A Brain Inspired Cognitive Architecture for Reinforcement Learning

Tyler Streeter
Iowa State University, Ames, IA USA
VR Applications Center, Brainpower Labs LLC
www.tylerstreeter.net
tylerstreeter@gmail.com

James Oliver
Iowa State University, Ames, IA USA
VR Applications Center
www.vrac.iastate.edu/∼oliver
oliver@iastate.edu

Abstract

We present a novel cognitive architecture, Sapience, inspired by the high-level organization of the mammalian brain. It is intended for reinforcement learning problems, is scalable to high-dimensional sensory and motor spaces, utilizes curiosity rewards to improve its own world model, and is directly implementable in software.

We aim towards the long-term goal of a general purpose "software brain" applicable to all kinds of real-time motor control tasks. How can this goal be achieved? Our general strategy is to engineer a cognitive architecture combining systems-level neuroscience with machine learning methods. We first take inspiration from the mammalian brain's high-level organization and function. Thus, our architecture includes five components, functional abstractions of the sensorimotor cortex, hippocampus, basal ganglia, cerebellum, and prefrontal cortex regions. (Note that our focus is brain-inspired engineering, not biologically plausible brain models.) To make the architecture concrete, we then choose specific machine learning algorithms for every component, each providing a unique computational benefit to the overall system. These include Bayesian networks, unsupervised learning kernel mixture models, supervised learning neural networks, and reinforcement learning.

What should be the learning objectives of this architecture? We use theoretical reinforcement learning (RL) as an organizing principle (Sutton and Barto 1998). The primary objective is to learn complex motor control tasks defined as RL problems. We distinguish between two related objectives: to solve externally given RL tasks, and to improve an internal world model (which implicitly helps external RL). Corresponding to these objectives are two types of reinforcements: external (programmer-defined) reinforcements which define the task, and internal "curiosity rewards" (Schmidhuber 1991) for world model improvements, which encourage active sensory exploration.

The resulting cognitive architecture, which we call "Sapience," represents a complete learning system which can handle arbitrary real-valued sensory (including reinforcements) and motor arrays. It is designed to be practical: directly implementable in software, scalable with available computing resources, and generally applicable (e.g., for video games and robotics). The entire architecture has already been implemented in software, along with many test programs and analysis tools. No aspect described here is merely a hypothetical mechanism; every part has been instantiated explicitly in C++. As the current implementation is relatively untested, most of our current effort involves evaluating it on various motor learning scenarios.

Here we describe the architecture at a high-level, illustrating its general learning objectives and high-level organization. More details will appear in the primary author's PhD thesis (in preparation).

Architecture

The Sapience architecture includes five components (Figure 1). All communication (among components and with the environment) uses a common format: real-valued arrays. The system is updated at a fixed rate (e.g., 10 Hz); on each update step it receives new inputs, performs internal processing, and produces new outputs.

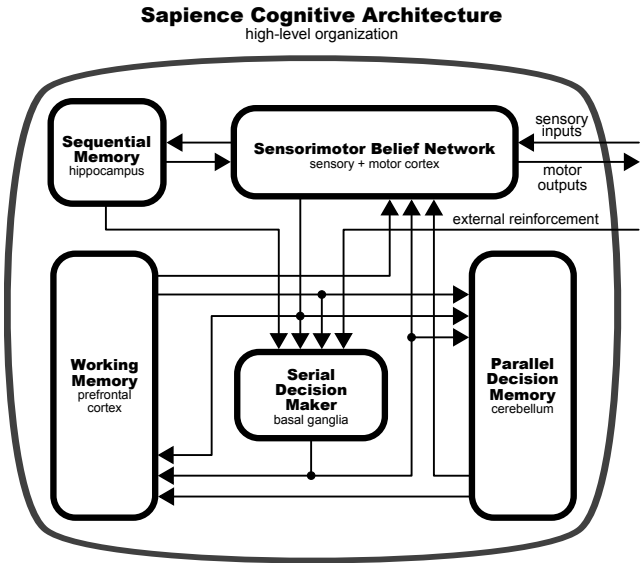

Figure 1: Components and corresponding brain regions.

Sensorimotor Belief Network This component is an internal world model. It learns a probabilistic model of incoming data patterns, produces motor control signals, and measures model improvements for curiosity rewards.

It is structured as a hierarchical empirical Bayesian network, combining the practical belief propagation algorithm of Pearl (Pearl 1988) with an unsupervised learning kernel mixture model (kernel-based Maximum Entropy learning Rule, or kMER (Van Hulle 2000)). Each node of the network uses kMER to learn online the conditional probability distributions relating it to other nodes. The network is structured as a hierarchy to combat the curse of dimensionality: high-dimensional input data spaces are subdivided into many smaller spaces, thus providing a fixed limit on each node's input dimensionality. In Bayesian network terms, likelihood messages come from raw input data (bottom-up), and priors come from the hierarchy's root node (top-down). The root node's prior distribution comes from the Sequential Memory component (described below). The posterior distributions computed at each node provide a distributed belief representation of the world. Furthermore, the concatenated posteriors should be more linearly separable than the input data since kMER can produce sparse codes. Other components in the architecture can influence this belief representation through "virtual evidence."

To measure improvements to this world model, we use the Jensen-Shannon divergence between the prior and posterior distribution at each node as a measure of information gain (similar to the KullbackLeibler divergence approach in (Schmidhuber, Storck, and Hochreiter 1995)). The mean information gain across all nodes then provides an internal curiosity reward to the Serial Decision Maker.

Sequential Memory This component learns sequential predictions which provide a prior distribution to the Bayesian network's root node. We use the dynamic reconstruction algorithm described in (Haykin 2008) (section 13.11). This involves a tapped delay line array representing a history trace of recent events (i.e. the root node's likelihood), coupled with a single-layer neural network predictor trained to predict the next pattern from the current history trace. Additionally, the history trace is provided to the Serial Decision Maker for making temporally precise reward predictions.

Serial Decision Maker This component performs reinforcement learning, using neural networks to represent context-dependent value and action selection (i.e. an actor-critic architecture). The Bayesian network's posteriors provide input context. Both types of reinforcement (external and internal) influence learning here. To generate a training signal for the neural networks, rather than using the well-known temporal difference (TD) algorithm (Sutton and Barto 1998), we opt for the "primary value learned value" (PVLV) dopamine model (O'Reilly et al. 2007), which aims to overcome TD limitations by using separate mechanisms for learning primary rewards and conditioned stimuli. Action selection is performed with a winner-take-all mechanism. Action choices influence a subset of nodes in the Bayesian network which model motor-related data. (The priors from these nodes then produce motor control signals). Action choices also modify the Working Memory component's contents.

Parallel Decision Memory This component automates well-learned actions. It watches the Serial Decision Maker's action selection and, over several trials, learns to assume control, freeing the Serial Decision Maker to focus on trial-and-error learning, not repetitive actions, as described in (Peck, Streeter, and Kozloski 2007). It uses single-layer neural networks (mapping context to actions) trained via supervised learning. Its parallel structure enables driving multiple motor and working memory targets simultaneously.

Working Memory This component provides a set of general purpose memory cells whose contents can be individually read and written. Essentially, this extends the set of RL choices beyond just motor actions to include working memory actions (active storage and recall), an arrangement inspired by the prefrontal cortex/basal ganglia/working memory (PBWM) model (Hazy, Frank, and O'Reilly 2007). This component augments the system's world model with contextual items not immediately present in sensory space. The recurrent combination of RL-based decision making with these memory cells (whose contents *influence* decision making) provides a sort of general RL-programmable computer.

Acknowledgements

This work is supported by AFOSR (FA9550-05-1-0384) and Brainpower Labs LLC. We are also grateful for the research infrastructure provided by the VRAC.

References

Haykin, S. 2008. *Neural Networks and Learning Machines*. Prentice Hall.

Hazy, T. E.; Frank, M. J.; and O'Reilly, R. C. 2007. Towards an executive without a homonculus: Computational models of the prefrontal cortex/basal ganglia system. *Philosophical Transactions of the Royal Society B* 362:1601–1613.

O'Reilly, R. C.; Frank, M. J.; Hazy, T. E.; and Watz, B. 2007. Pvlv: The primary value and learned value pavlovian learning algorithm. *Behavioral Neuroscience* 121:31–49.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Peck, C.; Streeter, T.; and Kozloski, J. 2007. An integrated cerebro-cerebellar model demonstrating associative learning and motor control. In *Proceedings of the 10th Tamagawa-Riken Dynamic Brain Forum*.

Schmidhuber, J.; Storck, J.; and Hochreiter, J. 1995. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of ICANN*, 159–164.

Schmidhuber, J. 1991. Curious Model-Building Control Systems. In *Proceedings of IJCNN*, 1458–1463.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Van Hulle, M. M. 2000. *Faithful Representations and Topographic Maps: From Distortion- to Information-Based Self-Organization*. Wiley.