# Apache Spark based Distributed Self-Organizing Map Algorithm for Sensor Data Analysis

Madhura Jayaratne, Damminda Alahakoon, Daswin De Silva
Research Centre for Data Analytics and Cognition
La Trobe University
Victoria, Australia.

Xinghuo Yu
School of Engineering
RMIT University
Victoria, Australia.

***Abstract*—The proliferation of Internets of Things (IoT) technologies in both industrial and non-industrial settings has led to the accumulation of Big Data sets. Analysis of these high-volume, high-velocity datasets require advanced processing techniques that incorporate parallel and distributed computations. In this paper, we present a novel distributed self-adaptive neural-network algorithm, the Distributed Growing Self-Organizing Map (DGSOM) algorithm to address the growing need for unsupervised machine learning of Big Data sets on distributed computing environments. The algorithm was tested on a Big Data set of sensor recordings of human activity collected from wearable devices, 2.8 million records. Results indicate that the distributed algorithm significantly reduces execution time compared to its serial counterpart. Moreover, the self-adaptive nature and controlled growth of the algorithm demonstrates data-driven structure adaptation and multi-granular pattern analysis. Overall, the proposed algorithm addresses the need for pattern discovery and visualization from Big Data sets generated by IoT devices which are increasingly commonplace in industrial scenarios.**

***Keywords—Growing Self-Organizing Maps, Resilient Distributed Dataset, Unsupervised Machine Learning, Internet of Things, Industrial IoT, Big Data analysis.***

## I. INTRODUCTION

Throughout the history of computing, researchers have attempted to draw inspiration from the human brain to improve computing to perform tasks that are effortless for humans. Cognitive computing refers to hardware and/or software that are strongly motivated by the working mechanisms of human brain [1]. In terms of software, this has led to the development of various machine learning and memory algorithms. Cognitive algorithms implement learning using Hebbian theory [2] on the adaptation of neurons in the brain during learning.

Artificial neural networks (ANN) are one such class of algorithms that draws inspiration from human nervous system and tries to model neurons and their interactions through synapses. Recurrent neural networks (RNN) are a class of ANN with connections between neurons forming a directed cycle. This facilitates internal memory in the network which is essential for task requiring temporal behavior such as speech recognition and language modeling [3]. RNNs are inspired by the fact that the neocortex is saturated with feedback connections. For example, between the neocortex and thalamus, connections going backward exceed the connections going forward by almost a factor of ten [4]. More recent developments such as the rise of Deep Neural Networks (DNN) have shown promising results in tasks such as image and speech recognition [5], [6]. While ANNs with a single hidden layer is a gross simplification of the neuronal activity in human brain, DNNs provide a better representation, however, at the expense of having to learn thousands of parameter values [7]. To date, ANNs are one of the most popular choices for classification and regression tasks in a vast array of fields such as manufacturing, automated driving, medical diagnosis, financial fraud detection, data mining etc.

Another class of ANNs uses competitive learning, a variant of Hebbian learning, in which neurons compete to respond to a subset of the input data in the winner-takes-all form. Vector quantization and Self-Organizing Maps (SOM) are the most popular cognitive computing algorithms that use competitive learning. SOMs are a very popular choice for dimensionality reduction, exploratory data analysis, and data clustering. SOM algorithm preserves the neighborhood relationships when adjusting the weights of its two-dimensional map to provide a non-linear projection of the higher dimensional data onto the two-dimensional map.

The recent Big Data phenomenon, driven by disruptive technological, social and economical forces, the amount of data available for analysis has grown exponentially. This is especially true in industrial settings where Industry 4.0 has ushered Industrial IoT (IIoT) systems generating large datasets. Serial algorithms that were being used in cognitive computing can no longer address Big Data problems. This has led to parallelization of cognitive computing algorithms to be able to run on distributed computing platforms which allows computing to be distributed over a number of computing resources, so the results are generated in acceptable time.

A major drawback of SOM algorithm is its time complexity. This makes SOMs unsuitable for processing large datasets in their current form. As the Big Data phenomenon ushers extremely large datasets, the SOM algorithm needs to be adapted to use distributed computing to be relevant in the Big Data era.

In this paper, we present a novel distributed algorithm to expand GSOM, a dynamic variant of the SOM algorithm. The GSOM algorithm has been successfully applied to numerous fields including engineering [8]–[11], text [12]–[14], biology [15]–[17]. The algorithm uses data parallelism to train GSOMs

978-1-5386-1127-2/17/$31.00 ©2017 IEEE

on subsets of data and finally merge them together to form the single two-dimensional map. We adapt the algorithm for the distributed computing paradigm Resilient Distributed Datasets (RDD). RDDs are immutable, fault-tolerant distributed dataset divided into logical partitions and can be created by deterministic operations on either data or other RDDs. Computations on these partitions may be carried out on different nodes in the cluster. We implement the algorithm for Apache Spark, the widely-used implementation of RDD.

We utilize the proposed Distributed GSOM algorithm to process a Big Data set of motion sensor data to analyze human activities. The choice of human activity sensor dataset is due to prevalence of IoT technology, the increased use of IoT systems consisting wearable devices and their numerous applications in industrial settings. Data from human worn sensors have been used for training car assembly line workers with sensor data being used for task tracking [18]. Moreover, activity recognition in industrial settings has been proposed for proactive instruction systems which track the completion of an activity and display the instructions for the next [19]. Quality control is another major use case of activity monitoring which verifies that the workers are correctly performing each procedure and tracks whether all the procedures have been completed [18], [19]. Industrial health and safety monitoring systems have also been proposed to monitor vibrations, sudden accelerations and generate necessary alerts [20].

The remainder of the paper is organized as follows. Section II presents a detailed background on SOM, GSOM, and the split and merge GSOM algorithm while Section III proposes the novel Distributed GSOM algorithm on Apache Spark. Human activity analysis using the proposed algorithm and results are presented in Section IV Finally, the paper concludes with the discussion of results and future enhancements.

## II. Background

### A. Self-Organizing Maps

The Self-Organizing Map (SOM) [21] algorithm is an artificial neural network algorithm which generates a non-linear mapping from higher dimensional input space to a lower (usually 2) dimensional output space. Trained using an unsupervised learning algorithm, SOMs facilitate dimensionality reduction while preserving the topological properties of the input space. Due to this topology preserving nature, SOMs are a popular candidate for exploratory analytics tasks.

### B. Growing Self-Organizing Maps

The main disadvantage of SOM is its requirement to specify the size of the map in advance. This may not be possible as knowledge of the underlying structure of data may not be available, especially for exploratory analytics tasks. Suboptimal choice of map sizes may lead to under or over representation of data on the map leading to poor topology preservation.

A dynamic variant of the SOM named Growing Self-Organizing Map (GSOM) has been proposed in [22] to address the above issue. The map is initialized with just 4 neurons organized in a 2×2 lattice and neurons are dynamically added as required. The algorithm consists of two phases and in the growing phase, input vectors are presented over a number of iterations and nodes are added to the map when the accumulated quantization error of a boundary node exceeds the growing threshold, *GT*. *GT*, as described in (1), is calculated based on the number of dimensions, *D*, and spread factor, *SF*, which controls the spread of the map.

$$GT = -D \times ln(SF) \tag{1}$$

In the case of a non-boundary node, the error is spread among neighboring neurons when the threshold is exceeded. The weight vector of the newly added neuron is initialized to match those of the neighbors. The smoothing phase is used to calibrate any existing quantization errors. Similar to the growing phase, inputs are presented and weights are adjusted, with no new neuron addition.

### C. Split and Merge GSOM Algorithm

A major drawback of SOM algorithm and its variants is time complexity. This makes SOMs unsuitable for Big Data applications in their current form. In [23], [24] a parallel version of GSOM algorithm which utilizes data parallelism and horizontal data splitting is proposed. Workings of the algorithm are outlined in Algorithm 1. Various strategies have been proposed for data partitioning, including random partitioning, class based partitioning and high-level clustering based partitioning. Once GSOMs are trained on data partitions an additional step is proposed to remove the redundant neurons among map. These redundant neurons may be present as the GSOMs are trained independently without any coordination among them. The main advantage of the proposed algorithm is that it preserves the final map of the whole dataset, the hallmark feature of the SOMs. Sammon's projection is used to generate the final map using the intermediate SOMs. Sammon's projection is based upon point mapping of higher dimensional vectors to lower dimensional space such that the inherent structure of the data is preserved. The algorithm does so by minimizing Sammon's stress *E* in (2), over a number of iterations.

$$E = \frac{1}{\sum_{i<j}[d^*_{i,j}]} \sum_{i<j}^{N} \frac{[d^*_{i,j} - d_{i,j}]^2}{d^*_{i,j}} \tag{2}$$

where, $d^*_{i,j}$ and $d_{i,j}$ are the distances between corresponding pairs of vectors in the higher and lower dimensional spaces.

**Algorithm 1**. Split and merge GSOM algorithm

1. Split dataset into partitions
2. Train a GSOM on each partition in parallel
3. Remove redundant neurons across partitions
4. Perform Sammon's projection on neurons in trained GSOMs

## III. Distributed GSOM Algorithm

In this paper, we present a distributed GSOM algorithm which is based on the framework of the split and merge GSOM algorithm outlined in Section II. We also implement the

Distributed GSOM algorithm using the Apache Spark distributed processing platform.

One of the issues of training separate GSOMs on partitions of data is that it leads to redundant neurons among maps as training on partitions of data happen without any coordination. The redundant neurons are undesirable as they lead to redundancies in the final map and increased time for merging GSOMs with Sammon's projection. An algorithm to identify and remove such redundant neurons has been proposed in [24]. However, the redundancy removal process requires all the GSOMs to be present to identify redundancies among them leading to serial execution. We identify that redundant neuron removal can be performed in a parallel manner. This is achieved by removing redundancies among subsets of map simultaneously by applying the reduce operation of functional programming. The reduce operation takes two objects of the same type and outputs an object of the same type. Here, the reduce operation will consume two maps at a time to output a single map with redundancies removed.

### A. Distributed GSOM on Apache Spark

We propose an Apache Spark based Distributed GSOM algorithm in this section. Apache Spark [25] is well-established open-source distributed computing framework. It implements Resilient Distributed Dataset (RDD) [26], a read-only, fault-tolerant, logically partitioned dataset distributed over a cluster of computers. Apache Spark supports fault-tolerance by keeping the lineage of operations, so it can regenerate RDDs in case of a fault. Apache Spark has seen huge levels of adoption since its first release in 2014, especially for data-intensive processing tasks. One major reason for this success is that Apache Spark does not require intermediate results to be written to slower HDFS for sharing among computing tasks opposed to Apache Hadoop. Apache Spark is increasingly used in a number of industrial applications including intelligent fault diagnosis of high-speed trains [27], predictive analytics of power system applications [28], IIoT framework for soil-less food production [29] due to the above reasons.

The algorithm uses a number of data transformation operations defined on RDDs such as `map`, `reduceByKey`, `treeReduce` etc. The `map` operation applies a provided function for each record while `reduceByKey` operation aggregates all the records for a particular key by applying a provided function, with both the operations resulting in new RDDs. On the other hand, `treeReduce` reduces the elements of the RDD using the supplied commutative and associative binary operator resulting in a single object of the same type of the input RDDs. Apache Spark supports `treeReduce` operation, which reduces the elements of the RDD in a multi-level tree pattern. Contrary to `reduce` operation where the driver node spends linear time on the number of partitions which can become a bottleneck when there are many partitions and the data from each partition is big, `treeReduce` uses a binary tree to speed up reductions [30]. This makes Apache Spark ideal to implement the distributed redundant neuron removal.

The Distributed GSOM algorithm on Apache Spark starts by reading the data file from the HDFS with the parameters specifying the minimum number of partitions to be the number of computing cores in the cluster. This operation returns an RDD of lines, $l \in L$, with each line relating to a record in the data file. The RDD is then subjected to two `map` operations of which the first parse lines to return an RDD of records, $d \in D$.

$$map\ (f: L \rightarrow D) \tag{3}$$

The second `map` operation assigns a random integer, $i \in I$, in the range of [1, $P$] where $P$ is the number of desired partitions, to each record in the RDD. This step is used to perform the random partitioning of the dataset and results in an RDD of key-value pairs. We propose to set $P$ to the number of computing cores in the cluster.

$$map\ (f: D \rightarrow I \times K) \tag{4}$$

The next operation `reduceByKey` collects all the records assigned with the same integer. Similarly, the number of computing cores in the cluster is used as the minimum number of partitions. The random partitioning is followed by a `map` operation which trains GSOMs on each data partition simultaneously.

$$reduceByKey\ (f: (I \times K, I \times K) \rightarrow I \times K) \tag{5}$$

$$map\ (f: I \times K \rightarrow M) \tag{6}$$

The next operation `treeReduce` is used for the redundant neuron removal. As outlined earlier, the proposed distributed redundancy removal mechanism does not require all the maps to be collected on a single machine and removes redundancies among subsets of maps simultaneously. The function provided to the `treeReduce` operation receives two GSOMs at a time, removes redundant neurons in the two maps using the redundancy reduction algorithm and outputs the other neurons. This operation generates a local list of neurons and finally, these neurons are merged using Sammon's projection to obtain the single map for the whole dataset.

$$treeReduce\ (f: M, M \rightarrow M) \tag{7}$$

## IV. Experiments and Results

### A. The Dataset

We used PAMAP2 Physical Activity Monitoring Data Set [31] for experimentation and demonstration of the DGSOM algorithm. The dataset consists of sensor readings captured from 9 subjects engaged in 12 different physical activities (such as *walking*, *cycling*, *vacuum cleaning*, etc.). Each subject is wearing a heart rate monitor and 3 Inertial Measurement Units (IMU), over the wrist on the dominant arm, on the chest, and on the dominant side's ankle. Each IMU captures temperature (in °C), 3D-acceleration data (in $ms^{-2}$) (with two scales: ±16g and ±6g resolutions), 3D-gyroscope data (in rad/s), 3D-magnertomerter data (in μT) and orientation. The IMUs capture data with a sampling frequency of 100Hz while the heart rate monitor uses a lower sampling frequency of ~9Hz.

The dataset contains 2,872,533 records and 52 attributes. The dataset was preprocessed to remove records with missing values resulting from loss of wireless connectivity as well as

records pertaining to transient activities coded with class '0'. We discarded 3D-acceleration data with the scale of ±6g resolution as the readings seem to get saturated sometimes due to high impacts caused by certain movements (e.g. during running) with acceleration over 6g. Further, we removed orientation data as it was indicated to be invalid for this data collection. Moreover, heart rate values were interpolated to compensate for the low sampling frequency of the heart rate monitor compared to IMUs. The metadata of the dataset contains resting heart rate for each participant and based on that we calculated the average heart rate increase for each activity, as shown in Table 1. We can see that the activities include a mix of low-intensity activities such as *lying*, *sitting*, *standing*, etc., moderated intensity activities such as *walking*, *Nordic walking*, *cycling*, etc. and high-intensity activities such as *running* and *rope jumping*.

We engineered additional features to improve the accuracy based on [32] by calculating from the triaxial signals the Euclidean magnitude and time derivatives (jerk, *da/dt* and angular acceleration, *dw/dt*). The complete set of features includes, for each IMU, temperature, triaxial acceleration, acceleration magnitude, triaxial acceleration jerk, acceleration jerk magnitude, triaxial angular speed, angular speed magnitude, triaxial angular acceleration, angular acceleration magnitude, and triaxial magnetism.

TABLE 1 AVERAGE HEART RATE INCREASE

| Activity | HR increase (bps) | Intensity |
|---|---|---|
| Lying | 9.02 | Low |
| Sitting | 13.21 | |
| Standing | 22.42 | |
| Ironing | 24.34 | |
| Vacuum cleaning | 37.69 | |
| Walking | 46.46 | Moderate |
| Nordic walking | 58.21 | |
| Cycling | 58.85 | |
| Descending stairs | 62.94 | |
| Ascending stairs | 63.01 | |
| Running | 81.62 | High |
| Rope jumping | 90.16 | |

### B. *Test Environment and Configurations*

Our experiments were conducted on a commercial cloud platform supporting Apache Spark and employed 8 virtual machines each with 8 virtual cores of 2.3 GHz clock speed. Apache Spark version 2.0.2 was used for the algorithm implementation.

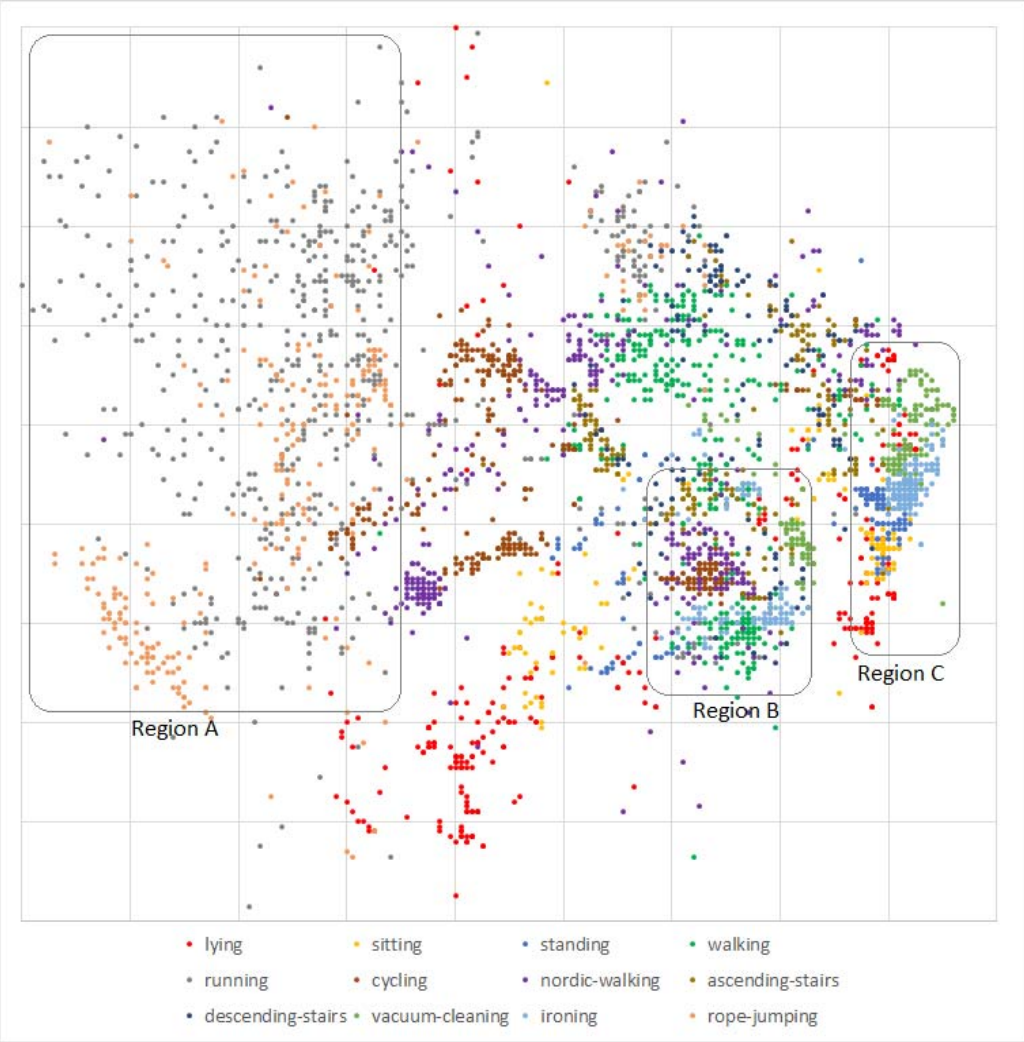


Fig. 1 Merged GSOM map generated by Distributed GSOM algorithm

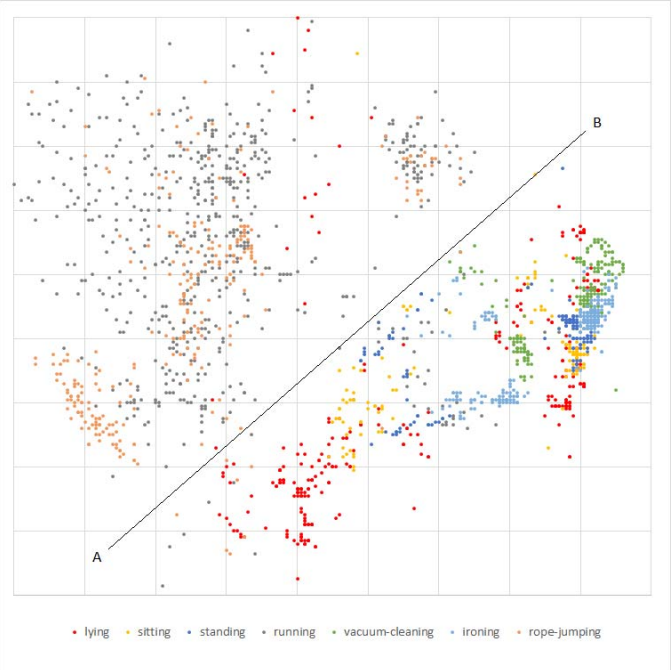

Fig. 2 Low-intensity and high-intensity activities clearly separated

To compare the execution times of the distributed and the serial version of GSOM, we implemented the serial version of GSOM and ran the algorithm on a single machine with a similar hardware specification. Total elapsed time which includes both the CPU and non-CPU time was chosen as the primary metric to compare performance between the distributed and the serial version of GSOM.

The parameters of GSOM algorithm were set as follows: spread factor, $SF = 0.1$, initial learning rate, $\alpha_0 = 0.3$ and initial neighborhood radius, $N_0 = 4$. We employed 100 growing iterations and 100 smoothing iterations in GSOM calculations. The number of data partitions, $P$ was set at 64 since the computing cluster contained 64 virtual computing cores.

### C. Results

The comparison of performance between Distributed GSOM algorithm on Apache Spark and the serial version of GSOM in terms of total elapsed time yielded significant results. The total elapsed time of the distributed version was 737s while its serial counterpart took 247,045s to execute. This is a 335.02-fold or 2 orders of magnitude improvement in the execution time.

A representative GSOM map generated by the merging process of Distributed GSOM algorithm is shown in Fig. 1. The nodes have been colored based on the data points mapped to them and we can clearly see clusters of nodes that have the same activity mapping. Moreover, similar activities have been mapped to close-by nodes and different activities have been mapped to nodes distant to each other. Plotting only the nodes that have low-intensity activities such as *lying*, *sitting*, *standing*, *ironing*, *vacuum cleaning* and high-intensity activities such as *running* and *rope-jumping* mapped to them in Fig. 2 shows that the nodes of two groups of activities are clearly separated by section AB. High-intensity activities show a greater spread while low-intensity activities are mapped more closely.

Moderate intensity activities such as *walking*, *ascending-stairs*, *descending-stairs*, *Nordic walking,* and *cycling* have been mapped to nodes in between the nodes mapped with low-intensity and high-intensity activities. Moreover, *cycling* and *Nordic walking* have been mapped to three separate clusters of nodes, the first cluster lying closer to high-intensity nodes, the second cluster lying closer to low-intensity nodes and the third cluster sitting on top of section AB which separates low and high-intensity activities. Manually inspecting the records mapped to each cluster shows that records are in fact different to each other and may pertain to different modes/sub-activities of the activity.

One of the main advantages of GSOM algorithm is multi-granular analysis using maps of varying resolution. This is facilitated by the spread factor parameter and by selecting a higher value for the parameter, the analyst can obtain a map with a higher resolution. Usually, such analysis is performed on an area of interest in the original map for finer cluster analysis. We identified 3 regions of the original map for fine-grained analysis, Regions A, B and C as marked on Fig. 1. Regions B and C were chosen as they had nodes mapped with multiple activities densely packed in them. While clusters could be clearly seen in both regions, the dense packing warranted further analysis at a higher resolution. In contrast, region A is sparsely packed with nodes mapped with high-intensity activities, *running* and *rope jumping*. These nodes are mostly intermingled. Hence, it was decided to perform a fine-grained analysis on the cluster to evaluate if the activities form their own cluster at a higher resolution.

We performed fine-granular analysis on the Region C marked on Fig. 1 using a higher spread factor, $SF = 0.3$. The clusters of activities can be seen more clearly in the spread-out map as shown in Fig. 3. The GSOM has a clearly separated

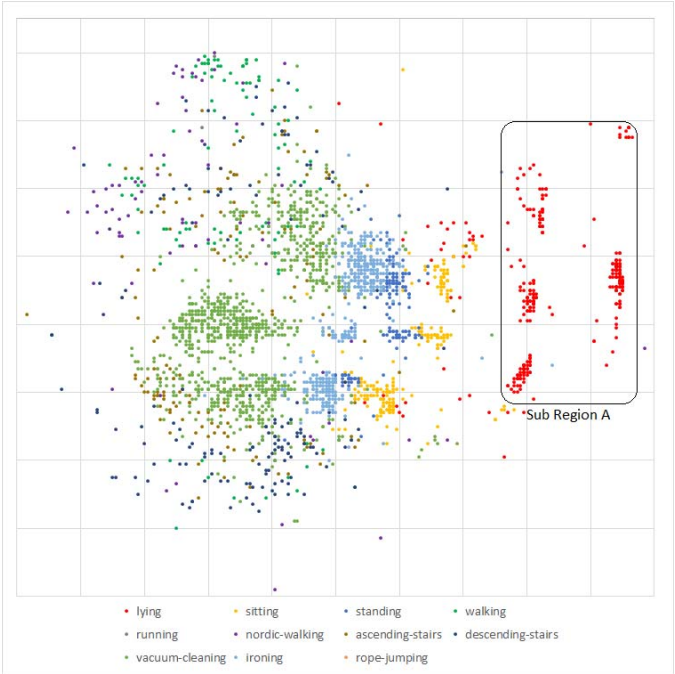

Fig. 3 Finer claster analysis of records mapped to Region C in Fig. 1, $SF = 0.3$

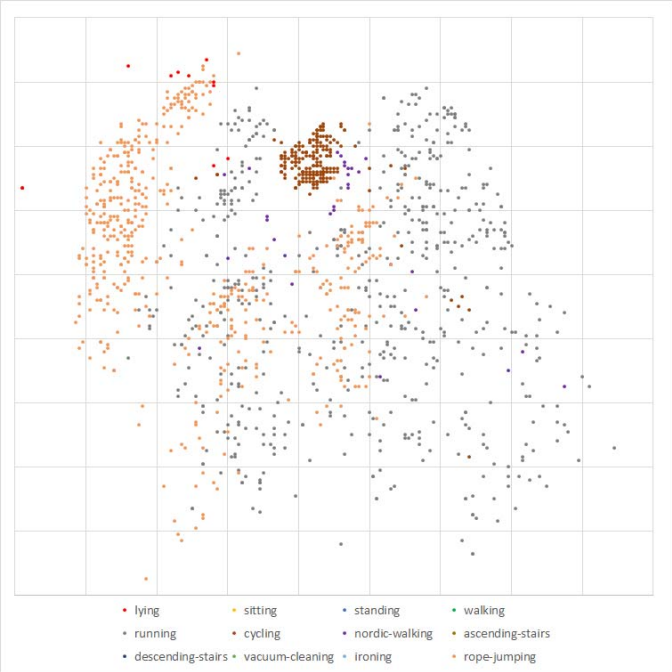


Fig. 4 Finer claster analysis of records mapped to Region A in Fig.1, *SF* = 0.3

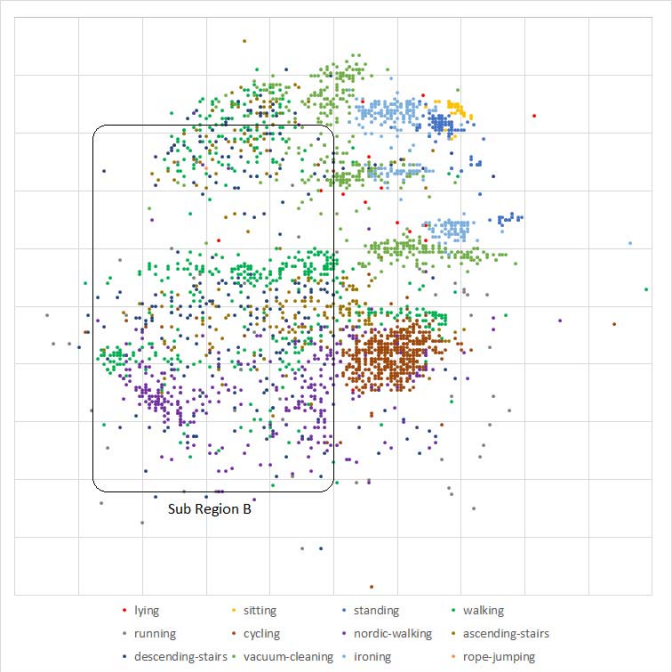


Fig. 5 Finer claster analysis of records mapped to Region B in Fig.1, *SF* = 0.3

region, Sub-Region A, which has all the records pertaining to *lying* mapped. Moreover, clusters of activities are organized in a logical order of activity intensity. The least intense activity, *lying* is mapped to the right most side of the map and moving towards left from the right edge increases the intensity of the activity mapped from *lying* to *sitting* to *standing* to *ironing* to *vacuum cleaning*. The bounding box of Region C included a small number of nodes mapped with moderately intense activities. However, these records have not formed clear clusters, rather have distributed with the higher spread factor. However, these are clearly separated from the nodes mapped with low intensive activities.

We undertook a similar fine-grained analysis of Region B with *SF* = 0.3 since the region was densely packed with nodes and the resulting map is presented in Fig. 5. The higher spread factor seems to have separated the clusters of nodes more clearly. Starting from the top right corner of the map and moving towards the bottom left corner, we can identify successive clusters of nodes mapped with *sitting*, *standing*, *ironing* and *vacuum cleaning*. Compared to the cluttered nature of Region B, these clusters are an improvement in the representation. Moreover, a distinct cluster of *cycling* can be identified towards the bottom right of the map. Although not distinctive as *cycling*, *Nordic walking* is mapped towards the bottom of the map. However, other moderate intense activities seem to have spread with each other and occupy the left half of the map. It would be an interesting exercise to analyze this region, marked as Sub-Region B, with an even higher spread factor.

Fig. 4 shows the sub-cluster analysis of the Region A of the original map. The records mapped to Region A were mostly high-intensity activities such as *running* and *rope-jumping*. The fine-grained analysis with *SF* = 0.3, has better separated these two activities with *rope-jumping* to the left of the map and *running* to the right of the map. The third activity included in the data mapped to Region A is *cycling* and this is mapped to a very distinctive cluster of its own in the top center of the map.

## V. Discussion and Conclusion

This paper proposed a new algorithm to address the growing need for scalability in cognitive computing algorithms for unlabeled Big Data sets, generated by IIoT systems in data–intensive environments. The proposed distributed GSOM algorithm is based on the RDD paradigm and implemented using the Apache Spark platform. Using a large sensor dataset on human activity containing more than 2.8 million records, we demonstrated reduced execution time leading to actionable insights from unlabeled data. Moreover, we carried out a multi-granular analysis of the dataset. The spread factor parameter of the algorithm facilitates fine-granular analysis of an area of interest of the original map at a higher resolution. This analysis showed that classes are better separated with a higher spread factor.

We engineered a number of features from the raw sensor readings. However, all features were point-based and did not take temporality of human activities into consideration. Inclusion of temporality may better represent human activities. For example [32], sampled the time-based features with a fixed-width sliding window of 2.56 seconds with 50% overlap between them. Hence, encoding temporality into features deserves further investigation.

A GSOM-based analysis is not limited to visual exploration of data. Trained GSOM can be used to classify human activities based on the winning node. Such classification may facilitate proactive instructions, interactive training for employees, quality control based on verification of procedures performed etc. as identified in the literature review. Moreover,

anomaly detection algorithms based on SOMs [33] could be used for monitoring and ensuring the safety of workers in industrial environments. The scalability of using RDD and Apache Spark, reduced execution time, unsupervised machine learning, topology preservation and multi-granular analysis of the Distributed GSOM algorithm amply demonstrate increased value for addressing Big Data problems in the cognitive computing era.

## ACKNOWLEDGMENT

This work was supported by an Australian Government Research Training Program Scholarship. Authors would also like to acknowledge the financial and in-kind support from the Data to Decisions Cooperative Research Centre (D2D CRC) as part of their analytics and decision support program.

## REFERENCES

[1] J. Kelly III and S. Hamm, *Smart Machines: IBM's Watson and the Era of Cognitive Computing*. Columbia University Press, 2013.

[2] D. O. Hebb, *The Organization of Behavior*. Wiley, 1949.

[3] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, 'Recurrent neural network based language model.', in *Interspeech*, 2010, vol. 2, p. 3.

[4] J. Hawkins and S. Blakeslee, *On Intelligence*. Macmillan, 2007.

[5] Q. V. Le, 'Building high-level features using large scale unsupervised learning', in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8595–8598.

[6] F. Seide, G. Li, and D. Yu, 'Conversational Speech Transcription Using Context-Dependent Deep Neural Networks.', in *Interspeech*, 2011, pp. 437–440.

[7] Y. Bengio, A. Courville, and I. Goodfellow, 'Deep Learning', 2016.

[8] D. D. Silva, X. Yu, D. Alahakoon, and G. Holmes, 'A Data Mining Framework for Electricity Consumption Analysis From Meter Data', *IEEE Trans. Ind. Inform.*, vol. 7, no. 3, pp. 399–407, Aug. 2011.

[9] S. M. Guru, A. Hsu, S. Halgamuge, and S. Fernando, 'An Extended Growing Self-Organizing Map for Selection of Clusters in Sensor Networks', *Int. J. Distrib. Sens. Netw.*, vol. 1, no. 2, pp. 227–243, Apr. 2005.

[10] D. D. Silva, X. Yu, D. Alahakoon, and G. Holmes, 'Incremental pattern characterization learning and forecasting for electricity consumption using smart meters', in *2011 IEEE International Symposium on Industrial Electronics*, 2011, pp. 807–812.

[11] D. D. Silva, X. Yu, D. Alahakoon, and G. Holmes, 'Semi-supervised classification of characterized patterns for demand forecasting using smart electricity meters', in *2011 International Conference on Electrical Machines and Systems*, 2011, pp. 1–6.

[12] N. Nathawitharana, D. Alahakoon, and S. Matharage, 'Improving the Decision Value of Hierarchical Text Clustering Using Term Overlap Detection', *Australas. J. Inf. Syst.*, vol. 19, no. 0, Sep. 2015.

[13] U. Gunasinghe and D. Alahakoon, 'The adaptive suffix tree: A space efficient sequence learning algorithm', in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[14] S. Matharage, H. Ganegedara, and D. Alahakoon, 'A scalable and dynamic self-organizing map for clustering large volumes of text data', in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[15] U. Gunasinghe, D. Alahakoon, and S. Bedingfield, 'Extraction of high quality k-words for alignment-free sequence comparison', *J. Theor. Biol.*, vol. 358, pp. 31–51, Oct. 2014.

[16] A. L. Hsu, S.-L. Tang, and S. K. Halgamuge, 'An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data', *Bioinformatics*, vol. 19, no. 16, pp. 2131–2140, Nov. 2003.

[17] C. Ck, H. Al, T. Sl, and H. Sk, 'Using growing self-organising maps to improve the binning process in environmental whole-genome shotgun sequencing.', *J. Biomed. Biotechnol.*, vol. 2008, pp. 513701–513701, Dec. 2007.

[18] P. Lukowicz, A. Timm-Giel, M. Lawo, and O. Herzog, 'WearIT@work: Toward Real-World Industrial Wearable Computing', *IEEE Pervasive Comput.*, vol. 6, no. 4, pp. 8–13, Oct. 2007.

[19] H. Koskimaki, V. Huikari, P. Siirtola, P. Laurinen, and J. Roning, 'Activity recognition using a wrist-worn inertial measurement unit: A case study for industrial assembly lines', in *2009 17th Mediterranean Conference on Control and Automation*, 2009, pp. 401–405.

[20] G. Kortuem *et al.*, 'Sensor Networks or Smart Artifacts? An Exploration of Organizational Issues of an Industrial Health and Safety Monitoring System', in *UbiComp 2007: Ubiquitous Computing*, 2007, pp. 465–482.

[21] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer Verlag, 1995.

[22] D. Alahakoon, S. Halgamuge, and B. Srinivasan, 'Dynamic self-organizing maps with controlled growth for knowledge discovery', *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 601–614, May 2000.

[23] H. Ganegedara and D. Alahakoon, 'Scalable Data Clustering: A Sammon's Projection Based Technique for Merging GSOMs', in *Neural Information Processing*, B.-L. Lu, L. Zhang, and J. Kwok, Eds. Springer Berlin Heidelberg, 2011, pp. 193–202.

[24] H. Ganegedara and D. Alahakoon, 'Redundancy reduction in self-organising map merging for scalable data clustering', in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[25] 'Apache Spark'. [Online]. Available: https://spark.apache.org/. [Accessed: 19-Apr-2017].

[26] M. Zaharia *et al.*, 'Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing', in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2012, pp. 2–2.

[27] h Hu, B. Tang, X. j Gong, W. Wei, and H. Wang, 'Intelligent fault diagnosis of the high-speed train with big data based on deep neural networks', *IEEE Trans. Ind. Inform.*, vol. PP, no. 99, pp. 1–1, 2017.

[28] J. Zheng and A. Dagnino, 'An initial study of predictive machine learning analytics on large volumes of historical data for power system applications', in *2014 IEEE International Conference on Big Data (Big Data)*, 2014, pp. 952–959.

[29] P. C. P. D. Silva and P. C. A. D. Silva, 'Ipanera: An Industry 4.0 based architecture for distributed soil-less food production systems', in *2016 Manufacturing Industrial Engineering Symposium (MIES)*, 2016, pp. 1–5.

[30] '[SPARK-2174][MLLIB] treeReduce and treeAggregate by mengxr · Pull Request #1110 · apache/spark', *GitHub*. [Online]. Available: https://github.com/apache/spark/pull/1110. [Accessed: 17-Apr-2017].

[31] A. Reiss and D. Stricker, 'Introducing a New Benchmarked Dataset for Activity Monitoring', in *2012 16th International Symposium on Wearable Computers*, 2012, pp. 108–109.

[32] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, R. Ortiz, and J. Luis, 'A public domain dataset for human activity recognition using smartphones', presented at the Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013, pp. 437–442.

[33] M. Ramadas, S. Ostermann, and B. Tjaden, 'Detecting Anomalous Network Traffic with Self-organizing Maps', in *Recent Advances in Intrusion Detection*, 2003, pp. 36–54.