Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 1516 – 1523

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Enhancement of Classifiers in HTM-CLA Using Similarity Evaluation Methods

Jahan Balasubramaniam[a], Gokul Krishnaa.C.B[a], Fangming Zhu[a,*]

[a] *Institute of Systems Science, National University of Singapore, Singapore*

Abstract

The recent development in the theory of Hierarchical Temporal Memory (HTM) - Cortical Learning Algorithms (CLA) which models the structural and algorithmic properties of neocortex has brought in a new paradigm in the field of machine intelligence. As the theory of HTM-CLA continues to evolve, the ways of inferring the patterns and structures recognized by the HTM-CLA algorithm are still a big challenge. Moreover, the existing methods used to infer the classification output from HTM-CLA are far from satisfactory. In this paper, we propose two new classifiers using similarity evaluation methods based on dot similarity (H-DS) and mean-shift clustering (H-MSC) to obtain classification from the sparse distributed representation (SDR) output of HTM-CLA. We validate and benchmark the performance of our proposed classifiers using three datasets from the UCI machine learning repository. The results show that the proposed classifiers enhance the classification performance of HTM-CLA and their performance is also comparable to other traditional machine learning technique such as decision tree.

Keywords: Hierarchical Temporal Memory; Cortical Learning Algorithms; HTM-CLA; Similarity Evaluation; Classifier

1. Introduction

The recent development in the theory of Hierarchical Temporal Memory (HTM) - Cortical Learning Algorithms (CLA) has brought in a new paradigm in the field of machine intelligence. HTM-CLA is a biomimetic model based on the memory-prediction theory of brain function proposed by Jeff Hawkins[1]. It is a method for discovering and inferring the high-level causes of observed input patterns and sequences, thus building an increasingly complex model of the world.

* Corresponding Author. Tel.: +65-6516-4681; fax: +65-6778-2571.
Email address: fangming@nus.edu.sg

doi:10.1016/j.procs.2015.08.261

HTM-CLA is based on the structure and function of the human neocortex. It offers the groundwork for building machines that approach or exceed human level performance for many cognitive tasks. It is implemented within the NuPIC, an open source project led by Numenta[2,3]. HTM-CLA is fundamentally a memory based prediction system. The networks are trained on time-varying data and store a large set of patterns and sequences. It has a hierarchical organization and is inherently time based[4].

HTM-CLA has been used in a wide array of domains and applications ranging from computer vision to financial forecasting. Typical applications include image analysis, robotics, data mining, intelligent traffic monitoring, weather prediction, etc. Some real world applications implemented using HTM-CLA are land use classification based on computer vision[5], song identification[6], trading of financial markets[7,12], retina analysis[8], music representation[10].

HTM-CLA learns and stores the structure and sequences of data in hierarchy of regions in unique representation called Sparse Distributed Representation (SDR)[9,11,13]. However, inferring output from these SDR's are proven to be difficult and existing methods perform poorly and take too much memory. While much research work is focused on establishing and evolving the core theories and concepts, very little work has been done on inferring final output based on the SDR output. Moreover, the existing methods used to infer the classification output from HTM-CLA are far from satisfactory. In this paper, we address this problem by proposing two new methods based on dot similarity (H-DS) and mean-shift clustering (H-MSC) to obtain classification from the SDR output of the HTM-CLA. We validate and benchmark the performance of our proposed classifiers for HTM-CLA using three datasets from the UCI machine learning repository[14].

The rest of the paper is organized as follows. In section 2, we give a brief overview of HTM-CLA. The proposed methods are described in Section 3. To illustrate the performance of our proposed methods, experimental results are provided in Section 4. Section 5 concludes the paper.

2. Overview of HTM-CLA

HTM-CLA is in essence a theory of how the human brain works. There are three characteristics of the brain that are of vital importance in designing HTM-CLA. Firstly, the brain is inherently a hierarchical system. There is flow of signal in both directions of the hierarchy. Also there is flow of signal within the region itself. Secondly, all information stored in the brain is temporal in nature. Time is a ubiquitous feature of all the aspects of the brain learning. Thirdly, the human brain is fundamentally a memory system. We try to remember and predict patterns over time. All the cells and their connections are in a way storing the patterns experienced over time.

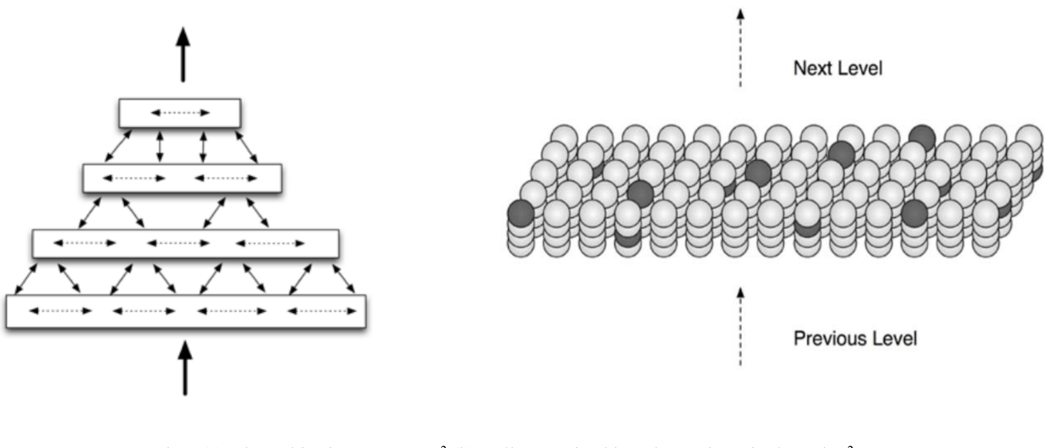

Fig.1 (a) Hierarchies in HTM-CLA[2] (b) Cells organized in columns in a single region[2]

Figure 1 shows the hierarchical structure of HTM-CLA. The basic unit of hierarchy is the cell in HTM-CLA. The cells are organized in columns. These columns combine to form a region and regions combine to form hierarchies. The cells can be connected with other cells in the same region or regions at upper or lower levels. The cells learn and store the temporal sequence of the data while columns denote the semantics of the data through SDR representations.

There are many advantages to learn by using a hierarchy. The regions at lower levels could learn some fundamental features and the regions higher up could be used to learn high level predictions. Generally regions higher up are more robust to noise. The input is represented as binary bits. Columns in the regions establish connections with these binary bits and each cell in-turn establishes connections with cells in others column based on context of temporal sequence of the sensory inputs. The output of lower region will be fed as input to region in upper hierarchy.

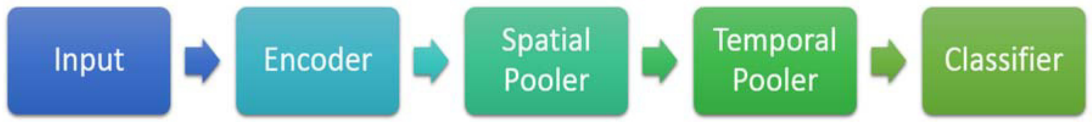

Fig. 2: NuPIC Pipeline Process flow

The implementation of HTM-CLA is available in the NuPIC framework[2]. Figure 2 shows the process flow of the NuPIC framework.

Encoders: Input data are first sent to encoders to convert them into binary bits. The encoders converts the data into binary bits based on the semantics of the variable. Similar values will have overlapping bits.

Spatial Pooler: The spatial pooler corresponds to learning function of the columns within the region. It tries to form a stable representation of the input patterns by establishing connection between input bits and columns.

Temporal Pooler: The temporal pooler enables the cells of a column to learn to represent the input in the context of time.

Classifier: Classifier tries to infer the output from active columns in the upper region in the hierarchy.

HTM-CLA aims to learn and represent structures and sequences based on memory predictions. However, the classifier used to infer the classification output from HTM-CLA are far from satisfactory. Classifiers used in the NuPIC framework are knnClassifier and CLAClassifier. The old knnClassifier maps the nearest neighbors and infers the output class. However, it requires every data point to be stored in memory which results in very high complexity for huge datasets. Furthermore, its performance is still not guaranteed. The latest CLAClassifier maps the SDR to the input region and tries to reconstruct the target variable. However, this method seems to be not so effective.

3. Proposed Classifiers for HTM-CLA

We propose two new classifiers based on different similarity evaluation methods. The first method is H-DS Classifier based on Dot Similarity and the second method is H-MSC Classifier based on Mean-Shift Clustering. We aim to make the classifiers in HTM-CLA more efficient and effective.

3.1 H-DS Classifier

The H-DS classifier employs dot product computation. The output from the region in the last level of HTM-CLA is a vector consisting of ones and zeros - an SDR. During the training phase, the SDR result for each input pattern is obtained and stored. Then these vectors are grouped according to their corresponding class labels. As a result, a class distribution matrix can be produced. Each row in the matrix represents the central position of each class, which is the average of all the SDR vectors belonging to the same class.

Let $V_{k\times n}$ be the class distribution matrix, where *k* is the number of classes, *n* is the dimension of SDRs;
$O_{1\times n}$ be the SDR vector for the new input pattern to be classified;
$P_{k\times 1}$ be the output class vector for the new input pattern;
C be the class label to be assigned to the new input pattern;

The output class vector for the new pattern can be computed as the inner/dot product of matrix *V* and vector *O*:

$$P = V \cdot O^T$$

We then identify the maximum value in *P*, and the class label *C* corresponding to the maximum value should be assigned to the new input pattern:

$$C = \underset{i\epsilon[1..k]}{argmax}\,\{P_i\}$$

Figure 3 shows a simple example of the H-DS classifier. It simplifies the SDR dimension as 6 and assumes a 3-class classification problem. After the dot product computation, the vector obtained indicates that class A gets the maximum value, which means the new pattern is most similar to those patterns belonging to class A, therefore the new pattern is classified as class A.

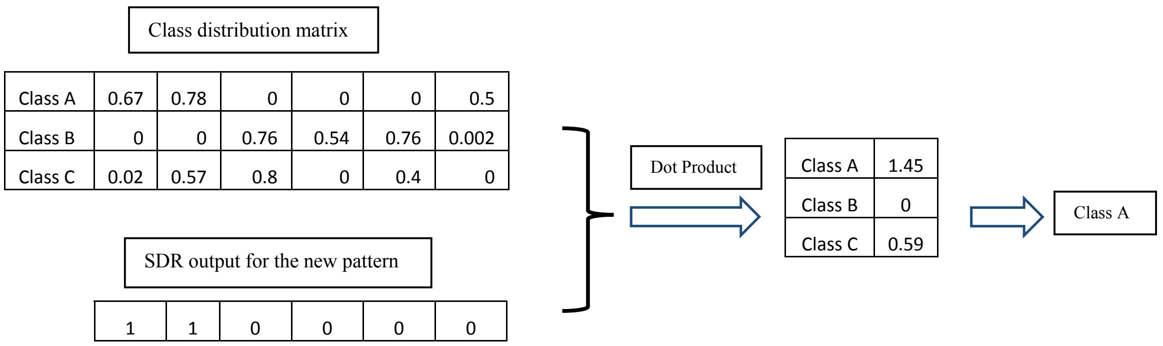
Class distribution matrix
Class A 0.67 0.78 0 0 0 0.5
Class B 0 0 0.76 0.54 0.76 0.002
Class C 0.02 0.57 0.8 0 0.4 0
SDR output for the new pattern
1 1 0 0 0 0
Dot Product
Class A 1.45
Class B 0
Class C 0.59
Class A

Fig. 3: A simple example of H-DS classifier

3.2 H-MSC Classifier

The H-MSC classifier employs mean-shift clustering. This method is based on the intuition that similar patterns will have similar SDR representation in HTM-CLA. One important feature of mean-shift clustering is that it is not required to specify number of cluster in advance, which is different from other clustering techniques such as *k*-means.

The H-MSC classifier uses the following four steps to classify a new input pattern:

Step 1: During the training phase, the training patterns are presented to train HTM-CLA, and their SDR results are obtained and stored.

Step 2: Mean-shift clustering is applied on the set of SDRs to form clusters. Then the members of each cluster are known and the cluster centers can be computed.

Step 3: Assign a class label to each cluster using majority voting scheme.

Step 4: During the testing phase, the nearest cluster is identified based on the similarity between the SDR output of the testing pattern and the cluster centers, and the class label suggested by the nearest cluster centers is assigned to the testing pattern.

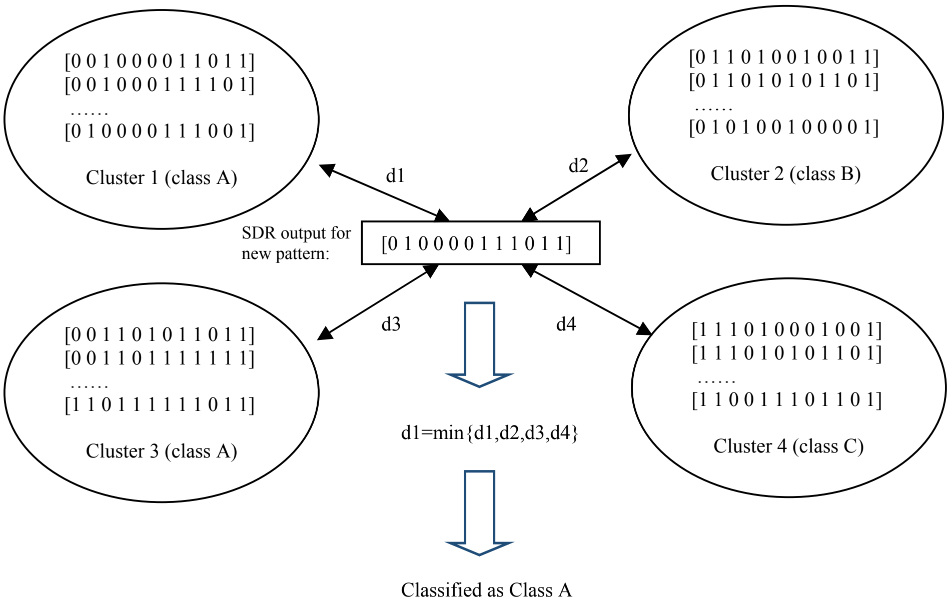

Fig. 4: A simple example of H-MSC classifier

Figure 4 illustrates a simple example of the H-MSC classifier. Four clusters have been formed, and each has been assigned a class label. We can find that each class can have one or more clusters associated with it. When a new input pattern needs to be classified, four distance values are computed between the SDR output of the new pattern and the four cluster centers. It is shown that d1 is the smallest among the four distance values, which means that the new pattern is most similar to the set of patterns belonging to cluster 1. Thus, the new pattern should be classified as Class A.

4. Experimental Results

Extensive experiments have been conducted to evaluate the performance of our proposed methods. We select three benchmark datasets from the UCI machine learning repository[14], i.e. Hebberman's Survival Dataset, Iris Dataset, and Diabetes Dataset. All the three datasets are about classification problems.

For HTM-CLA, we use three level hierarchy of spatial poolers. The number of columns in each level is 500, 250 and 100 respectively. The number of active columns per inhibition area was kept to 1. These settings are consistently maintained throughout all the experiments.

In each experiment, the dataset is randomly partitioned into 70% as training set and 30% as testing set. We use the training data to train HTM-CLA with three different classifiers, ie. H-DS classifier, H-MSC classifier, and

CLAClassifer. Then we apply them on the testing data to compute their accuracy. Each experiment is run ten times to get average accuracy and standard deviation. We also compare their performance with another machine leaning technique - decision tree C5. The results on the testing data are presented in Tables 1, 2 and 3 for the three datasets respectively.

Table 1: Results on Heberman's Survival Dataset

Run	**H-DS Classifier**	**H-MSC Classifier**	**CLA Classifier**	**Decision Tree**
1	0.751	0.737	0.728	0.716
2	0.740	0.751	0.713	0.715
3	0.723	0.779	0.712	0.703
4	0.735	0.741	0.725	0.702
5	0.738	0.779	0.713	0.715
6	0.743	0.733	0.714	0.709
7	0.749	0.759	0.722	0.706
8	0.751	0.763	0.715	0.718
9	0.756	0.771	0.717	0.696
10	0.749	0.739	0.725	0.715
Accuracy (Mean ± Standard Deviation)	**0.743±0.010**	**0.755±0.018**	**0.718±0.006**	**0.709±0.007**

Table 2: Results on Iris Dataset

Run	**H-DS Classifier**	**H-MSC Classifier**	**CLA Classifier**	**Decision Tree**
1	0.867	0.894	0.866	0.904
2	0.862	0.902	0.863	0.905
3	0.862	0.879	0.842	0.907
4	0.873	0.876	0.878	0.908
5	0.867	0.875	0.839	0.903
6	0.881	0.893	0.866	0.918
7	0.863	0.877	0.840	0.908
8	0.889	0.897	0.848	0.909
9	0.879	0.875	0.861	0.905
10	0.862	0.903	0.881	0.914
Accuracy (Mean ± Standard Deviation)	**0.870±0.010**	**0.887±0.012**	**0.858±0.015**	**0.908±0.005**

Table 3: Results on Diabetes Dataset

Run	**H-DS Classifier**	**H-MSC Classifier**	**CLA Classifier**	**Decision Tree**
1	0.681	0.708	0.731	0.720
2	0.730	0.734	0.749	0.714
3	0.685	0.710	0.721	0.696
4	0.685	0.701	0.754	0.723
5	0.718	0.721	0.721	0.696
6	0.713	0.748	0.745	0.745
7	0.665	0.714	0.737	0.744
8	0.687	0.712	0.750	0.715
9	0.722	0.708	0.739	0.699
10	0.668	0.741	0.728	0.703
Accuracy (Mean ± Standard Deviation)	**0.695±0.023**	**0.720±0.016**	**0.738±0.012**	**0.716±0.018**

The experimental results show that our proposed H-DS classifier and H-MSA classifier outperform the CLAClassifier on the Heberman's Survival Dataset and Iris dataset and achieve comparable results on the Diabetes dataset. Moreover the performance of the three HTM-CLAs are in general comparable to that of decision tree. Between the proposed two methods, the H-MSC Classifier has shown better performance than the H-DS Classifier.

5. Conclusion

HTM-CLA is a new technique in the field of machine intelligence. In this paper, we aim to enhance the performance of the classifiers for HTM-CLA based on the similarity evaluation methods. We propose two new classifiers: H-DS based on dot similarity and H-MSC based on mean-shift clustering. We validate and benchmark the performance of our proposed classifiers for HTM-CLA using three datasets from the UCI machine learning repository. The results show that the proposed classifiers can enhance the classification performance of HTM-CLA and their performance is also comparable to other traditional machine learning technique such as decision tree. Our future work will be on further improving the classifier design and exploring the use of HTM-CLA for regression problems.

References

1. J. Hawkins and S. Blakeslee, *On Intelligence*: Times Books, 2004.
2. Numenta, Hierarchical Temporal Memory including HTM Cortical Learning Algorithms, HTM CLA white paper - VERSION 0.2.1, SEPTEMBER 12, 2011.
3. Numenta, Learning Center, available: http://numenta.com/learn/
4. D. George and J. Hawkins, Towards a mathematical theory of cortical micro-circuits, *PLoS Computational Biology*, vol. 5, issue 10, 2009.
5. A.J. PereaI, J.E. MeroñoII, M.J. AguileraI, Application of Numenta® Hierarchical Temporal Memory for Land-Use Classification, *South African Journal of Science*, ISSN 0038-2353, vol.105 n.9-10 Pretoria Sep./Oct. 2009.
6. Nathan C. Schey, Song Identification Using the Numenta Platform for Intelligent Computing, Department of Computer Science and Engineering Honors Theses, Ohio State University. 2008. http://hdl.handle.net/1811/32025
7. Patrick Gabrielsson, Rikard König and Ulf Johansson, Hierarchical Temporal Memory-Based Algorithmic Trading of Financial Markets, IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012.

8. Boone, A.R.W., Karnowski, T.P., Chaum, E., Giancardo, L., Li, Y., Tobin, K.W., Jr., Image Processing and Hierarchical Temporal Memories for Automated Retina Analysis, IEEE, Biomedical Sciences and Engineering Conference (BSEC), 2010.
9. Michael Galetzka, Intelligent Predictions: an empirical study of the Cortical Learning Algorithm, University of Applied Sciences Mannheim, 2014.
10. J. Maxwell, et al., Hierarchical Sequential Memory for Music: A Cognitive Model, In *Proceedings of the 10th International Society for Music Information Retrieval Conference*. 2009, p. 429--434.
11. D. Rozado, et al., Optimizing Hierarchical Temporal Memory for Multivariable Time Series, in Artificial Neural Networks – ICANN 2010. vol. 6353, K. Diamantaras, et al., eds., Springer Berlin / Heidelberg, 2010, p. 506-518.
12. F. Åslin, Evaluation of Hierarchical Temporal Memory in algorithmic trading, Department of Computer and Information Science, University of Linköping, 2010.
13. Fergal Byrne, *Real Machine Intelligence with Clortex and NuPIC*, available: https://leanpub.com/realsmartmachines/read
14. UCI machine learning repository, http://archive.ics.uci.edu/ml/