Continual General Chunking Problem and SyncMap

Danilo Vasconcellos Vargas[1,2]**,Toshitake Asabuki**[3]

[1]Kyushu University, Fukuoka, Japan
[2]The University of Tokyo, Tokyo, Japan
[3]Okinawa Institute of Science and Technology Graduate University, Okinawa, Japan
vargas@inf.kyushu-u.ac.jp, toshitake.asabuki2@oist.jp

Abstract

Humans possess an inherent ability to chunk sequences into their constituent parts. In fact, this ability is thought to bootstrap language skills and learning of image patterns which might be a key to a more animal-like type of intelligence. Here, we propose a continual generalization of the chunking problem (an unsupervised problem), encompassing fixed and probabilistic chunks, discovery of temporal and causal structures and their continual variations. Additionally, we propose an algorithm called SyncMap that can learn and adapt to changes in the problem by creating a dynamic map which preserves the correlation between variables. Results of SyncMap suggest that the proposed algorithm learn near optimal solutions, despite the presence of many types of structures and their continual variation. When compared to Word2vec, PARSER and MRIL, SyncMap surpasses or ties with the best algorithm on 66% of the scenarios while being the second best in the remaining 34%. SyncMap's model-free simple dynamics and the absence of loss functions reveal that, perhaps surprisingly, much can be done with self-organization alone.

Introduction

Humans are able to rapidly detect patterns in sequences (Bekinschtein et al. 2009; Strauss et al. 2015). By detecting and chunking together patterns found, even without a supervised signal, humans are able to classify sounds, images and other information signals (Bulf, Johnson, and Valenza 2011), (Orbán et al. 2008). Therefore, this unsupervised learning process is thought to bootstrap many of the initial cognitive capabilities such as natural language processing, speech recognition and even image recognition.

Here, motivated by this general learning ability we propose the continual general chunking problem. To evaluate the quality of an algorithm in the proposed problem a set of tests are defined, including chunks with fixed time series, chunks based on graphs with probabilistic transitions, overlapping chunks, chunks with probabilistic cycles, temporal and causal structures identification and continual scenarios. In other words, the continual general chunking problem generalizes the chunking problem to the identification of temporal and causal structures. All this taking into consideration continual learning (change of the underlying structure as well

as the probabilistic distribution of variables throughout the experiment).Thus, the proposed problem joins the neuroscientific/psychologic chunking problem to the discovery of causal/temporal structure and unsupervised feature learning of time series.

To tackle the continual general chunking problem, we propose an algorithm, that without any parameter adjustment between problems, works for all the continual general chunking problems and achieves near optimal solutions. In other words, it can cope with probabilistic and fixed chunks as well as causal structures. This algorithm is inspired by Hebbian learning and negative feedback signals. It works by first converting inputs into spikes with slow decaying rate and creating a map on which signals self-organize through neuron group attraction/repeal forces. By creating a dynamic in which signals that activate together or deactivate together are attracted to a common center, the self-organizing dynamics are able to create clusters of correlated signals. It is worth noticing that attraction of the activated signals (which is closely related to Hebbian learning) by itself is not enough. Attraction of both activated and deactivated signals are necessary for the dynamics to reach the cohesive behavior described here.

Our Contributions

In this paper, a general problem is proposed called Continual General Chunking Problem as well as an algorithm to solve it called SyncMap. The key contributions are as follows:

- We generalize various problems from neuroscience and computer science (i.e., chunking problem, discovery of causal/temporal communities, unsupervised feature learning of time series and their continual variations) into a problem called Continual General Chunking Problem (CGCP). CGCP is defined formally and experiments are developed to evaluate an algorithm's performance. Chunking problem alone encompasses problems from learning image features to sounds as shown in (Asabuki and Fukai 2020). It originates from detecting repetitive patterns of neural spike sequences, but its primitives are thought to be widely used in the brain. Discovery of causal and/or temporal communities was explored in (Schapiro et al. 2013) with applications in (Jiao et al. 2018). Here we shown how all these problems and their continual variations can be seen as a single one.

- We propose an algorithm for tackling CGCP called SyncMap. SyncMap is a different type of self-organizing map with dynamics of attraction between all nodes that activate or deactivate at the same time. It shares with other self-organizing systems such as Self-Organizing Map (Kohonen 2013) and Novelty Map (Vargas, Takano, and Murata 2015) only the idea of using a map as all other dynamics and intent differ. Moreover, its self-organizing dynamics enables it to flexibly respond to changing environments which is a challenge for most algorithms that optimize loss functions.

- Beyond generalizing the problems, we consider (perhaps for the first time) continual variations of them in the CGCP. The challenge here is to respond quickly to the environment, adapting previous learned structures and correlations that have changed. This is motivated by the constant adaptation spotted in neural cells which can rapidly switch behavior according to environmental changes (Dahmen et al. 2010).

- Experiments on fixed chunks, probabilistic chunks and temporal structures suggest that SyncMap reaches near optimal solutions. The same is true for continual variations of them, i.e., when such probabilistic chunks or temporal structures change throughout the experiment. Moreover, we extend the tests for detecting temporal structures of real world scenarios.

Related Work

Chunking. Through the process, called "chunking", the brain attains compact representation of sequences, which is thought to reduce the complexity of temporal information processing and associated cost (Ramkumar et al. 2016). Chunking in the brain computation is crucial to achieve high-order functions that require hierarchical sequence processing, such as motor-skill learning (Graybiel 1998; Jin, Tecuapetla, and Costa 2014) and language acquisition (Buiatti, Peña, and Dehaene-Lambertz 2009; Gentner et al. 2006). Cognitive experiments suggests that children learn words as chunks (Saffran and Wilson 2003). This process is thought to contribute to higher-order learning process, and be fundamental mechanisms that children identify words from speech (Dehaene et al. 2015; Hay et al. 2011). Recent study with human subjects has shown that chunking occurs even when the sequence has co-occurring structure rather than a repetitive pattern (Schapiro et al. 2013). The influential chunking (word segmentation) algorithm, PARSER has been proposed, which extract the all frequently appeared n-grams within the sequence (Perruchet and Vinter 1998). Despite PARSER works well for simple chunking tasks and computational linguistic applications (Goldwater, Griffiths, and Johnson 2009; Feldman et al. 2013) , yet fail to detect chunks if the transition probability over all elements are uniform(Schapiro et al. 2013). Recently, biology-inspired sequence learning model, called Minimization of Regularised Information Loss (MRIL) has been proposed (Asabuki and Fukai 2020). MRIL is applicable to broad range of chunking task including uniform transitions. These studies suggest that chunking is a fundamental process, yet the mechanism of which is still elu-

sive. Albeit the multitude of applications of chunking and the wide interest of neuroscientists on the subject, this subject was not fully explored from a machine learning perspective. Chunking can be found in some articles focusing on natural language processing. Sometimes chunking is modeled as a supervised pre-processing step (Zhai et al. 2017). In other cases, it is used for unsupervised grammar induction (Ponvert, Baldridge, and Erk 2011). In both cases the problem taking into account differs from the original task independent one defined in neuroscience.

Unsupervised Learning for Sequences. Unsupervised learning for sequences usually extract features which can predict future input (Clark, Livezey, and Bouchard 2019; Hyvarinen and Morioka 2016; Lei et al. 2019; Mikolov et al. 2013; Wu, Zeng, and Yan 2018). These features, albeit descriptive of the sequences and useful for classification of sequences, do not uncover the chunks present. Related to unsupervised learning for sequences, contrastive predictive encoding (CPE) is a peculiar learning algorithm which makes use of a probabilistic contrastive loss, inducing the latent space to encode maximally useful information in its representation (Hjelm et al. 2018). It requires the sequence of samples to have some sort of order and could use a general chunking algorithm to present images that are coherent to a certain class. In this manner, CPE and its applied problem formulation is complementary rather than competing with the problem proposed here. Moreover, self-organizing maps and their variations tackle a related but different problem (Fortuin et al. 2019). They can learn topological representations of time series and static data while chunks are temporal correlations between variables, therefore, their intent differ.

Latent Variable Estimation. Latent variable estimation (Fox et al. 2011; Pfau, Bartlett, and Wood 2010; Qian and Aslin 2014; Rabin 1963) resemble some of the chunking problems defined here. However, their objective differ. Chunking is a clustering over the variables of the problem respecting their temporal correlation. Even if chunks of variables are abstracted as meta-variables (a set of variables) there is still an inherent difference between chunks and latent variables.

Word Embeddings. To transform words and paragraphs into vectors of numbers, word embeddings are used in natural language processing (Mikolov et al. 2013), (Khattak et al. 2019). Some of them are enriched with information specific to natural language processing such as FastText (Bojanowski et al. 2017) or are contextualized word embeddings such as ELMo (Peters et al. 2018) and BERT (Devlin et al. 2018). However, prediction-based word2vec embeddings (Mikolov et al. 2013) and co-occurrence matrix based GloVe embeddings (Pennington, Socher, and Manning 2014) can be also used for more general problems similar to the one presented in this paper.

Causal Graphs. As part of the general chunk problem, this paper aims to discover temporal and causal structures in sequences which are related to causal graphs. The detection

of temporal structures has many practical applications and can be used as well for facilitating the learning of causal graphs by identifying confounders, identifying correlated variables, among others (Cai and Xie 2019; Kocaoglu et al. 2019). Interestingly, some terms from causality studies also take place in chunking problems albeit in different ways. For example if variables x and y are confounded by a given variable z, if both x and y pertains to a different chunk, z is an overlapping variable. This overlapping variable can be identified as a different chunk which should facilitate the learning of causal graphs.

Continual General Chunking Problem

We define continual general chunking as the problem of extracting co-occurring states from a time series of which the underlying generation process changes depending on the task as well as throughout the task. Here, we first describe the input time series structure considered in this problem.

Our input sequence consists of state variables, transitioning by first-order Markov chain, of which the element of transition matrix is defined as: $P_{ab} = Pr[s_{t+1} = b|s_t = a]$, where s_t is the state variable at time t; and a and b are the labels of states. Here, note that each state s_t is a vector. In our sequence, each state belongs to either a fixed chunk or a probabilistic one. In a fixed chunk, given the state at the current time, the state at the next time step is selected deterministically. Let a and b be elements of a fixed chunk, with the direction a to b, then the transition matrix should satisfy: $P_{ab} = 1$. Note that since $\Sigma_b P_{ab} = 1$, the state i has only one possible next state.

In real situations, the deterministic chunks explained above is unrealistic, and the stochastic transition even within the chunks should be allowed. In order to include such cases, let us first define a structure of the input sequences. Our input can be generated by a random walk over graphs characterized by the two types of degrees- internal and external degrees. The internal degree k_a^{int} of state a is the number of connections with nodes that belong to the same chunk. The external degree k_a^{ext} is the number of links that node a connects to nodes belonging to other chunks. For all states, we assumed that the following relation holds:

$$k_a^{int} > k_a^{ext} (\forall a \in A), \tag{1}$$

where A is the set of states in the sequences. The above relations are satisfied if the graph has dense connections within chunks yet sparse connections between nodes in different chunks. The graph structure defined above is also known as communities (Radicchi et al. 2004). In our setting, the transition probabilities were assigned to all links, hence stochastic transitions were allowed in all connected states. Note that both P_{ab} and P_{ba} can be assigned in a link between a pair of states. A special case of our case was investigated in a cognitive experiment (Schapiro et al. 2013). In their setting, they assumed that transition probabilities P_{ab} are uniform over all states, which is not the case in our setting.

In our continual setting, the causal structure is behind the generation process, hence the transition matrix can change over time. In this paper, we assume that the input

sequence can take m graph structures. The label of transition matrixes, which determines the causal structure, is termed as "tasks". Assume the set of transition matrixes, $\mathbf{P} = \{P^{(1)}, P^{(2)}, \ldots, P^{(m)}\}$. At the time when the tasks are switched, one task is randomly selected from the set $\mathbf{P}$, and the time series is continuously generated. We set $m = 2$ and since the generation process itself changes, the set of state variables that constitute each chunk also change.

SyncMap

Inspired by how Hebbian volume learning complements Hebbian, taking into consideration the effects of nearby neurons learning (Mitchison and Swindale 1999) as well as how feedback systems affect the learning of algorithms for chunking problems (Asabuki and Fukai 2020); here we propose a map in which neurons in a group can learn together to represent interrelated concepts . The main idea here is to use a simplified Hebbian learning together with feedback dynamics to create a projection that encodes the probability of two variables activating at the same time with their distances in this new generated space, i.e., encoding the relative probabilistic correlation between variables as distances between them. Thus, variables that activate together will have their respective projections drawn to their middle point while variables that do not activate together will be, consequently, pulled away from each other.

SyncMap is divided into two steps: (a) the activation of nodes and their update inside the map (b) a clustering phase on the learned map, revealing their clusters. The following subsections explain these steps in detail.

Input Encoding

Let $s_t = s_{1,t}, s_{2,t}, ..., s_{n,t}$ be the set of state values at time t for $s_t \in \{0, 1\}^n : \sum_{i=1}^n s_{i,t} = 1$. The input encoding is an exponentially decaying vector x_t with the same size as the number of states:

$$x_{i,t} = \begin{cases} s_{i,ta} * e^{-0.1*(t-ta)}, & ta < m * tstep \\ 0, & otherwise \end{cases} \tag{2}$$

where ta is the most recent state transition to state s_i. State transitions happen every $tstep$ steps and variables which have their time of activation greater than $m * tstep$ are set to 0. Consequently, the system can only remember the last m variables presented. All experiments here are conducted using $m = 2$ and $tstep = 10$.

This representation of input can also be encoded differently without any change in result, e.g., by storing the last m inputs directly. In other words, the details of implementation for the input is not important for the method itself. It suffices to remember the last m states.

Dynamics

First all inputs $x_{i,t}$ have a corresponding weight $w_{i,t}$ initialized to a random position in the map $w_{i,t} \in \mathbb{R}^k$, creating a pair $(x_{i,t}, w_{i,t})$. $w_{i,t}$ is a k dimensional variable and k is a parameter defined a priori (the dimension of the map).

Every time a new input x_t is presented, each of its constituents $x_{i,t}$ are divided into activated or recently activated

(positive) and non-recently activated (negative) inputs. Specifically, all inputs are converted into two sets: positive PS_t and negative NS_t sets. Inputs with value greater than 0.1 are a member of PS_t. Otherwise, inputs are a member of NS_t. In other words, the following holds true:

$$PS_t = \{i | x_{i,t} > 0.1\} \tag{3}$$
$$NS_t = \{i | x_{i,t} \leq 0.1\} \tag{4}$$

If the cardinality of both sets are greater than one, i.e. $|PS_t| > 1$ and $|NS_t| > 1$; then the center of both sets are calculated as follows:

$$cp_t = \frac{\sum_{i \in PS_t} w_{i,t}}{|PS|} \tag{5}$$

$$cn_t = \frac{\sum_{i \in NS_t} w_{i,t}}{|NS|}, \tag{6}$$

in which cn_t and cp_t are respectively the centers of NS_t and PS_t sets. If the cardinality of both sets are not greater than nothing is updated in this iteration.

After the center of both sets are calculated, the position $w_{i,t}$ of each input is updated.

$$\phi(i,t) = \begin{cases} 1, & i \in PS_t \\ 0, & i \in NS_t \end{cases} \tag{7}$$

$$w_{i,t+1} = w_{i,t} + \alpha * \frac{\phi(i,t)*cp_t + (1-\phi(i,t))*cn_t}{\|w_{i,t} - cp_t\|} \tag{8}$$

where α is the learning rate. Subsequently, all values of $w_{i,t+1}$ are normalized to be within a hypersphere of radius $r = 1$.

Clustering

The clustering step happens after the dynamics described in the previous section is repeated for each input. In the clustering step, the projected map w is clustered to determine effectively the chunks/communities. Here, DBSCAN (Schubert et al. 2017) is used for this procedure because it does not require the number of clusters as input. The required parameter is the density of clusters eps, which is somewhat fixed for a given hypersphere of radius r, and the minimum size of each cluster mc which can also be set independent of problem. eps and mc are set to respectively 3 and 2. Having said that, other clustering algorithms together with the use of clustering analysis techniques for discovering number of clusters should be equally or even more effective. For simplicity we are narrowing the scope of this paper to DBSCAN alone.

Experiments

The experiments compose a total of nine different tests encompassing fixed chunks, mixed structures, their continual variations, long chunks, overlapped chunks and real world scenarios. Both overlapped chunks and real world scenarios have two distinctive tasks.

Settings. In all experiments, the learning algorithm is first exposed to 100000 samples of the problem and followed by an extraction of the chunks predicted. This is also true

for continual variations, where the problem changes after 100000 samples are inputted, with the second problem also presenting 100000 samples before the run is finished. All tests are run on a MacBook Pro 10.15.5 2.3Ghz 16GB laptop as they demand little computational effort.

Here we compare four distinct algorithms: SyncMap (proposed one), MRIL (Asabuki and Fukai 2020), PARSER and Word2vec. SyncMap's parameters α and k are fixed to respectively 0.1 and 3. Regarding the PARSER algorithm, since it finds possible n-grams, hence not whole chunks, we first excluded the unnecessary long n-grams ($n > 6$), and concatenated the rest of the short segments, of which share the same element. These resultant segments were regarded as "chunks" that PARSER extracted. To evaluate how a word embedding algorithm would fair in CGCP, we include a skip-gram Word2vec embedding modified to work in the context of CGCP. A dense deep neural network model was used as model for the Word2vec with a latent dimension of 3 and an output layer with softmax and size equal to the number of inputs. The chosen training parameters are 10 epochs, $1e - 3$ learning rate and 64 batch size with a mean squared error as loss. These parameters were the best performing without unnecessary slowdown after a dozen trials. A window of 100 steps was used to calculate the output probability of skip gram. The input was kept the same as SyncMap, since variations of non-decaying ones performed (perhaps surprisingly) worse. Therefore, the window for Word2vec include 10 state transitions; equivalent of 100 time steps. Regarding MRIL, we used five output neurons for all tasks, with the learning rate $1e-3$. For comparison, we used the same decaying input as SyncMap, rather than the Poisson spiking input used in the original setting of MRIL. We grouped the output neurons showing correlation larger than 0.5, and determining chunk by assigning an index of groups that maximally respond to each input.

Fixed Chunks. The problem considered here has four fixed chunks each containing three different variables. Transition between chunks happen at the end of the chunk sequence, i.e., after the third variable inside the chunk is presented. A chunk can transition to any other chunk with equal probability.

Overlapped and Long Chunks. In one hand, overlapped chunks evaluate the capability of systems to perceive chunks that share some variables. On the other hand, long chunks evaluate if the frequency of chunks affect the system. Both overlapped and long chunks are probabilistic chunks.

For the overlapped chunks, two problems are tested: overlap1 and overlap2. Overlap1 have two chunks composed respectively of variables a,b,c,d,e and d,e,f,g,h while overlap2 has chunks with respectively variables a,b,c,d,e and a,b,c,d,e,f,g,h,i,j. In other words, they either share variables of are a subset of each other. For the long chunk experiment, each chunk has four non-repeating variables a,b,c,d and e,f,g,h presented randomly each step, however, the duration of the first chunk is four transitions while the second has a stochastic duration of $5 + unif(0, 20)$ transitions.
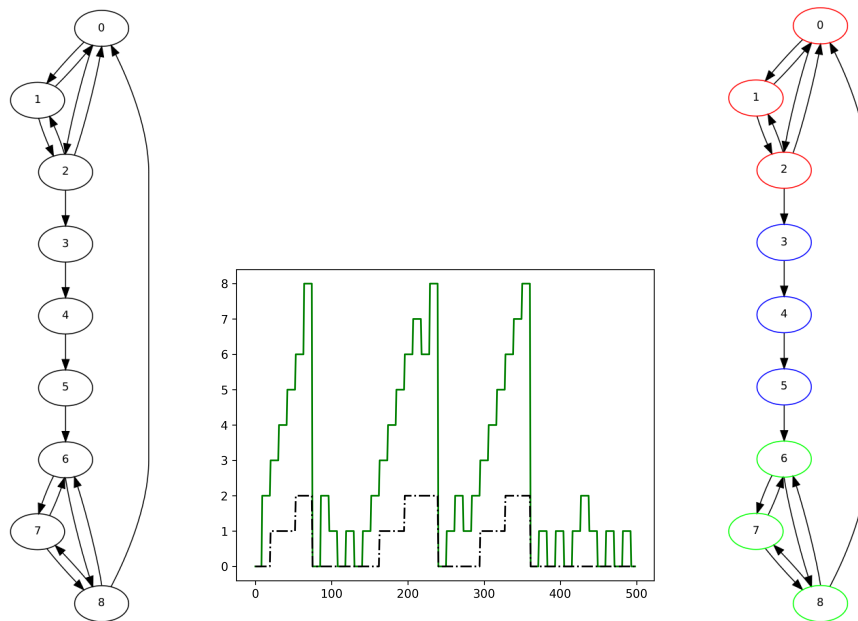
Figure 1: Problem description (up left), learned input-output map (upper middle) and respective learned clustering (upper right). The problem is a cyclic directed graph with node numbers as indicated. Each node number consists of a given input which is activated and inputted by a random walk over the graph (see the Section for more details). The learned input-output map is as follows: input (red line) and learned output (dash-dotted line). In direct correspondence to the learned output, it is possible to cluster the nodes with the colors shown (upper right). The learned map is shown in Fig 2.

$unif(min, max)$ defines a uniform distribution within the half-open interval $[min, max)$.

Mixed Structures. In this problem, both probabilistic chunks together with fixed one are present in the system. The graph with the structure and transitions of the problem is shown in the left of Fig. 1. It has 2 probabilistic chunks and one fixed chunk.

Figure 2: The learned map by SyncMap together with colors showing the chunks learned for problem described in Fig1.

Continual Variations. Two continual variations of previous problems are proposed: continual fixed and continual mixed. They are variations of respectively the fixed chunk and the mixed structures. In both variations, variables are permuted between chunks when tasks are changed. For the continual variation of the fixed chunk, in the first task the configuration of chunks is (a,b,c), (d,e,f), (g,h,i), (j,k,l). In the second task, the fixed chunks become respectively (a,k,i), (g,e,j), (d,h,c) and (f,b,l). Regarding the continual mixed problem, Fig. 3 shows how the structure of the problem changes throughout.

Real World Scenarios. We test in two variations of a real world scenario. Specifically, the recognition of probabilistic chunks in the first-order Markov model of theme transitions for humpback whales' song types (Garland et al. 2017). Since the transition between nodes are not given, they are considered equally probable. Sequence 1 and 2 are defined as respectively the graphs A and B from Fig 1 in the supplementary materials.

Results and Analysis

In this paper, we define the optimality of solutions by the degree of correlation with the ground truth. For all tests, as a correlation metric, we measured the normalized mutual information scores for Wor2vec, MRIL, PARSER and SyncMap (Tables 1, 2 and 3). Here, the normalized mutual information
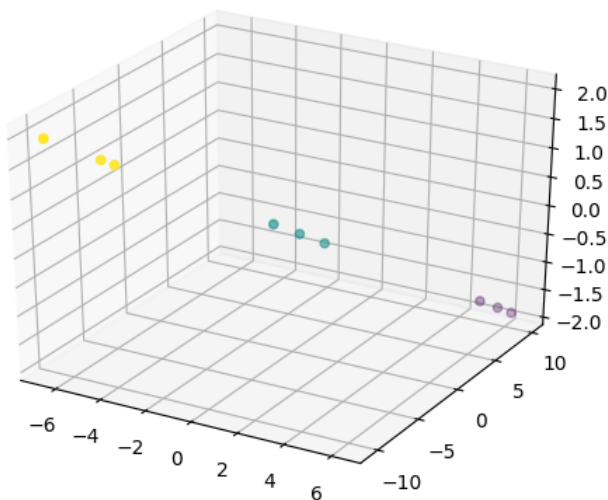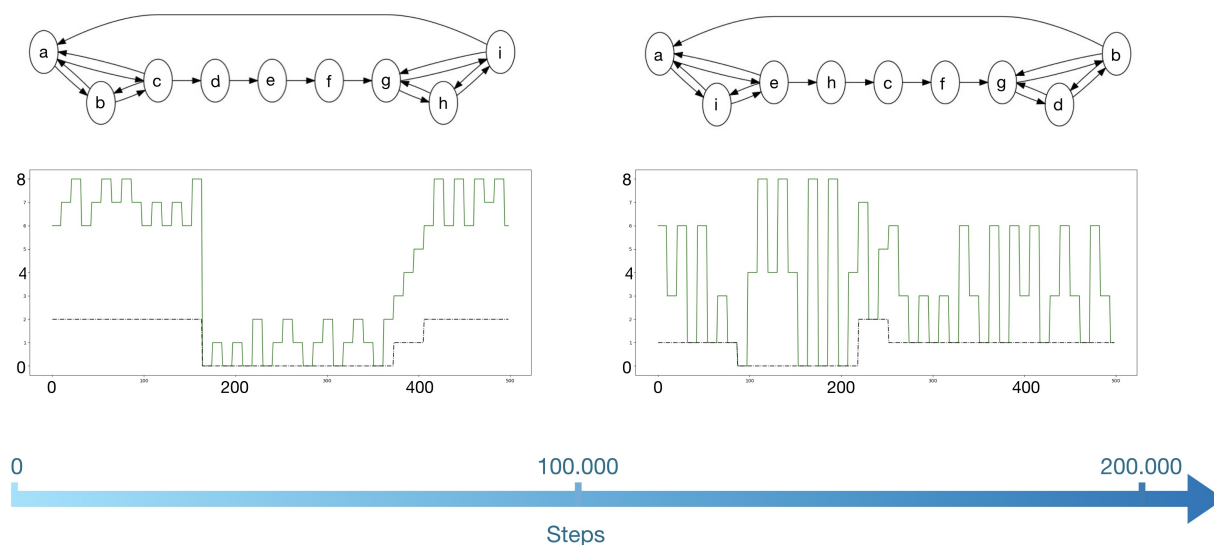
Figure 3: Illustration of the continual version of the mixed structures problem and an example of SyncMap's learning output. The structure changes after 100000 steps and then the experiment ends after 200000 steps. The upper graphs show the start (left) and end (right) problem structures and their respective variables. The start and end outputs of SyncMap are shown at the bottom.

Model	Fixed Chunks	Long Chunks	Mixed Structures
PARSER	0.95±0.07	0.16±0.27	0.36±0.40
Word2vec	**1.0±0.0**	0.68±0.22	0.78±0.07
MRIL	0.86±0.13	0.76±0.17	0.85±0.16
Ours: SyncMap	0.97±0.09	**0.97±0.06**	**0.95±0.06**

Table 1. Mutual information comparison over SyncMap, PARSER, Word2vec and MRIL in Fixed chunks, Long chunks and Mixed structures settings. Best mean and statistically similar results (results with p<0.05 in a t-test with null hypothesis of having the same mean as the best mean) are in bold.

score is defined as:

$$NMI(\hat{Y}, Y) = 2\frac{I(\hat{Y}; Y)}{H(\hat{Y}) + H(Y)}, \qquad (9)$$

where $\hat{Y}$ and Y are the output of algorithms and the ground truth, respectively. $I(\hat{Y}; Y)$ is the mutual information between $\hat{Y}$ and Y, and $H(\cdot)$ is the entropy. This NMI measure takes value between a minimum of 0 (no mutual information) and a maximum of 1 (perfect correlation).

The proposed algorithm SyncMap performed better overall. It surpassed or performed within the standard deviation of other algorithms in 7 out of the 9 tests. PARSER and Word2vec performed similarly, but had only 3 out of the 9 results that were comparable with the others. MRIL was better in only one of the tests but if SyncMap is removed from the comparison it performs slightly better than both PARSER and Word2vec. In other words, SyncMap and MRIL are both

general algorithms while Word2vec and PARSER have some specific problems they are very good at.

Regarding SyncMap, it performs similarly for long chunks and mixed structures (Table 1). Long chunks frequency is less of an issue because once variables are sufficient far away in the projected map, the update becomes weaker as well as deactivating together has also a similar attraction force. Moreover, SyncMap considers only one-to-one correlations and create groups through the emergent attraction/repulsion behavior, consequently, the nature of the chunk or structure do not matter in this regard. Continual variations of the problems (Table 2) suggests that SyncMap is capable of adapting to changes in the structure without any observed deleterious effect. This is expected since SyncMap steadily updates the correlation between variables projected into the map w. Once the dynamics reach an equilibrium the updates affect less the map distribution, however, a change in the underlying problem structure affects the place of attractors and

Model	Continual fixed		Continual mixed	
	Task 1	Task 2	Task 1	Task 2
PARSER	0.97±0.06	**0.96±0.07**	0.47±0.40	0.28±0.38
Word2vec	1.0±0.0	0.29±0.20	0.76±0.08	0.0±0.0
MRIL	0.80±0.16	0.53±0.12	0.85±0.15	0.32±0.16
Ours: SyncMap	0.93±0.13	0.89±0.15	0.97±0.04	**0.99±0.03**

Table 2. Mutual information comparison over SyncMap, PARSER, Word2vec and MRIL in continual fixed and continual mixed settings. In task 2, Best mean and statistically similar results (results with p<0.05 in a t-test with null hypothesis of having the same mean as the best mean) are in bold.

Model	Overlap1	Overlap2	Sequence1	Sequence2
PARSER	**0.77±0.42**	0.30±0.46	0.27±0.11	0.57±0.1
Word2vec	0.21±0.28	0.06±0.09	0.28±0.03	**0.79±0.08**
MRIL	0.03±0.18	0.0±0.0	**0.38±0.11**	0.51±0.10
Ours: SyncMap	**0.70±0.19**	**0.64±0.10**	**0.39±0.16**	0.61±0.06

Table 3. Mutual information comparison over SyncMap, PARSER, Word2vec and MRIL in Overlapped chunks and Real world scenarios. Sequence1 and sequence2 correspond to parts A and B in Figure 1 in the supplementary materials, respectively (problem defined in (Garland et al. 2017)). Best mean and statistically similar results (results with p<0.05 in a t-test with null hypothesis of having the same mean as the best mean) are in bold.

therefore naturally put the system in an unstable state initiating the adaptation. SyncMap performs better than the other algorithms in overlapped chunks problems (Table 3). However, there is still ground for improvement here as it cannot represent a hierarchical structure. In real world scenarios, SyncMap performs equally to all other in Sequence1 and is the second best in Sequence2. Increasing past state memory m should enable better performance.

Word2vec usually generates a map in which variables are more dispersed than the one produced by SyncMap which makes clustering difficult. For long chunks and mixed structures, the map becomes fuzzier and therefore the MI score drops accordingly. Moreover, it does not have a built-in adaptation which makes changes in the problem structure cause probabilities of nearby nodes to even out. Big overlaps do a similar effect, probabilities of nearby nodes are similar, it ends up recognizing all nodes as a single chunk. Real world sequences have mixed results, being the best in one and second worse on the other.

Regarding PARSER, it extracts chunks based on the bias of transition probabilities, therefore, performance deteriorates in time series with equal probability such as our long chunk and mixed structures. In the continual variations, PARSER performs well in both tasks involving the fixed problem since forgetting phase during learning enables it to adapt to the new environment (i.e., second task). Unlike the fixed case, it failed to learn in the mixed structures even in the first task, as shown previously. MI score for overlap2 becomes less than half of overlap1. We speculate this is because sequence of overlap2 has higher fraction of overlaps than sequence1. Therefore, the transition becomes much uniform which dete-

riorates the performance of PARSER. Since real world tasks have uniform transition part, the MI was lower than the fixed case.

Since MRIL detects spatio-temporal correlation over inputs, it showed better performance than PARSER and Word2vec if the sequence has uniform probability. MRIL learns not only the feedforward weights, but also lateral inhibition weight with the spike train correlation of output. Since each state in our sequence was presented only 10 time steps, MRIL failed to detect spike correlation by its slower timescales, hence showed MI less than 0.9 in all problems. Similarly, the MI of the second task in continual problems was lower than that of first one because lateral inhibition could not be trained efficiently. In the overlap tasks, since MRIL creates chunks including both non-overlapped and overlapped states, the MI was significantly low. In the real world problems, the MI in sequence1 was almost comparable to that of SyncMap, yet lowest in sequence2.

Conclusions

In this paper, we proposed both a general problem called CGCP and an algorithm called SyncMap which outperforms or ties with competitive algorithms from neuroscience and machine learning in 7 out of 9 tests. We expect that variations of SyncMap should further better its performance relative to other algorithms in CGCP and will probably become a strong alternative in applications from natural language processing to image recognition. Future directions will investigate problems with noise, hierarchy and causal relations as well as tasks specific to language processing and image/action recognition.

Acknowledgements

Broader Impact

In this paper, we proposed the continual general chunking problem which merges many problems from neuroscience and machine learning into a continual general one. This should promote the exchange of knowledge between distant fields for both the development of new more animal-like intelligent algorithms as well as to aid in the understanding of intelligence. The proposed algorithm, SyncMap, is an example of this with features inspired by neuroscience and the simplicity of machine learning. Therefore, we believe that many researchers beyond machine learning may benefit from this research. There is nobody put at disadvantage with this research. Both the problem and the proposed solution are new developments which aim at increasing overall generality and robustness of AI systems. Failure is expected to decrease if more importance is giving to such general problems as proposed here. CGCP aims at evaluating algorithms in many variations of problem types and settings. Therefore algorithms are not able to take advantage from biases even from specific problem classes.

References

Asabuki, T.; and Fukai, T. 2020. Somatodendritic consistency check for temporal feature segmentation. *Nature communications* 11(1): 1–13.

Bekinschtein, T. A.; Dehaene, S.; Rohaut, B.; Tadel, F.; Cohen, L.; and Naccache, L. 2009. Neural signature of the conscious processing of auditory regularities. *Proceedings of the National Academy of Sciences* 106(5): 1672–1677. ISSN 0027-8424. doi:10.1073/pnas.0809667106. URL https://www.pnas.org/content/106/5/1672.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5: 135–146. doi:10.1162/tacl_a_00051.

Buiatti, M.; Peña, M.; and Dehaene-Lambertz, G. 2009. Investigating the neural correlates of continuous speech computation with frequency-tagged neuroelectric responses. *Neuroimage* 44(2): 509–519.

Bulf, H.; Johnson, S. P.; and Valenza, E. 2011. Visual statistical learning in the newborn infant. *Cognition* 121(1): 127–132.

Cai, R.; and Xie, F. 2019. Triad constraints for learning causal structure of latent variables. *Advances in neural information processing systems* .

Clark, D.; Livezey, J.; and Bouchard, K. 2019. Unsupervised discovery of temporal structure in noisy data with dynamical components analysis. In *Advances in Neural Information Processing Systems*, 14267–14278.

Dahmen, J. C.; Keating, P.; Nodal, F. R.; Schulz, A. L.; and King, A. J. 2010. Adaptation to stimulus statistics in the per-

ception and neural representation of auditory space. *Neuron* 66(6): 937–948.

Dehaene, S.; Meyniel, F.; Wacongne, C.; Wang, L.; and Pallier, C. 2015. The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees. *Neuron* 88(1): 2–19.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Feldman, N. H.; Griffiths, T. L.; Goldwater, S.; and Morgan, J. L. 2013. A role for the developing lexicon in phonetic category acquisition. *Psychological review* 120(4): 751.

Fortuin, V.; Hüser, M.; Locatello, F.; Strathmann, H.; and Rätsch, G. 2019. SOM-VAE: Interpretable discrete representation learning on time series. In *ICLR 2019*.

Fox, E. B.; Sudderth, E. B.; Jordan, M. I.; and Willsky, A. S. 2011. A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics* 1020–1056.

Garland, E. C.; Rendell, L.; Lamoni, L.; Poole, M. M.; and Noad, M. J. 2017. Song hybridization events during revolutionary song change provide insights into cultural transmission in humpback whales. *Proceedings of the National Academy of Sciences* 114(30): 7822–7829.

Gentner, T. Q.; Fenn, K. M.; Margoliash, D.; and Nusbaum, H. C. 2006. Recursive syntactic pattern learning by songbirds. *Nature* 440(7088): 1204–1207.

Goldwater, S.; Griffiths, T. L.; and Johnson, M. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition* 112(1): 21–54.

Graybiel, A. M. 1998. The basal ganglia and chunking of action repertoires. *Neurobiology of learning and memory* 70(1-2): 119–136.

Hay, J. F.; Pelucchi, B.; Estes, K. G.; and Saffran, J. R. 2011. Linking sounds to meanings: Infant statistical learning in a natural language. *Cognitive psychology* 63(2): 93–106.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* .

Hyvarinen, A.; and Morioka, H. 2016. Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In *Advances in Neural Information Processing Systems*, 3765–3773.

Jiao, P.; Yu, W.; Wang, W.; Li, X.; and Sun, Y. 2018. Exploring temporal community structure and constant evolutionary pattern hiding in dynamic networks. *Neurocomputing* 314: 224–233.

Jin, X.; Tecuapetla, F.; and Costa, R. M. 2014. Basal ganglia subcircuits distinctively encode the parsing and concatenation of action sequences. *Nature neuroscience* 17(3): 423–430.

Khattak, F. K.; Jeblee, S.; Pou-Prom, C.; Abdalla, M.; Meaney, C.; and Rudzicz, F. 2019. A survey of word embeddings for clinical text. *Journal of Biomedical Informatics* 10(4): 100057.

Kocaoglu, M.; Jaber, A.; Shanmugam, K.; and Bareinboim, E. 2019. Characterization and learning of causal graphs with latent variables from soft interventions. In *Advances in Neural Information Processing Systems*, 14346–14356.

Kohonen, T. 2013. Essentials of the self-organizing map. *Neural networks* 37: 52–65.

Lei, Q.; Yi, J.; Vaculin, R.; Wu, L.; and Dhillon, I. S. 2019. Similarity preserving representation learning for time series clustering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2845–2851. AAAI Press.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mitchison, G. J.; and Swindale, N. V. 1999. Can Hebbian volume learning explain discontinuities in cortical maps? *Neural computation* 11(7): 1519–1526.

Orbán, G.; Fiser, J.; Aslin, R. N.; and Lengyel, M. 2008. Bayesian learning of visual chunks by human observers. *Proceedings of the National Academy of Sciences* 105(7): 2745–2750.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. *Glove: Global vectors for word representation*. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).

Perruchet, P.; and Vinter, A. 1998. PARSER: A model for word segmentation. *Journal of memory and language* 39(2): 246–263.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, 2227–2237.

Pfau, D.; Bartlett, N.; and Wood, F. 2010. Probabilistic deterministic infinite automata. In *Advances in neural information processing systems*, 1930–1938.

Ponvert, E.; Baldridge, J.; and Erk, K. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 1077–1086. Association for Computational Linguistics.

Qian, T.; and Aslin, R. N. 2014. Learning bundles of stimuli renders stimulus order as a cue, not a confound. *Proceedings of the National Academy of Sciences* 111(40): 14400–14405.

Rabin, M. O. 1963. Probabilistic automata. *Information and control* 6(3): 230–245.

Radicchi, F.; Castellano, C.; Cecconi, F.; Loreto, V.; and Parisi, D. 2004. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences* 101(9): 2658–2663. ISSN 0027-8424. doi:10.1073/pnas.0400054101. URL https://www.pnas.org/content/101/9/2658.

Ramkumar, P.; Acuna, D. E.; Berniker, M.; Grafton, S. T.; Turner, R. S.; and Kording, K. P. 2016. Chunking as the result of an efficiency computation trade-off. *Nature communications* 7(1): 1–11.

Saffran, J.; and Wilson, D. 2003. From syllables to syntax: Multilevel statistical learning by 12-month-old infants. *Infancy* 4: 273–284. doi:10.1207/S15327078IN0402_07.

Schapiro, A. C.; Rogers, T. T.; Cordova, N. I.; Turk-Browne, N. B.; and Botvinick, M. M. 2013. Neural representations of events arise from temporal community structure. *Nature neuroscience* 16(4): 486.

Schubert, E.; Sander, J.; Ester, M.; Kriegel, H. P.; and Xu, X. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42(3): 1–21.

Strauss, M.; Sitt, J. D.; King, J.-R.; Elbaz, M.; Azizi, L.; Buiatti, M.; Naccache, L.; van Wassenhove, V.; and Dehaene, S. 2015. Disruption of hierarchical predictive coding during sleep. *Proceedings of the National Academy of Sciences* 112(11): E1353–E1362. ISSN 0027-8424. doi:10.1073/pnas.1501026112. URL https://www.pnas.org/content/112/11/E1353.

Vargas, D. V.; Takano, H.; and Murata, J. 2015. Novelty-organizing team of classifiers in noisy and dynamic environments. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2937–2944. IEEE.

Wu, J.; Zeng, W.; and Yan, F. 2018. Hierarchical Temporal Memory method for time-series-based anomaly detection. *Neurocomputing* 273: 535–546.

Zhai, F.; Potdar, S.; Xiang, B.; and Zhou, B. 2017. Neural models for sequence chunking. In *Thirty-First AAAI Conference on Artificial Intelligence*.