Compression Progress-Based Curiosity Drive for Developmental Learning

Hung Ngo, Mark Ring, and Jürgen Schmidhuber
IDSIA / University of Lugano / SUPSI / 6928 Manno-Lugano, Switzerland
Email: {hung, mark, juergen}@idsia.ch

I. Introduction

A continual-learning agent [1], which accumulates skills incrementally, benefits by improving its ability to predict the consequences of its actions, learning environmental regularities even when external reward is rare or absent. A principled way of motivating such agents is to use subjective compression progress [2] as an intrinsic reward for actions generating learnable but as-yet-unknown regularities in the observation stream.

Here we study *pure* curiosity behavior with the aid of a simple environment that features initially unknown regularities represented by a set of functions. At every time step the autonomous agent chooses which function to learn better, using a predictor dedicated to that function. Intrinsic reward for any given function depends on the corresponding predictor's progress. Our experiments exhibit the agent's developmental stages. Initially it learns to focus attention on an easily learnable constant function, then on a harder-to-learn linear function, finally on a hard-to-learn nonlinear function. The results illustrate how artificial curious systems can learn to deal with the unavoidable limitations of their predictor learning algorithms, by temporally focusing computational resources on those parts of the world that make learning easy, given their previously learnt knowledge.

II. Artificial Curiosity based on Compression/Prediction Progress

Basic Setup: The environment features a set of functions $f_i, i = 1 \dots N$, initially unknown to the agent. Each f_i maps an observed feature vector $\mathbf{x} = (x^1, ..., x^m)$ to a *discrete* outcome $y = f_i(\mathbf{x})$. The developmental stages of the agent reflect the acquisition of various environmental regularities represented by these functions. Multi-Layer Perceptrons (MLPs) are used as predictors, and standard error back-propagation as the online learning method, with only one training iteration for each new sample $(\mathbf{x}_k, y_k)$. Due to the stochastic nature of online learning, learning progress (see below) on the whole observation history is not always positive. Hence the agent keeps training $\mathbf{w}_i$ (the weight vector of MLP i) while maintaining $\mathbf{w}_i^\star$ as the current best predictor for function f_i.

To maximize cumulative learning progress, the agent uses an action-value function $Q(i)$ to keep track of the estimated progress of each function f_i, like in the $n-$Armed Bandit problem [3]. It uses an ϵ-greedy policy for balancing exploration and exploitation: with probability $1-\epsilon$ it chooses to learn from function f_i, where $Q(i) \geq Q(j), \forall j$; with probability ϵ it selects i at random, uniformly. The Q-function is updated using the current curiosity reward r_i: $Q'(i) \leftarrow (1-\alpha)Q(i) + \alpha[r_i - Q(i)]$, where the constant step size parameter $0 < \alpha \leq 1$ is used to cope with the non-stationarity of learning progress [3, §2.6], as shown in Figure 1.

Curiosity Reward for Online Interactive Learning: In principle, the agent's learning/compression progress is the number of bits saved [2] when encoding the historical data $h_i(n) = \{\mathbf{x}_k, y_k\}^{k=1:n}$, taking into account the *description length* of the predictor $\mathbf{w}_i$ [4]. If logistic sigmoid activation functions are used, the MLP outputs can be interpreted as conditional probabilities of the possible observations [5]. Then the *entropy* error $\mathrm{C}_i(n) = -\sum_{k=1}^{n} \log P(y_k|\mathbf{x}_k; \mathbf{w}_i)$ can be viewed as the number of extra bits needed to encode $h_i(n)$ using $\mathbf{w}_i$ [4]. To encode $\mathbf{w}_i$, we assume quantized weights obeying the same zero-mean Gaussian distribution with precision λ. The description length of $\mathbf{w}_i$ becomes the weight decay regularizer [6]; the description length of both $h_i(n)$ and $\mathbf{w}_i$ is $\xi(h_i(n), \mathbf{w}_i) = \mathrm{C}_i(n) + \frac{\lambda}{2}\mathbf{w}_i^T\mathbf{w}_i$. More sophisticated model-coding schemes combined with efficient predictor learning algorithms exist but are beyond the scope of this paper.

At each discrete time step t the agent chooses a function f_i to learn from according to the action policy on $Q(i)$, then proceeds as follows:

1. Observe $\mathbf{x}_n$, predict outcome y_n using $\mathbf{w}_i^\star$, and update history $h_i(n)$.
2. Compute $\xi(h_i(n), \mathbf{w}_i)$. Train $\mathbf{w}_i$ online, using sample $(\mathbf{x}_n, y_n)$ to get $\mathbf{w}_i^{'}$. Compute $\xi(h_i(n), \mathbf{w}_i^{'})$.
3. Compute $r_i := \xi(h_i(n), \mathbf{w}_i) - \xi(h_i(n), \mathbf{w}_i^{'})$. If $r_i < 0$, set $r_i = 0$; otherwise replace $\mathbf{w}_i^{\star'} := \mathbf{w}_i^{'}$. Finally compute $Q'(i)$.

III. Simulations and Analysis

The agent may actively choose between four types of functions embedded in the environment: constant, linear, nonlinear, and pseudo-random. Since they have distinctive learning complexity with respect to the MLP predictors used, they can serve to illustrate the effectiveness of the framework (see Figure 1). Each observation-outcome sample is represented by a binary

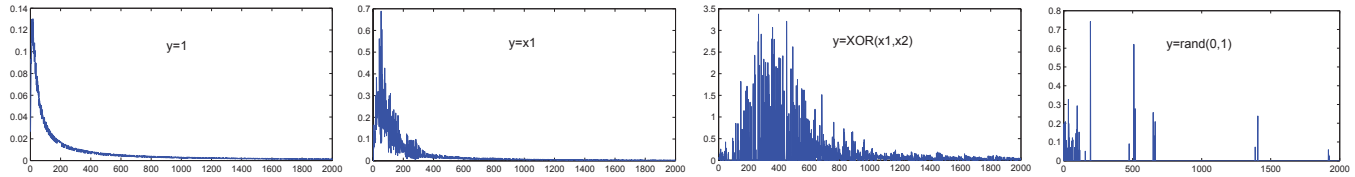

Fig. 1. Learning/compression progress vs. number of training samples for individual functions with different learning complexity. Only one training iteration was used for each new sample. The learning progresses are clearly non-stationary.

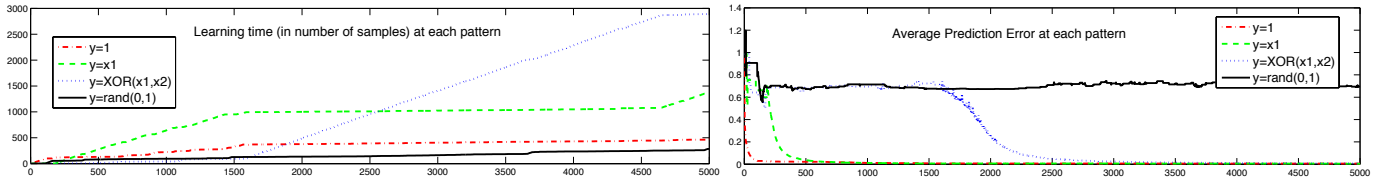

Fig. 2. Explorative behavior of the agent

vector $\mathbf{x} = (x^1, x^2)$ and a binary outcome $y \in \{0, 1\}$, with binary features x^1, x^2 randomly sampled from $\{0, 1\}$. We use $f_1 : y = 1, \forall \mathbf{x}$; $f_2 : y = x^1$; $f_3 : y = \text{XOR}(x^1, x^2)$; and f_4 is based on the binary, pseudo-random generator of Matlab.

The MLP predictors have 2 input units, 5 hidden units, and 1 output unit; the learning rate η is 0.1, and λ is 1. For the Q-function, $\alpha = 0.1$, and $\mathbf{Q}_0 = \mathbf{0}$.

Figure 1 plots learning/compression progress of each function's MLP predictor with respect to the number of training samples for a typical run. For simple patterns defined by constant and linear functions, the predictors can learn quickly: progress is fast in the beginning, then diminishes rapidly. For harder patterns like nonlinear XOR, the predictor needs more samples to start making progress. Once learning gets going, however, the curiosity reward increases rapidly. For pseudo-random observations, the predictor remains unable to learn much.

Figure 2 clearly exhibits the developmental stages of the learning agent. In the first 100 steps, after a bit of initial random exploration, it achieves quick progress on the simplest constant patterns. Then it spends most of its time on learning the linear function, from steps 100 to 1500. During this time, the XOR pattern also is tried out on occasion, due to the ϵ-greedy policy, but for a long time XOR seems random to the agent, despite XOR's deterministic regularity. After 1500 interactions, however, the agent starts making progress on XOR, suddenly experiencing a *Wow* effect—a sudden increase in intrinsic reward and a quick shift of attention to this pattern. The pseudo-random patterns always remain incompressible, never causing significant intrinsic reward, and never becoming a long-term focus of learning.

IV. Conclusion

We studied the purely curious behavior of an autonomous agent trying to learn environmental regularities, using the minimum description length principle to quantify its online learning/compression progress during interactive learning. After some initial steps of random exploration, the agent shifts its attention towards data expected to become more predictable, hence more compressible, through additional learning. Only observations with learnable but as-yet-unknown algorithmic regularities are temporarily novel and interesting, while subjectively random, arbitrary or fully predictable data quickly becomes boring. Ongoing work concentrates on automatically decomposing complex behaviors into meaningful sub-behaviors, and assigning (more powerful) prediction modules to them.

Acknowledgments

The authors would like to thank Leo Pape and Jonathan Masci for helpful discussions. This work was partially funded by the EU project FP7-ICT-IP-231722 (IM-CLeVeR) and SNF Sinergia Project CRSIKO-122697.

References

[1] M. B. Ring, "Continual learning in reinforcement environments," Ph.D. dissertation, University of Texas at Austin, Austin, Texas 78712, August 1994.
[2] J. Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990-2010). IEEE Transactions on Autonomous Mental Development, 2(3):230-247, 2010.
[3] R. S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press (1998).
[4] J. Rissanen. Stochastic Complexity in Statistical Inquiry. Hackensack, NJ:World Scientific (1989).
[5] C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press (1995).
[6] G. E. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. Proceedings of the COLT'93, (Santa Cruz, California, USA, July 26-28), pp. 5-13, 1993.