Comparing Reinforcement Learning Methods for Computational Curiosity through Behavioural Analysis

Nadia M. Ady
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2R3
nmady@ualberta.ca

Patrick M. Pilarski
Department of Medicine &
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2R3
pilarski@ualberta.ca

Abstract

Curiosity, a desire to know or learn more, appears to motivate many of the decisions made by biological systems. Numerous researchers, curious about a computational equivalent, have experimented with the idea of using computational reinforcement learning to produce curious behaviour in learning agents. Their efforts have resulted in a rich foundation for future work on curiosity, but the relationships between existing methods remain poorly understood. We suggest that one way to solidify this foundation is through a comparison of the behaviours resulting from different curiosity methods. In a domain with clear properties, the same agent, when motivated by different computational curiosity methods, will follow different behavioural trajectories. Tracking the underlying changes in such a curious agent's computations allows us to clarify why its behaviours differ and better understand how agents motivated by the tested methods might behave overall. Given the clear importance of understanding curiosity in understanding the decision-making behaviour of both biological and artificially intelligent systems, we emphasize the relevance of a systematic study of computationally curious behaviours and suggest that it is natural to begin with reinforcement learning methods.

Keywords: Computational curiosity, motivation, behavioural analysis, machine learning, intelligent agents

Acknowledgements

This research was undertaken, in part, thanks to funding from the Canada Research Chairs program, the Canada Foundation for Innovation, and was supported by the Alberta Machine Intelligence Institute (Amii), Alberta Innovates – Technology Futures (AITF), the Government of Alberta, and the Natural Sciences and Engineering Research Council (NSERC).

1 Introduction

From the multitude of 'why' questions asked by children to the multitude of projects undertaken by researchers, human behaviour is marked by curiosity. Many other animals also engage in behaviour that is well-explained as motivated by curiosity [2]. Curiosity's pervasiveness has lead us to believe it to be a crucial component of intelligent behaviour, yet it is still poorly understood [3]. One way we hope to develop our understanding of curiosity is to model and utilize curiosity in computational systems. Since curiosity appears to *motivate* many of the decisions made by biological systems, reinforcement learning is a strong candidate for achieving this goal.

Reinforcement learning (RL) allows machines and biological systems to learn, through trial and error, the value of situations and choices. In computational RL, we formalize this idea by requiring that a signal known as *reward* is delivered to the learner during its interactions with its environment. We generally associate a notion of likely future cumulative reward with situations and actions, and call it their *value*. There are existing algorithms that can be used to efficiently learn which actions maximize future reward. We can therefore design different motivations for our systems by designing their reward signals.

Researchers have developed different methods to modify the reward delivered to a learner or to modify other parts of an RL algorithm so as to evoke curious behaviours in their systems. Many of their methods have shown promise in real-world or simulated domains.

However, at present, there is no unified way to compare different curiosity methods. In both humans and machines, curiosity is challenging to measure due to its the variety and the individuality of the ways it is exhibited. One way we would like to be able to compare different curiosity methods is in how they impact agent behaviour.

To create a clear comparison of different curiosity methods, we want to hold both the domain and the majority of the agent's internal workings constant, varying only the curiosity method used to motivate the agent.

For our initial experiments, we hoped to gain insight using a simple, clearly understandable setting. We chose to design for decision process domains. By the term *decision process*, we mean the domain is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and a rule for determining the next state and reward given the current state and action.

Time is considered discrete rather than continuous. At each timestep, the agent is in some state $s \in \mathcal{S}$, takes an action $a \in \mathcal{A}$, receives a reward $r \in \mathbb{R}$, and arrives in another state $s' \in \mathcal{S}$, which begins the next timestep. The agent would take a next action a', repeating the process. The agent receives signals differentiating the state it is in and the reward it receives, and those signals are determined by the domain, usually dependent on the action chosen by the agent.

2 Experimental Design

Domain: We devised the curiosity bandit, depicted in figure 1, to showcase the behaviour elicited by variations in *domain-delivered reward*. The curiosity bandit has a single state ($\mathcal{S} = \{s_0\}$), but its three actions ($\mathcal{A} = \{a_1, a_2, a_3\}$) provide different rewards.

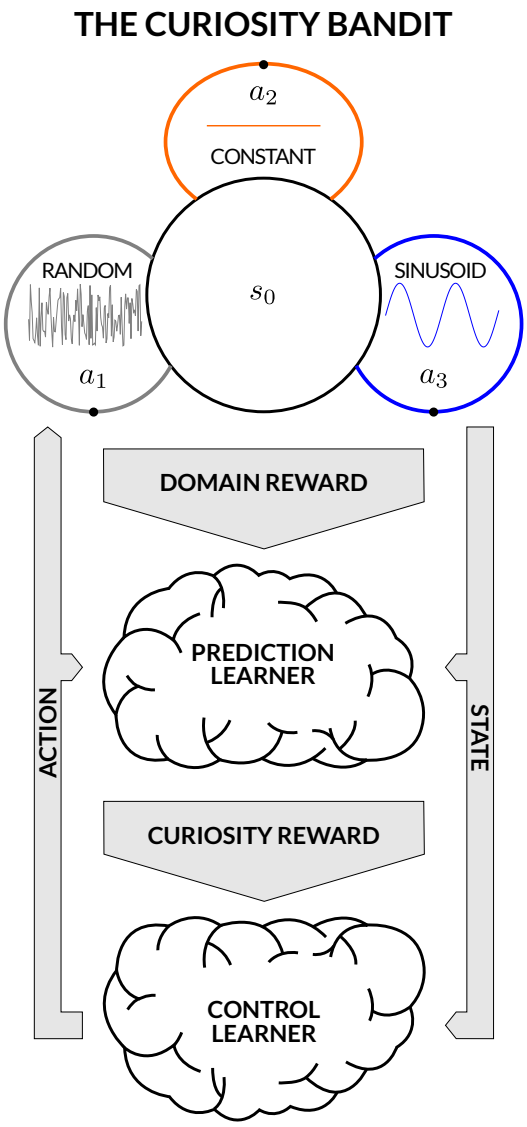

Figure 1: The Curiosity Bandit domain (top) and its interaction with a two-part agent suitable for error-based curiosity methods.

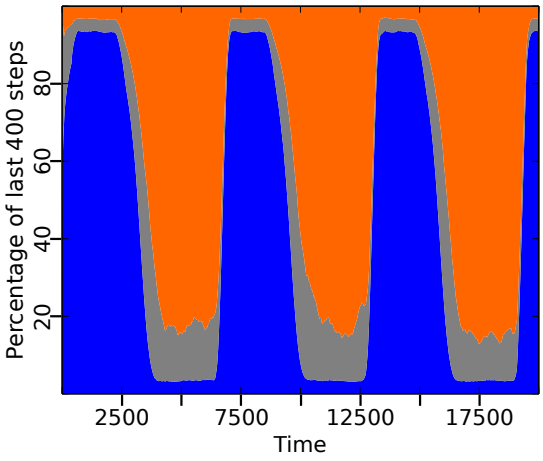

Figure 2: The usual RL agent: behaviour of an agent using standard Q-learning control to maximize future domain-delivered reward.

If the agent takes a_1, its reward is drawn uniformly randomly from $[-1, 1]$. If the agent takes a_2, it always receives a reward of 0. If the agent takes a_3, then it receives a reward of $\sin(c \cdot t)$, where c is a small constant (set to $c = 0.001$ in the included experiments) and t is the current timestep (starting at $t = 0$ in our experiments).

One typical view of curiosity is that it motivates agents to maintain an intermediate level of "arousal" (cf. the Wundt curve as described by Berlyne [1, pp. 200–201]). Therefore, providing the sinusoidal action as one form of regularity between the most simple (constant) and the most complex (random) seemed like a natural starting point in exploring the agent's possible reactions to variations in domain-delivered reward.

Agent: The design of our agent was motivated by emphasizing simplicity and consistency in the inner workings of the agent while allowing for the computations required for a variety of curiosity methods.

Many RL control algorithms rely on maintaining an estimate of the value of each action. Therefore, one component of the agent was a prediction learner, which used the TD(0) algorithm [7, pp. 133–138] to estimate the value $Q(s, a)$ of each action a, given it is taken from state s. At each timestep, given s, a, r, s', a' as described in the introduction, the prediction learner computed the *temporal difference error* (TD-error), δ, as

$$\delta \leftarrow r + \gamma\, Q(s', a') - Q(s, a) \tag{1}$$

where γ is a *continuation probability* (set to $\gamma = 0.9$ in our experiments), which determines how much more we value the reward achieved during the current timestep than that achieved in future timesteps.

Essentially, the TD-error is the difference between our *predicted value*, $Q(s, a)$, of taking action a from state s, and our *sample value*, $r + \gamma\, Q(s', a')$, which combines our sample reward r and the value of our sample action a', as taken from our sample state s'.

Because there may be an element of randomness to the domain's rule for determining the next state and reward given the current state and action, we do not necessarily want to change our new estimated value to the sample value—we only move it towards that value, so the estimated value for action a from state s is then updated as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\delta \tag{2}$$

where α is the learning rate (kept constant at $\alpha = 0.1$ in our experiments). We initialized $Q(s, a)$ to 0 for all states and actions.

Since we are already computing a form of prediction error—the TD-error—we can conveniently implement several curiosity methods based around prediction error. For the purposes of this initial experiment, we limited the tested methods to several which computed a new *curiosity reward* from the prediction error for each transition. This meant we could control the overall agent by simply using a control learner aiming to maximize cumulative future curiosity reward (*curiosity value*).

For control, we used ϵ-greedy Q-learning [7, p. 140]. Q-learning maintains estimates of the value (in this case, curiosity value) of each action, assuming the agent will choose the action with the highest value in the next step. If we had perfect estimates of our values, we would want to always act *greedily*, taking the action with the highest value. But to learn and maintain our estimates of each action-value, the agent must try every action occasionally. With probability ϵ, the agent will choose a random action, but otherwise, it really does choose the action with the highest value (hence the name, ϵ-greedy). In our experiments, we set $\epsilon = 0.1$.

Curiosity Methods: We selected four curiosity methods for our initial experiments. Each of these methods relies on prediction error as a component of its computation of a new motivation signal, which we call *curiosity reward*, in contrast with the reward signal provided by the domain, which we refer to as *domain reward*.

(a) **Absolute prediction error**, referring to the absolute value of some measure of prediction error, was one of the earliest learning signals optimized specifically for curiosity [6]. The initial intuition: to improve prediction, the agent should spend more time in areas of high error. Unfortunately, such an agent might get stuck repeatedly choosing areas that are highly unpredictable or random.

(b) Schembri et al. [5] specifically aimed to maximize **(signed) TD-error in domain value**, as opposed to general prediction error. The intuitive benefit of maximizing TD-error is that the agent's choices should favour areas which seem to be better than expected (and so afford positive TD-error) and avoid areas which seem to be getting worse (affording negative TD-error) in terms of the usual RL goal of maximizing cumulative future domain reward.

(c) **Learning progress**, as defined by Oudeyer et al. [4] refers to the decrease in error over a recent window of timesteps. Intuitively, an agent achieves high learning progress as its prediction improves.

(d) **Unexpected Demon Error** (UDE) was designed by White et al. [8] to measure the surprisingness of an observation and can be used as a curiosity reward. UDE is the ratio of the moving average of the prediction error and the variance. If the prediction error is consistently large then the samples have high variance, and we expect high error. If the error becomes larger than expected, something about the observation is surprising, and should be explored further.

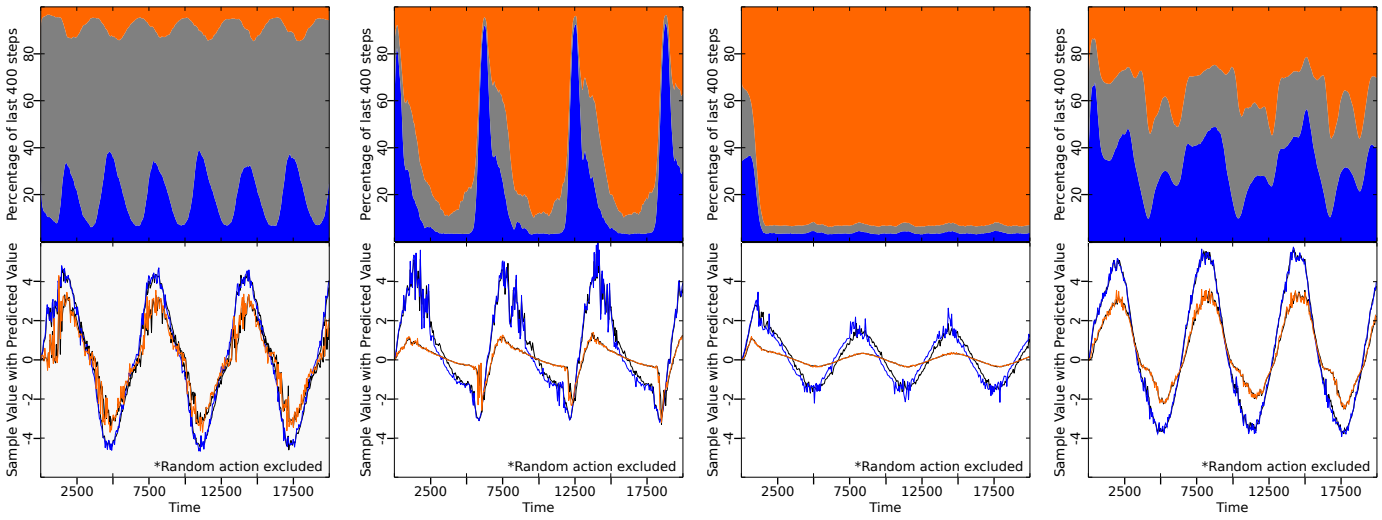

Figure 3: (A) As predicted, when the agent is motivated by **absolute prediction error**, we observe an attraction to randomness. Though one might expect the sinusoidal action to be chosen most (hardest to predict) when it experiences the largest change, around zero, the sinusoidal action actually shows periodic spikes exactly at its crests and troughs. These spikes occur because α is small; it is at these peaks that the value estimate lags most. (B) When the agent is aiming to maximize its **(signed) TD-error**, it can guarantee the greatest error when it chooses the sinusoidal action at its crests, similar to when it was aiming to maximize absolute error. In contrast, at the troughs for the sinusoidal action, the constant action is under-estimated. This under-estimation occurs because the prediction learner is taking the possibility of the sinusoidal action into account in its estimate. This results in the constant action giving the best error. (C) In taking the constant action, the agent always shifts the value of the constant action closer to zero by increasingly smaller amounts. Therefore, the sinusoidal action only ever shows more **learning progress** at the crests and troughs, where the predicted and sample values cross, resulting in corresponding blips. (D) In the behaviour of an agent maximizing **Unexpected Demon Error** (UDE), we see two peaks of the constant action either side of each trough and two peaks of the sinusoidal action either side of each crest.

3 Results and Discussion

For comparison, in figure 2 we show an illustration of the behaviour of an agent without a specific curiosity method, simply aiming to maximize future domain reward. This agent is using the ϵ-greedy Q-learning control learner described above, but it is estimating and making choices based on the domain reward. The plot shows, at each timestep, the percentage of the last 400 actions (or up to 400 actions, for timesteps earlier than 400)

We initially thought that tracking the TD-error might afford us some insight into the behaviour of the agent in relation to its computational curiosity method. However, it turned out that comparing the sample value to the predicted value (as shown in each lower plot of figures 3–6) provided better insight, as described in the caption of figure 3.

4 Conclusions

These initial experiments have already shown that we can more clearly discern differences in the behaviour motivated by different computational curiosity methods by controlling other aspects of the experiment. We have initial insight into how regularities in the environment can impact the behaviour of agents motivated by different curiosity methods. We believe that the principled understanding of computational curiosity will make significant contributions to our understanding of curiosity as a whole, and to the development of general machine intelligence.

Not only can curiosity benefit computational systems, allowing them to learn more effectively about non-stationary, specialized environments while using a general learning algorithm, but they could also be used to pique human curiosity, making things more engaging for the users of a wide range of computation-enabled technologies. We can *design* for curiosity by *modelling* curiosity.

References

[1] Berlyne, D. E. (1960). *Conflict, arousal, and curiosity.* New York: McGraw-Hill.

[2] Glickman, S. E., & Sroges, R. W. (1966). Curiosity in zoo animals. *Behaviour*, 26(1), 151-187.

[3] Gottlieb, J., Oudeyer, P. Y., Lopes, M., & Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences*, 17(11), 585-593.

[4] Oudeyer, P. Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation, 11*(2), 265-286.

[5] Schembri, M., Mirolli, M., & Baldassarre, G. (2007, September). Evolution and learning in an intrinsically motivated reinforcement learning robot. In *European Conference on Artificial Life* (pp. 294-303). Springer Berlin Heidelberg.

[6] Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In Meyer & Wilson (Eds.) *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior* (pp. 15-21).

[7] Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An Introduction* (Vol. 1, No. 1). Cambridge: MIT press.

[8] White, A., Modayil, J., & Sutton, R. S. (2014, July). Surprise and curiosity for big data robotics. In *AAAI-14 Workshop on Sequential Decision-Making with Big Data*, Quebec City, Quebec, Canada.