2013

Cortical Learning Algorithms with Predictive Coding for a Systems-Level Cognitive Architecture

R. McCall

S. Franklin

Cortical Learning Algorithms with Predictive Coding for a Systems-Level Cognitive Architecture

Ryan J. McCall RMCCALL@MEMPHIS.EDU
Stan Franklin FRANKLIN@MEMPHIS.EDU
Fedex Institute of Technology 301, The University of Memphis, Memphis, TN 38152 USA

Abstract

Human-level intelligent agents must autonomously navigate complex, dynamic, uncertain environments with bounded time and memory. This requires that they continually update a hierarchical, dynamic, probabilistic (uncertain) internal model of their current situation, via approximate Bayesian inference, incorporating both the sensory data and a generative model of its causes. Such modeling requires suitable representation at multiple levels of abstraction from the subsymbolic, sensory level to the most abstract conceptual representation. To guide our approach, we identify principles for perceptual representation, perceptual inference, and the associated learning processes. Based on these principles, a predictive coding extension to the HTM Cortical Learning Algorithms (CLA), termed PC-CLA, is proposed as a foundational building block for the systems-level LIDA cognitive architecture. PC-CLA fleshes out LIDA's internal representations, memory, learning and attentional processes; and takes an initial step towards the comprehensive use of distributed and probabilistic (uncertain) representation throughout the architecture.

1. Introduction

In this work we describe how modern treatments of perception might be integrated with a broad systems-level cognitive architecture, LIDA. This work is presented in the context of the LIDA model; however, it does not depend on the commitments of this model (Franklin, Strain, McCall, & Baars, 2013). As such, much should be widely applicable to other cognitive architectures. We view solutions to the problem of perception or internal representation as central to building intelligent machines, since accurate modeling of the world underlies many of the faculties needed for generally intelligent agents, including action selection, motivations or goal-directedness, higher-level cognition, etc.

The outline of the paper is as follows: We first review several inspirational models of perception and data analysis. Some, as models of cortical computation, have biological inspiration to their credit. Others feature equally compelling information-theoretical and mathematical justifications. We next identify key principles, across these various approaches, to guide the research and development of networks capable of modeling complex, "real-world" sensory data. While such an approach might have more narrow applications to pattern recognition problems, we focus here on PC-CLA's potential application as a building block for the implementation of the current representations, memory, and learning and attentional processes of a systems-level cognitive architecture. Goertzel (2012) also discusses integrating symbolic and subsymbolic processing in a cognitive architecture and favors a tight integration between separate algorithms for the two. In contrast, we view PC-CLA as a single algorithm for integrating both the symbolic and the subsymbolic. After identifying these principles, we then focus on the umbrella-like free-energy principle and its application to model inference for complicated hierarchical dynamic models in an approach termed Generalized Filtering (GF). GF provides the theoretical guide for the proposed PC-CLA. Next, we give a brief description of the LIDA model

of cognition and discuss some of the theoretical issues surrounding the integration of PC-CLA with LIDA. Finally, we describe the initial implementation of PC-CLA, report on some initial tests, and discuss directions for future work.

2. Models of Perceptual Analysis and Learning

Several existing theories of perception and pattern recognition provide inspiration and motivation for the PC-CLA model. Hierarchical Temporal Memory (Hawkins & Blakeslee, 2004; Hawkins, Ahmad, & Dubinsky, 2011) postulates that the neocortex builds a model of the world using a spatiotemporal hierarchy of basic computational units, each performing the same learning and inference algorithm, which entails storing co-occurrence patterns and then sequences of such patterns. Rao & Ballard's (1999) "predictive coding" models the visual cortex as a hierarchical network with units at each hierarchical level continually predicting the responses the units in he next lower level via top-down, feedback connections. Through feed-forward connections the lower level units send back the errors between the top-down predictions and the actual activity. These error signals are then used to correct the higher-level representation of the cause of the input signal, ideally improving future top-down predictions. Generalized Filtering (Friston et al., 2010) is an online recursive Bayesian estimation scheme[1] for nonlinear state-space models in continuous time. It provides a method for estimating the posterior (or conditional) probability density functions on the hidden states (current representations) and unknown parameters (e.g., connections implementing memory) generating the observed input data (e.g., audio signals or human kinematics). Finally, we mention, but will not focus on, some closely related theories including HMAX (Riesenhuber & Poggio, 1999), Leabra (O'Reilly & Munakata, 2000), DeSTIN (Arel, Rose, & Coop, 2009) and rRNN (Bitzer & Kiebel, 2012).

Taken together, these theories call for the use of hierarchy, non-linear dynamics, approximate Bayesian inference, and biological inspiration from the primate neocortex to confront the problem of data analysis or data modeling, a problem we view as central to the development of intelligent agents. Based on these ideas, we next identify guiding principles for the PC-CLA model.

3. Perceptual Principles for Systems-Level Cognitive Architectures

Here we identify key principles for perceptual representation, processing, and learning in cognitive systems. We focus on ideas supported by evidence from neurobiology and neuroscience as well as information theory and mathematics. What evidence is there that common perceptual principles might exist? Mountcastle (1978) first proposed that a common function is performed throughout the entire neocortex. In support of this are experiments that successfully rerouted sensory stimuli to foreign cortical areas in mammals suggesting neocortical adaptability across modalities (e.g., Métin & Frost, 1989). Bedney et al. (2011) found congenitally blind humans perform language processing in their visual cortices during verbal tasks, and concluded that so-called "visual" cortical areas could take on language processing. Finally, several researchers have suggested that the neo-cortex may be implementing a large number of similar, hierarchically arranged "cortical circuits" (e.g., Douglas & Martin, 2004; Felleman & van Essen, 1991; Phillips & Singer, 1997).

3.1 Autonomy and Agency

Autonomous agents require online learning without stoppage or interruption. However, this still allows for a human-like developmental period, in which the agent learns autonomously from its environment. In addition, agents must be able to learn autonomously without the aid of labeled

[1] Recursive Bayesian estimation is an approach for estimating an unknown probability density function recursively over time using incoming measurements and a mathematical process model.

data, i.e., to perform unsupervised learning.

3.2 Hierarchical decomposition

Hierarchical decomposition allows a computational task to "be reduced to a small number of activities at the next lower level so the computational cost of finding the correct way to arrange those activities [...] is small" (Russell & Norvig, 2010). In the context of internal representations, a multitude of representations can be constructed from different combinations of the same representations in the next lower level. This achieves substantial pattern reuse and such decomposition can be repeated recursively. Hierarchical decomposition via nonlinear operations is a major foundation of the "deep learning" movement (Bengio, 2009).

3.3 Sparse Distributed Representation

With distributed representations, a number of units represent each pattern, and each unit can participate in the representation of many patterns. Each unit in a distributed representation can be thought of as representing a single "microfeature" (Hinton, McClelland, & Rumelhart, 1986), with the information being encoded by particular combinations of such features, and not by a single bit or feature. With distributed representation, the explicitness in representation is lost; however, it is more efficient than the localist one. Additionally, similar elements have similar representations, because they may share multiple features.

The idea of sparse representation can be traced to the infomax principle (Linsker, 1990), which suggests the brain maximizes the mutual Shannon information between sensations and representations. Similarly, the sparse coding hypothesis (Olshausen & Field, 1996) suggests that sensory patterns are represented by the strong activation of a relatively small set of neurons. Kanerva (1998) demonstrated that high-dimensional sparse representations are highly unique and noise robust.

3.4 Prediction

Modern views of perception see the human brain as a constructive or predictive organ actively generating predictions of its sensory inputs using a generative model[2]. Bar (2009) describes a "proactive" brain continuously generating predictions anticipating the relevant future. HTM theory (Hawkins, Ahmad, & Dubinsky, 2011) suggests that temporal predictions are useful because they help produce temporally invariant representations and can help with the recognition of noisy temporal patterns. Here we distinguish *top-down prediction*, a prediction coming from a given hierarchical level and predicting the pattern of the next lower level, from *temporal prediction*, one coming from a given level and predicting future patterns of the same level.

3.5 Approximate Bayesian Inference

A close correspondence has been shown between human decision making and judgments made by Bayesian decision theory, which defines optimal behavior in uncertain environments with noisy signals (Kording & Wolpert, 2004). Lee & Mumford (2003) suggested the visual cortex performs hierarchical Bayesian inference where "top-down contextual priors and bottom-up observations [...] implement concurrent probabilistic inference." Knill & Pouget (2004) popularized the "Bayesian coding hypothesis" suggesting that the brain represents information probabilistically, by coding and computing with probability density functions or approximations to them.

[2] A generative model is a model for probabilistically generating observable data from some distribution, typically given some hidden parameters.

3.6 The Free-Energy Principle

The free-energy principle (Friston, 2010) is a theory of brain function that accounts for action, perception and learning. The principle unifies a number of existing theories including predictive coding, the Bayesian coding hypothesis, the infomax principle, associative plasticity, optimal control, and value learning. The principle suggests that the brain changes to minimize its (variational) free energy, an upper bound on the surprisal[3] in sampling sensory data, given an approximate probabilistic model of the data. If an agent minimizes free energy, it implicitly minimizes surprisal since free energy is an upper bound. Unlike surprisal, free energy can be evaluated because it is a function of the agent's sensory states and its approximate internal probabilistic model of the causes of the sensory data.

4. Applying the Free-Energy Principle

Here we overview the application of the free-energy principle to a hierarchical dynamic generative model showcasing an approach termed Generalized Filtering (Friston et al., 2010). Attractively, Generalized Filtering makes biologically plausible assumptions (Feldman & Friston, 2010) and conforms to the free-energy principle. It operates online and unsupervised, inferring model states (current representations), parameters (memory), and hyperparameters (uncertainties). The approach can also be viewed a form of hierarchical Bayesian inference. Embodying our identified principles, we adopt Generalized Filtering as a theoretical guide for PC-CLA.

Generalized Filtering is a method for the approximate Bayesian inversion (estimation) of a statistical model with a quite sophisticated and general form. Specifically, Generalized Filtering considers a hierarchical, dynamic, generative model comprised of hidden state variables, the dynamics of which can be influenced by an appropriate non-linear function (equation of motion). Furthermore, at each level, these hidden state variables may be subject to (random) stochastic fluctuations representing the variables' statistical uncertainty. We will discuss uncertainty in terms of *precision*, the multiplicative inverse of the variance[4]. High statistical precision implies a probability distribution that is highly concentrated about its mean suggesting that the mean is known with a high degree of precision in the ordinary sense. In terms of statistical model inference, precision is a hyperparameter to be estimated from the data. We will refer to such hyperparameters that parameterize estimations of precision as *precisions*. Precisions in Generalized Filtering's form of generative model may themselves undergo state-dependent changes in amplitude and can have arbitrary autocorrelation[5] functions influencing their time evolution. An important aspect of this model is its hierarchical form, which induces top-down priors (predictions) into lower levels. Figure 1 illustrates these features for one hierarchical level.

The next key feature of Generalized Filtering is the representation of internal states and hidden parameters in a generalized coordinate system. That is, one where the coordinates describe the configuration of the system relative to some reference configuration under the constraint that the set of coordinates *uniquely* defines the configuration of the system relative to the reference configuration. For example, the position of the end of a pendulum can be defined in terms of a coordinate that represents the angle of the pendulum's arm from its stationary position, which is the reference configuration. This simplifies calculations about the arm's dynamics. Generalized Filtering uses generalized coordinates of *motion* where the coordinates include the values of the model variables, and the infinite set of the higher order derivatives of each variable's equations of motion. In this system a point can be viewed as encoding a variable's current trajectory since

[3] Surprisal is a measure of the information content associated with the outcome of a random variable.
[4] For a multivariate distribution, the precision takes the form of a precision matrix, which is the matrix inverse of the covariance matrix, if such an inverse exists.
[5] Autocorrelation is the correlation between a signal and itself at some offset in time as a function of the offset.

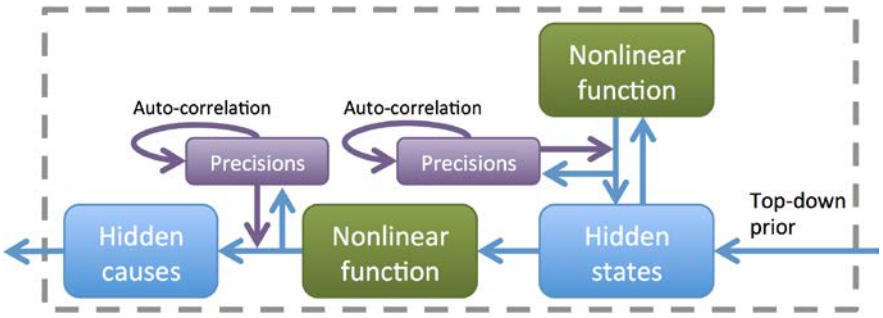

Figure 1. A single hierarchical level in Generalized Filtering. The dashed lines bound one level with the lower level to the left and the higher level to the right. The level's state variables are depicted by the blue boxes. Two non-linear functions and their parameters (green) govern both top-down and dynamical priors (or predictions). Two corresponding sets of precisions (purple) affect the hidden states and vice versa. The dynamics of precisions are governed by some autocorrelation function.

future variable values can be extrapolated using the higher order terms. Concretely, if *x* represents a set of state variables, then such a coordinate system is parameterized by $\tilde{x} = \{x, x', x'', \ldots\}$. With generalized coordinates of motion, instead of simply estimating the conditional density of hidden states (and parameters), one optimizes the conditional density on their generalized motion to an arbitrarily high order. In practice, this can be truncated to a relatively low order.

The third main feature of Generalized Filtering is a fixed-form, Gaussian, approximation to the internal probabilistic representations of the causes of sensory data. While the Gaussian distribution has several advantages, a critical one is that, when free energy is minimized, the conditional covariance of hidden states becomes an analytic function of the conditional mean. This implies that only the mean need be optimized since the covariance can be derived from it. It also simplifies the mathematics of model optimization to a single ordinary differential equation describing the motion of the conditional mean of states (Friston et al., 2010).

Given the above form of a generative model, the treatment of variables in generalized coordinates of motion, and Gaussian assumptions on the form of probabilistic representation of the causes of data, Generalized Filtering prescribes a set of ordinary differential equations, governing recognition dynamics, that update the conditional density on the model states (e.g. hidden state variables), parameters (e.g., weighted links), and hyperparameters (e.g., precision parameters). Next, we discuss these equations conceptually.

4.1 State Optimization via Prediction Error Minimization

The update equation for the model's hidden state suggests that instances of two kinds of computational units send and receive messages at each hierarchical level (Figure 2). *State units* (blue) encode the mean vector of the multivariate Gaussian distribution over unknown states generating sensory data, while *error units* (red) encode prediction error. The activity of each error unit is a function of state units (or the input), while state unit activity is a function of error units. State units provide top-down and temporal predictions to same-level and lower-level error units. Hierarchical inference requires only the prediction error from the lower level, which drives the conditional expectations toward a better prediction, so as to minimize the prediction error in the level below. This recurrent message passing between hierarchical levels and its processing optimizes free energy by minimizing prediction error, and constitutes perceptual inference. It is also known as (hierarchical) predictive coding (Rao & Ballard, 1999). Note that prediction errors can be based on top-down predictions (r^{TD} in Figure 2) or on temporal predictions (not shown). Temporal prediction error would be based on a temporal prediction and the actual future state.

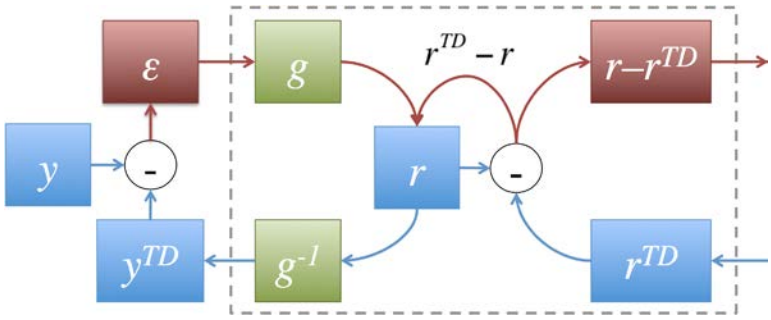

Figure 2. Hierarchical Predictive Coding. Each hierarchical level (dashed box) receives top-down predictions, r^{TD}, of the level's current representation, r, of the input signal, y. Feed-forward pathways carry errors (e.g., $r–r^{TD}$) between the predictions and the actual activity. Errors ($r^{TD}–r$) are used to correct the current representation, r, and, hopefully, future top-down predictions, y^{TD}. g represents a set of feedforward connection weights and g^{-1} a set of feedback connection weights. Redrawn from Rao & Ballard (1999).

4.2 Parameter Optimization via Associative Plasticity

Parameter optimization[6] may also reduce a model's free energy. Parameters modulate the effects of the nonlinear functions shown in Figure 1. In a computational network, a parameter would be some connection between units, and its value would be the weight of a connection. GF supplies parameter update equations similar to models of associative plasticity based on correlated pre- and post-synaptic activity. The update has two forms: The structural parameter update is illustrated in Figure 3a, and considers a currently active hidden variable at a given level (e.g., Level 2) as the connection's source, and a currently active bottom-up prediction error variable as the sink. Such an update changes future top-down predictions, to hopefully better minimize future bottom-up prediction error. Figure 3b illustrates a dynamical update. Initially, there was a state variable, p, active at time t from bottom-up prediction error. This update takes a state variable, q, at the same level whose activity preceded q's at time $t – 1$, and updates a dynamical connection from q to p. At a future time $t + n – 1$, the re-activation of q produces a temporal prediction of p (shaded), to hopefully minimize future prediction error. In either case, the parameter value is updated as a function of 1) the product of the q^{th} pre-synaptic input and post-synaptic response of the p^{th} error-unit, 2) the product of the parameter value and its function's associated precision, and 3) a value-dependent decay, e.g., sigmoidal decay (Friston & Feldman, 2010, p. 8).

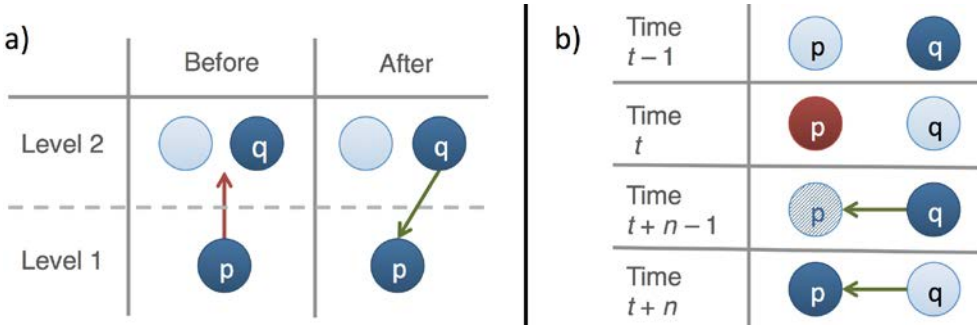

Figure 3. Structural and dynamical parameter updates. Both updates are performed in response to bottom-up prediction errors (red). a) A structural update modifies (or adds) a structural connection (parameter), from a higher to a lower level, which changes future top-down predictions, to minimize future bottom-up prediction error. b) A dynamical update modifies (or adds) a dynamic connection between hidden state variables in the same level changing future temporal predictions to also minimize future prediction error.

[6] In Bayesian statistics, parameters are continually estimated and updated; they are not assumed fixed.

4.3 Hyperparameter Optimization via Gain Control

GF also provides an update equation for the model's hyperparameters that estimate the precision of the mean encoded by a level's hidden states. Such precisions weight the magnitude of their associated prediction errors and are estimate from the state via prediction error (Figure 4). Precision parameters change as a function of 1) the precision-weighted sum of squared prediction error term, and 2) a decay in proportion to the current precision value. Such precision parameters are thought to correspond to *gain control*, the control of synaptic gain or post-synaptic responsiveness. Synchrony and neurotransmitters have been suggested as the neurobiological implementation of gain control (Friston & Feldman, 2010, p. 8).

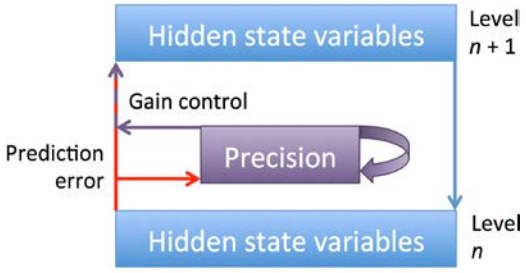

Figure 4. A single precision hyperparameter in a hierarchical predictive coding network. It is updated based on the sum of squared bottom-up prediction (red) and a value-dependent decay (purple arrow). It modulates the magnitude of the same prediction error signal from which it is estimated. Precisions can similarly exist for temporal prediction errors at each hierarchical level.

We have just described Generalized Filtering, an application of the free-energy principle to complicated, hierarchical dynamic uncertain statistical models yielding update equations for model states, parameters, and hyperparameters. We adopt GF as a theoretical guide for PC-CLA. Next, we review the LIDA cognitive architecture leading up to PC-CLA, which is proposed as a foundational building block for LIDA and provides detailed data structures and algorithms for LIDA's current representations, memory, and attention and learning processes.

5. The LIDA Model and its Cognitive Cycle

The LIDA model is a comprehensive, conceptual and computational model that covers a large portion of human cognition while implementing and fleshing out Baars' Global Workspace Theory (1988). Early versions of the model have seen successful real-world applications (Franklin, Keleman, & McCauley, 1998), while, more recently, the model has replicated some experimental findings (Faghihi, McCall, & Franklin, 2012; Madl, Baars, & Franklin, 2011; Madl & Franklin, 2012). The model and its ensuing architecture are grounded in the LIDA cognitive cycle. The cycle is based on the fact that every autonomous agent (Franklin & Graesser, 1997), be it human, animal, or artificial, must frequently sample (sense) its environment and select an appropriate response (action). A cognitive cycle can be thought of as a cognitive "moment." Higher-level cognitive processes are composed of many of these cognitive cycles.

LIDA hypothesizes a rich inner structure for its cognitive cycles (Franklin, Baars, Ramamurthy, & Ventura, 2005). The cycle (see Figure 5) begins with sensory stimuli from external and internal sources in the agent's environment. Low-level feature detectors in Sensory Memory begin the process of making sense of the incoming stimuli. These low-level features are passed on to Perceptual Associative Memory where higher-level features, such as objects, feelings, events, categories, relations etc. are recognized. These entities that have been recognized preconsciously make up the percept that passes asynchronously to the Workspace, where a model of the agent's current situation is updated. This percept serves as a cue to two forms of episodic

memory, transient and declarative. Responses to the cue, called local associations, consist of remembered events from these two memory systems that were associated with the various elements of the cue. Besides the current percept, the Workspace contains recent percepts and portions of the structures assembled from them that haven't yet decayed away.

A new model of the agent's current situation is assembled from the percepts, the local associations, and the undecayed parts of the previous model. This assembling process will typically require structure-building codelets. These are small, special purpose processors, each of which has some particular type of structure it is designed to build. To fulfill their task these codelets may draw upon Perceptual Associative Memory, to enable the recognition of relations and situations. They may also draw on the Conscious Contents Queue, which stores the conscious contents of the past few seconds. The newly assembled model forms the agent's understanding of its current situation within its world. It has made sense of the incoming stimuli.

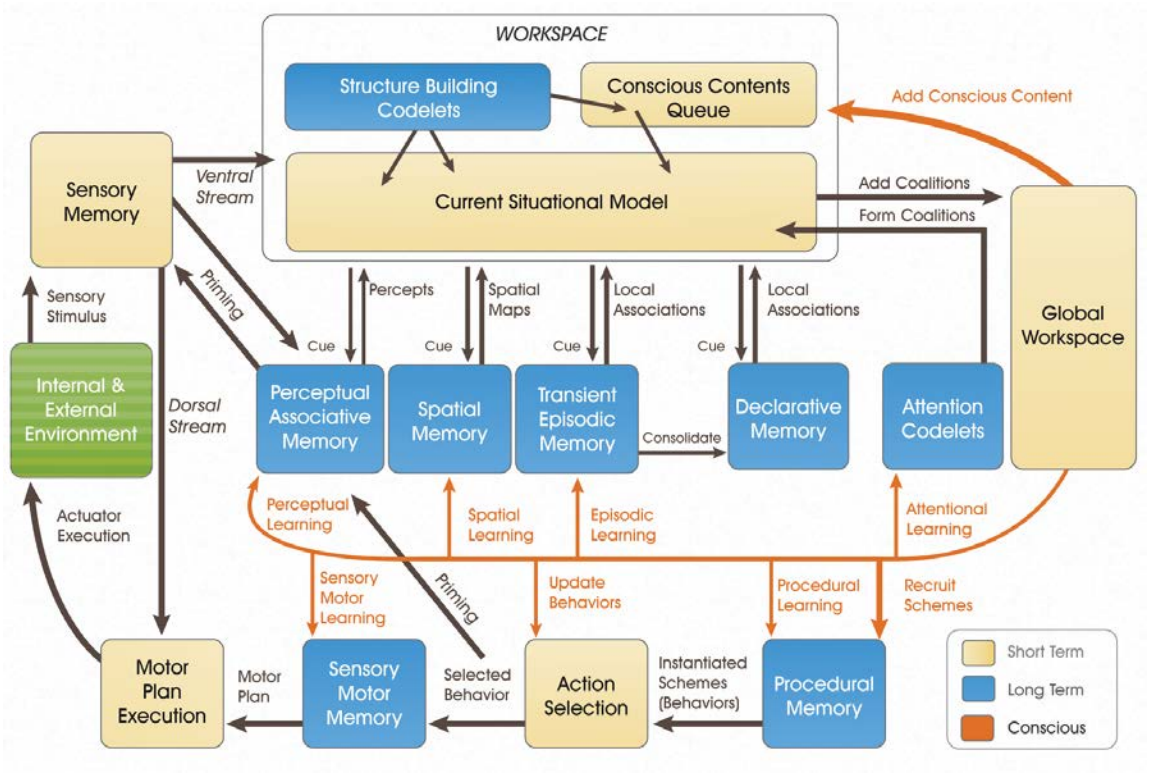

Figure 5. The LIDA Architecture and its Cognitive Cycle.

For an agent operating within a complex, dynamically changing environment, this Current Situational Model (CSM) may well be much too much for the agent to consider all at once in deciding what to do next. It needs to selectively attend to a portion of the model, the most salient portion. Portions of the CSM compete for attention. These competing portions take the form of coalitions of structures from the CSM. Such coalitions are formed by attention codelets, whose function is to try to bring certain structures to consciousness. When one of the coalitions wins the competition, the agent has effectively decided on what to attend.

One purpose of this processing is to help the agent decide what to do next. To this end, a representation of the contents of the winning coalition is broadcast globally. LIDA's multiple modes of learning all occur continually, simultaneously, and online using the contents of consciousness of each broadcast (Franklin & Patterson, 2006). Consciousness in LIDA refers to *functional* consciousness; no assumption is made with regard to phenomenal consciousness. While conscious contents are available globally, a primary recipient is Procedural Memory, which

stores schemes, which are templates of possible actions including their contexts and possible results. It also stores an activation value that attempts to measure, for each such template, the likelihood that an action taken within its context produces the expected result. Templates whose contexts intersect sufficiently with the contents of the conscious broadcast instantiate copies of themselves with their variables specified to the current situation. Instantiated templates remaining from previous cycles may also continue to be available. These instantiations are passed to the action selection mechanism, which chooses a single action from one of them. The chosen behavior then goes to Sensory-Motor Memory, where an appropriate algorithm, called a motor plan, executes it. The relevant actuators affect the environment, and the cycle is complete.

6. Cortical Learning Algorithms with Predictive Coding for LIDA

Generalized Filtering provides a general-purpose, biologically plausible, Bayes-approximate mathematical prescription for an advanced filtering network. We are interested in a computational model and see promise in the computational HTM Cortical Learning Algorithms (Hawkins, Ahmad, & Dubinsky, 2011), which integrate sparse distributed representation, clustering based on temporal coincidence, the representation of high-order temporal dynamics, and online learning in a single algorithm. However, not all of the features suggested by GF are implemented by the Cortical Learning Algorithms (CLA) including 1) hierarchical prediction coding, the optimization of states and structural (top-down) parameters based on prediction error minimization, and 2) precision hyperparameters. While we have discussed precision parameters (hyperparameters) theoretically we will not approach this aspect computationally in this work. Instead our aim is to extend the CLA by incorporating hierarchical predictive coding, an approach we term the Predictive Coding CLA (PC-CLA). This is a first step towards a long-term goal to develop a network with the capacity to process a broad range of sensory signals, estimating complex models of the form approached by Generalized Filtering from sensory data. While GF does not restrict the types of state variables used, PC-CLA, following CLA, uses binary variables.

Again, while PC-CLA is a data analysis algorithm, we also propose it here as a potential building block for cognitive systems. As with the CLA, the algorithm is recursive, and can be repeated in a large tree-structured hierarchy, possibly following Fuster's (2006) outline of two intertwined perceptual and executive memory hierarchies. We conjecture that, out of such a network, we will be able implement the larger, more global, more cognitive features typically addressed by LIDA and similar cognitive systems. Additionally, an agent built using a PC-CLA network would have the capacity to deal with high-dimensional "real-world" sensory data in a grounded manner (Barsalou, 1999). Currently, neither the CLA nor any other high-dimensional pattern recognition algorithm has been implemented within LIDA for its perception.

A number of aspects of PC-CLA correspond to previously conceived entities in LIDA: the notion of hidden state variables and their values corresponds to the current representations in the Current Situational Model of LIDA's Workspace. The long-term synapse-like connections of PC-CLA could potentially implement one or more of LIDA's memories including Perceptual Associative Memory, Procedural Memory, Declarative Memory, etc. PC-CLA's learning procedures are akin to LIDA's structure-building codelets, which build new nodes, links, and other representations in the Workspace. With this view, PC-CLA then suggests some general structure-building codelets for proposing and reinforcing links. Structural links connecting hidden units in neighboring hierarchical levels serve to implement memory for co-occurrence patterns, and are used in producing top-down predictions. Similarly, dynamical links connecting hidden units in the same hierarchical level are the foundations for temporal predictions.

If PC-CLA were to incorporate precision parameters, then their update could be performed by attention codelets concerned with the activity of prediction-error units and performing the aforementioned update operation to estimate precision. Previously in LIDA, attention codelets did not have such general concerns. Recall that Generalized Filtering suggested that precisions be

used to weight prediction errors. Feldman and Friston (2010) suggest that attention can be seen as the "selective sampling of sensory data that have high precision in relation to the model's predictions." In LIDA terms, this suggests that precision modulates the *salience* of current representations. It also suggests that top-down predictions also contribute to salience as they can create an attentional "bottleneck" by biasing particular representations, preventing others from being expressed. LIDA has defined attention as the process of bringing content to consciousness based on salience. PC-CLA then suggests that current salience involves: 1) prediction error that updates current representations, 2) precision that weights prediction error, and 3) both top-down and temporal predictions. Each is influential, in the short term, in determining which pattern, representing the cause(s) of the sensory input, is expressed.

7. Computational Implementation

Building from the HTM Cortical Learning Algorithms (CLA) described by Hawkins, Ahmad, & Dubinsky (2011), and incorporating the ideas of hierarchical predictive coding, we developed an initial implementation of the PC-CLA. The algorithm, like CLA, is defined recursively allowing for hierarchical arrangement whereby instances of the algorithm appear at each level, each sending and receiving Boolean vectors. At the first level, the original input is processed, while higher levels process the output of their immediately lower level. For best performance, input patterns should be, on average, as uniformly distributed throughout the input dimensions as possible. This constraint arises because the CLA is limited in its ability to produce more distributed representations than it receives, and insufficiently distributed representations do not enjoy the advantages of sparse high-dimensional spaces. "Natural" auditory and visual stimuli fit this distribution requirement quite easily, while, for human-generated data, a preprocessing step is often required to produce distributed inputs. While proposed for LIDA, due to PC-CLA's complexity, we first focus here on it as a stand-alone algorithm, not integrated with LIDA.

PC-CLA adds the following to the CLA model: 1) Multiple hierarchical levels, 2) predictive coding message passing involving the feed-forward passing of only prediction error between levels, and 3) Being initially interested in visual data streams, we focus on a 2D arrangement of receptive fields and internal elements.

7.1 Cortical Region

The description in this section applies to both the original CLA and PC-CLA unless otherwise noted. To start, the fundamental data structure of the CLA is called the *Cortical Region* (Figure 6). A Cortical Region consists of a set of *columns*, which are based on the functionality of cortical minicolumns. In this PC-CLA implementation, we focus on a square 2D column arrangement, although other arrangements are possible. Each column is comprised of multiple *cells* (defined below), and has one *proximal dendrite segment*, a functional approximation to a neuronal dendrite segment. The proximal dendrite segments integrate the activity of artificial *proximal synapses* (described later), implementing connections in close *proximity* to the column. Proximal synapses can transmit the input activity received by the region.

For each column, the input locations of its proximal synapses are modeled using a bivariate (2D) Gaussian distribution. Figure 7 depicts a single column, its proximal dendrite segment, and the segment's proximal synapses. The activity of the proximal synapses on a proximal dendrite segment determine its column's *overlap score*[7], a scalar measure of column activity from the bottom-up input. Each column also has a *boost* attribute that weights the overlap score and is based on the column's recent history of activity. Boost is used to make under-utilized columns more salient. Each column also has an *activity state*, which may take a value of *inactive*, not sufficiently active to compete with other columns, *competitive*, sufficiently active to compete

[7] The original CLA called both overlap score and boosted overlap score "overlap." Here we distinguish between them.

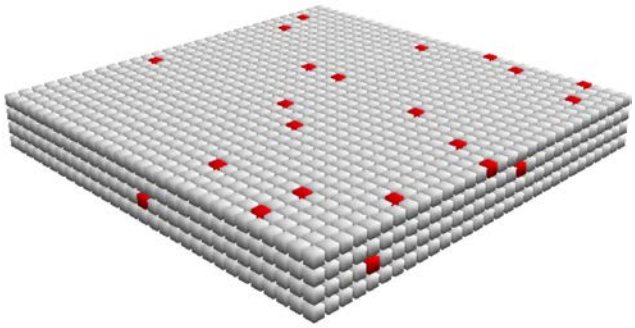

Figure 6. A single Cortical Region with 900 (30x30) columns, arranged in a 2D fashion, with each column having 4 cells. Typical implementations have 1024–2048 columns.

with other columns, or *active*, competitive and having won a competition with its neighboring columns to determine the most salient columns. A column's activity state is based on its boosted overlap score. For PC-CLA, the activity state also depends on the *predicted column activation*, a scalar measure of salience based on current temporal predictions (see below) of the column's activity. Predicted column activation ensures predicted columns remain competitive during an inter-column inhibition step (Step 2c below) since, in predictive coding, the column being predicted effectively inhibits its receiving bottom-up input. A predicted column must remain competitive because its cell(s) are likely encoding the current temporal context.

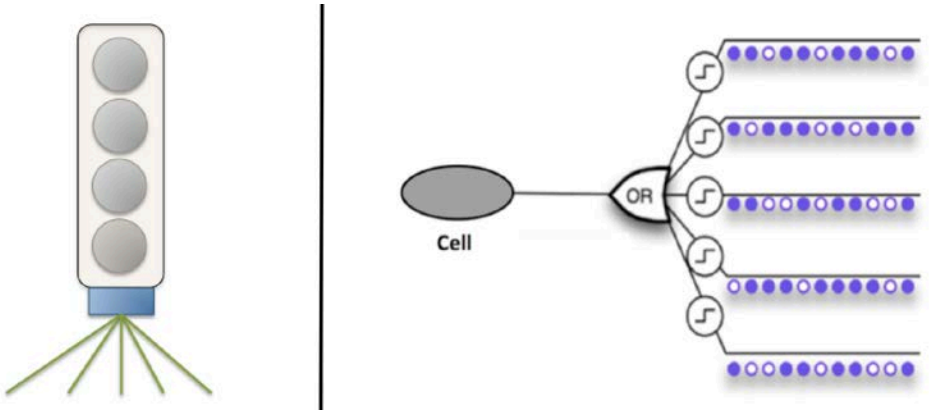

Figure 7. Left: A single column having four cells (gray). Each column has a single proximal dendrite segment (blue) that integrates the activity of many proximal synapses (green). Right: A single cell with five distal dendrite segments, which are depicted by the circles containing a step-function. Each segment has multiple distal synapses, shown as open and filled blue circles. Each circle represents a possible connection with a neighboring cell in the region. A cell becomes predictive whenever one or more segments become active, while a segment's activity depends on its synaptic activity (filled blue circles).

Moving on, a *cell* (gray circles in Figure 7 left) is a representational unit belonging to exactly one column. If a column is currently active, one or more of its cells may become active. Active cells constitute the current representation of the temporal context. Long-term information about context is encoded in CLA using a combination of *distal synapses* and *distal dendrite segments*. Each *distal synapse* is a synapse that has a cell as its source, and has, as its sink, a distal dendrite segment, which is connected to another cell, with the constraint that these two cells must be from different columns. *Distal dendrite segments* maintain a set of distal synapses, and integrate the synapses' activity. Distal dendrite segments also have an activation threshold, measured in the number of *active synapses* (explained shortly). If the number of active synapses is above this

threshold, the segment is considered active. Finally, each cell has multiple distal dendrite segments (Figure 7 right) whose activity, in part, determines the cell's state. Specifically, they determine whether a cell state is *predicted* to become active in the future. Alternatively, cell state may be *active* or *inactive*.

All synapses, proximal or distal, have a source, a sink, a binary weight, and a *permanence* (defined below). A *potential synapse* is one with a weight of 0, while a *connected synapse* is one with a weight of 1. A synapse is *active* if, and only if, its source is active *and* it is connected. Synapse weight is determined by *permanence*, a scalar attribute of all synapses modified during learning. If the permanence is above the *synapse connection threshold*, then the synapse is *connected* with a weight of 1. If the permanence is below this threshold, but above 0, then the synapse is *potential* with weight 0. Finally, if a synapse's permanence drops to 0 or less, then it is removed. In review, the source of a *distal synapse* is some cell in the Cortical Region, and the sink is a distal dendrite segment of a neighboring cell. In contrast, the source of a *proximal synapse* is a bit in the Cortical Region's input and the sink is the proximal dendrite segment of a column.

7.2 Predictive Coding Cortical Region Process

In this implementation of PC-CLA, each Cortical Region is driven by its own *Cortical Region Process*, which runs a serial cycle updating the region's state and performing learning. The Cortical Region Process can be roughly subdivided into two sub-processes, *spatial pooling* and *temporal pooling*, to reuse terminology from the original CLA description. Spatial pooling refers to the process of grouping similar inputs into the same (or nearly the same) sparse distributed set of active columns representing the input. Spatial pooling approximately corresponds to Step 2 in the detailed description below. Temporal pooling (Step 3) occurs sequentially after spatial pooling. It takes the active columns representation produced by spatial pooling, and the current temporal context encoded by the cells predicted in the previous cycle, and produces the current active cell state. From the current active cell state, the temporal pooler then produces a current predicted cell state, the context for the next cycle. Both the current active cell state and the current predicted cell state comprise the temporal pooler's output. Representing multiple time steps, the union of these two states ideally exhibits some temporal invariance, hence the name *temporal pooler*.

Predictive coding brings additional steps to the Cortical Region Process. Briefly, it requires us to compute and process the prediction error between 1) the top-down prediction of a Cortical Region and 2) the region's input, not to just process the original input. Additionally, the Cortical Region Process must incorporate top-down predictions with the Cortical Region's active cell representation. Compared to the original CLA, this corresponds to the addition of Steps 1, 4, and 5 (below) and includes modifications to Steps 2ab and 6a. We now summarize the main loop of the Predictive Coding Cortical Region Process (Figure 8) at an arbitrary cycle *t* as follows:

Step 1. Compute the current bottom-up prediction error, ε_v, between the current bottom-up Boolean input, *y*, and the previous cycle's top-down prediction, $\boldsymbol{y}^{TD}$.
Step 2. Compute the active columns of the Cortical Region for cycle *t*, *L1*.
a) Perform process *g* taking the bottom-up prediction error, ε_v, and the columns' proximal dendrites and associated proximal synapses, and outputting the columns' overlap score.
b) Add each column's (bottom-up) overlap score to its *predicted column activation*, a scalar measure of column activation from temporal predictions for the column for this cycle (computed in Step 5a of cycle $t - 1$), to obtain the *overall column activity*.
c) For columns with overall column activity greater than a threshold, perform a local k-winners-take-all procedure to determine the active columns, *L1*. The constraint, *k*, limits the number of possible active columns within a given area ensuring that the active columns are distributed.
Step 3. Compute the active cells at cycle *t*, *L2*, the current cells predicted to be active at *some*

future cycle, $PL2_t$, and their union, U.
a) Based on the active columns, $L1$ and the currently predicted cells, $PL2_{t-1}$ (computed in Step 3b of cycle $t - 1$), compute the current active cells, $L2$.
b) Based on the active cells, $L2$, and the region's distal dendrites and synapses, perform process, f, producing the region's current predicted (for some future cycle) cells, $PL2_t$. Based on only the cells predicted for the next cycle $t + 1$, determine the columns, predicted this cycle, to be active next cycle, $PL1_t$ (used later in Step 5).
c) Compute the union, U, of the active cells, $L2$, and the current predicted cells, $PL2_t$.

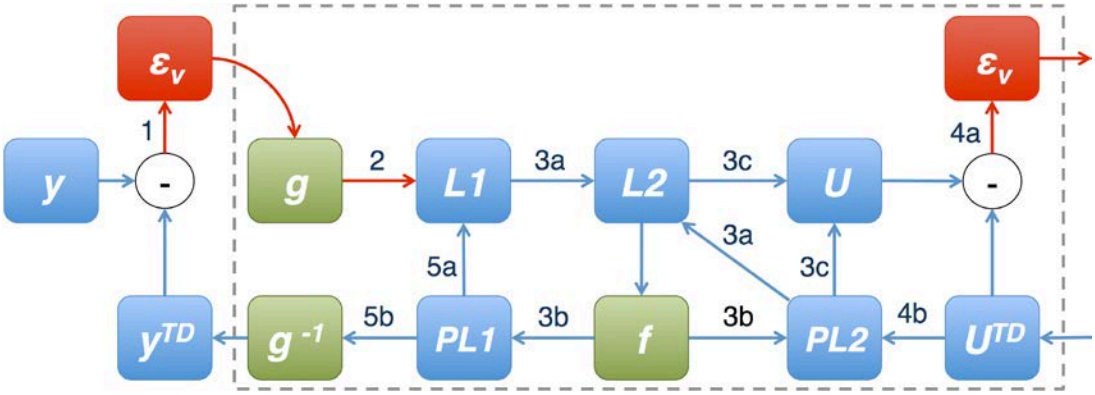

Figure 8. The predictive coding Cortical Region Process for a single hierarchical level with an input. All the components of single Cortical Region appear within the gray dashed box.

Step 4. Process the current received top-down prediction, U^{TD}.
a) Compute the error between U and the current received top-down prediction, U^{TD}, and send the error to the next hierarchical level.
b) Update $PL2_t$, the current cells predicted to be active at *some* future cycle, by adding in those cells predicted in U^{TD}.
Step 5. Based on the columns predicted to be active next cycle, $PL1_t$, (found in Step 3b):
a) Compute each column's predicted column activation (used in 2b of next cycle).
b) Perform process g^{-1} to generate the region's current top-down prediction, y^{TD}.
Step 6. Perform the learning processes.
a) Perform spatial learning, updating the permanence of proximal synapses based on bottom-up prediction error. Also update each column's boost attribute based on its activity history.
b) Perform temporal learning, updating the permanence of distal synapses, and possibly adding new distal synapses. Temporal learning is driven by both unpredicted columns and predicted columns that did not actually become active. We give more details of learning in the next section.

7.3 Learning

Her we further detail the learning processes of Step 6. In the original CLA, the spatial learning process iterates over each active column updating the permanence values of the column's synapses based on the synapse's input. Synapses with active inputs have their permanence incremented, while those with inactive inputs have their permanence decremented. This has the effect of tuning columns to be more selective to their current bottom-up inputs. In PC-CLA, an alternate version of spatial learning is used since the region's bottom-up input is an error signal. This process iterates over each column, not just active columns, strengthening all synapses connected to an active input (to minimize future false negative error), and weakens all synapses whose input was predicted, but did not become active (false negatives).

For boosting, there are two separate mechanisms to bias underrepresented columns. If a column's *active history*, the recent history of being an active column, is not a sufficient

percentage of its neighbors' active history, its boost attribute is increased. Increased boost makes it more likely for a column to win in the inhibition step and become an active column. Additionally, if a column's *competition history,* the recent history of whether its overall column activity was sufficient to enter the inhibition competition, is not a sufficient percentage of its neighbor's competition history, the permanence of each of its proximal synapses is increased.

For the temporal pooler, learning updates distal dendrites segments. For computational reasons, we introduce an attribute unique to *distal* dendrite segments called *prediction order*. The prediction order of a distal dendrite segment represents the number of cycles in the future in which the segment's sink is predicted to be active. Initially, the prediction order of all distal dendrite segments is one. Whenever the temporal learning process uses a segment in an attempt to add a temporal prediction with an order greater than one, the segment's prediction order is changed to the new order. Given this, the temporal learning can be summarized in three cases: 1) *New first-order* learning attempts to add a first-order temporal prediction (via synaptic modifications) for each unpredicted active column. In particular, based on the active cells from the previous cycle, the distal dendrite segment best predicting the column's activity is selected for learning in which the segment's synapses are positively updated, and new synapses, also predictive of the column's activity, are added to the segment. 2) *Active prediction* learning occurs for each active distal dendrite segment (determined in Step 3b). For each such segment a segment update is stored for processing at the cycle in which the prediction's validity can be verified, which is termed the segment's *verification time*. If this type of update is performed, new synapses are not added. 3) *Extending prediction* learning, like active prediction learning, involves currently predicting distal dendrite segments. For every such segment, *d1*, another distal dendrite segment, *d2*, which predicts the cell's activity *one cycle earlier* than *d1*, is selected and a segment update is stored for possible implementation. If performed, this type of update adds new synapses to bolster the newfound prediction of higher-order.

Not all segment updates are actually performed as we also wish to keep the number of connected distal synapses, and the predicted cells they produce, sparse. If a column is already well predicted by an existing first-order dendrite segment, it is not necessary, and likely detrimental, to update synapses to bolster a similar prediction. In order to enforce sparsity in the temporal pooler, one *learning cell* is always designated during each cycle for each active column. A cell is marked learning if 1) a sufficient number of learning cells predicted it last cycle or, 2) it was the most strongly predicted by distal dendrite segments.

Segment updates of type one are always performed the same cycle they are created and always positively update the segment's synapses. Updates of types two and three are processed at their verification time based on the state of their associated cell during that cycle. If the cell is active at the verification time, and is a learning cell during that cycle, the update is performed positively. This positively updates correct predictions concerning learning cells. If the cell is not learning, the update is not performed, and is discarded, keeping temporal learning minimal. Finally, if the associated cell is inactive at the verification time, then the update is performed and negatively modifies the segment's synapses. This weakens predictions that do not come true.

We have delved into the details of PC-CLA, which adds predictive coding message passing to CLA allowing the algorithm to be deployed hierarchically. We hypothesize that PC-CLA will be generally useful in implementing representations, memory, and processing in systems-level cognitive architectures.

8. Testing

Here we report on the results of initial tests of our implementation of PC-CLA as a stand-alone algorithm, not yet integrated within the LIDA architecture. The implementation is based on the ideas of the original CLA, but adds predictive coding and 2D receptive fields as mentioned. We use randomly generated 2D Boolean patterns, and sequences of such patterns, as an initial means

of obtaining data. One advantage of such patterns is that they are generic and introduce no bias. However, they may not produce patterns challenging to discriminate, since random patterns are likely to be quite different from one another.

While the CLA has been around for some years, the algorithm has not seen much in the way of published work reporting its abilities, or the effects of parameters. The two tests shown below constitute a representative sample of a larger body of tests that explore the spatial pooling operation (Steps 2, 6), temporal pooling (Steps 2, 3, 6), and the full PC-CLA. To our knowledge, hierarchy with the CLA has not been previously studied.

8.1 Noise Robustness Test

Recall that the spatial pooling operation attempts to produce a sparse distributed set of active columns representing the current bottom-up input. One benefit of sparse distributed representation is the noise robustness it affords. To determine *how* robust the CLA's active columns representations are to noise, we tested the effect of varying amounts of noise, from 0% to 50%, added to input patterns, on the inputs' resulting sets of active columns. We also varied the *input activity*, or the percentage of bits in the input patterns that were true, from 1% to 5%, since the CLA works best with, and tends to produce outputs having, around a 2% true bit rate.

Concretely, for each input activity condition, *i*, a single Boolean input, with 256 dimensions, was generated with *i* percent of its bits randomly set to true. Next, in a developmental period, the input was shown to a Cortical Region performing just the spatial pooling operation for 250 cycles. After this period the final set of active columns was recorded. Then, for a range of added noise amounts, the original input pattern was corrupted by that amount. Noise was added uniformly to true bits and false bits alike, turning the former false and the latter true. For a particular input activity and noise amount, 500 trials were performed, each generating a noisy version of the input, showing it to the same Cortical Region for a time, and comparing the "noisy" set of active columns with the original set. We computed the normalized taxicab distance between the two sets, which, here, is the total number of errors in the active columns representation, both false positives and false negatives, divided by total number of columns in the region. The scores were averaged across all trials for a given experimental condition. Figure 9 summarizes the results of this test.

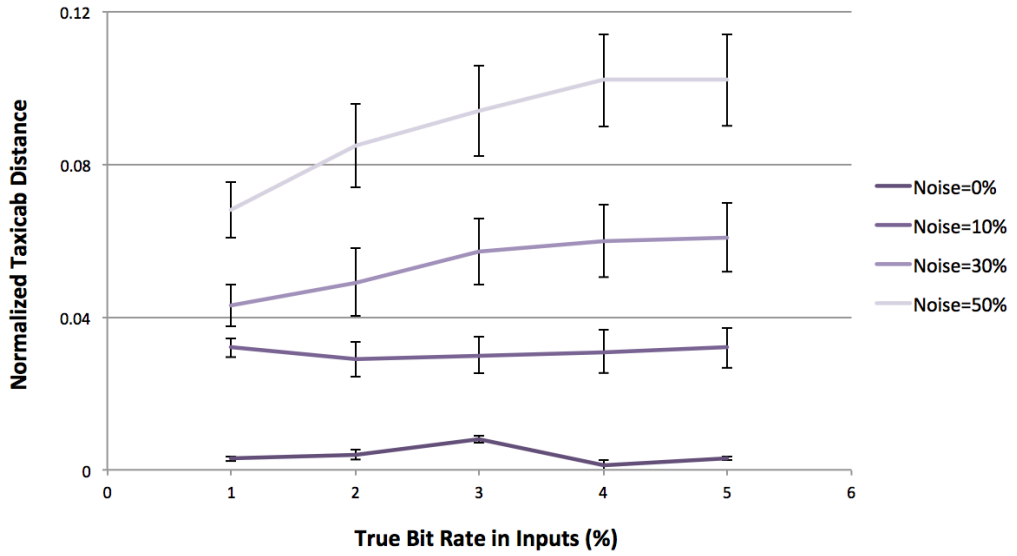

Figure 9. The average normalized taxicab distance, between the original active columns representations of inputs and noisy versions of the same inputs, as a function of the rate of true bits in the inputs. The legend to the right shows some different noise conditions. We identify noise robustness in these results, e.g., the

addition of 10% noise yields less than 4% error in the active columns representation, while 50% noise gave about 10% error or less. Also, the data suggest a benefit in keeping the true bit rate low, which is especially true for higher amounts of added noise.

8.2 Two Hierarchical Levels Test

This test looks at the effects of having top-down predictions from a higher Cortical Region influence the cell activity of a lower region (Step 4b). We compare a two-level two-region network in which the higher region sends influencing top-down predictions into the lower level, with the same network that doesn't send top-down predictions. We run the network on sequences of 2D Boolean patterns, each pattern having 529 dimensions, 2% of which are randomly set true.

In both conditions, we first create two Cortical Regions with the same parameters except for the *columns per input* parameter. While the lower Cortical Region had 4 times as many columns as inputs while the second Cortical Region had 1/*cellsPerColumn* as many to compensate for the fact that the first Cortical Region's output is *cellsPerColumn* times greater than its input.

In each of 100 trials, a sequence of 2D patterns having length 8 was generated. Then, in a developmental period, the sequence was shown 200 times to the network. Both regions performed spatial and temporal pooling operations either with or without top-down predictions. We controlled for the effects of the Cortical Regions processing prediction errors by having both regions process only their respective input. The effect of processing prediction errors is the subject of another test.

After the developmental period, the regions' current states, which encode context, are cleared, and the same sequence is again shown to the network, in the same manner as just described, except with learning turned off. For this test presentation, for each pattern in the sequence (except the first), the first-order temporal prediction accuracy of the active columns is recorded for both regions in terms of the F-score. Similarly, we also assess the top-down prediction accuracy of each region with respect to its input at each step in the sequence. Across each sequence and all sequence trials these two measures were averaged. We measure the accuracy of both top-down and temporal predictions using the *F-score* measure (van Rijsbergen, 1979) with beta[8] set to 10. Figure 10 summarizes the results.

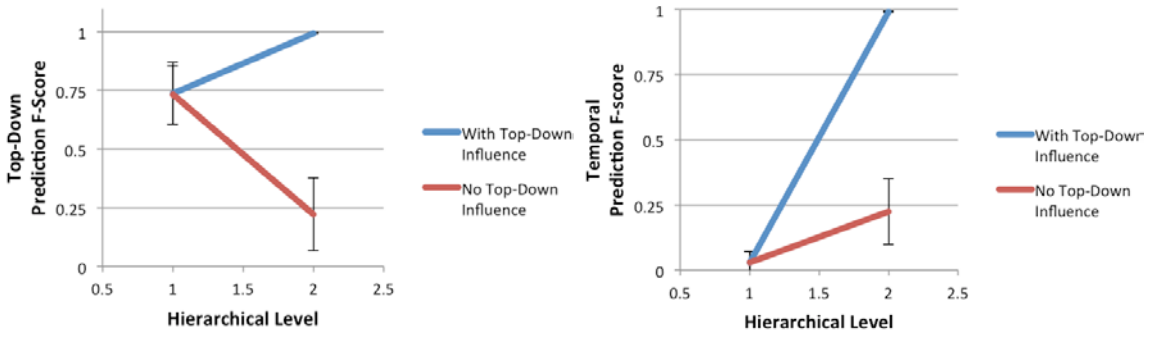

Figure 10. The left graph plots the average F-score of the accuracy of top-down predictions as a function of hierarchical level. Top-down influence did not affect the first region's accuracy, but significantly improved the second. The right graph is similar except the dependent variable is the temporal prediction accuracy of a given level. Again the top-down influence did not change accuracy for the first level but did for the second.

Conclusions

In the future, once well understood, PC-CLA must be extensively tested on real-world data

[8] The F-score measures the effectiveness of retrieval with respect to a user who attaches *beta* times as much importance to recall as precision. This implies we tolerate false positives 10 times more than false negatives.

streams including visual, auditory, etc. It must also be studied on a larger scale with more than two hierarchical levels, and with high-dimensional input. Future work includes the addition of precision estimation to model the uncertainty in information sources based on accumulated prediction error. Implementing a cognitive system like LIDA with a PC-CLA network brings additional challenges, such as interfacing a PC-CLA network with Sensory-Motor Memory for action execution, and developing methods to build in preferences for goal-directed agents.

We identified several guiding principles on the nature of perceptual representation, perceptual inference, and the associated learning processes. Guided by these principles, in particular the free-energy principle, we presented a predictive coding extension to the HTM Cortical Learning Algorithms, termed PC-CLA. We propose PC-CLA as a potential building block for the systems-level LIDA cognitive architecture that fleshes out LIDA's internal representations, memory, learning and attentional processes, and takes an initial step towards the comprehensive use of distributed and probabilistic (uncertain) representation throughout the architecture. Finally, we presented some results of initial tests of the algorithm.

Acknowledgements

The authors would like to thank Tamas Madl and Pulin Agrawal as well as the anonymous reviewers for their useful feedback.

References

Arel, I., Rose, D., & Coop, R., (2009). DeSTIN: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. *Proceedings of AAAI Workshop on Biologically Inspired Cognitive Architectures.*

Baars, B. (1988). *A Cognitive Theory of Consciousness*. Cambridge: Cambridge Univ. Press.

Bar, M. (2009). The proactive brain: memory for predictions. *Philosophical Transactions Royal Society B: Biological Sciences, 364*(1521), 1235–1243, doi:10.1098/rstb.2008.0310.

Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Science, 22*(04), 577–660.

Bedny, M., Pascual-Leone, A., Dodell-Feder, D., Fedorenko, E., & Saxe, R. (2011). Language processing in the occipital cortex of congenitally blind adults. *PNAS, 108*(11), 4429–4434.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning, 2*(1), 1–127.

Bitzer, S., & Kiebel, S. (2012). Recognizing recurrent neural networks (rRNN): Bayesian inference for recurrent neural networks. *Biological Cybernetics, 106*(4–5), 201–217.

Douglas, R. J., & Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci., 27*, 419–451.

Faghihi, U., McCall, R., & Franklin, S. (2012). A Computational Model of Attentional Learning in a Cognitive Agent. *Biologically Inspired Cognitive Architectures, 2*, 25–36.

Feldman, H., Friston, K. (2010). Attention, Uncertainty, and Free-Energy. *Frontiers in Human Neuroscience, 4*(215).

Felleman, D. J., & van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex, 1*(1), 1–47.

Franklin, S., A. Kelemen, & L. McCauley. (1998). IDA: A Cognitive Agent Architecture. *IEEE Conference on Systems, Man, and Cybernetics*, IEEE Press.

Franklin, S., Baars, B., Ramamurthy, U., & Ventura, M. (2005). The Role of Consciousness in Memory. *Brains, Minds, and Media, 1*, 1–38.

Franklin, S., & Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Intelligent Agents III* (p. 21–35). Berlin: Springer Verlag.

Franklin, S., & Patterson, F. (2006). The LIDA Architecture: Adding New Modes of Learning to an Intelligent, Autonomous, Software Agent. *Proceedings of IDPT-2006*.

Franklin, S., Strain, S., McCall, R., & Baars, B. (2013). Conceptual Commitments of the LIDA Model of Cognition. *Journal of Artificial General Intelligence, 4*(2), 1–22.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience, 2*, 127–38.

Friston, K., Stephan, K., Li, B., & Daunizeau, J. (2010). Generalised Filtering. *Mathematical Problems in Engineering*, Article ID 621670, 34 pages, doi:10.1155/2010/621670

Fuster, J. (2006). The cognit: A network model of cortical representation. *International Journal of Psychophysiology, 60*, 125–132.

Goertzel, B. (2012). Perception Processing for General Intelligence: Bridging the Symbolic/Subsymbolic Gap. In *Artificial General Intelligence* (pp. 79-88). Springer Berlin.

Hawkins, J., Blakeslee, S. (2004). *On Intelligence*. New York, NY: Henry Holt.

Hawkins, J., Ahmad, S., & Dubinsky, D. (2011). Hierarchical Temporal Memory including HTM Cortical Learning Algorithms. Retrieved from www.numenta.org

Hinton, G., McClelland, J., & Rumelhart, D. (1986). Distributed representations. In D. Rumelhart & J. McClelland (Eds.), *PDP: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, (77–109). Cambridge, MA: MIT Press.

Kanerva, P. (1988). *Sparse Distributed Memory*. Cambridge, MA: The MIT Press.

Knill, D., & Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences, 27*(12), 712–719.

Kording, K. & Wolpert, D. (2004). Bayesian integration in sensorimotor learning. *Nature, 427*, 244–247.

Lee, T., & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A, 20*(7), 1434–1448.

Linsker, R. (1990). Perceptual neural organization: some approaches based on network models and information theory. *Annual Review Neuroscience, 13*, 257–281.

Madl, T., Baars, B., & Franklin, S. (2011). The Timing of the Cognitive Cycle. *PLoS ONE, 6*(4), e14803.

Madl, T., & Franklin, S. (2012). A LIDA-based model of the attentional blink. *ICCM 2012 Proceedings, 283*.

Métin, C. & Frost, D. (1989). Visual responses of neurons in somatosensory cortex of hamsters with experimentally induced retinal projections to somatosensory thalamus. *PNAS, 86*, 357–361.

Mountcastle, V. (1978). An Organizing Principle for Cerebral Function: The Unit Model and the Distributed System, In G. Edelman & V. Mountcastle, (Eds.), *The Mindful Brain*, Cambridge, MA: MIT Press.

Olshausen, B. & Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature, 381*, 607–609.

O'Reilly, R., Munakata, Y. (2000). *Computational explorations in cognitive neuroscience*. Cambridge: MIT Press.

Phillips, W. A., & Singer, W. (1997). In search of common foundations for cortical computation. *Behavioral and Brain Sciences, 20*(4), 657–683.

Rao, R., & Ballard, D. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience, 2*(1), 79–87.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience, 2*(11), 1019–1025.

Van Rijsbergen, C. (1979). *Information Retrieval*, Retrieved from: http://www.dcs.gla.ac.uk/Keith/Preface.html