvisited state-actions, the tree will predict that the outcome is the same

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002



as that of similar state-actions (ones in the same leaf of the tree). It will continue to refine itself until it has leaves for individual states where the transition dynamics differ from the global dynamics.

Using decision trees to learn the model of the MDP provides TEXPLORE with a model that can be learned quickly with few samples. However, each tree represents just one possible hypothesis of the true model of the domain, which may be generalized incorrectly. Rather than planning with respect to this single model, our algorithm plans over a distribution of possible tree models (in the form of a random forest) to drive exploration. A random forest is a collection of 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. Random forests have been proven to converge with less generalization error than individual tree models [34].

Each of the m decision trees in the forest is trained on only a subset of the agent's *experiences* ($\langle s, a, s', r \rangle$ tuples), as it is updated with each new experience with probability w. To increase stochasticity in the models, at each split in the tree, the best input is chosen from a random subset of the inputs.

TEXPLORE plans greedily with respect to a distribution of m model hypotheses. TEXPLORE's action-values are then:

$$Q(s,a) = \frac{1}{m}\sum_{i=1}^{m} R_i(s,a) + \gamma \frac{1}{m}\sum_{i=1}^{m}\sum_{s'} P_i(s'|s,a)\max_{a'} Q(s',a'). \tag{3}$$

Each decision tree in the random forest generalizes transitions differently, resulting in different hypotheses of the true MDP. As each tree model's predictions differ more, the predictions from the aggregate model become more stochastic. For example, if each of five trees predict a different next state, then the aggregate model will have a uniform distribution over these five possible next states. The aggregate model includes some probability of transitioning to the states and rewards predicted by the optimistic models as well as those predicted by the pessimistic ones. Thus, planning on the aggregate model makes the agent balance the likelihood that the transitions predicted by the optimistic and pessimistic model will occur. The agent will explore towards state-actions that some models predict to have higher values while avoiding those that are predicted to have low values.

Another benefit of planning on this aggregate model is that it enables TEXPLORE to explore multiple possible generalizations of the domain, as it can explore state-actions that are promising in any one of the hypotheses in the aggregate model. In contrast, if TEXPLORE acted using a single hypothesis of the task model, then it would not know about state-actions that are only promising in other possible generalizations of its past experience. Fig. 2 shows a diagram of how the entire model learning system works. The variance of the different trees' predictions can be used as a measure of the uncertainty in the model.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

4.2. Intrinsic motivation

The main contribution of this article is a method for extending the model-based TEXPLORE algorithm for learning specific RL tasks to the intrinsically motivated TEXPLORE-VANIR algorithm. The best intrinsic rewards to use to improve the efficiency of model-learning are highly dependent on the type of model being learned. With the random forest models TEXPLORE uses, we hypothesize that the following two intrinsic motivations will perform the best: 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. TEXPLORE-VANIR calculates a measure of the variance in the predictions of each state feature for a given state-action:

$$D(s,a) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} D_{KL}(P_j(x_i|s,a)||P_k(x_i|s,a)), \tag{4}$$

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. $D(s,a)$ measures 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, the VARIANCE-REWARD, is incorporated into the agent's model for planning:

$$R(s,a) = vD(s,a), \tag{5}$$

where v is a coefficient determining how big this reward should be.

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 dynamics. However, there will still be cases where all of the agent's models make incorrect predictions. For the random forest model that TEXPLORE-VANIR uses, the model is more likely to be incorrect when it has to generalize its predictions farther from the experiences it is trained on. Therefore, TEXPLORE-VANIR uses 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. This distance is calculated separately for each action. For an action a, X_a is the set of all the states where this action was taken. Then, $\delta(s,a)$ is the L_1 distance from the given state s to the nearest state where action a has been taken:

$$\delta(s,a) = \min_{s_x \in X_a} ||s - s_x||_1, \tag{6}$$

where each feature is normalized to range from 0 to 1. For boolean features, the distance between true and false is assumed to be 1. A reward proportional to this distance, the NOVELTY-REWARD, drives the agent to explore the state-actions that are the most novel compared to the previously seen state-actions:

$$R(s,a) = n\delta(s,a), \tag{7}$$

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 agent will take the actions with the highest long-term rewards, causing it to trade-off between exploring faraway states with large intrinsic rewards and closer states with smaller intrinsic rewards. As the agent explores the states with the highest intrinsic rewards, other features of the domain will become more rewarding and the agent will move to explore them.

The TEXPLORE with Variance-And-Novelty-Intrinsic-Rewards algorithm (TEXPLORE-VANIR) is completed by combining these two intrinsic rewards. They can be combined with different weightings of their coefficients (v and n), or with an external reward defining a task. Pseudo-code for TEXPLORE-VANIR's model learning algorithm is shown in Algorithm 1. A combination of the two intrinsic rewards should drive the agent to learn a model more efficiently, as well as explore in a developing and curious way: seeking out novel and interesting state-actions, while exploring increasingly complex parts of the domain.

5. Empirical results

We empirically evaluate the TEXPLORE-VANIR algorithm on both a simulated domain called LIGHT WORLD and on a physical robot. In the simulated domain, we are able to perform detailed experiments and comparisons with other state-of-the-art methods. The experiments on the robot demonstrate the algorithms ability to work on a physical robot. Intrinsically motivated learning fits robots well as they must learn their own dynamics and environment, which are then applicable to any task the robot may want to perform.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

Algorithm 1 MODEL.

procedure INIT-MODEL(n) ▷ n is the number of state variables
  for $i = 1 \rightarrow n$ **do**
    $featModel_i \Rightarrow$ INIT() ▷ Init random forest to predict feature i
  end for
  for $i = 1 \rightarrow nactions$ **do**
    $X_i \leftarrow \emptyset$ ▷ Init visited state set for action i
  end for
end procedure

procedure UPDATE-MODEL($list$) ▷ Update model with *list* of experiences
  for all $\langle s, a, s' \rangle \in list$ **do**
    $s^{rel} \leftarrow s' - s$ ▷ Calculate relative effect
    for all $s_i^{rel} \in s^{rel}$ **do**
      $featModel_i \Rightarrow$ UPDATE($\langle s, a \rangle$, s_i^{rel}) ▷ Train feature model
    end for
    $X_a \leftarrow X_a \cup s$ ▷ Add s to visited set for action a
  end for
end procedure

procedure QUERY-MODEL(s, a, v, n) ▷ Get prediction of $\langle s', r \rangle$ for s, a
  for $i = 1 \rightarrow$ LENGTH(s) **do**
    $s_i^{rel} \leftarrow featModel_i \Rightarrow$ QUERY($\langle s, a \rangle$) ▷ Sample prediction for feat i
  end for
  $s' \leftarrow s + \left\langle s_1^{rel}, \ldots, s_n^{rel} \right\rangle$ ▷ Get absolute next state
  $D(s, a) \leftarrow \sum_{i=1}^{n} featModel_i \Rightarrow$ VARIANCE($\langle s, a \rangle$) ▷ Calculate variance.
  ▷ Each forest returns $\sum_{j=1}^{m} \sum_{k=1}^{m} D_{KL}(P_j(x_i|s,a)||P_k(x_i|s,a))$
  $\delta(s, a) \leftarrow \min_{s_x \in X_a} ||s - s_x||_1$ ▷ Calculate model novelty
  $r_{var} \leftarrow v D(s, a)$ ▷ Calculate variance reward
  $r_{nov} \leftarrow n\delta(s, a)$ ▷ Calculate novelty reward
  $r \leftarrow r_{var} + r_{nov}$
  return $\langle s', r \rangle$ ▷ Return sampled next state and reward
end procedure

5.1. Light world

Evaluating the benefits of intrinsic motivation is not as straightforward as evaluating a standard RL agent on a specific task. Rather than attempting to accrue reward on a given 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 efficiently learn a transition model that is useful for performing tasks in the domain.

The agent is tested on the LIGHT WORLD domain [35], shown in Fig. 3. 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 pick up 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 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 color 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 0.9 and to either side with probability 0.05 each. The PRESS and PICKUP actions are only effective when the agent is on top of the lock and the key, respectively, and then only with probability 0.9. The agent starts in a random state in the top left room in the domain, and can proceed through the rooms indefinitely.

This domain is well-suited for this task because the domain has a rich feature space and complex dynamics. There are simple actions that move the agent, as well as more complex actions (PICKUP and PRESS) that interact with objects in

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

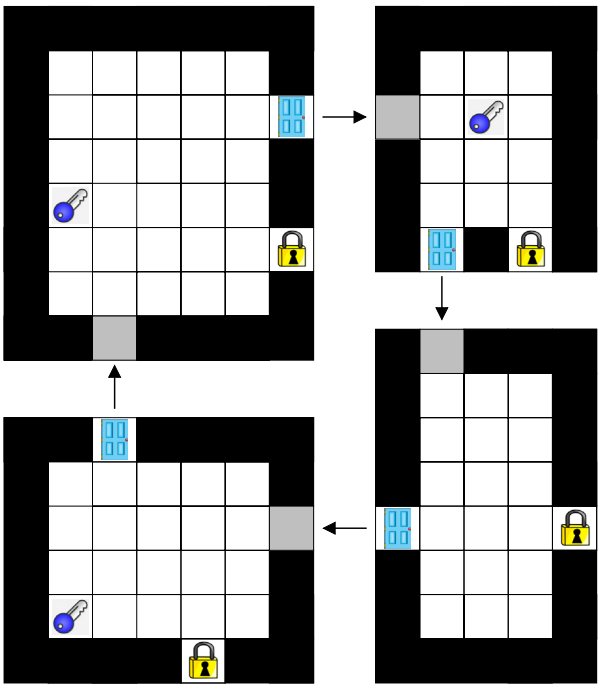


Fig. 3. The LIGHT WORLD domain. In each room, the agent must navigate to the key, PICKUP the key, navigate to the lock, PRESS it, and then navigate to and exit through the door to the next room.

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.

Based on informal testing, we set TEXPLORE-VANIR's parameters to $v = 1$ and $n = 3$. TEXPLORE-VANIR is tested against the following agents:

1. Agent which selects actions *randomly*
2. Agent which is given an intrinsic motivation for regions with more *competence progress* (based on R-IAC [5])
3. Agent which is given an intrinsic motivation for regions with more *prediction errors*
4. Agent which uses R-MAX style rewards (terminal reward of R_{max} for state-actions with fewer than m visits)
5. Agent which acts randomly with a *tabular* model
6. R-MAX algorithm [4]

These six algorithms provide four different ways to explore using TEXPLORE-VANIR's random forest model, as well two approaches using a tabular model. The tabular model is initialized to predict self-transitions for state-actions that have not been visited.

We have created a method based on R-IAC to compare with our approach (the *Competence Progress* method). We chose to compare against R-IAC (presented in Section 3) because it is one of the most well-known intrinsic motivation algorithms and maps to reinforcement learning well. This method splits the state space into random regions at the start, maintains error curves in each region, and provides intrinsic rewards based on competence progress within a region. These intrinsic rewards are combined with the same TEXPLORE model learning approach as the other methods. As another comparison, the *Prediction Error* method uses the same regions, but rewards areas with high prediction error.

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. All of the algorithms use the RTMBA parallel architecture [24] and take 2.5 actions per second.

First, we examine the accuracy of the agent's learned model. After every 25 steps, 5000 state-actions from the domain are randomly sampled and the variational distance between the model's predicted next state probabilities are compared with the true next state probabilities. Fig. 4 shows the variational distance between these distributions, averaged over the 5000 sampled state-actions. This figure shows that TEXPLORE-VANIR learns significantly more accurate models than the other methods ($p < 0.025$). The next best algorithm is R-MAX. However, using R-MAX style reward with the TEXPLORE model strategy is worse than acting randomly. This result illustrates our point that the best intrinsic reward is dependent on the particular model learning approach that is used. The method rewarding visiting regions with high prediction error performs poorly, possibly because it is not visiting the right state-actions within these regions.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

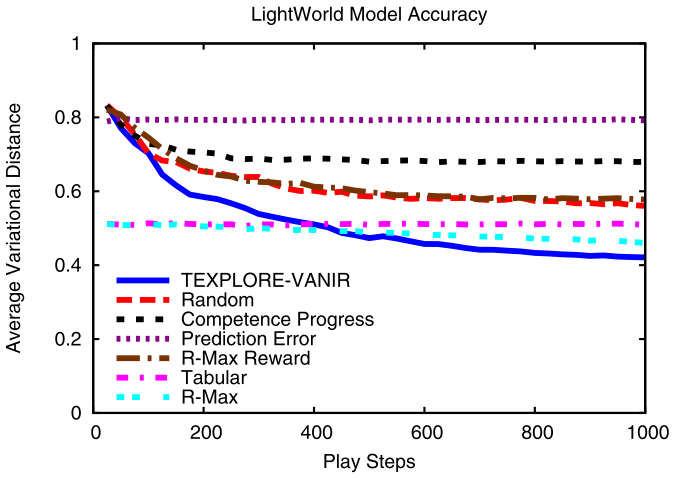


Fig. 4. Accuracy of each algorithm's model plotted versus number of steps the agent has taken, averaged over 30 trials and 5000 randomly sampled state-actions. TEXPLORE-VANIR learns the most accurate models.

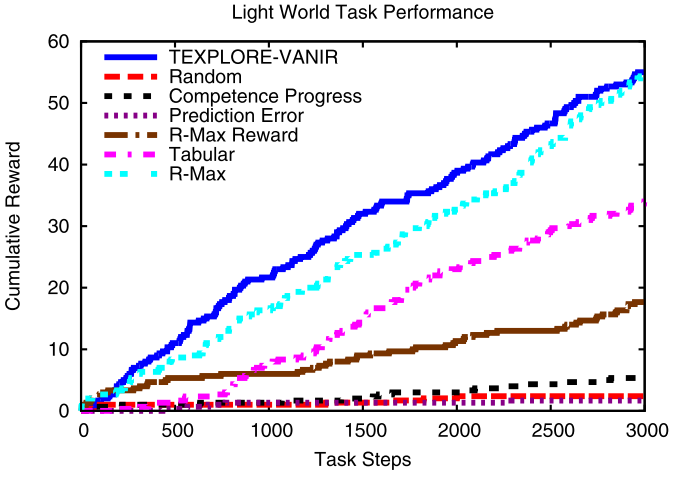


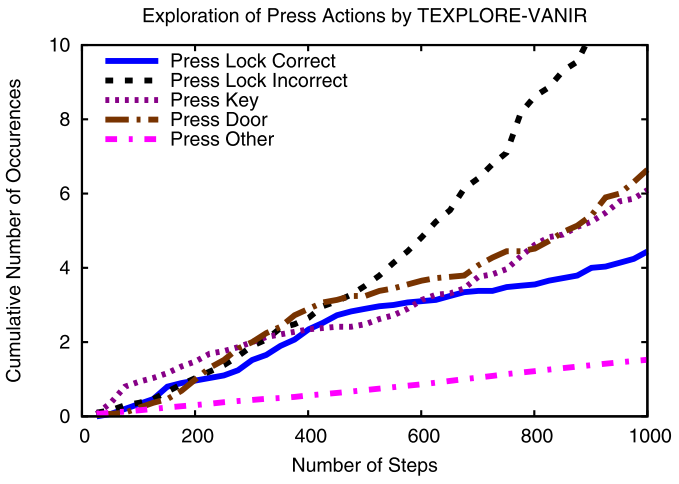
Fig. 5. Cumulative rewards received by each algorithm over the 3000 steps of the task, averaged over 30 trials. Agents act greedily with respect to their previously learned transition model and the given external reward function. TEXPLORE-VANIR receives the most reward.

While TEXPLORE-VANIR and R-MAX appear to learn fairly accurate models, it is more important for the algorithms to be accurate in the interesting and useful parts of the domain than for them to be accurate about every state-action. Therefore, we next 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 10 for moving from one room to the next, and a reward of 0 for all other actions. In this second phase, the agents act greedily with respect to their previously learned transition models and the given external reward function with *no* intrinsic rewards for 3000 steps.

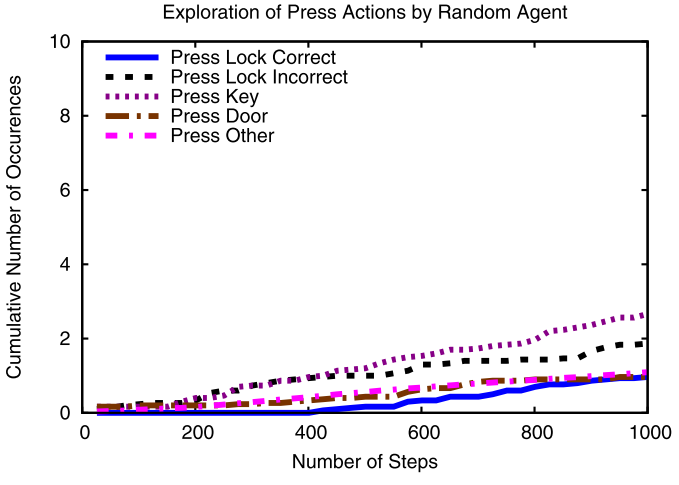
Fig. 5 shows the cumulative external reward received by each algorithm over the 3000 steps of the task. Again, TEXPLORE-VANIR performs the best, slightly out-performing R-MAX and significantly out-performing the other methods ($p < 0.001$). Learning an accurate transition model appears to lead to good performance on the task, as both TEXPLORE-VANIR and R-MAX perform well on the task. For comparison, an optimal policy (with full a priori knowledge of the transition and reward functions) would be expected to receive a cumulative reward of 1800.6 over the 3000 steps. Thus, the performance of all of these learning agents is still far below optimal performance in this domain.

Next, the exploration of the TEXPLORE-VANIR agent is examined. In addition to learning an accurate and useful model, 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. Figs. 6(a) and 6(b) show the cumulative number of times that TEXPLORE-VANIR 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 shows that TEXPLORE-VANIR calls the PRESS action many more times than the random agent. Fig. 6(a) also shows that TEXPLORE-VANIR tries PRESS on objects more often than on random

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002



(a) TEXPLORE-VANIR



(b) Random Agent.

Fig. 6. This plot shows the cumulative number of times that TEXPLORE-VANIR 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.

states in the domain. In contrast, Fig. 6(b) shows that the random agent tries PRESS on arbitrary states more often than it uses it correctly.

Analyzing the exploration of TEXPLORE-VANIR further, Fig. 6(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. 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. The agent moves from easier to harder tasks as it learns due to its variance reward. For particularly difficult state transitions, the agent's model may make an incorrect prediction in all of its forest trees, and as some of the trees start to learn the transition, the variance increases and the agent moves to explore the true dynamics of how it works.

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 3000 steps with their intrinsic rewards added to the previously used external reward function that rewards moving between rooms. Instead of an agent acting randomly, we instead have one agent acting using only the external rewards, and one performing Boltzmann, or soft-max, exploration with temperature $\tau = 0.2$. Fig. 7 shows the cumulative external reward received by each agent over the 3000 steps of the task. TEXPLORE-VANIR receives significantly more reward than the other

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

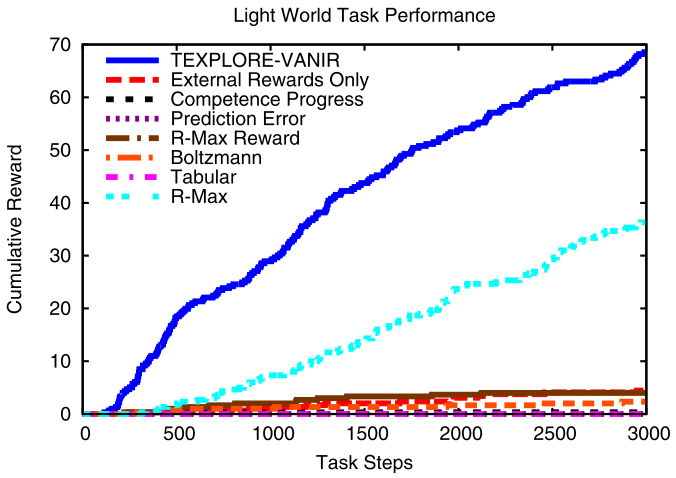


Fig. 7. Cumulative rewards received by each algorithm, using intrinsic and external rewards combined, over the 3000 steps of the task, averaged over 30 trials. TEXPLORE-VANIR receives the most reward, while the agent using only external rewards performs very poorly.

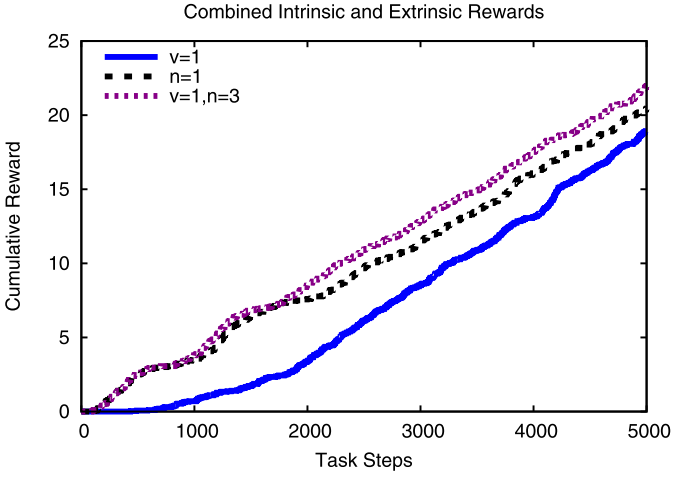


Fig. 8. Cumulative rewards received by each algorithm, using intrinsic and external rewards combined, over the 5000 steps of the task, averaged over 30 trials. TEXPLORE-VANIR with $v = 1$, $n = 3$ receives the more reward than versions using either intrinsic reward alone.

algorithms ($p < 0.001$), followed by R-MAX. Now that exploration and exploitation are no longer separated into separate phases, the exploration of R-MAX is too aggressive and costs it external reward.

Next, we compare versions of TEXPLORE-VANIR using one or both of its intrinsic rewards combined with the external rewards. Fig. 8 shows the cumulative reward accrued by each algorithm over the 3000 steps. Here, the version of TEXPLORE-VANIR using both variance and novelty intrinsic rewards performs better than the versions using just one of the intrinsic rewards. Having novelty rewards is important at the start to drive the agent to expand its knowledge about unseen parts of the world, and early on adding variance rewards to this has little effect. However, after the agent gets back to the first room again, it becomes useful to take actions to disambiguate its various models' hypotheses of how the objects work, and then it becomes important to have variance rewards as well.

These results show that TEXPLORE-VANIR's intrinsic rewards out-perform other exploration approaches and intrinsic motivations combined with the TEXPLORE model. TEXPLORE-VANIR performs similarly to R-MAX when exploration and exploitation are split into separate phases, but out-performs R-MAX significantly when combining intrinsic and external rewards together. TEXPLORE-VANIR explores the domain in a curious manner progressing from state-actions with easier dynamics to those that are more difficult. Finally, in a task with external rewards, TEXPLORE-VANIR can use its intrinsic rewards to speed up learning with respect to an algorithm using only external rewards.

It is important to note that the best intrinsic rewards are dependent on the learning algorithm and the domain. For example, the competence progress rewards used by R-IAC are intended to be used in complex high-dimensional domains where learning is slow. It takes quite a few samples in one region to get a reasonable estimate of the derivative of the error. In the LIGHT WORLD domain, by the time the algorithm has determined error is improving in a region, the agent has already learned a model of that region and no longer needs to explore there. When using other model learning methods, the best

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

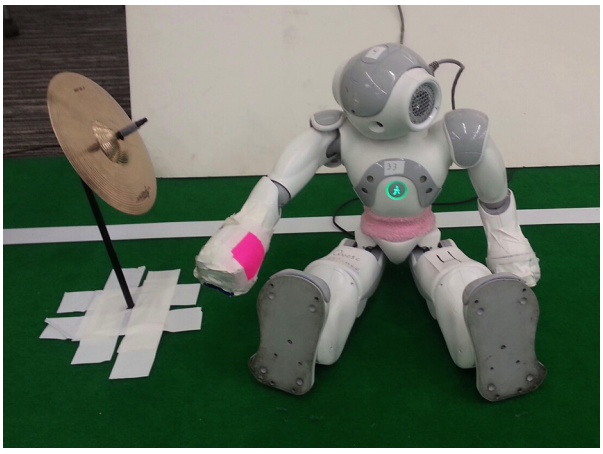


Fig. 9. The experimental setup with an Aldebaran Nao robot. The robot is placed in a sitting position, with a cymbal within arm's reach, and a square of pink taped to its hand. The learning agent can move either of the robot's right shoulder joints and can potentially hit the cymbal, push the button on its right foot, or move more or less pink into the view of its camera.

intrinsic reward will vary as well, for example, the R-MAX reward works well for a tabular model, but not for a random forest model.

In this particular domain, the agent is drawn to explore the areas of the statespace that are most different and complicated, meaning that it explores the corners of the rooms, the dynamics of the objects, and the different boolean values of having or not having the key, or locking or unlocking the door. While we demonstrated the agent learns to use the objects to get through the rooms, this exploration enables it to perform simpler tasks as well, such as simply navigating across the room. Depending on the task to later be performed, the bias given by the VANIR exploration towards a particular definition of "interesting" domain dynamics could hinder learning. For example, if the eventual goal is only to learn to navigate around one room, the time spent learning about the key and the lock are wasted. However, the agent still learns to navigate the room as well, even as it learns how to use the objects.

5.2. Robot learning

Next, we evaluate the ability of TEXPLORE-VANIR to learn to control the arm of a humanoid robot. Due to the difficulty of performing experiments on a real robot, instead of the detailed evaluations we performed on the LIGHT WORLD task, we only examine two aspects of the learning agent on the robot. First, we look at the amount of the state space that the agent was able to explore. Second, we test whether the learned model can be used to perform tasks in the domain when given a reward function. 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 efficiently learn a transition model that is useful for performing tasks in the domain.

The agent is tested controlling the arm of an Aldebaran Nao robot, shown in Fig. 9. A video of the robot learning is available at http://www.cs.utexas.edu/~AustinVilla/?p=research/vanir. The agent has control of the robot's two right shoulder joints. It can take one of five actions: Increase the angle of either joint by 8 degrees, decrease the angle of either joint by 8 degrees, or do nothing. The agent's state is made up of 8 state features: the angle of both shoulder joints, the 3-dimensional location of the robot's hand in mm relative to its chest, how many pink pixels the robot can see in its camera image, whether its right foot button is pressed, and the amount of energy it hears on its microphone. The robot is placed in a sitting position where it can reach its right foot button, and has a cymbal within reach to its right. It also has a pink square attached to its arm which it can move in front of its camera. The robot takes actions at 3 Hz to give the arm time to reach its new position after each action.

This domain was created as an example of an open-ended robotic task where there are many rich dynamics that the robot can learn about. In addition to learning about how the position of its hand changes as it moves its shoulder joint, it can learn how its foot button works, how the cymbal makes noise, and how to move the pink square on its arm in front of its camera.

Based on informal testing, we set TEXPLORE-VANIR's parameters to $v = 3$ and $n = 10$. We tested TEXPLORE-VANIR against performing random exploration. The robot acted following one of these two exploration strategies for 400 steps, with no external rewards. 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. Then we tested whether the model the robot learned could be used to perform tasks in the domain. We provided the agent with the reward function for 3 different tasks: maximizing the amount of pink it could see in its camera, pushing the button on its right foot, and maximizing the energy it received on its microphone.

For the viewing pink task, the reward was equal to the number of pink pixels the robot saw in its camera image. The robot has 153,600 pixels in its camera image, so if every pixel was pink, it would receive a reward of +153,600. However,

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

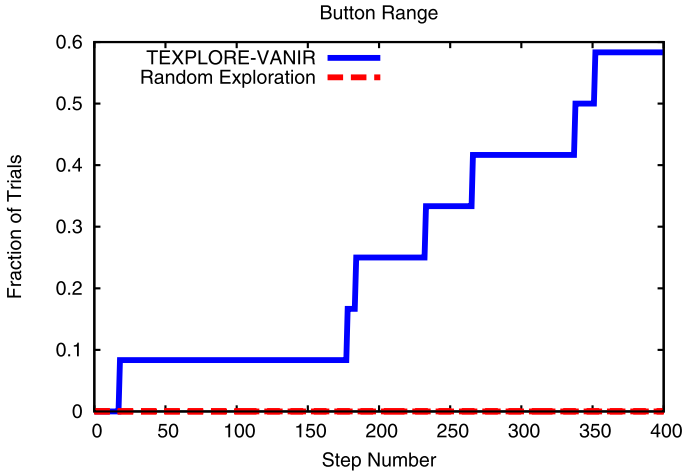


Fig. 10. The fraction of the 13 trials where the robot pushed its right foot button over the 400 exploration steps.

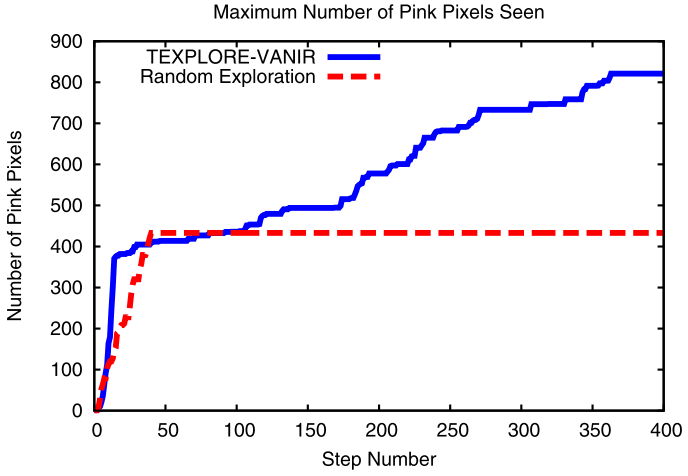


Fig. 11. The maximum number of pink pixels seen in the robot's camera image, averaged over 13 trials.

it is not possible for the robot to bring the pink square close enough to its camera to take up the entire field of view. In addition, as the pink square gets closer to the robot's head it gets shadowed and is no longer classified as pink. The highest number of pink pixels seen in our experiments was 5434. For the button pushing task, the reward was +200.0 when the robot's foot button was pushed, and 0 otherwise. For the microphone task, the reward was equal to the power of the signal received on the microphone. We capped this value to a maximum of 4000 as that was the range of values typically seen in the room during our experiments.

First, we can look at how much the agent explored during the 400 step exploration phase. Fig. 10 shows the fraction of trials that had discovered how to push the robot's right foot button over the 400 steps. By the end of the 400 steps, the TEXPLORE-VANIR agent had pushed the foot button on 7 of the 13 trials. In contrast, when doing random exploration, none of the trials pushed the foot button. Similarly, Fig. 11 shows the maximum number of pink pixels the robot had seen over the 400 steps, averaged over the trials. The TEXPLORE-VANIR agent brings the robot's hand much closer to its face and sees nearly twice as many pink pixels in its camera image.

While it appears that TEXPLORE-VANIR explores more of the statespace and discovers more of the complexities of the domain than random exploration, it is important for the agent to be able to perform actual tasks in the domain. After the algorithms learned models without rewards for 400 steps, they are provided with a reward function for a task. We tested 3 different tasks. The first task was to maximize the number of pink pixels seen in the robot's camera image. The second task was to push the robot's right foot button. The third task was to maximize the energy sensed on the robot's microphone. Each task was run for 200 steps.

Fig. 12 shows the average external reward received by both exploration strategies over the 200 steps of the task to maximize pink in the camera image, after the 400 exploration steps. The agent that explored using TEXPLORE-VANIR performs better on this task, receiving more average reward on every time step.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

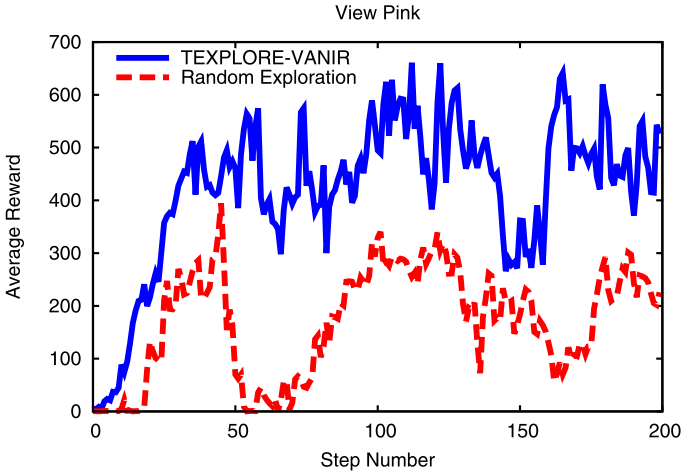


Fig. 12. Average rewards received by the agents over the 200 steps of the task to maximize the number of pink pixels seen in the robot's camera image, averaged over 13 trials. Agents act greedily with respect to their previously learned transition model and the given external reward function. TEXPLORE-VANIR receives the most reward.

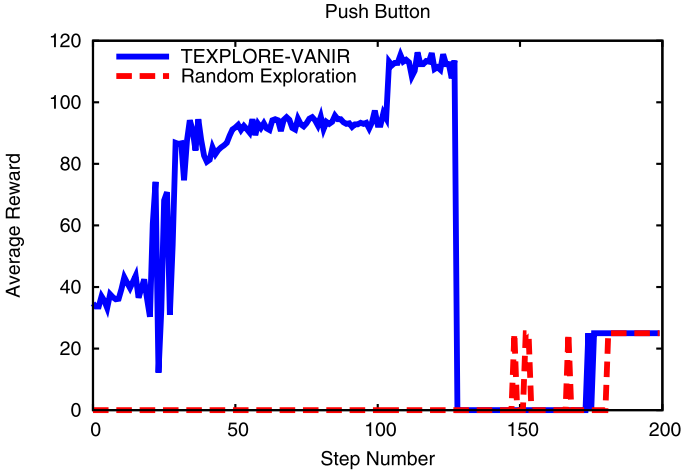


Fig. 13. Average rewards received by the agents over the 200 steps of the task to push the robot's right foot button, averaged over 13 trials. Agents act greedily with respect to their previously learned transition model and the given external reward function. TEXPLORE-VANIR receives the most reward.

Fig. 13 shows the average external reward received by both agents on the task of pushing the robot's foot button. Again, the agent that explored with TEXPLORE-VANIR performs better than the agent that did random exploration. However, the algorithm does have some difficulty keeping the button pressed for the entire 400 steps, eventually moving its hand off the button in all the trials.

The final task that we evaluated was for the agent to maximize the energy sensed on the robot's microphone. The idea was for the robot to learn how to make noise by hitting the cymbal with its arm, although it could also make quieter noises by hitting its arm into itself. This task proved to be more difficult than the other two tasks. For the agent that explored randomly, the agent never hit the cymbal during the task phase. For the agent that explored using TEXPLORE-VANIR, the agent learned to hit the cymbal in just one trial out of 5. Fig. 14 shows the reward received by the TEXPLORE-VANIR agent on this one trial. Rewards over approximately 800 represent noise made by the robot hitting the cymbal with its arm.

These results show that TEXPLORE-VANIR's intrinsic rewards drive the agent to explore more of the state space and discover more affordances in the task than if the agent explored randomly. Exploring with TEXPLORE-VANIR enables the robot to learn a more accurate model of more of the state space and perform better on possible tasks given to the robot after exploring the domain.

6. Code release

The TEXPLORE-VANIR algorithm is available as an open source robot operating system (ROS) [36] package at: http://www.ros.org/wiki/rl-texplore-ros-pkg. ROS is a popular robot middleware package that provides a framework for many nodes to communicate with each other through defined messages. With the code released as an ROS package, TEXPLORE can be easily

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

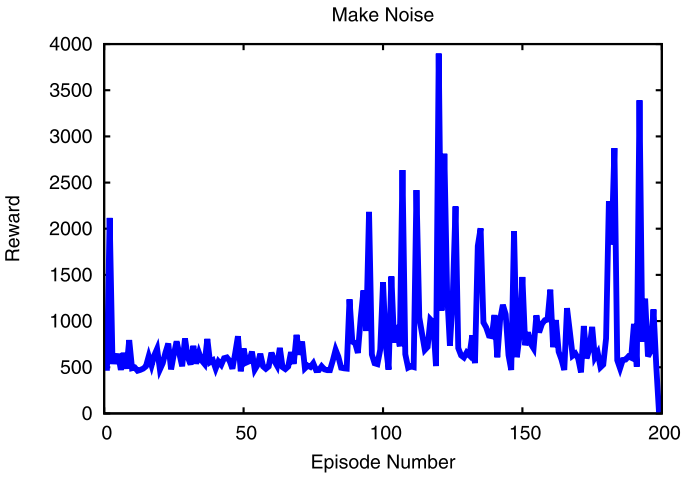


Fig. 14. Rewards received by the one successful trial of TEXPLORE-VANIR on the task of maximizing the energy received by the robot's microphone. Rewards greater than 800 represent times that the robot hit the cymbal with its arm.

downloaded and applied to a learning task on any robot running ROS with minimal effort. The goal of this algorithm and code release is to encourage more researchers to perform learning on robots using state-of-the-art algorithms.

The code release enables users to run the TEXPLORE-VANIR algorithm as well as the LIGHT WORLD domain. The TEXPLORE-VANIR domain can be run with different exploration parameters for both the variance and novelty exploration rewards.

7. Conclusion

This article presents the TEXPLORE-VANIR algorithm for intrinsically motivated learning. This algorithm combines random forest 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.

This article presents three main contributions:

1. Novel methods for obtaining intrinsic rewards from a random-forest-based model of the world.
2. The TEXPLORE-VANIR algorithm for intrinsically motivated model learning, which has been released open-source as an ROS package: http://www.ros.org/wiki/rl-texplore-ros-pkg.
3. Empirical evaluations of the algorithm, both in a simulated domain and on a physical robot.

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 use cases of this type of learning is learning on robots. Robots must learn their own dynamics and environment affordances, which are then applicable to tasks the robot may want to perform in its environment. We demonstrated TEXPLORE-VANIR learning to move the arm of an Aldebaran Nao humanoid robot. First, the agent learned solely from intrinsic motivation, exploring the state space, and then we showed that it was able to use its learned model to perform various tasks in the domain.

Our main interest in the pursuit of this research is to apply it to developmental learning on robots. One area where TEXPLORE-VANIR's performance on robots can be improved is handling continuous actions. While TEXPLORE-VANIR selects from a set of discrete actions, robots typically take a vector of continuous commands. For example, controlling the Aldebaran Nao robot requires a continuous vector of either desired velocities or positions for each of the robot's 25 joints. TEXPLORE-VANIR's tree models should already be able to handle multi-dimensional continuous actions as input in making predictions about the next state and reward. Thus, extending TEXPLORE-VANIR to use multi-dimensional continuous actions mainly requires extensions to the UCT planning algorithm for sampling and selecting from a multi-dimensional continuous action space. One possible approach to this problem is to utilize recent work [37,38] adapting the HOO algorithm for continuous bandit problems [39] to action selection at each level of the UCT tree.

The TEXPLORE-VANIR algorithm represents an important step towards fully autonomous developmental learning on robots. The TEXPLORE-VANIR algorithm could be utilized on robots in multiple ways. It can be used without an external reward on robots to create curious robots that learn and develop. It can also be combined with external rewards to speed up learning of tasks. By releasing TEXPLORE-VANIR as an open-source ROS package, we hope many other researchers are able to apply and extend it on their robots to improve learning.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002

References

[1] M. Lopes, P.-Y. Oudeyer, Guest editorial: active learning and intrinsically motivated exploration in robots: advances and challenges, IEEE Trans. Auton. Ment. Dev. (TAMD) 2 (2010) 65–69.
[2] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[3] T. Hester, P. Stone, Real time targeted exploration in large domains, in: Proceedings of the Ninth International Conference on Development and Learning, ICDL, 2010.
[4] R. Brafman, 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, 2001, pp. 953–958.
[5] A. Baranes, P.Y. Oudeyer, R-IAC: robust intrinsically motivated exploration and active learning, IEEE Trans. Auton. Ment. Dev. (TAMD) 1 (2009) 155–169.
[6] J. Schmidhuber, Curious model-building control systems, in: Proceedings of the International Joint Conference on Neural Networks, IEEE, 1991, pp. 1458–1463.
[7] M. Lopes, T. Lang, M. Toussaint, P.-Y. Oudeyer, Exploration in model-based reinforcement learning by empirically estimating learning progress, in: Neural Information Processing Systems, NIPS, Tahoe, USA, 2012.
[8] A. Jonsson, A.G. Barto, Active learning of dynamic Bayesian networks in Markov decision processes, in: SARA, 2007, pp. 273–284.
[9] C.M. Vigorito, A.G. Barto, Intrinsically motivated hierarchical skill learning in structured environments, IEEE Trans. Auton. Ment. Dev. (TAMD) 2 (2010).
[10] S. Singh, A.G. Barto, N. Chentanez, Intrinsically motivated reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 17, NIPS, 2005.
[11] O. Şimşek, A.G. Barto, An intrinsic reward mechanism for efficient exploration, in: ICML, 2006, pp. 833–840.
[12] C. Watkins, Learning from delayed rewards, Ph.D. thesis, University of Cambridge, 1989.
[13] A. Stout, A. Barto, Competence progress intrinsic motivation, in: Proceedings of the Ninth International Conference on Development and Learning, ICDL, 2010, pp. 257–262.
[14] S.P. Singh, R.L. Lewis, A.G. Barto, J. Sorg, Intrinsically motivated reinforcement learning: an evolutionary perspective, IEEE Trans. Auton. Ment. Dev. (TAMD) 2 (2010) 70–82.
[15] J. Sorg, S.P. Singh, R.L. Lewis, Internal rewards mitigate agent boundedness, in: J. Fürnkranz, T. Joachims (Eds.), ICML, Omnipress, 2010, pp. 1007–1014.
[16] J. Sorg, S.P. Singh, R.L. Lewis, Optimal rewards versus leaf-evaluation heuristics in planning agents, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2011.
[17] J. Bratman, S.P. Singh, J. Sorg, R.L. Lewis, Strong mitigation: nesting search for good policies within search for good reward, in: Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2012, pp. 407–414.
[18] P. Sequeira, Socio-emotional reward design for intrinsically motivated learning agents, Ph.D. thesis, Universidad Técnica de Lisboa, 2013.
[19] P. Sequeira, F.S. Melo, A. Paiva, Emotion-based intrinsic motivation for reinforcement learning agents, in: S.K. D'Mello, A.C. Graesser, B. Schuller, J.-C. Martin (Eds.), ACII (1), in: Lecture Notes in Computer Science, vol. 6974, Springer, 2011, pp. 326–336.
[20] J. Randløv, P. Alstrøm, Learning to drive a bicycle using reinforcement learning and shaping, in: Proceedings of the Fifteenth International Conference on Machine Learning, ICML, 1998.
[21] A. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: theory and application to reward shaping, 1999.
[22] M.G. Sam Devlin, D. Kudenko, An empirical study of potential-based reward shaping and advice in complex, multi-agent systems, Adv. Complex Syst. 14 (2011) 251–278.
[23] T. Hester, P. Stone, Intrinsically motivated model learning for a developing curious agent, in: Proceedings of the Eleventh International Conference on Development and Learning, ICDL, 2012.
[24] T. Hester, M. Quinlan, P. Stone, RTMBA: a real-time model-based reinforcement learning architecture for robot control, in: Proceedings of the 2012 IEEE International Conference on Robotics and Automation, ICRA, 2012.
[25] L. Kocsis, C. Szepesvári, Bandit based Monte-Carlo planning, in: Proceedings of the Seventeenth European Conference on Machine Learning, ECML, 2006.
[26] T. Degris, O. Sigaud, 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, 2006, pp. 257–264.
[27] N. Jong, P. Stone, Model-based function approximation for reinforcement learning, in: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2007.
[28] B. Leffler, M. Littman, T. Edmunds, Efficient reinforcement learning with relocatable action models, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007, pp. 572–577.
[29] C. Guestrin, R. Patrascu, D. Schuurmans, Algorithm-directed exploration for model-based reinforcement learning in factored MDPs, in: Proceedings of the Nineteenth International Conference on Machine Learning, ICML, 2002, pp. 235–242.
[30] A. Strehl, C. Diuk, M. Littman, Efficient structure learning in factored-state MDPs, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007, pp. 645–650.
[31] D. Chakraborty, 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, 2011.
[32] T. Hester, P. Stone, An empirical comparison of abstraction in models of Markov Decision Processes, in: Proceedings of the ICML/UAI/COLT Workshop on Abstraction in Reinforcement Learning, 2009.
[33] R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[34] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[35] G. Konidaris, A.G. Barto, Building portable options: skill transfer in reinforcement learning, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, IJCAI, 2007, pp. 895–900.
[36] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.
[37] C.R. Mansley, A. Weinstein, M.L. Littman, Sample-based planning for continuous action Markov decision processes, in: ICAPS, 2011.
[38] A. Weinstein, M.L. Littman, Bandit-based planning and learning in continuous-action Markov decision processes, in: Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS, 2012.
[39] S. Bubeck, R. Munos, G. Stoltz, Pure exploration in finitely-armed and continuous-armed bandits, in: Algorithmic Learning Theory (ALT 2009), Theor. Comput. Sci. 412 (2011) 1832–1852.

Please cite this article in press as: T. Hester, P. Stone, Intrinsically motivated model learning for developing curious robots, Artificial Intelligence (2015), http://dx.doi.org/10.1016/j.artint.2015.05.002