The Context-Aware Learning Model: reward-based and experience-based Logistic Regression Backpropagation

Joohee Suh
Robotics, Evolution, Adaptation, and Learning Laboratory
School of Computer Science
Gallogly College of Engineering
University of Oklahoma
Norman, OK 73019–1101
Email: joohee.suh@ou.edu

Dean F. Hougen
Robotics, Evolution, Adaptation, and Learning Laboratory
School of Computer Science
Gallogly College of Engineering
University of Oklahoma
Norman, OK 73019–1101
Email: hougen@ou.edu

***Abstract*— To deal with uncertain environments, autonomous agents need to be able to learn without supervision. However, reward-based interactive learning often exhibits limitations handling both generalizations and exceptions. For these reasons, this research introduces the Context-Aware Learning Model (CALM) and two different learning algorithms. CALM-rLRB combines logistic regression backpropagation in artificial neural networks with hyperbolic reward-based learning. CALM-eLRB adds an empirical knowledge base that enables experience-based learning. CALM is evaluated using four metrics on six synthetic data sets and shows promising performance.**

I. Introduction

To be truly autonomous, a system needs to be able to learn without supervision. To be an efficient and effective autonomous learner, the system must quickly generalize from scarce experiences so as to avoid unnecessary errors. At the same time, the system must allow for distinctions to be learned in environments in which exceptions or data commingling exist.

A. Overview

The Context-Aware Learning Model (CALM) is inspired by context-aware computing, where responses should differ based on circumstances [1]. We define *context* as any combined, compressed, or encoded information that can be reasonably used as input for an embedded artificial neural network (ANN). In a robot, for example, a context might be based on currently sensed data or a time history of such data. CALM is a general learning model using such contexts. This paper describes the first implementation of this model.

Supervised learning, such as logistic regression backpropagation (LRB) in ANNs, is used to allow a system to generalize from known training data to similar data in its deployment environment [2]. Reinforcement learning and other reward-based learning methods allow a system to learn from interactions with its environment so long as there is a mechanism for evaluating performance in that environment [2], [3]. Combining these learning methods can provide an interactive system that can generalize its neural weights based on current context and can dynamically adapt to new environment that might change in the future. This is the basis of the CALM with reward-based Logistic Regression Backpropagation (CALM-rLRB).

However, ANN weights are an amalgam of initial (generally random) weights and the adjustments made to them throughout the learning process—there are no discrete "memories" of any particular learning event, or experience, to be found. An agent's *experience* is a combination of (1) a given context, (2) the response in that context, and (3) the evaluative feedback received as a result of the response given. An experience-based knowledge-base (EKB), similar to the hippocampus in vertebrates, could record experiences and recall them when the agent encounters a new situation. Using this EKB to bolster learning transforms CALM-rLRB to its experience-based counterpart CALM-eLRB, which is an advanced version of CALM-rLRB that optimizes its neural networks based on its accumulated experiences as inspired by the hippocampus.

B. Related Work

Artificial neural networks (ANNs) are computational models inspired by biological neural systems with a wide variety of topologies and adaptation mechanisms [2]. One biological inspiration is that of *Hebbian plasticity*, which is based on the discovery that "when a presynaptic neuron repeatedly participates in firing of a postsynaptic neuron, the strength between pre- and postsynaptic neurons increases" [4].

Reward-based Hebbian plasticity adds rewards from sources external to the pre- and post-synaptic neurons. It is known that additional chemical signals affect synaptic changes, which is the basis of a modulated Hebbian model [5], [6]. From a neurobiological perspective, neurotransmitters play important roles in changing synaptic plasticity and some are related to reward information so that the short-lived synaptic association between pre- and post-synaptic neuron can be retained for a longer time. In this case, *modulatory Hebbian learning* is a

kind of reward-based Hebbian learning where a reward value (e.g., ± 1) is applied as a modulatory signal.

In Hebbian and reward-based Hebbian learning, the value of each postsynaptic node can be increased or decreased infinitely; *reward-based hyperbolic Hebbian learning* avoids this by using a hyperbolic tangent activation function [6].

Numerous approaches for combining ANNs or other generalization mechanisms with some form of reward-based learning, often for robotic applications, are common in the literature but no single architecture has emerged as the default approach and many open questions remain [7]–[13]. Indeed, questions as fundamental as whether to use model-based or model-free methods are still strongly debated [9], [13].

In the area of cognitive robotics, there are memory-related approaches for adaptive and autonomous learning. These can be classified into three approaches. First, animal behavioral learning is embodied based on instrumental learning or conditioning learning [14]. Second, in the neurobiological approach, the memory in the learning model is based on the hippocampus [15]. Third, in the knowledge-based approach, memory has a structure such as a frame so that a controller can infer appropriate behaviors [16]–[19]. These approaches show how a robot can be successfully controlled by inferring or shaping its own memory. The primary distinction between these and CALM is that CALM does not start from prior knowledge—it shapes its EKB from scratch.

II. System Architecture

CALM consists of 10 components: (1) Sensory System, (2) Context Supplier, (3) CALM-ANN, (4) Motor System, (5) Observer, (6) Reward Policy Storage, (7) CALM-Learner, (8) Learning Rule Storage, (9) Experience-based Knowledge Base (EKB), and (10) EKB Manager. Figure 1 gives the overall system structure. The EKB and EKB Manager are optional components that are not used in CALM-rLRB and thus are shown using dashed lines.

The *Sensory System* gathers input data directly from sensory devices such as cameras. The *Context Supplier* processes sensory data through feature scaling, input encoding, and/or data compression. Input encoding and feature scaling reduce computational complexity and promote efficient learning [20].

The *CALM-ANN* is a layered, feed-forward, fully-connected ANN that learns using LRB. On each time step, it takes a context as input and the output node with the highest activation value is the selected response.

The *Motor System* takes the selected response and carries out the corresponding action in the environment.

The *Observer* computes the reward value based on the environmental state resulting from the action of the Motor System. This value is based on the reward policy saved in the Reward Policy Storage. Note that it is also possible for the Observer to get reward values by directly interacting with other sources such as a trainer.

The *Reward Policy Storage* maintains the *reward policy*, which is a mapping from (action, effect) pairs to reward values based on the expected effects of executing each behavior. For example, in robotics each behavior could have its own goal in affecting the environment. If the robot executes GO_FORWARD, it is expected that the position of the robot will change. In this case, a reward policy rule for GO_FORWARD could be: If position changed, reward $r \leftarrow +1$; otherwise $r \leftarrow -1$. This would reward the robot for recognizing that the current context, based at least in part on current sensory data, allowed the robot to move forward (e.g., it moved forward when its path was unobstructed) or penalize the robot if it failed to recognize the opposite (e.g., it attempted to move forward when its path was blocked).

The *CALM-Learner* adjusts the CALM-ANN based on weight update rule(s) in Learning Rule Storage. Also, it utilizes the EKB if its algorithm utilizes experiences. The details of the CALM algorithms are described in their own sections.

The *Learning Rule Storage* contains the weight update rules used by the CALM-Learner. Having the storage separate from the learner allows learners to be developed that reuse rules used by other learners.

The *EKB* stores past positive experiences (those for which the Observer assigned a positive reward). The EKB serves the same functions as the Empirical Context-based Knowledge Base in our previous work [21]. Note that CALM-rLRB is not experience-based so it does not use the EKB.

The *EKB Manager* is the interface of the EKB, which carries out three major tasks: (1) it stores rewarding experiences in the EKB, (2) it retrieves experience information from the EKB, and (3) it performs knowledge optimization by reducing knowledge redundancy. *Knowledge redundancy* occurs when there are experiences that are almost alike. Knowledge redundancy can be avoided by saving one composite experience to represent multiple similar experiences. Mechanisms for accomplishing this will constitute future work.

III. CALM Learning

CALM uses "quasi-target outputs" to optimize neural weights instead of depending on a teacher saying which output is correct for a given input. A *quasi-target output* is the desired output inferred by CALM during online learning based on the action taken and reward received.

Quasi-target output is similar to target output in supervised ANN learning; both are used to adjust the ANN weights to produce desired outputs. However, there are distinctions between them. First, in supervised learning, target output comes from labeled training data—the "right answers" for the given input given from outside the learning process. Quasi-target output is inferred by the learning model during the learning process. Second, in supervised learning, target output can in general have any combination of values while quasi-target output values are restricted based on the inference method used, as described after the basic learning steps.

The basic steps of any LRB CALM system are (1) take input $context(t)$, (2) do forward propagation, (3) select the output node with the maximum output, this node is denoted $son(t)$,

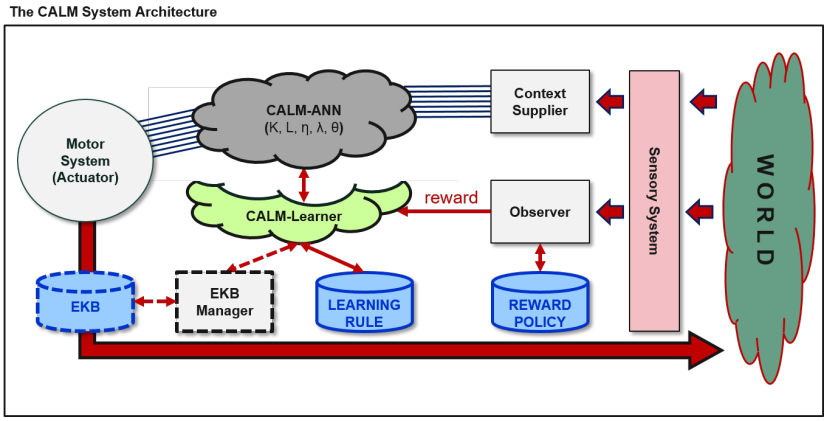


Fig. 1: CALM System Architecture

(4) perform the action corresponding to $son(t)$, (5) get reward value $r(t)$ from the Observer, (6) generate quasi-target output $R(t)$ based on reward value $r(t)$, (7) set input $LRBX(t)$ and desired output $LRBY(t)$ for LRB, and (8) perform LRB by taking $LRBX(t)$, $LRBY(t)$, and current neural weights $\theta(t)$ as inputs. In this last step, the outputs are the updated $\theta(t+1)$ and a cost function value $J(t)$ that measures the error between the desired and actual outputs; these are calculated at the end of some number of iterations ITR of the LRB process. $\theta(t+1)$ is used in the next learning step with $context(t+1)$; $J(t)$ is used for only evaluation (see Section V-B).

Here, quasi-target output $R(t)$ is generated in one of two ways depending on reward $r(t)$.

First, if $r(t) = 1$, the $son(t)$ is a good choice for $context(t)$. In this case, the element of the quasi-target output vector that corresponds to $son(t)$ is set to 1 (that is, $R_{son(t)}(t) \leftarrow 1$) and the other elements of the quasi-target output vector are set to 0 (that is, $R_k(t) \leftarrow 0$ where $k \neq son(t)$). The implications here are that if the agent earned a reward by taking an action in this context, then that action was correct and should be repeated in the same or similar contexts in the future *and* that all of the other actions were incorrect and should be avoided in the same or similar contexts in the future.

Second, if $r(t) = 0$, the quasi-target output value corresponding to $son(t)$ is set to 0 ($R_{son(t)}(t) \leftarrow 0$) while the others are set to 1 ($R_k(t) \leftarrow 1$ where $k \neq son(t)$). This quasi-target output vector implies that the action performed was a bad choice for the current context (so it should be avoided) and other actions are likely to be better choices (so they should have increased opportunities to be selected in the future).

After setting the quasi-target output $R(t)$, CALM generates learning input $LRBX(t)$ and learning output $LRBY(t)$ to optimize the ANN with logistic regression backpropagation. Note that each algorithm in CALM has its own way of generating $LRBX(t)$ and $LRBY(t)$.

A. CALM-rLRB

CALM-rLRB is the simplest CALM implementation. In CALM-rLRB, $LRBX(t)$ is simply set equal to $context(t)$ and $LRBY(t)$ is set equal to $R(t)$. This provides a simple baseline to which to compare innovations.

B. CALM-eLRB

The primary innovation of CALM-eLRB is to integrate past positive experiences into the optimization process as well as using the current context and reward information. Recall that for learning CALM-rLRB uses the current $context(t)$ as its input vector $LRBX(t)$ and the quasi-target output vector $R(t)$ as the desired output vector $LRBY(t)$. CALM-eLRB saves its positive experiences into its EKB then uses $context(t)$ along with prior context vectors saved in the EKB to create an input matrix $LRBX(t)$ and, correspondingly, $R(t)$ along with prior quasi-target output vectors saved in the EKB to create an output matrix $LRBY(t)$.

This means that each time CALM-eLRB receives a new experience, it not only adjusts its ANN weights to try to accommodate what it has just learned (as with CALM-rLRB) but actively cycles through all of its past positive "memories." The hypothesis here is that using these memories will improve system performance in static environments by ensuring that the system doesn't forget what it already knew.

Figure 2 helps us to visually understand the overall steps of CALM-eLRB; the difference from CALM-rLRB is the EKB that saves rewarding experiences.

The steps followed by CALM-eLRB are: (1) take $context(t)$, (2) do forward propagation, (3) select the output node with maximum value as $son(t)$, (4) take action corresponding to $son(t)$, (5) get reward $r(t)$ from Observer examining world, (6) generate quasi-target output $R(t)$ based on $r(t)$, (7) set $LRBX(t)$ and $LRBY(t)$ for LRB based on $context(t)$, $R(t)$, and saved experiences in EKB, (8) optimize neural weights by calling CALM-LRB-CORE-GEN, which performs logistic regression optimization with gradient descent weight update rule by taking $LRBX(t)$, $LRBY(t)$, and current neural weights $\theta(t)$ as an input, and (9) saves current experience if its reward value is positive. More specifically, in this last step, the EKB Manager takes necessary learning values to generate synthetic information ($context(t)$, $son(t)$, and $r(t)$) as the definition of experience.

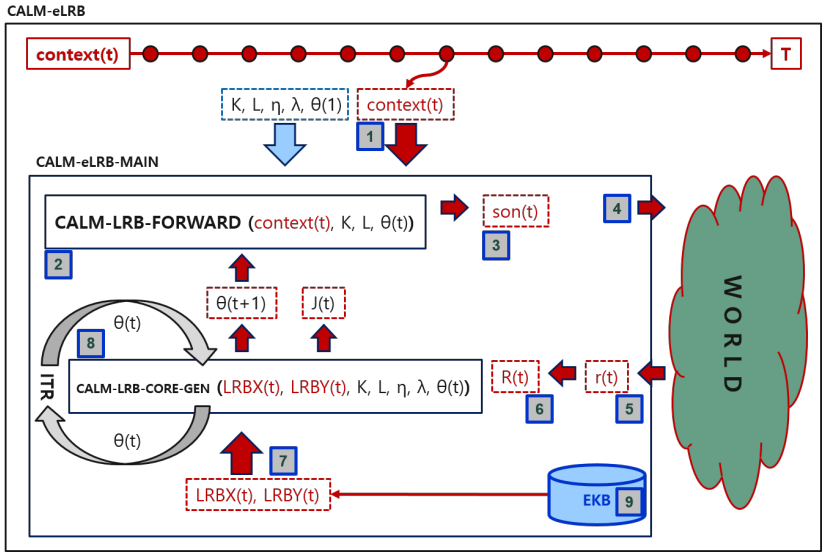


Fig. 2: CALM-eLRB Algorithm Diagram

Note that, compared to CALM-rLRB, only step (7) and step (9) are different whereas the other steps are the same.

IV. Experimental Setup

We test CALM-rLRB and CALM-eLRB using ANNs with depths ranging from 1 to 5 layers of weights. These 10 learning systems are each tested on 6 synthetic data sets with 200 to 980 data points in different distributions. All data sets have 2D "sensory" input contexts and 1D target outputs with 7 possible output values. In sets Data 1 to Data 4, the data is clustered into 7–14 clusters, meaning some sets have 2 clusters of points for the same output value. Different sets have more or less overlap between clusters to test performance with more or less data commingling. Data 5 has no clusters—the data is entirely random to give a validation baseline (no learning algorithm can find a pattern where none exists). In Data 6, the data is clustered but after all the data are presented to the system, the output value for each cluster changes and the data is presented to the system a second time. This simulates a dynamic environment in which there is an abrupt environmental change.[1]

On each learning step t, the Sensory System takes a different data point at random from the data set as an input/output pair. Therefore, the number of points in a data set signifies the maximum learning step T and the data is not sequentially ordered by target output to simulate an unpredictable environment.

Note that the target output value of a pair is not provided as a typical target output for supervised learning. Rather, if the ANN's selected output node is the same as the given target output, the Observer gets a reward $r(t)$ of 1, otherwise $r(t)$ is 0. This simulates reward-based learning in an interactive (e.g., robotic) environment.

Given that each data point is presented only once, a state-based reinforcement learner would do no better than random guessing, selecting the correct output and earning reward on average once every 7 steps, since there are 7 possible outputs. This gives a baseline to which to judge system generalization.

In order to increase the reliability of the results, each algorithm is evaluated on each data set (except for Data 6) using 5-fold cross-validation with the ratio of 80% training and 20% testing. Learning parameters are: $\eta = 1.0, \lambda = 0.0, \epsilon = 0.3, \gamma = 0.5$, and $ITR = 200$.

V. Results

This section shows experimental results on four evaluation metrics: (1) accuracy, (2) cost, (3) accumulated rewards, and (4) dynamic accuracy. Most static results focus on Data 4 as it has the most points, the most clusters, and the most overlap between clusters, making it the most challenging static data set. Dynamic accuracy uses the dynamic data set Data 6.

A. Accuracy

Accuracy measures correct classifications of contexts during learning on static data sets.[2] Accuracy is measured on both training and testing data.

Training accuracy is measured each learning step as the percentage of experienced contexts that are correctly classified. Note that over learning steps, the number of processed contexts increases since CALM iteratively takes one context at each step. For example, if the current learning step is 10 ($t = 10$), this means CALM processed 10 contexts and training accuracy at this learning step checks how many of the past 10 contexts the current ANN can correctly classify; therefore if at learning step 10 it can successfully classify 8 past contexts, the accuracy is 80% at that learning step.

[1] Further details of these data sets can be found elsewhere [22].

[2] Note that this could be referred to as "static accuracy" but we use the term "accuracy" for brevity. We use "dynamic accuracy" explicitly for the dynamic counterpart.

On the other hand, *testing accuracy* is acquired by applying the ANN with its current weights to the whole test set on each learning step.

Accuracy gives us an indicator of the degree to which the system has learned the particular data points presented thus far (training accuracy) and how well it can generalize to untrained data (testing accuracy). Note that presentation of the data to the ANN for measuring accuracy does not change the learned weights.

Table I shows accuracy results on Data 1 to Data 4 and Figure 3 shows the accuracy of the CALM algorithms on Fold 1 of Data 4. Results from training and testing show similar trends across all folds of all data sets, which points to the reliability of the learning process, although the systems do better on data sets with fewer clusters and less overlap between clusters, and they're unable to find useful patterns in Data 5, as one would expect.

The results make clear that CALM-rLRB with 2 weight layers learns useful generalizations. However, it struggles with Data 4, the most difficult data set, and its exclusive reliance on current data causes erratic shifts in learned responses. In contrast, CALM-eLRB learning is vastly more consistent and greatly outperforms simple reward-based learning. The value of the EKB is clear.

B. Cost Function Values

Cost function values indicate whether each learning algorithm optimizes its ANN effectively. If generalized learning is to be successful, cost function values should decrease over learning steps since errors between actual outputs and quasi-target outputs should decrease as a consequence of learning. Note that costs are calculated during learning, thus these results are based on training data not testing data.

Figure 4 shows cost J on Fold 1 of Data 4 (note different y scale). Other results are similar.

CALM-eLRB mostly shows cost decreases over learning steps. This means it learns appropriately generalized mappings between input contexts and quasi-target outputs. However, for CALM-rLRB, costs don't diminish over learning steps. Instead, costs in CALM-rLRB are consistently low after training on each context, showing that it is able to adapt its weights to each new input-output pair, even though its degree of generalization cannot be determined by this metric (since its cost function only considers the current data point).

C. Accumulated Rewards

The rate at which a learner accumulates rewards gives a measure of how well it is able to generalize from the training instances it has seen in order to give correct responses for inputs it has not yet seen. Figure 5 shows the accumulated rewards on Fold 1 of Data 4. Performance on Data 1 to Data 3 is generally higher for both algorithms, as these sets have fewer clusters and/or less overlap between clusters. Naturally, these algorithms do not outperform random guessing on Data 5.

While CALM-rLRB generally outperforms the baseline (by up to 1.5 times on Data 4; by nearly 3.5 times on Data 2), CALM-eLRB greatly outperforms CALM-rLRB on Data 1 through Data 4, earning up to 5 times the baseline on Data 4 and up to 6 times on Data 2.

Note that high accuracy tends to be correlated with high reward accumulation, as one would expect, because the points used in these measures, while distinct, were drawn from the same distributions.

D. Dynamic Accuracy

Dynamic accuracy measures how algorithms adapt to dynamic environments. In order to provide dynamic changes in synthetic data, Data 6 has 7 data clusters each with 70 data points (identical to Data 2). After all 490 points are presented to the learner, the correct output value changes for each cluster. This means that everything the system has learned before the change is wrong. The points are then re-presented to the learner, one on each learning step, and the learner should unlearn what it already knew in order to gain new knowledge.

Dynamic accuracy is measured on both the original and the changed data in order to see learning then unlearning on the original data as well as non-learning then learning on the changed data. Figure 6 shows dynamic accuracy on Data 4, Fold 1. The left column represents accuracies of each algorithm on the original data. The right column shows accuracies on the changed data.

The left half of Figure 6a shows CALM-rLRB, particularly using 2 weight layers, generally improving its accuracy on the original data as more of that data is presented to it, as one would expect. The right half of that figure shows its accuracy on that data set declining as more of the changed data is presented to it, again as one would hope. This shows that CALM-rLRB both learns and unlearns as appropriate for a dynamic environment.

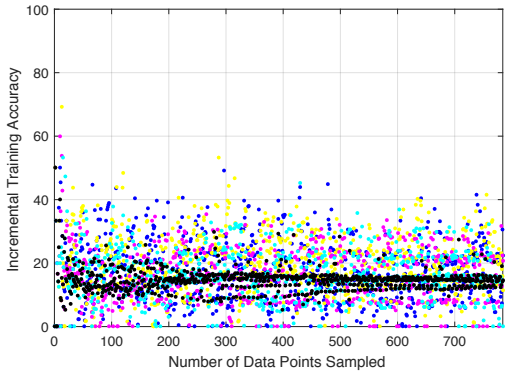
The left half of Figure 6b shows CALM-rLRB accuracy staying constant or declining for the changed data as more of the original data set is presented to it. Again, this is as one would expect, since it is learning different outputs than are considered correct for the changed data. The right half of the figure shows the accuracy, particularly using 2 weight layers, generally increasing as more of the changed data is presented to it, as one would hope. The original data does not appear to interfere with CALM-rLRB learning the new data.

CALM-eLRB, particularly with 2 and 3 weight layers, shows increasing accuracy before the environment changes and unchanged accuracy after the environment changes on the original data as shown Figure 6c. On the other hand, CALM-eLRB shows almost no learning on the changed data after the environment changes.[3] This means CALM-eLRB successfully learned the original data but could not unlearn what it knew in

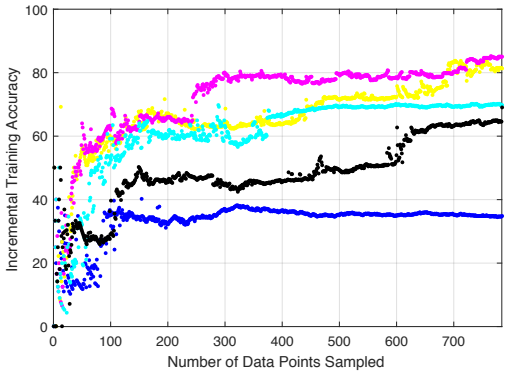
[3]The primary exception to this is for the case of 5 weight layers. Here, CALM-eLRB quickly learns to correctly classify points from 2 of the original clusters but fails to learn the others. After the data changes, it builds from correct responses on 1 cluster up to 4 before the end. It is likely that because that the EKB did not fill up with positive experiences for most clusters before the data changed, that it was able to learn most of the new clusters after the change.

TABLE I: Accuracy results for static data sets Data 1 to Data 4. Mean±standard deviation.

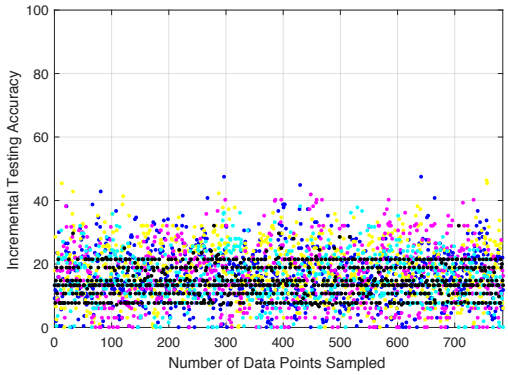
		Data 1		Data 2		Data 3		Data 4	
		training	testing	training	testing	training	testing	training	testing
rLRB	1	13.4±4.5	17.0±6.7	22.8±12.0	20.8±10.9	18.6±8.8	17.6±4.1	18.5±8.1	14.8±8.7
	2	26.8±15.1	19.0±9.5	69.2±8.0	66.5±7.6	48.3±11.0	48.3±12.6	18.8±7.8	16.4±10.9
	3	18.8±11.0	17.0±14.1	17.7±6.7	15.3±6.0	13.7±4.3	13.7±3.6	19.0±6.4	19.4±8.2
	4	13.3±5.1	17.0±8.6	11.3±6.4	12.0±7.8	10.0±0.5	10.0±1.8	13.3±8.1	13.7±7.9
	5	10.1±1.3	9.5±5.1	15.6±1.0	13.5±5.7	12.5±5.0	10.0±2.5	15.3±1.5	13.9±5.0
eLRB	1	32.6±4.8	30.5±9.7	43.0±1.8	42.2±7.3	42.5±5.2	40.1±3.8	30.4±2.7	28.3±7.3
	2	71.1±7.8	66.0±13.6	100.0±0.0	100.0±0.0	96.0±4.0	95.4±2.6	82.6±3.1	80.5±4.0
	3	29.6±21.9	29.5±20.0	91.3±7.8	91.2±8.2	89.5±17.1	91.2±15.1	91.8±6.0	90.8±7.6
	4	68.4±11.3	60.5±16.2	79.9±18.5	79.2±20.7	76.5±15.0	70.9±18.0	75.7±8.5	76.0±10.7
	5	47.3±15.2	44.5±21.2	82.7±5.3	81.4±8.1	87.6±6.1	88.4±8.4	57.4±9.4	55.0±10.0



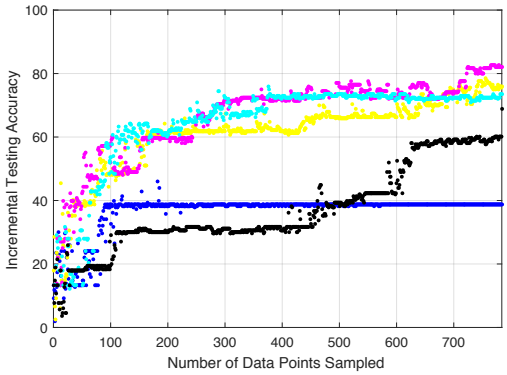
(a) CALM-rLRB Training Accuracy



(b) CALM-eLRB Training Accuracy



(c) CALM-rLRB Testing Accuracy



(d) CALM-eLRB Testing Accuracy

Fig. 3: Accuracy on Data 4, Fold 1. Colors show ANN depth: Blue 1, yellow 2, magenta 3, cyan 4, black 5.

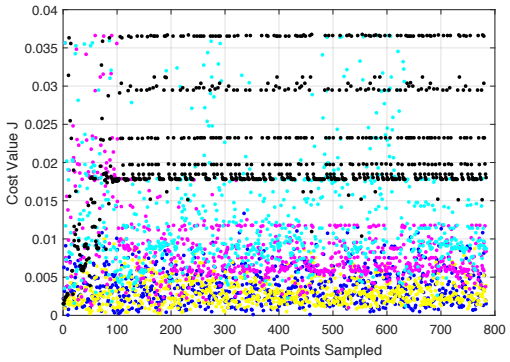
order to adapt to the changed environment. This drives home an important point: The EKB of CALM-eLRB is not dynamic.

VI. Discussion

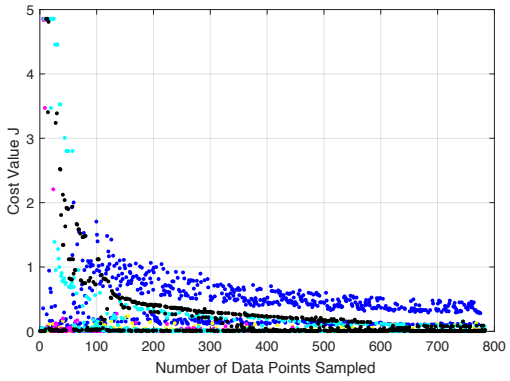
These experiments show that CALM-eLRB learns more stably, shows higher static accuracy, and accumulates more reward in static environments than CALM-rLRB but cannot adapt to dynamic environments. In contrast, CALM-rLRB learns more erratically, shows lower static accuracy, and accumulates less reward in static environments but can adapt to dynamic environments by unlearning and relearning.

Additionally, we can see clearly that a single weight layer is not sufficient for the ANN in these experiments. This is not surprising, as an ANN with a single weight layer can only handle linear classification [2] and the data here are not, in general, linearly separable.

Moreover, for CALM-rLRB, 2 weight layers generally seem to work best. This is likely to be because learning with only the latest data point doesn't effectively propagate strong weights back through more layers. Note that for Data 4, which is the largest data set, 3 layers seems to work at least as well as and perhaps better than 2, so perhaps the greater number of data points helps to propagate changes farther back in the network. These general trends seem to hold for CALM-eLRB as well, although the effects seem less pronounced, perhaps

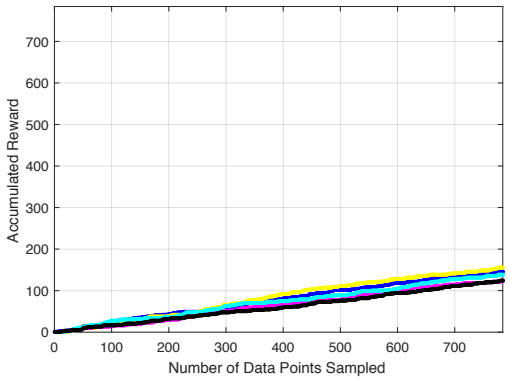


(a) CALM-rLRB Cost J

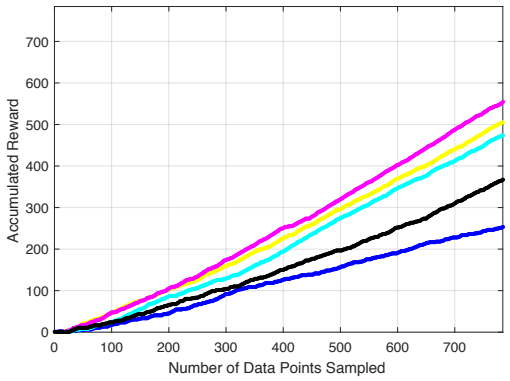


(b) CALM-eLRB Cost J

Fig. 4: Cost J on Data 4, Fold 1. Colors show ANN depth: Blue 1, yellow 2, magenta 3, cyan 4, black 5.



(a) CALM-rLRB Accumulated Rewards



(b) CALM-eLRB Accumulated Rewards

Fig. 5: Accumulated rewards on Data 4, Fold 1. ANN depth: Blue 1, yellow 2, magenta 3, cyan 4, black 5.

because using all past positive experiences helps to push useful weights back farther through the ANN. We should note that having more layers means more weight vectors which means a larger search space and more opportunities to get trapped in local optima.

VII. Conclusions and Future Work

This research introduces the Context-Aware Learning Model which is a novel learning model that combines supervised learning using logistic regression backpropagation and hyperbolic reward-based learning. CALM-rLRB introduces quasi-target outputs and CALM-eLRB adds an empirical knowledge base of past positive experiences to enhance its learning. These innovations contribute to successful learning as shown through evaluations with four metrics on six synthetic data sets.

Given the much greater performance of CALM-eLRB on static data sets but its inability to handle dynamic environments, both due to its EKB, a fundamental next step in this research is to find mechanisms that will enable the EKB to deal with dynamic data as does CALM-rLRB.

CALM should be evaluated in other environments, including robotics (starting in simulation). In particular, CALM should be tested with higher-dimensional input spaces and with noisy data. Along with this, several types of input should be processed as contextual information from the Context Supplier in a more sophisticated way in order to demonstrate high-quality neural context-awareness. In this paper, the Context Supplier directly takes the input from the Sensory System as a context.

CALM should also be evaluated with different values for its learning parameters. Experiments with different parameter values can give an understanding of the relationships between parameters and system performance.

Also, a neurobiological study of the hippocampus of a vertebrate brain should be accomplished to compare the role of the Experience-based Knowledge Base with the functions of the hippocampus; such a study could help support development of the learning system not only computationally but also neurobiologically. This includes investigating mechanisms for dealing with knowledge optimization to reduce redundancy.

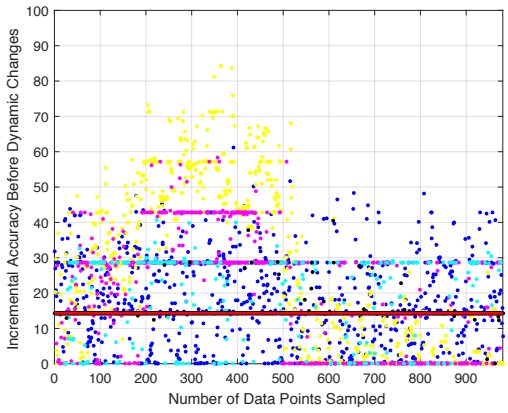
References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–104, September 1991.

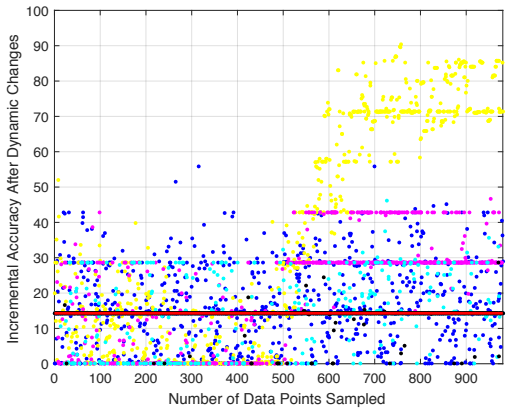
[2] A. P. Engelbrecht, *Computational Intelligence*, 2nd ed. Wiley, 2007.

[3] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, USA: MIT Press, 1998.

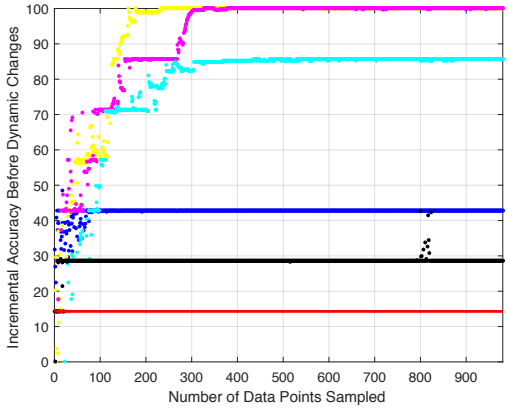
[4] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley, 1949.



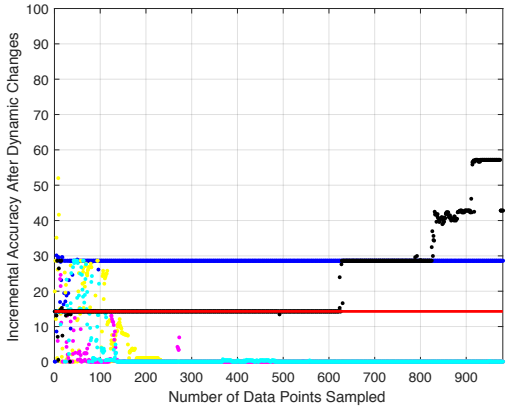
(a) CALM-rLRB on original clusters.



(b) CALM-rLRB on changed clusters.



(c) CALM-eLRB on original clusters.



(d) CALM-eLRB on changed clusters.

Fig. 6: Dynamic accuracy on Data 6. ANN depth: Blue 1, yellow 2, magenta 3, cyan 4, black 5.

[5] C. Pennartz, "Reinforcement learning by Hebbian synapses with adaptive threshold," *Neuroscience*, vol. 81, no. 2, pp. 303–319, 1997.

[6] A. Soltoggio and K. O. Stanley, "From modulated Hebbian plasticity to simple behavior learning through noise and weight saturation," *Neural Networks*, vol. 34, pp. 28–41, October 2012.

[7] H. van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 207–251. [Online]. Available: https://doi.org/10.1007/978-3-642-27645-3_7

[8] A. Ghanbari, Y. Vaghei, S. Noorani, and S. M. Reza, "Reinforcement learning in neural networks: A survey," *International Journal of Advanced Biological and Biomedical Research*, vol. 2, no. 5, pp. 1398–1416, 2014. [Online]. Available: http://www.ijabbr.com/article_7340.html

[9] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey," in *Learning Motor Skills*. Cham: Springer International Publishing, 2014, vol. 97, pp. 9–67. [Online]. Available: http://link.springer.com/10.1007/978-3-319-03194-1_2

[10] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0893608014002135

[11] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation: The state of the art," *arXiv preprint arXiv:1701.08878*, 2017. [Online]. Available: https://arxiv.org/abs/1701.08878

[12] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274v3*, p. 66 pages, Jul. 2017.

[13] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, May 2017. [Online]. Available: http://link.springer.com/10.1007/s10846-017-0468-y

[14] L. M. Saksida, S. M. Raymond, and D. S. Touretzky, "Shaping robot behavior using principles from instrumental conditioning," *Robotics and Autonomous Systems*, vol. 22, no. 3-4, pp. 231–249, December 1997.

[15] J. G. Fleischer and G. M. Edelman, "Brain-based devices: An embodied approach to linking nervous system structure and function to behavior," *IEEE Robotics and Automation Magazine*, vol. 6, no. 3, pp. 33–41, September 2009.

[16] L. N. Soldatova, A. Clare, A. Sparkes, and R. D. King, "An ontology for a robot scientist," *Bioinformatics*, vol. 22, no. 14, pp. e464–e471, July 2006.

[17] W. Hwang, J. Park, H. Suh, H. Kim, and I. Suh, "Ontology-based framework of robot context modeling and reasoning for object recognition," in *Fuzzy Systems and Knowledge Discovery*. Springer Berlin Heidelberg, 2006, pp. 596–606.

[18] R. Salgado, F. Bellas, P. Caamaño, B. Santos-Díez, and R. J. Duro, "A procedural long term memory for cognitive robotics," in *Evolving and Adaptive Intelligent Systems (EAIS), 2012 IEEE Conference on*. Evolving and Adaptive Intelligent Systems (EAIS), May 2012, pp. 57–62.

[19] P. Moore and H. V. Pham, "Predicting intelligence using hybrid artificial neural networks in context-aware tunneling systems under risk and uncertain geological environment," in *Complex, Intelligent, and Software Intensive Systems (CISIS)*. IEEE, July 2012, pp. 989–994.

[20] I. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[21] J. Suh and D. F. Hougen, "Context-based adaptive robot behavior learning model (carb-lm)," in *IEEE Symposium Series on Computational Intelligence*. IEEE, December 2014, pp. 206–211.

[22] J. Suh, "The context-aware learning model," Ph.D. dissertation, University of Oklahoma, Norman, OK, Dec. 2017, forthcoming.