Artificial Curiosity with Planning for Autonomous Perceptual and Cognitive Development

Matthew Luciw, Vincent Graziano, Mark Ring, and Jürgen Schmidhuber
IDSIA / University of Lugano / SUPSI / 6928 Manno-Lugano, Switzerland
Email: {matthew, vincent, mark, juergen}@idsia.ch

***Abstract*—Autonomous agents that learn from reward on high-dimensional visual observations must learn to simplify the raw observations in both space (i.e., dimensionality reduction) and time (i.e., prediction), so that reinforcement learning becomes tractable and effective. Training the spatial and temporal models requires an appropriate sampling scheme, which cannot be hard-coded if the algorithm is to be general. Intrinsic rewards are associated with samples that best improve the agent's model of the world. Yet the dynamic nature of an intrinsic reward signal presents a major obstacle to successfully realizing an efficient curiosity-drive. TD-based incremental reinforcement learning approaches fail to adapt quickly enough to effectively exploit the curiosity signal. In this paper, a novel artificial curiosity system with planning is implemented, based on developmental or continual learning principles. Least-squares policy iteration is used with an agent's internal forward model, to efficiently assign values for maximizing combined external and intrinsic reward. The properties of this system are illustrated in a high-dimensional, noisy, visual environment that requires the agent to explore. With no useful external value information early on, the self-generated intrinsic values lead to actions that improve both its spatial (perceptual) and temporal (cognitive) models. Curiosity also leads it to learn how it could act to maximize external reward.**

I. Introduction

This paper is concerned with the holistic problem faced by an autonomous reinforcement learning (RL) agent, in which it is not only a problem to find externally valuable places (where external drive, such as hunger, can be satisfied), but also to arrange experience leading to valuable observations, which improve its internal model of the world.

A core challenge in extending RL to real-world developmental agents lies in uncovering how an agent might select actions to autonomously build an effective *perceptual mapping* — a simplifier of its sensory experience — through its interactions with the environment. This mapping is necessary since RL is generally intractable on an agent's raw input. Developmental agents with hand-coded non-adaptive perceptual maps are limited, and suited to specific environments only. Instead, an adaptive system must build a perceptual layer using observations particular to its environment. We propose that a major purpose of exploration for an agent is to learn *where* it can find and *how to reliably get to* certain sensory inputs, which are expected to *improve* its perception of the world. It remains an open challenge to show, in a practical computational sense, how autonomous agents can shape their own interactions with the environment to best improve their perceptual ability, while at the same time learning the value-based information for RL.

In the context of classical RL, the problem of designing or learning what is analogous to a perceptual mapping — a value function approximation — has been well-studied [21], [15], [10], but mainly in the context of continuous but low-dimensional state spaces. The issues related to handling an agent's raw, real-world, high-dimensional observations have not been addressed until recently. There are still few works where unsupervised learning (UL) methods have been combined with RL [2], [9], [4]. These recent approaches deal with the UL *separately* from the RL, by alternating the following two steps: (1) improving the UL layer on a set of observations, modifying its encoding of observations, and (2) developing the RL on top of the encoded information. It is assumed that the *implicit* feedback inherent in the process leads to both useable code and an optimal policy. But there is no reason in general to expect such a result.

A major issue with the integration of UL and RL for *autonomous* agents is exactly how to deal with the mutual dependence of these layers, which poses a bootstrapping problem. For the UL to produce a useful code, the agent's policy needs to provide it with informative observations, while the RL must use the current code as a basis for the policy to find these observations. Exploration methods based on randomness cannot address this problem, since they are are not scalable to larger environments. Indeed, exploration itself is a skill that must be learned, as the solution to how best explore depends on both the environment and the learning capabilities of the agent.

The system in this paper is based on Schmidhuber and collaborators' comprehensive theory of artificial curiosity [18], developed since the early 1990's. Our goal is to utilize a curiosity signal to both (1) shape the sampling of the environment leading to perceptual layer on which tabular RL methods are tractable, and to (2) make exploration efficient so that optimal, or negligibly sub-optimal policies are quickly discovered.

This paper explores an application of curiosity to developmental agents for the purpose of actively learning effective perceptual (encoding raw observations) and cognitive (prediction) systems from vision-based observations. Neto [22] combined a growing Kohonen-type network to build a model of observations, and used error from the mature network as curiosity signal to detect novel visual data, but the curiosity

was not used for actually building the model itself. We will show the advantage of models built through artificial curiosity, with respect to how well they enable an agent to potentially achieve external rewards, over models built through random exploration or externally-focused exploration. Further, we examine the critical role that planning capabilities [17], [3], [5] play, due to the non-stationary intrinsic reward signal, in curious agents.

The rest of this paper is organized as follows. We discuss the context of this work with respect to the field (focused on computer science) in Section II. Our system and methods are presented in Section III. The algorithm is presented as modules in Algorithms 1, 2, 3, and 4. Experiments and results are shown in Section IV. Section V concludes the paper.

II. Background

The classical RL framework has an agent interacting with an environment. With each observation the agent takes an action to produce the next observation, and the agent also receives a reward signal. The agent's goal is to act so that it maximizes the reward it receives during its lifetime. *Real-world* RL situates an agent, which manifests itself as a robot or android, in situations that are proto-typical for humans, an requires the handling of raw (high-dimensional, noisy) observations in a continuous environment.

A. Developmental Systems

We use the term, *developmental system*, to describe a system that can accomplish the goals of real-world RL both autonomously and without prior knowledge of the environment. That is, the system is expected to adapt successfully, *ab ovo*, to any environment in which it finds itself. This process involves a continual building of increasingly complicated skills on top of already developed skills, with no final stage of development in mind. This open-ended, incremental development of a reinforcement-learning agent was first specifically addressed by Ring [14], who termed it *continual learning*. It is a very general RL problem that invokes another significant challenge of artificial intelligence research—how does an agent go about *exploring* its environment so that its knowledge of the environment is sufficient to solve the RL task?

Here is considered a (memory-less) developmental agent's mapping from observation to action in two layers: a *perceptual map*, and a *value map*. At each step, the agent receives a sensory observation from a discrete high-dimensional space and maps it to an internal representation. The perceptual activation is then mapped, via a value function, to a distribution over the agent's actions. An action is taken by sampling from the distribution.

B. Methods for Exploration

In small low-dimensional environments, where there are both few states and actions, guided exploration is not needed. Sampling of the important parts of these small environments is easily achieved by the most basic of exploration methods– random selection of actions (e.g., ϵ-greedy).

In tabular settings, where there is *a priori* knowledge of the environment, fancier methods are available. For example, by keeping track of visits to state-action pairs their associated values can be learned as a function that includes a factor which is inversely proportional to the number of times the pair has occurred. Since the extent of the environment (i.e., how many states) is known beforehand, this method can drive the agent to places where it has relatively little experience. A method that suits certain non-stationary environments is Dyna-Q+ [20], which assigns interest to state-actions pairs in proportion to the time elapsed since last taken.

Real-world settings require different exploration methods since the agent does not have pre-wired, distinct knowledge of the *states* it will encounter. One way to get the agent to explore in a systematic and non-uniform way is by broadening the reward signal so that the agent is drawn to things it finds interesting. The reward signal is modified to contain two distinct components: an *intrinsic* or internal component and an extrinsic or external component. The external component is the reward signal in the classical RL sense, while the intrinsic component is a reward signal that is based on some measure of *interestingness* and is used as a motivational system to speed learning.

The intuitive notion of interestingness [16] can be summarized as the 'potential for the discovery of novel patterns.' A sound quantitative measure of interestingness must assign low values to patterns the observer already knows, and patterns the observer cannot learn; and high values to patterns the observer does not yet know, but can still discover. A developmental system equipped with a subjective notion of interestingness, a measure based on the *internal state of the agent*, can be used to drive the agent's exploration of the environment in an informed, systematic fashion when used as an intrinsic reward signal.

In this paper, we use the framework of artificial curiosity [18] to generate the intrinsic reward signal. Artificial curiosity is based on *compression progress*, a measure of interestingness. Compression progress relies on a learning compressor, e.g., a recurrent neural network, to measure how the length of the compressed history of observations changes when a better compressor of the history is learned. Artificial curiosity then is the drive to go to the parts of the environment where the most compression progress is expected to be made. Implicitly, implementing artificial curiosity requires a module to predict (which implies learning to predict) how much compression progress will be made for each of its actions with respect to current state of the agent. We shall refer to this as the *cognitive map* and discuss it further in Section III.

C. Critical Issues

As discussed by Oudeyer et al. [13] integrating an intrinsic motivation system with a developmental system requires some care. Here we overview some of the critical issues that arise.

1) Suitable Sensory Representations: In real-world settings, where we turn to developmental systems, the 'states' or perceptual layer must be learned. RL takes the form of learning

value-functions on the learned internal representation of the world. It is possible to learn the perceptual map completely based on value-based feedback from the RL component. There are many works [12], [7] that develop perceptual systems in this way. However, there are good reasons to utilize an unsupervised learning approach. The value-based information can be sparse, hidden, or unreliable and lessening the reliance on information such as TD-error can often speed the development of the perceptual layer.

That said, if the two layers are learned independently, without the sharing of error signals, there is no way to guarantee that the perceptual map is suitable for the value map. For example, even in a Markovian environment the perceptual system could very well learn a representation that produces a non-Markovian problem for the RL algorithm. We leave this issue for another study, choosing in this paper a method for learning a perceptual map that is well-suited to the test-environment.

2) RL and Intrinsic Reward: Unlike the external reward the intrinsic reward is inherently non-stationary. This causes problem with classical RL methods that assume a stationary, noise-free reward signal. Consider here, without loss of generality, an environment with no external reward. Given the internal state of the agent it is possible to formalize the goal of maximizing intrinsic reward as a reinforcement learning problem. The policy that results from such a calculation is only (approximately) correct for a single action, after which, since additional knowledge of the environment has been gained, it most-likely changes. Further, many learning methods, e.g., incremental TD-learning, can also produce behaviors that fail to maximize future expected discounted intrinsic reward.

Another potentially serious over-simplification results when the RL task is reduced to maximizing the expected intrinsic reward *at the next time step*, rather than using the expected future discounted reward [19]. Though seemingly innocuous, this simplification can undermine the intrinsic reward signal. For example, in any given state the agent will always prefer the actions that are interesting to those which are uninteresting, even in cases where all the interesting actions always lead to uninteresting places (the so-called *'57-ways-to-end-it-all'* phenomenon).

III. Artificial Curiosity for Autonomous Developmental Agents

The developmental agent here is composed of three parts [23]: (1) the sensory or *perceptual* map, for encoding raw observations, (2) the *cognitive* map, used to predict subsequent states in the encoded space, and (3) the *value* system, which assigns values to the immediately available actions. The values are derived from both the external and the intrinsic rewards, allowing the agent to collect external reward while gaining experiences that will improve its perceptual and cognitive systems. A graphical depiction of the overall architecture is shown in Figure 1, and is described in the caption.

A. Perceptual System

The observation space $\mathcal{O}$ is huge, it has many elements and is high-dimensional. Each observation is mapped to an internal, self-generated state $s \in \mathcal{S}$. The number of internal states n is kept small to make prediction and value learning practical. To each state s there is an associated prototype vector $p_s \in \mathbb{R}^M$, where M is the dimension of the observation space.

Growing Collection of Internal States: This internal state collection $\mathcal{S}$ is built and modified in an on-line fashion with a method based on adaptive vector quantization (VQ). A new state is generated when no existing state prototype is similar enough to the current observation [1]. When the agent receives an observation $\mathbf{o}'$ the closest prototype $\mathbf{p}_s$ is identified (using the function $dist$). If $err^{per} = dist(\mathbf{o}', \mathbf{p}_s)$ is greater than a fixed value κ a new state s' is adjoined to the set $\mathcal{S}$ with the prototype vector $\mathbf{p}_{s'} \equiv \mathbf{o}'$ associated to it. The internal state is then set to s'.

Prototype Adaptation: If $\mathbf{o}'$ is within κ of $\mathbf{p}_s$ then s is set as the current internal state, and the point $\mathbf{p}_s$ is moved towards $\mathbf{o}'$ along the line connecting the points in a Hebbian-like update:

$$\mathbf{p}_s \leftarrow (1 - \eta_s^{per})\mathbf{p}_s + \eta_s^{per}\mathbf{o}', \tag{1}$$

where η^{per} is a dynamic learning rate (based on Lobe Component Analysis [24]). Each internal state s has an associated age h_s, which is simply the number of times that the state was set as the current state. To achieve a balance between memory and plasticity the learning rate η_s^{per} is given as a function of the age h_s, e.g., $\eta_s^{per} = 1/h_s$. As a result units with less experience can be made to adapt more quickly than those with more. To maintain lifetime plasticity the learning rate is kept away from 0 (e.g., if $1/h_s \leq \omega > 0$ then $\eta_s^{per} = \omega$).

Adding states allows the system to adapt to its environment as the number of states does not have to be determined beforehand. Some difficulties with on-line training are mitigated, namely when an observation arrives that no existing prototype is similar enough to, a new state is simply created, instead of potentially corrupting an existing state.

B. Cognitive System

The cognitive system updates transition probabilities between the internal states (stored within $\mathbf{C}^{state}$). It also updates predictions of immediate external and intrinsic rewards. With respect to the transitions, it is desired that

$$c_{s,a,s'}^{state} \approx \Pr\{s(t+1) = s' | s(t) = s, a(t) = a\}.$$

The transition probabilities from state s under action a can be represented by a vector $\mathbf{c}_{s,a}^{state}$ where the i-th entry in the vector represents the probability of transitioning to the i-th internal state, given some index on the internal states. These probabilities for all next states can be updated at once, in Hebbian form, similar to Equation 1:

$$\mathbf{c}_{s,a}^{state} \leftarrow (1 - \eta_{s,a}^{cog})\mathbf{c}_{s,a}^{state} + \eta_{s,a}^{cog}\mathbf{y}', \tag{2}$$

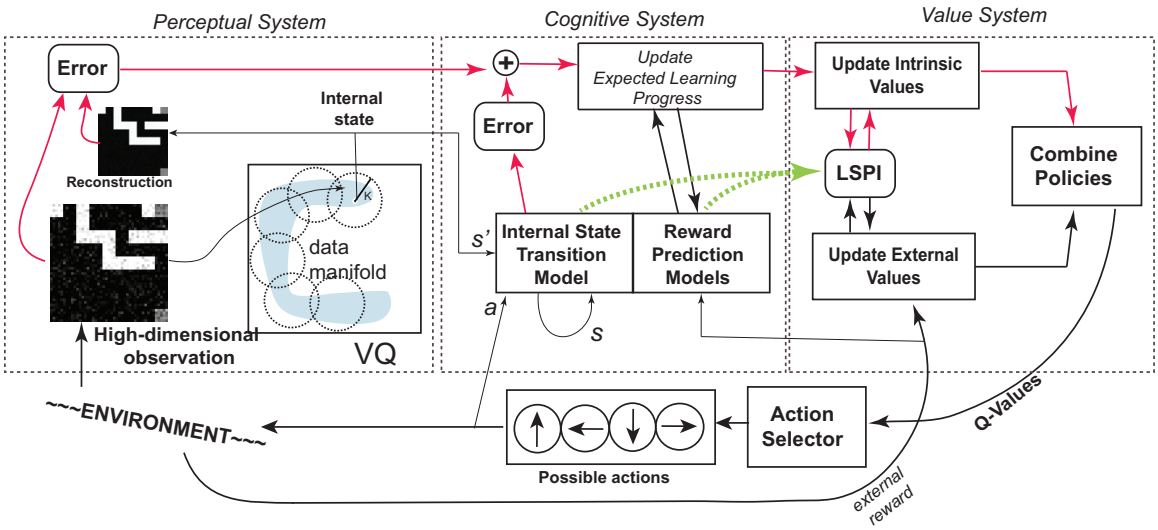


Fig. 1. System Architecture. The agent's decision making is limited to a value-maximizing forward pass from observation to action, therefore this system can only handle Markovian environments. Not only must it learn its values, but also self-generated (internal) states and a predictive model of internal state transitions and expected rewards. There are three parts to the system. The perceptual system controls state generation and adaptation of internal states to better fit the observations. It is improved by reducing observation reconstruction error. The cognitive system controls the adaptation of a model of expected next state and expected rewards. It is improved by reducing prediction error. The value system assigns expected future discounted rewards to internal state-actions so the agent can take actions that lead to better actual rewards in the future. The values are based on external and intrinsic rewards. The intrinsic rewards (expected learning progress) are high for a state-action if the observed recent errors (reconstruction and predictive) associated with the next (outcome) state are higher than normal (expected) error. These intrinsic rewards are not stable, unlike traditional external rewards, since the perceptual and cognitive systems reflect the accuracy of the model. LSPI is used to compensate for the quickly changing values, since it is able to adapt the values of all the state-actions without having to re-visit them. The intrinsic values in a practical sense allow the agent to select actions that direct its sampling so that its perceptual and cognitive systems can make quick improvement.

where $\mathbf{y}'$ is a vector whose entries are identically 0 except for at the entry corresponding to the current state s' where it is 1. The learning rate $\eta^{cog}_{s,a}$ is similar to η^{per}_{s}, but using the state-action "age".

The cognitive error, $err^{cog} = dist(\mathbf{c}^{state}_{s,a}, \mathbf{y}')$, is currently based only on the state model (though it would be straightforward to extend to both state and reward).

Expected external reward expectations are kept in $\mathbf{C}^{ext}$. This is updated as an expectation of actual rewards received after (s, a), so that $c^{ext}_{s,a} \approx \mathrm{E}\{r^{ext} | s(t) = s, a(t) = a\}$.

Expected Intrinsic Reward: Intrinsic reward expectations are kept in $\mathbf{C}^{int}$. The perceptual and cognitive errors are the basis for generating the intrinsic reward signals. However, we cannot simply use the prediction errors of the model or the reconstruction errors of the perceptual map as the intrinsic reward signal—a measure of interestingness should not assign high values to noise, which would be the result of a signal proportional to the prediction or reconstruction errors. Instead we use a measure of the *expected learning progress*, which is calculated using the *expected reducible error*.

To measure intrinsic reward, each internal state-action pair is associated with two error predictors, $\xi^{long}_{s,a}$ and $\xi^{short}_{s,a}$. These are calculated using a weighted combination of the sensory reconstruction errors: $err^{tot} = err^{per} + \alpha err^{cog}$. The first, $\xi^{long}_{s,a}$, keeps an error average with a *long* memory, while the second, $\xi^{short}_{s,a}$, keeps an error average with a *short memory*. The first is a measure of the baseline or irreducible error, while the second corresponds to the recent average error. They are calculated as follows:

$$\begin{aligned} \xi^{long}_{s,a} &= (1 - \eta^{long})\xi^{long}_{s,a} + \eta^{long} err^{tot}, \\ \xi^{short}_{s,a} &= (1 - \eta^{short})\xi^{short}_{s,a} + \eta^{short} err^{tot}, \end{aligned}$$

where the constant η^{long} induces a slower forgetting rate than η^{short}. Finally, the expected intrinsic reward is the difference of the two averages:

$$\mathbf{c}^{int}_{s,a} \leftarrow g(\xi^{short}_{s,a} - \xi^{long}_{s,a}), \tag{3}$$

with g enforcing zero as the lower bound. See Figure 2.

This formulation allows for continual adaptation of the system, which is important since sudden changes, e.g., the addition of a new internal state, to the underlying system may alter the error encountered in the future. Accordingly, $\xi^{long}_{s,a}$ must adapt to this "new normal".

Initial Values when a State is Created: Since the cognitive updates are incremental, the initial values given, when a state is created, can heavily influence the behavior of the agent. An optimistic initialization (OI) of these values is a beneficial addition to a curious system. It can be used to encourage the agent to try things it has yet to try by assigning high intrinsic value to new state-action pairs. This is leads to the so-called *frontier exploration* behavior. More explicitly, this behavior

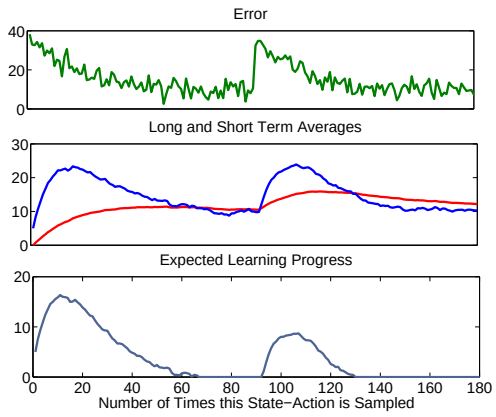


Fig. 2. Illustration of expected learning progress for a particular state-action pair. Top: The errors (e.g., reconstruction, prediction) after taking this state-action. The x-axis is the number of times this state-action was visited. Middle: The long term and short term averages for the error associated with this pair. An "optimistic" initialization puts the initial long average at zero and the short average high. Bottom: the expected learning progress, which corresponds to expected intrinsic reward. At about 90 visits, the error spikes and the intrinsic reward drives the agent to visit the state more often, since it assumes it can make learning progress (reduce this error). If the new error cannot be reduced, the long-term error increases, reflecting the new baseline.

can be achieved by setting $\mathbf{c}^{ext}_{n,a_j}$ and ξ^{short}_{n,a_j} to high values, and ξ^{long}_{n,a_j} to 0, for all j whenever a new internal state is added to the perceptual layer.

The complexity of behavior that a curious system exhibits cannot be captured solely by OI. The OI helps to generate important new samples for the development of the sensory system, in particular the perceptual layer. However, as the model matures OI plays an increasingly incidental role—the expected errors correspond more to the true expected learning progress as calculated by cognitive system. Curiosity then drives the agent to select the actions that will contribute the most useful samples to refine the model. Overall, this can be viewed as *novelty detection* followed by a *preferential sampling* where the novelty occurred.

C. Value System

The internal values (Q-values) are the basis for decision making. The potential action with highest value is most likely to lead to highest reward down the road. Assuming values can be tractably represented by a table on $\mathcal{S}$ means that a separate Q-value can be kept and updated for each state-action pair. The policy is then given by $\pi(\mathbf{o}) = \text{argmax}_a\, q_{s,a}$. Let r be the combined reward signals. Then, the Q-value for pair $\{s, a\}$ is $q_{s,a} \approx \text{E}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s(t) = s_0, a(t) = a_0\}$, which is the expected total discounted reward by taking action a in state s and then following the current policy thereafter.

Both external and intrinsic rewards are used to define the Q-values. This combination is important for balancing exploration and exploitation, even during development. Real-world agents cannot learn everything about their environments, and their exploration must be mediated by a pull to achieve

Algorithm 1: AGENT($\eta^{long}, \eta^{short}, \eta^{val}, \nu, \kappa, \omega, \alpha, \beta, \tau, \gamma$)

$\{n, s, a\} \leftarrow 1$
$\mathbf{o} \leftarrow$ SENSE($sysstate$)
NEWINTERNALSTATE ($\mathbf{o}$) //Birth
//Do until end of life
for $t \leftarrow 1$ *to* T **do**
  $sysstate \leftarrow$ ENVIRONMENT($sysstate$, a)
  ($\mathbf{o}'$, r^{ext}) $\leftarrow$ SENSE($sysstate$)
  //Classify as internal state
  $s' \leftarrow \text{argmin}_i\, dist(\mathbf{o}', \mathbf{p}_i)$
  //Take value maximizing action
  $a' \leftarrow \text{argmax}_i\, q_{s',i}$
  if $t > 1$ **then**
    PERCEPTUALADAPTATION()
    COGNITIVEADAPTATION()
    VALUEADAPTATION()
  end
  $\mathbf{o} \leftarrow \mathbf{o}'$, $s \leftarrow s'$, $a \leftarrow a'$
end

Algorithm 2: PERCEPTUALADAPTATION

$err^{per} \leftarrow dist(\mathbf{o}', \mathbf{p}_s)$ //Reconstruction error
if $err^{per} < \kappa$ **then**
  //Update internal state
  $\mathbf{p}_s \leftarrow (1 - \eta^{per}_s)\, \mathbf{p}_s + \eta^{per}_s\, \mathbf{o}'$
  $h_s \leftarrow h_s + 1$ //State ages
else
  $n \leftarrow n + 1$
  NEWINTERNALSTATE ($\mathbf{o}'$)
  $err^{per} \leftarrow 0$ //new state is perfect match
end

external goals. On the other hand, agents that are concerned only with external reward might not learn enough about the environment to maximize the collection of external reward. An ϵ-greedy approach for exploration through random action selection is not *informed*. Informed exploration is exploitative in the sense of maximizing the future expected discounted of a combined reward signal, i.e., by taking the action associated with the largest Q-value. This approach succeeds since these values are both informed (due to the learned model of intrinsic rewards) and appropriately *biased* (via optimistic initialization).

Planning through LSPI [8]: Every τ steps, the system enters a planning phase. The Q-values are adapted using LSPI and the tabular model of the transitions and the expected rewards. Without a model, LSPI requires collecting data in the form of $\{s, a, r, s'\}$. But if a forward state model and model of expected reward are available, then no data collected is needed. Within each iteration the system samples every known state-

Algorithm 3: COGNITIVEADAPTATION

$\mathbf{y}' \leftarrow$ VECTORIZE(s')
$err^{cog} \leftarrow dist(\mathbf{c}^{state}_{s,a}, \mathbf{y}')$ //model error
$err^{tot} \leftarrow err^{per} + \alpha\ err^{cog}$ // $\{s,a\}$ total error
//Update internal state f.model for LSPI
$\mathbf{c}^{state}_{s,a} \leftarrow (1 - \eta^{cog}_{s,a})\ \mathbf{c}^{state}_{s,a} + \eta^{cog}_{s,a}\ \mathbf{y}'$
//Update expected reward model for LSPI
$c^{ext}_{s,a} \leftarrow (1 - \eta^{cog}_{s,a})\ c^{ext}_{s,a} + \eta^{cog}_{s,a}\ r^{ext}$
$\eta^{cog}_{s,a} \leftarrow \eta^{cog}_{s,a} + 1$
//Update recent error avg.
$\xi^{short}_{s,a} \leftarrow (1\text{-}\eta^{short})\ \xi^{short}_{s,a} + \eta^{short}\ err^{tot}$
//Update long term error baseline
$\xi^{long}_{s,a} \leftarrow (1\text{-}\eta^{long})\ \xi^{long}_{s,a} + \eta^{long}\ err^{tot}$
//Update expected int. rew. model for LSPI
$c^{int}_{s,a} \leftarrow g(\xi^{short}_{s,a} - \xi^{long}_{s,a})$

Algorithm 4: VALUEADAPTATION

$r^{int} \leftarrow g(err^{tot} - \xi^{long}_{s,a})$ //{s,a} actual int. rew.
//SARSA updates
$q^{ext}_{s,a} \leftarrow (1 - \eta^{val})\ q^{ext}_{s,a} + \eta^{val}\ (\gamma\ q^{ext}_{s',a'} + r^{ext})$
$q^{int}_{s,a} \leftarrow (1 - \eta^{val})\ q^{int}_{s,a} + \eta^{val}\ (\gamma\ q^{int}_{s',a'} + r^{int})$
//LSPI updates every τ steps
if $cnt = 0$ **then**
  $\mathbf{Q}^{ext} \leftarrow$ LSPI-MODEL ($\mathbf{C}^{state}$, $\mathbf{c}^{ext}$, $\mathbf{Q}^{ext}$, γ)
  $\mathbf{Q}^{int} \leftarrow$ LSPI-MODEL ($\mathbf{C}^{state}$, $\mathbf{c}^{int}$, $\mathbf{Q}^{int}$, γ)
  $cnt \leftarrow \tau$
else
  $cnt \leftarrow cnt - 1$
end
$\mathbf{Q} \leftarrow$ COMBINEVALUESYSTEMS ($\mathbf{Q}^{ext}$, $\mathbf{Q}^{int}$, β)

action pair once, and uses only the three most likely transitions to build the linear system. Sparsifying the transition models reduces the potential computational burden and succeeds for low-branching factors. LSPI iterates until the policy converges.

SARSA for immediate value adaptation/correction: LSPI does not need to be carried-out after each interaction with the environment. It is possible to continue to update the model though temporal difference (TD) learning. Since the model is updated, so too should the expected learning progress associated with the newly visited state-action pairs. We therefore use use a simple SARSA update after each action is taken to update the external and internal values systems.

The off-line LSPI calculations and the on-line SARSA updates complement each other. As the agent interacts with the environment, and the model is refined the intrinsic reward signals tend to drop off. This simple SARSA update is computationally cheap and essentially provides the agent with a dynamic plan while the system awaits its next planning cycle. As a result plans need to be computed less frequently.

Please see Algorithms 1 - 4 for a concise description of the system. For details of LSPI, refer to Lagoudakis and Parr, 2003 [8], specifically the LSPI-Model algorithm.

IV. EXPERIMENTS AND RESULTS

A. Vision-Based Markovian Explorer

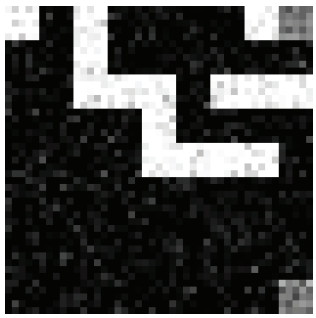
The system was evaluated in an environment type recently introduced by Lange and Reidmiller [9], in which an agent's observations are high-dimensional, noisy images, each of which shows the agent itself and its world (imagine a camera overhead pointed at at a mobile robot in a maze below). A sample observation, 45×45 pixels (2025 dimensions), is shown in Figure 3(a). There is slight (0.1 variance) gaussian noise on each pixel. In this particular environment, the agent always starts in the upper left. There are four actions, each of which moves it 5 pixels in one direction (up, down, left, or right in the image — orientation does not matter). When it reaches a gray square ("goal state"), it gets positive external reward and teleports back to the starting position in the upper left. Other transitions lead to slight punishment.

This environment was chosen for several reasons: (1) Due to high-dimensionality and noise, direct RL is intractable on the visual observations. An encoding of the images is needed[1]. (2) Before finding a goal, external value information is not available to guide the agent's policy. One method to aid data collection for UL+RL systems is via teleportation, in which the agent is transitioned to a nearly random position after some time (e.g., 20 steps of following its current policy or taking random actions). But we wish for the agent to explore in an informed way, thus the teleportation we used does not assist exploration (since it moves the agent back to the beginning); rather it must be overcome. (3) Finding the lower right goal will influence the agent to keep going there. It must value potential intrinsic reward more than external reward in order to explore the upper right region. (4) The places where the agent expects to learn something are changing as the agent learns and are not always near its current position, as a result planning becomes useful.

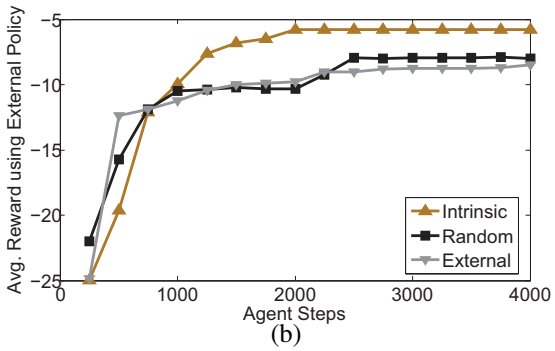
B. Setup

Agents used the following general setup: the distance function $dist$ was the normalized inner product (n.i.p.) subtracted from one, thus the perceptual and cognitive errors were both bounded from 0 to 1, and $\alpha = 1$ allowed both to be weighted equally. The error averaging rates were $\eta^{long} = 0.1$ and $\eta^{short} = 0.2$. Optimistic initialization was enabled: initial long-term errors were $\nu = 1$. Internal states were created when $1-$n.i.p. distance was greater than $\kappa = 0.05$. The learning rate lower bound was $\omega = 0.05$. The TD-learning rate was $\eta^{value} = 0.25$. Discount factor γ was set to 0.9. For agents using LSPI, the number of steps between planning phases $\tau = 20$. The external and intrinsic policies were combined through weighted addition of L1-normalized Q-value vectors per state (first shifted positive).

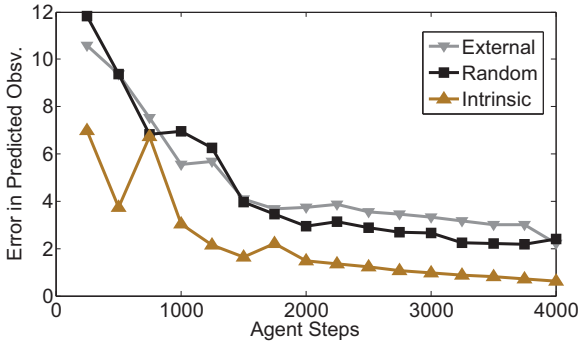
[1]Of course, some useful encoding scheme could be completely hand-designed for any particular environment, including this one. For a developmental agent the schemes for all possible environments cannot be hand-designed into the program, rather an appropriate one must be learned.



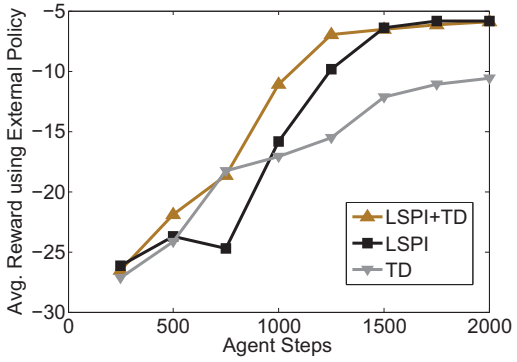
(a)



(b)



(c)



(d)

Fig. 3. Model building in "duck-world". (a). Example high-dimensional observation image (> 2000 dimensional), with the agent as the square in the upper left and positive external rewards at squares in the upper and lower right. Interior walls are the same color as the agent. (b). Average path cost from all possible starting points. (c). Average distance from the predicted next observation to the actual over all possible position-actions. (d). Effectiveness of planning in curious model-building.

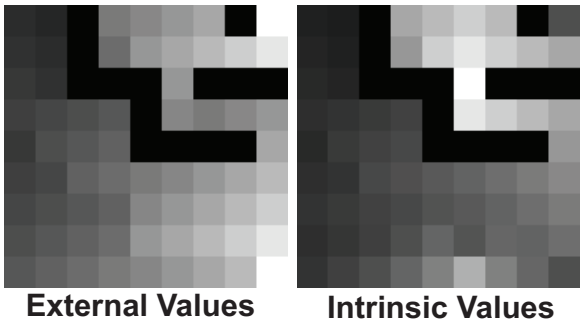


Fig. 4. Snapshots of external and intrinsic state values for a sample intrinsically motivated agent after 2500 steps, superimposed on an observation image, where white indicates high value. At this point, external values are stable (both goals have been found), and the intrinsic ones are changing. After it gains experience in the high value area, some other region becomes interesting, which could be far from its current position.

We compared three types of agents: random, external-random, and intrinsically-biased. The random agents simply always performed random actions. The external-random agents were random until any goal was found, then shifted to pulling from the external policy with 25% chance of a random action. The intrinsically-motivated agents placed a 0.8 weight on intrinsic policy and 0.2 on the external. Among the intrinsically-motivated agents, we tested the effect of LSPI vs. TD. To show the effect of planning, we compared agents using both LSPI and the TD-based SARSA, LSPI without SARSA, or SARSA without LSPI. In the TD agents, lines 5 and 6 were disabled in Alg. 4. In the LSPI agents, lines 2 and 3 disabled. In the LSPI+TD agents, nothing was disabled.

We measured performance of each agent's external policy as it gained experience. A "testing phase" was done every 250 steps, with learning frozen during the tests. In the test phase, an agent used its external policy only, and, starting in each possible non-goal position, got up to 30 steps to reach any goal. Second, to measure the success of model-building, we measured agents' error between predicted and actual next *observation*, over every position with every possible action. Predicted observation state-action pairs uses the state prediction probabilities as weights in a linear combination of prototype vectors of possible next states.

C. Results

Results are shown in Figure 3(b)-(d). First, the intrinsically-motivated agents always found the second goal position very early-on. The upper right goal is not likely to be found through an exploration policy based on randomness, and an externally-biased explorer remains attracted to the easy to find lower right square. The intrinsically motivated agent sees the benefits associated with the intrinsic reward signal as worth the cost in steps to reach the interesting regions. It is worth noting that this type of agent incurs more model errors early-on, compared to the others, since it is rewarded more for reaching newer regions. The LSPI-based intrinsic system is able to,

eventually, model the entire environment better than the other systems. Lastly, this system explores more efficiently, and is aided by the addition of TD-learning. TD-learning alone is not as effective of a method.

This system is meant to be practical for use on exploring robots, but a potential concern is that LSPI updating is too costly for real-world applications. The system runs quickly enough to watch it explore visually on a machine with an Intel Core i7 CPU (2.8GHz) and 8 GB RAM, with only small delays noticeable during the planning phase once it has grown to include roughly more than 400 internal state-actions.

V. Conclusions

We have presented a novel developmental system, which deals with high-dimensional noisy visual observations, and is driven to improve and adapt its perceptual and cognitive capabilities through artificial curiosity. It is intrinsically guided to improve how well it perceives and predicts in its world. It therefore explores its environment in an informed sense, avoiding a reliance on random action selection. Furthermore, we showed the importance of planning in curiosity-driven exploration, necessary due to the nonstationary intrinsic reward landscape. Systems that explore based on random actions, one-step look-aheads, or even TD-based intrinsic value updates are not fully equipped to deal with the challenges of the developmental RL problem.

Future directions: we will further explore this system in other environments. The agent should probably not re-plan at fixed intervals, but instead in response to some event (i.e., when surprised, such as when an expected reward is gone). Regarding scalability, a system completely based on VQ will not deal with real vision problems, e.g., requiring attention. Future work looks to first pre-process the visual information using the deep autoencoder networks currently in fashion [11]. The system should be scaled up to include an internal memory as well. A possibility is that the code from the smallest output layer of a deep network is then passed, rather than to the VQ perceptual map used in this paper, to the recently introduced TNT system [6]. The recurrent connection allows the TNT to discern underlying-states that VQ cannot, and is easily used in any system that already uses VQ. The reconstruction error from the autoencoder and the learning updates inside the TNT will form the basis for the curiosity signal. Another direction is to explore incorporation of top-down aspects (i.e., for adaptation of κ).

Acknowledgments

We would like to thank Faustino Gomez, Sun Yi, Leo Pape and Hung Ngo for enlightening discussions. We'd also like to thank the anonymous reviewers, who provided valuable feedback. This work was funded by SNSF grant CRSIKO-122697 (Sinergia), SNF project 200020-122124, and EU project FP7-ICT-IP-231722 (IM-CLeVeR).

References

[1] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks*, 4(6):759–771, 1991.

[2] F. Fernndez and D. Borrajo. Two steps reinforcement learning. *International Journal of Intelligent Systems*, 23(2):213–245, 2008.

[3] Baldassarre G. *Adaptive Behaviour in Anticipatory Learning Systems*, chapter Forward and bidirectional planning based on reinforcement learning and neural networks in a simulated robot, pages 179–200. Berlin: Springer Verlag, 2003.

[4] Linus Gisslén, Matt Luciw, Vincent Graziano, and Jürgen Schmidhuber. Sequential Constant Size Compressors and Reinforcement Learning. In *Proceedings of the Fourth Conference on Artificial General Intelligence*, 2011.

[5] Alexander Gloye, Fabian Wiesel, Oliver Tenchio, and Mark Simon. Reinforcing the driving quality of soccer playing robots by anticipation. *Information Technology*, 47(5):250–257, 2005.

[6] Vincent Graziano, Jan Koutnìk, and Juergen Schmidhuber. Unsupervised modeling of partially observable environments. In *European Conference on Machine Learning*, 2011.

[7] P.W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 449–456. ACM, 2006.

[8] M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[9] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8, July 2010.

[10] L.J. Lin. *Reinforcement learning for robots using neural networks*. School of Computer Science, Carnegie Mellon University, 1993.

[11] Jonathan Masci, Ueli Meier, Dan Ciresan, and Juergen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, 2011.

[12] I. Menache, S. Mannor, and N. Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

[13] Pierre-Yves Oudeyer, Frédéric Kaplan, and Véréna Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(6), 2007.

[14] Mark B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, Austin, Texas 78712, August 1994.

[15] J.C. Santamaria, R.S. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive behavior*, 6(2):163, 1997.

[16] Tom Schaul, Leo Pape, Tobias Glasmachers, Vincent Graziano, and Jürgen Schmidhuber. Coherence Progress: A Measure of Interestingness Based on Fixed Compressors. In *Proceedings of the Fourth Conference on Artificial General Intelligence*, 2011.

[17] J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 253–258, 1990.

[18] J. Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *Autonomous Mental Development, IEEE Transactions on*, 2(3):230–247, 2010.

[19] Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *AGI'11*, 2011.

[20] R. S. Sutton. First results with DYNA, an integrated architecture for learning, planning and reacting. In *Proceedings of the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, 1990.

[21] R.S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, pages 1038–1044, 1996.

[22] Hugo Vieira Neto and Ulrich Nehmzow. Visual novelty detection with automatic scale selection. *Robot. Auton. Syst.*, 55:693–701, September 2007.

[23] J. Weng. Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, 1(2):199–236, 2004.

[24] J. Weng and M. Luciw. Dually optimal neuronal layers: Lobe component analysis. *Autonomous Mental Development, IEEE Transactions on*, 1(1):68–85, 2009.