Cortical learning algorithms with predictive coding for a systems-level cognitive architecture

Ryan McCall and Stan Franklin
Computer Science & Institute for Intelligent Systems, The University of Memphis

Introduction

- The mind might be partly understood as a generic statistical model inference (Friston, 2010) based on coincidence patterns and sequences of such patterns in sensory data
- We present a predictive coding extension to the HTM Cortical Learning Algorithms (CLA) termed PC-CLA as an initial step
- PC-CLA is suggested as a building block for representation, memory, and learning in the systems-level LIDA cognitive architecture

Perceptual Principles

Mountcastle (1978) first proposed that the entire neocortex performs a similar function. Bedny et al. (2011) found congenitally blind humans perform language processing in their visual cortices during verbal tasks. We seek an algorithm employing the following:
- Autonomy and Agency (Online and Unsupervised)
- Hierarchical Decomposition
- Sparse Distributed Representation
- Prediction
- Approximate Bayesian Inference
- The Free-Energy Principle (Friston, 2010)

Methods

- Cortical Learning Algorithms (Hawkins, Ahmad, & Dubinsky, 2011) integrate:
  - Sparse distributed representation
  - Coincidence memory
  - Variable-order temporal memory
  - Online, unsupervised learning
- Our predictive coding extension, PC-CLA, incorporates hierarchical predictive coding:
  - Each level sends top-down predictions
  - Pass prediction error feed-forward
  - Allows for multiple hierarchical levels
- Initially not integrated with LIDA

Results

PC-CLA’s Step 2 produces a sparse distributed representation (SDR) of the current input.
To determine the noise robustness of these SDRs, we tested the effect varying amounts of added noise had on an input’s SDR.
- For different true bit rates in input patterns, r, ranging from 1% to 5%
  - Generate a single Boolean input with 2^r dimensions having r bit randomly chosen to be true
  - Perform Step 2 using the input 250 times recording the final SDR
- For different noise amounts, n, from 0% to 50%, randomly generate 500 noisy versions of original pattern with the noise added uniformly to true bits and false bits alike.
  - For each noisy version:
    - Perform Step 2 using the noisy pattern 250 times recording the final SDR
  - Compare the “noisy” SDR with the original using the normalized taxicab distance, here, the total number of errors between the two SDRs divided by total number of representational units


References


Acknowledgements

We thank Tamas Madl and Pulin Agrawal as well as the anonymous reviewers for their useful feedback.