OpenBrain: The Story So Far

April 22, 2016

Overview

Current Work Session Organization

We currently have three sets of separate meetings throughout the week.

1. Algorithms Meeting - Tuesday Evenings - talk strictly about the different pieces of the OpenBrain model.
2. Reading group - Friday Afternoons - First hour spent on presenting the paper to the group; Second hour spent on discussing what elements could be incorporated into the OpenBrain model.
3. Software Meeting - Monday afternoons. Discuss implementation details and write code.
    1. These have unfortunately been infrequent

Theory: Important Concerns

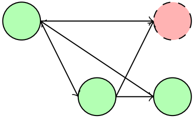
1. Intelligence - definition and measurement
   - UIM incompatable:

$$\Upsilon(\pi) = \sum_{\mu \in E} 2^{-K(\mu)} \mathbb{E}\left(\sum_{i=1}^{\infty} r_i\right) \tag{1}$$

   requires meaningful "timesteps", π_O runs asynchronously!
2. Mathematical framework for asynchronicity.
   - How do we talk about representation theory for agent on different timeline.
3. Conwayian learning rules for emergent intelligence
   1. Especially stressing the need to keep emergent rules simple.

Theory: A Rigorous Mathematical Framework

How do we account for asynchronousity? Ignore it!



Definition. A **neuron** $n \in N$ is defined by

- a voltage $V_n(t)$
- a decay time τ_n
- a refactory period ρ_n
- a voltaic threshold θ_n

Definition. A **connection** $c \in C$ is a tuple $(n_i, n_j, w_{ij}) \in N \times N \times \mathbb{R}$ where n_i is the **anterior neuron**, n_j is the **posterior neuron**, and w_ij is the standard synaptic weight.

Theory: A Rigorous Mathematical Framework

Definiton. A neuron n is said to **fire** if it is not in its refractory period and $V_n(t_k) = V_n[k] > \theta_n$. Then for all $m \in P_n$,

$$V_m[k+1] \mathrel{+}= w_{nm}\sigma(V_n[k]);$$

that is, voltage is propagated to the posterior neurons.

Immediately after neuron n fires, it enters a **refractory period** until time $t_k + \rho_n$, or iteration $k + \frac{\rho_n}{\Delta t}$.

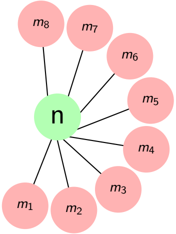


Figure: A neuron n firing into its posterior neurons, $P_n = \{m_1, \ldots, m_8\}$

Theory: Continuous Time Universal Intelligence

- Theoretical problems evaluating our agent.
- UIM defines an environment, μ, as a probability measure on sequences of actions and perceptions. Typically

$$o_1 r_1 a_1 o_2 r_2 a_2 \dots o_n r_n a_n \tag{2}$$

where r_i are rewards!

- What does it mean to have a reward at time step k when $\Delta t \to 0$? $\boldsymbol{r_i \to 0}$ **:(**
- Actions become sparse as $\Delta t \to 0$.

$$a_1 \emptyset \dots \dots \dots \emptyset a_{100000} \emptyset \dots \dots \emptyset a_{2500605} \dots \tag{3}$$

Theory: Continuous Time Universal Intelligence

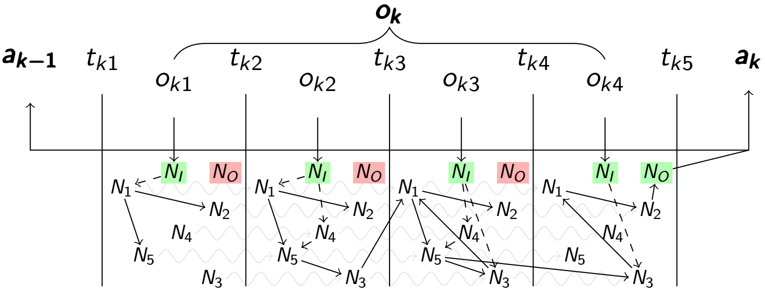


Figure: The diagram of sub-observation neural interaction for π_O.

Solution: Subobservations

- Actions determine what it means to be an observation.
- A visualization of asyncronous neural activity.
- More details in our paper at github.

Theory: Other progress!

1. Hutter's AIXI
2. Biological parallels:
    1. Synaptogenesis
    2. Asynchronicity - important
    3. Refractory periods
3. Sparse Distributed Representation
4. LSTM Models: Rewards associated with actions taken in the world.
5. In the process of writing paper to be presented at NIPS (2017) conference.

Theory: Goals

1. Develop a strong representation theory for the algorithm.
    1. What class of functions can OpenBrain approximate?
2. Figuring out how to measure learning quantitatively
3. Expanding framework to include specialized neurons to more closely mimic biology
    1. Wide variety of applications to all sorts of problems if specificity can be cracked.
    2. Cracking neuron specificity and learning $\rightarrow$ solution to general AI problem as $t \rightarrow \infty$.

Software: Overview

The pieces of the OpenBrain software project

1. Brain
2. Environment
3. Visualization

Progress

1. Design Doc Mostly Complete
2. Prototyping Framework
    1. Make sure that we can input learning rules
    2. Return empirical measurements to gauge usefulness of learning rules.
3. Migrate to IDE
    1. More efficient build process
    2. Code completion is nice

DEMO!!

Software: Obstacles

1. Determining how to scale across many different computers
    1. How to scale program
    2. Designing for scale
2. Saving state
3. Maintaining inheritence patterns for neuron structures.
4. Erlang is hard to learn

References

Yaniv Taigman et. al. (2014)
DeepFace: Closing the Gap to Human-Level Performance in Face Verification
Facebook AI Research

Michael Nielsen (2014)
Neural Networks and Deep Learning
http://neuralnetworksanddeeplearning.com/

The End