Dynamic learning rates for Continual Unsupervised Learning

José David Fernández-Rodríguez [a,b,*], Esteban José Palomo [a,b],
Juan Miguel Ortiz-de-Lazcano-Lobato [a,b], Gonzalo Ramos-Jiménez [a,b] and Ezequiel López-Rubio [a,b]

[a] *Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur, 35, 29071, Málaga, Spain*

E-mails: josedavid@lcc.uma.es, ejpalomo@lcc.uma.es, jmortiz@lcc.uma.es, ramos@lcc.uma.es, ezeqlr@lcc.uma.es

[b] *Biomedic Research Institute of Málaga (IBIMA), C/ Doctor Miguel Díaz Recio, 28, 29010, Málaga Spain*

Abstract. The dilemma between stability and plasticity is crucial in machine learning, especially when non-stationary input distributions are considered. This issue can be addressed by continual learning in order to alleviate catastrophic forgetting. This strategy has been previously proposed for supervised and reinforcement learning models. However, little attention has been devoted to unsupervised learning. This work presents a dynamic learning rate framework for unsupervised neural networks that can handle non-stationary distributions. In order for the model to adapt to the input as it changes its characteristics, a varying learning rate that does not merely depend on the training step but on the reconstruction error has been proposed. In the experiments, different configurations for classical competitive neural networks, self-organizing maps and growing neural gas with either per-neuron or per-network dynamic learning rate have been tested. Experimental results on document clustering tasks demonstrate the suitability of the proposal for real-world problems.

Keywords: continual learning, unsupervised learning, competitive neural network, self-organizing map, growing neural gas, document clustering

1. Introduction

Humans are able to adapt to the changing environments around them in an efficient and robust way through a sequence of experiences that allow them to incrementally learn what they need from these non-stationary environments and retain vital information for the future. This feature has not been emulated by machine learning models yet. They include a wide variety of methods that have been successfully applied to such diverse tasks as job scheduling [1], natural disaster prediction [2], material production [3, 4], structure inspection and defect detection [5–8], biomedical tasks [9–11] and economic ones [12–14]. Furthermore, they excel in tasks that are dependent on image processing and analysis [15–17] due to the improvement of deep learning models. However, machine learning training is typically based on a data set that is assumed to maintain the same data distribution over time.

In recent times, continuous learning has emerged as one of the areas of study with the greatest potential for endowing learning systems with the ability to work with non-stationary data distributions [18, 19]. It is essential for the learning models to be sufficiently plastic to incorporate more current knowledge while avoiding forgetting information acquired at the beginning of their training, which in its most extreme case would lead to a situation known as catastrophic forgetting [20, 21], in which the models would continually discard information learned in the past and would only retain the latest data presented to them. However, it is not easy to find a compromise between plasticity and the stability necessary for the model to converge to a steady state that maintains the meaningful information extracted throughout the whole training [22]. Although multiple continuous learning models based on supervised and reinforcement learning have been proposed [23–26], the development of unsupervised models that attempt to cluster input data whose distribution changes over time is still a challenging field.

[*] Corresponding author. E-mail: josedavid@lcc.uma.es.

Unsupervised learning methods do not need an input data set labeled by a human expert. They usually attempt to detect similar features or common patterns that allow them to group the input data into different clusters or categories [27]. Knowledge about the data similarity within each cluster can be used to find association rules between features [28]. Furthermore, each cluster information may be useful for representing data that belongs to that cluster in a reduced form [29]. For that purpose, only the most significant features are retained. Those that provide little or no information are removed. The present work is focused on a specific paradigm within unsupervised learning: competitive learning. This paradigm is represented by a family of classical neural networks where neurons compete to represent data samples. The early stages of the training process greatly influence the information learned by the neural network. A change in the distribution of data when the network is practically trained does not greatly affect its final configuration. In order to learn those new data properties, the neural networks would have to be retrained. The aim of this work is to propose a framework in which competitive networks could learn continuously. Within this family, we experiment with three network architectures: competitive neural network, self-organizing map, and growing neural gas. Each new model has been applied to the problem of online document clustering and its performance has been tested.

The remainder of the paper is structured as follows. Section 1.1 comprises works related to competitive neural networks and online document clustering. The competitive and self-organized continuous learning models are presented in Section 2, while Section 3 exposes our proposal to adapt them to continual learning, as well as the metrics used to evaluate the performance. Section 4 aims to present the results of the experiments. Subsequently, Section 5 is devoted to discussing the results. Finally, Section 6 presents some conclusions.

1.1. Related works

A competitive neural network is designed to perform vector quantization of the input space once it is trained [30] Despite being relatively simple models, competitive networks have achieved a good performance in applications with high-dimensional data such as images or video sequences. Color image segmentation methods where competitive models are an essential component are presented in [31, 32]. Regarding the analysis of video sequences, [33] proposes the use of rival penalized competitive learning [34] to categorize the body postures of human beings appearing in the video frames, which allows the system to extract behavioral patterns. Finally, a probabilistic competitive neural model to compress multispectral data by estimating each cluster's intrinsic dimension is presented in [33].

Self-organizing maps can be considered a refinement of competitive learning networks, organizing the neurons into a graph representing a low-dimensional grid . Specialized variants of self-organizing maps have been adapted for many different engineering tasks, such as foreground detection in video sequences [35], brain-computer interfaces [36], intrusion detection in sensor networks [37], monitoring of industrial processes [38], or object detection in controlled environments [39].

A growing neural gas is another kind of competitive learning model where neurons are organized into a graph whose topology and size are dynamic. The number of neurons and the connections among them may be increased or decreased as it is required to adapt the number of clusters accordingly to the distribution of the training data. This model and various variants have also been applied to a range of engineering tasks, such as temporal video segmentation [40], foreground detection in video sequences [41], automatic recognition of relevant landmarks in medical images [42], modeling of biodiversity data [43], 3D structure reconstruction [44], or vehicle classification [45].

The aim of this work is to provide a dynamic learning rate framework for unsupervised learning models in order to adapt them for continual learning. This way, these models can learn from non-stationary distributions in a continual way, as in many real-world problems, addressing the stability and plasticity dilemma. This framework is tested with an array of experiments to cluster CORA [46], a dataset of scientific articles. While this dataset has been used to test supervised continual learning for Graph Neural Networks using the dataset's citation network [47–49], our approach is the first application to continual learning with unsupervised competitive learning using the vocabulary metadata in the dataset. Other works for online clustering of documents use probabilistic models [50], various online variants of K-means clustering [51, 52], or fuzzy clustering [53], but no competitive learning approaches such as the self-organizing map o the growing neural gas, whose update rules update neurons (clusters) not in isolation but in neighborhoods, therefore providing a mechanism to avoid individual clusters to get stuck on low local optima [54]. Besides online document clustering, there are also works developing online clustering of images or image patches [55–58], online clustering of sound samples [59], or the more generic multi-armed bandit problem [60].

2. Preliminaries

The three unsupervised learning algorithms used in this paper are competitive learning, Kohonen's Self-Organizing Map (SOM), and Growing Neural Gas (GNG). They are briefly reviewed in this section to present the notation that will be used throughout the paper.

2.1. Competitive neural networks

In a competitive learning network composed of K neurons, the training phase can be understood as a sequence of steps during which the network learns to represent the distribution of the input data. Each neuron i has a weight vector $\mathbf{w}_i(t)$, which estimates the centroid of the cluster which the neuron represents. At each step t, a sample (data vector) $\mathbf{x}(t)$ from the input data is presented to the network, and the selected neuron s is the one with the closest weight vector in the input space, i.e., the neuron whose weight vector is most similar to the input vector. The Euclidean distance $\|\cdot\|$ is chosen as the measure to estimate the similarity between a neuron weight vector $\mathbf{w}_i(t)$ and the input sample $\mathbf{x}(t)$ at training step t:

$$s = winner(\mathbf{x}(t)) = \arg \min_{i\in\{1,\ldots,K\}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\| \tag{1}$$

Then, only the winning neuron s is updated according to the following rule:

$$\mathbf{w}_s(t+1) = \mathbf{w}_s(t) + \alpha(t)\left(\mathbf{x}(t) - \mathbf{w}_s(t)\right) \tag{2}$$

In that rule, $\alpha(t)$ is a scalar factor called learning rate that represents the size of the correction and is monotonically decreasing with respect to the training step t:

$$\alpha(t) = \frac{\alpha(t-1)}{d(t)} \tag{3}$$

where $d(t)$ is a decay function. Typically, the training phase is divided into two stages. In the first one, the neurons are widely but roughly distributed to model the distribution of the input data. For that purpose, the learning rate usually decreases linearly with respect to the time step t, i.e., $d(t) = 1 + at$, or exponentially, i.e., $d(t) = e^{at}$, with $a \in \mathbb{R}^+$. After that, a second stage is intended to fine-tune the positions of the weight vectors, and the learning rate is held constant at a low value.

Typically, the time steps in the training phase are grouped into a number of *epochs*: in each epoch, all samples from the training dataset are presented (in random order) to the training network. After the training phase, the weights are fixed, and each new input vector is assigned to the cluster corresponding to the neuron with the closest weight vector. If the training was successful, the weights of the neurons are distributed so that they represent the cluster centroids of a Voronoi tesselation. Please note that other competitive learning networks such as SOM and GNG share most of the basic setup described here for competitive networks.

2.2. The Self-Organizing Map (SOM)

The Self-Organizing Map (SOM) is an unsupervised clustering technique designed to learn a low-dimensional representation of higher-dimensional data. This representation is frequently a 2D neuron grid (but the dimensionality and size of the grid may be changed), and it aims to preserve the topology of the input data. Each neuron represents a cluster of input samples. Formally, in a SOM with a 2D grid, K is the number of neurons, which are arranged in a lattice of size $a \times b$, where $K = ab$. The topological distance between neurons i and j at positions $\mathbf{r}_i, \mathbf{r}_j \in \mathbb{R}^2$ in the grid is defined as:

$$d(i, j) = \left\|\mathbf{r}_i - \mathbf{r}_j\right\| \tag{4}$$

For each neuron i, its weight vector $\mathbf{w}_i$ is a point in the input space, and it represents the centroid of the associated cluster. At each time step t during training, a sample $\mathbf{x}(t)$ is presented to the network and the closest neuron s is declared as the winner (Eq. 1).

Since the SOM is a self-organizing neural network, not only is the winner updated as in the competitive neural network, but also the rest of the neurons are adjusted, for $i \in \{1, \ldots, K\}$:

$$\mathbf{w}_i\,(t+1) = \mathbf{w}_i\,(t) + \alpha\,(t)\,\Lambda\,(i,s)\,(\mathbf{x}\,(t) - \mathbf{w}_i\,(t)) \tag{5}$$

where $\alpha\,(t)$ is a learning rate monotonically decreasing with respect to the training step t as shown in (3). Λ is a neighborhood function depending on $\Delta\,(t)$, a monotonically decreasing *neighborhood radius* which is also decreased as in (3):

$$\Lambda\,(i,s)) = \exp\left(-\left(\frac{d\,(i,s))}{\Delta\,(t)}\right)^2\right) \tag{6}$$

$$\Delta\,(t+1) \leqslant \Delta\,(t) \tag{7}$$

According to these training dynamics, please note that the neurons are not necessarily placed in the data's higher dimensional space as a smooth 2D grid. At the beginning of the training, the neurons' positions are initialized by taking the mean of a random sample of data points; the resulting initial embedding is typically a random assemblage around the dataset's centroid. Each neuron will influence (and be influenced by) the position of their topological neighbors, and for well-behaved datasets, this will translate into the 2D grid being embedded as a smooth sheet modeling the data's distribution, but this will not always be the case, generally speaking.

2.3. *The Growing Neural Gas (GNG)*

The Growing Neural Gas (GNG) is another self-organizing neural network. The main difference with the SOM is that it learns a dynamic graph with a variable number of neurons and connections instead of a two-dimensional fixed topology. This set of connections will be noted $A \subseteq \{1,...,K\} \times \{1,...,K\}$. Each connection has an associated age, which is a non-negative integer. Every neuron i has an error variable $e_i \in \mathbb{R}$, $e_i \geqslant 0$, in addition to a weight vector $\mathbf{w}_i$ which represents a cluster of input samples. In this model, when a new input sample $\mathbf{x}(t)$ is presented to the network, the nearest unit s_1 and the second nearest unit s_2 are computed:

$$s_1 = \arg\min_{i\in\{1,...,K\}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\| \tag{8}$$

$$s_2 = \arg\min_{i\in\{1,...,K\}-\{s_1\}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\| \tag{9}$$

After incrementing the age of all connections emanating from s_1, the squared Euclidean distance between the input sample $\mathbf{x}(t)$ and the nearest neuron s_1 is added to the error variable e_{s_1}:

$$e_{s_1}(t+1) = e_{s_1}(t) + \left\|\mathbf{x}(t) - \mathbf{w}_{s_1}(t)\right\|^2 \tag{10}$$

Then the weights of the nearest unit s_1 and its direct topological neighbors j in the graph are updated:

$$\mathbf{w}_{s_1}(t+1) = \mathbf{w}_{s_1}(t) + \epsilon_b\,(\mathbf{x}(t) - \mathbf{w}_s(t)) \tag{11}$$

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \epsilon_n\,(\mathbf{x}(t) - \mathbf{w}_s(t)) \tag{12}$$

where every neuron j is a direct topological neighbor of the nearest neuron s_1 ($(j, s_1) \in A$), ϵ_b is the learning rate of s_1, and ϵ_n is the learning rate of every neuron j. Therefore, the GNG has two learning rates, ϵ_b for the nearest neuron s_1, and ϵ_n for the neurons directly connected to the activated neuron (s_1) in each training step.

The GNG can create or remove neurons and connections. Therefore, the connection between s_1 and s_2 is set to zero if this connection already exists. If such a connection does not exist, then it is created. Also, connections with an age larger than a parameter a_{max} are removed, and neurons without connections are also removed. Then if the number of steps is multiple of a parameter λ, a new neuron r is added to the network between the neuron q with the largest accumulated error and its neighbor f with the largest error value. The weight vector of the new neuron r (w_r) is given as follows:

$$\mathbf{w}_r = 0.5(\mathbf{w}_q + \mathbf{w}_f) \tag{13}$$

where w_q is the weight vector of the neuron q with the largest accumulated error, and w_f is the weight vector of its neighbor f with the largest error value. Connections between the new neuron r and neurons q and f must be inserted, and the original connection between q and f is removed. Finally, the error variables of all neurons are updated.

3. Methodology

Our goal is to adapt the models presented in Section 2 to make them able to learn continuously: instead of training ahead of time to learn a specific clustering and then performing inference, the neural network should be able to adapt and change the configuration of its weight vectors to represent a different input distribution if it is required, in order to be able to learn continuously. Accordingly, the training regime should be changed so that it can be run in parallel to inference. Therefore, we propose to make the learning rate a function based on the reconstruction error.

Regarding our methodology, first, we present the modifications performed to adapt the unsupervised learning algorithms shown in Section 2 to continual learning. Second, the multi-learning rate approach is introduced, as a further adaptation to continual learning. Finally, the clustering metrics used to measure the performance are given.

3.1. Adaptation to continual learning

To achieve the goal of adapting competitive models to train indefinitely, we propose a different scaling strategy during training: instead of using a learning rate α purely dependent on t as shown in (3), a function that relies on the Euclidean distances between samples and the corresponding winning neuron weights $\|\mathbf{x}(t) - \mathbf{w}_s(t)\|$ in the last N time steps, from t to $t - N + 1$, is defined. We will note the Euclidean distance as $e(t)$ because it can be seen as the error at each time step t.

The core idea is that if the input distribution changes, the $e(t)$ distances are expected to increase, thus increasing the scaling factor α, which allows the neural model neurons to update their internal structures to a greater extent. As a consequence, the neurons will be redistributed in the input space, and the neural model will adapt to the new input distribution effectively. To reduce the effect of noise and outliers, which could hinder the learning process of the network, due to an excess of change in neurons in a sudden way, the median of these reconstruction errors $e(t)$ (from the last N time steps in the training process) is proposed as a consensus measure, noted as $C(t)$. Then, that value is used as input for a monotonically increasing function f:

$$C(t) = median_{t-N+1 \leqslant j \leqslant t}(e(j)) \tag{14}$$

$$\alpha(t) = f(C(t)) \tag{15}$$

This new way to update the learning rate is intended to be used from the beginning of the training, with no warm-up period using the standard way described in the previous section. Different monotonic function types are considered in the experiments:

- Linear: $\alpha(t) = aC(t)$
- Quadratic: $\alpha(t) = a\,(C\,(t))^2$
- Inverse: $\alpha(t) = a\,(C\,(t))^{-1}$
- Constant: $\alpha(t) = a$

While a constant learning rate is not adaptive, it is included for comparison with the other ones. In the context of the updating rule, it only makes sense for $\alpha(t)$ to take values in the $[0 \ldots 1]$ range. Therefore, the results of the first three functions are clamped to that interval. All the functions have been modeled with just a single parameter a to limit the parameter space to search, as well as to avoid non-monotonic function shapes. Note that the number of parameters remains the same since parameter a replaces the parameter of the same name from Eq. (3).

3.2. Multi-learning rate

Note that, as described above, the $e(t)$ distances from all winning neurons in the last N steps contribute to a **single** learning rate applied to the winning neuron in step t. It is possible, however, that the input distribution changes only in limited ways. We define a variant of the model with multiple learning rates (noted as **multi** in Section 4), one per neuron, so that the learning rate of each neuron i can change independently and is determined as a monotonic function of the median of the previous N errors $e(t)$. Note, however, that in this approach the considered distances are from the previous N times that neuron i was the winner, not (necessarily) from the last N time steps in the training process. Although a different type of monotonic function could be employed for each neuron, for the sake of model simplicity, we decided to use the same type of function for all neurons of the neural network. As the learning rate can be fine-tuned for each neuron , those ones representing relatively infrequent parts of the input distribution can keep a learning rate commensurate to the error distances within its receptive field without being affected by error distances from other neurons.

3.3. Clustering Performance Measures

Next, we review the clustering performance measures that will be used in this work.

The first one is the Mean Squared Error (MSE), which measures the average of the squares of the errors, that is, the average squared difference between the estimated values (weight vectors) and the actual values (input samples), lower is better:

$$MSE = \frac{1}{M}\sum_{k=1}^{M} \|\mathbf{w}_i - \mathbf{x}_k\|^2 , \quad i = 1, \cdots, K \tag{16}$$

where M is the number of samples in the distribution, K is the number of clusters, $\mathbf{x}_k$ is the k-th input sample, and $\mathbf{w}_i$ the prototype of the winning neuron i corresponding to $\mathbf{x}_k$.

The Davies-Bouldin index [61] is a well-known measure which favors compact and well-separated clusters [62, 63]. It is given by (lower is better):

$$DB = \frac{1}{K}\sum_{i=1}^{K} \max_{j:i\neq j} \frac{\sigma_i^2 + \sigma_j^2}{\left\|\mathbf{w}_i - \mathbf{w}_j\right\|^2} \tag{17}$$

where σ_i^2 measures the spread of cluster C_i,

$$\sigma_i^2 = \frac{1}{|C_i|}\sum_{\mathbf{x}_k \in C_i} \|\mathbf{w}_i - \mathbf{x}_k\|^2 \tag{18}$$

and $|C_i|$ stands for the cardinality of cluster C_i.

The original Dunn index [64] tries to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within-cluster variance. This index has been improved in several ways to make it more robust and computationally efficient. The particular version that we will consider here is one of those advocated in [65] (higher is better):

$$Dunn = \min_{i\in\{1,\ldots,K\}} \left\{ \min_{j:i\neq j} \frac{\left\|\boldsymbol{w}_i - \boldsymbol{w}_j\right\|}{\Delta} \right\} \tag{19}$$

$$\Delta = \max_{i\in\{1,\ldots,K\}} \frac{1}{|C_i|\left(|C_i| - 1\right)} \sum_{\mathbf{x}_k \in C_i} \sum_{\mathbf{x}_l \in C_i - \{\mathbf{x}_k\}} \|\mathbf{x}_l - \mathbf{x}_k\| \tag{20}$$

Table 1

Configuration values used in the experiments for the competitive learning networks modified for online learning. The first one is the type of competitive learning network. All other ones are specific to our proposal for online learning in these networks. See Section 4.2 for details.

Parameter description	Values
Network model	competitive, SOM, GNG
history size N	10, 50, 100
learning rate mode	single, multi
function type $\alpha(t)$	$a \cdot C(t)$, $a \cdot C(t)^2$, $a \cdot C(t)^{-1}$, a
function parameter a	40 logarithmically spaced values in $[0.001, 5]$ (30 log. spaced v. in $[0.001, 1]$ for $\alpha(t) = a$)

The Calinski-Harabasz criterion [66] measures the similarity of a sample with respect to its own cluster as compared to the separation between clusters. The distances from the cluster's centroid $\mathbf{w}_i$ to each one of its samples $\mathbf{x}_k$ are used to estimate the similarity. Separation is estimated using the distance of the cluster centroids $\mathbf{w}_i$ to the global centroid $\bar{\mathbf{w}}$. The criterion is defined as (higher is better):

$$CH = \left[\frac{\sum_{i=1}^{K} |C_i| \, \|\mathbf{w}_i - \bar{\mathbf{w}}\|^2}{K-1}\right] \Big/ \left[\frac{\sum_{i=1}^{K}\sum_{k=1}^{|C_i|} \|\mathbf{w}_i - \mathbf{x}_k\|^2}{N-K}\right] \tag{21}$$

Silhouette values are often used to assess the quality of a clustering [67, 68]. They are a measure of how similar an input sample is to its own cluster (cohesion) compared to other clusters (separation). Let $\sigma(\mathbf{x}_k)$ be the mean distance from sample $\mathbf{x}_k$ to the other points in its own cluster, and $\sigma_j(\mathbf{x}_k)$ the mean distance from $\mathbf{x}_k$ to points in another cluster j. The silhouette value for a sample $SV(\mathbf{x}_k) \in [-1, 1]$ (higher is better) and the mean silhouette value MSV are defined as:

$$SV(\mathbf{x}_k) = \frac{-\sigma(\mathbf{x}_k) + \min_j \sigma_j(\mathbf{x}_k)}{\max\left\{\sigma(\mathbf{x}_k), \min_j \sigma_j(\mathbf{x}_k)\right\}} \tag{22}$$

$$MSV = \frac{1}{M}\sum_{k=1}^{M} SV(\mathbf{x}_k) \tag{23}$$

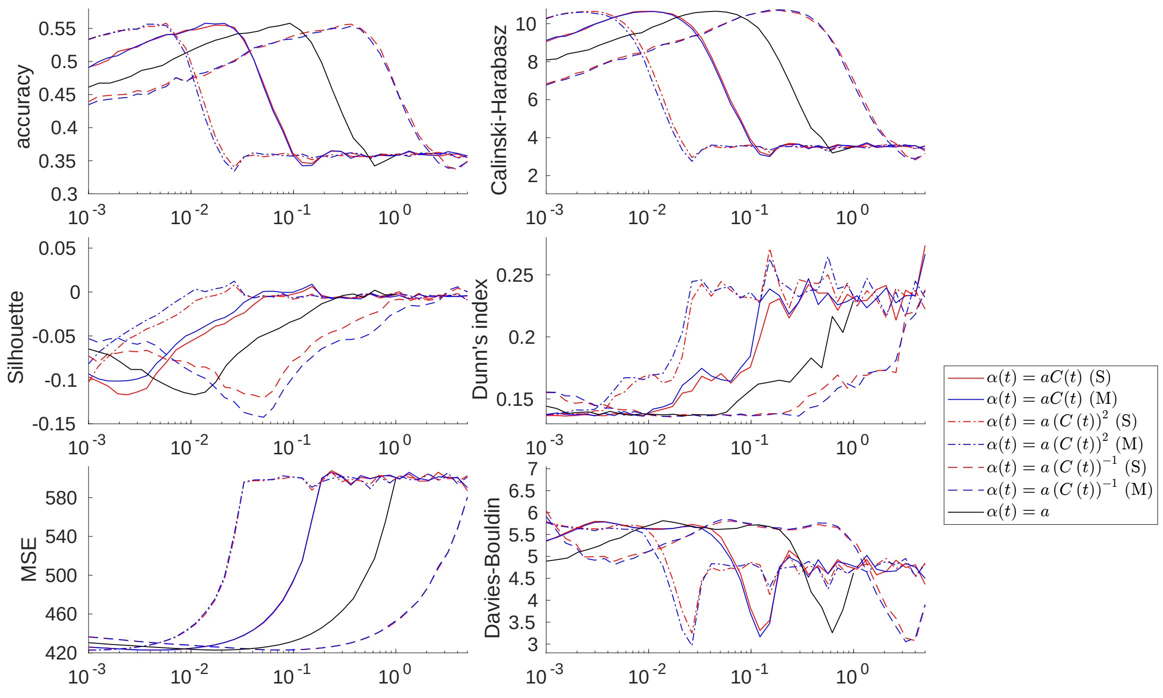

Fig. 1. Results for Competitive learning network, window size $N = 10$, regular training (no batches). X axes: parameter a in each $\alpha(t)$. Y axes: mean value of the corresponding performance metric. (S) stands for single learning rate, (M) for multiple learning rates. See text for details.

4. Experimental Results

In this section, the computational experiments that we have carried out are described, and their results are reported.

4.1. Dataset

The proposed model has been tested to compute clusterings of the CORA dataset [46]. This dataset has 2708 samples, each consisting of metadata about a scientific article on machine learning: its citation network, its sub-field, and the presence of words within the article from a standardized vocabulary of 1433 words, obtained by stemming the text corpus, removing stop words and filtering out words appearing less than ten times [46] with absolute frequency less than 10. From this metadata, only the vocabulary data is used to compute clusterings of the dataset, considering each sample (each article) as a Boolean vector of length 1433, each value signifying the presence or absence of each word in the vocabulary. While not directly used in the training processes (since competitive learning networks are unsupervised), the dataset also assigns one of seven possible sub-fields to each paper. The sub-fields are genetic algorithms, reinforcement learning, theory of machine learning, rule learning, case-based learning, probabilistic methods, and neural networks. Sub-fields are used to compute accuracy (see next section). The dataset also includes the paper's citation network, but citation information is not used at all in this work.

4.2. Setup

All experiments were done using the Matlab programming environment in a computer with an Intel i7 processor; because of the nature of competitive learning networks, memory requirements are modest and mostly dependent on the size of the training dataset. An array of experiments with different configurations has been run to test the proposed model. In this subsection, all the tested parameters and training modes are described.

As described in Section 3, a variety of performance metrics are used to measure the goodness of the clusterings: accuracy, Calinski-Harabasz criterion, Silhouette coefficient, Dunn's index, mean squared error, Davies-Bouldin index, and (for self-organizing maps and growing neural gas) topographic error. Accuracy is computed with respect to the sub-fields of the samples in each cluster's receptive field. Please note that

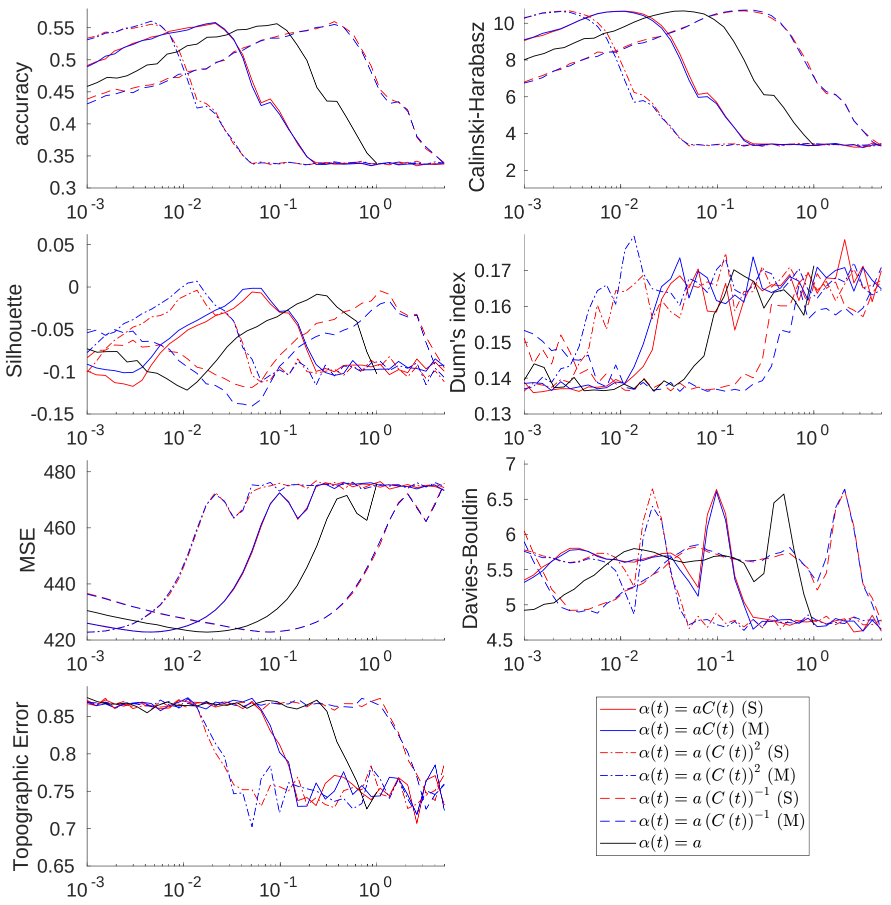

Fig. 2. Results for self-organizing map, $N = 10$, regular training (no batches). X axes: parameter a in each $\alpha(t)$. Y axes: mean value of the corresponding performance metric. (S) stands for single learning rate, (M) for multiple learning rates. See text for details.

all models are unsupervised: the sub-field of each sample is not part of the training data, and the model does not directly predict the sub-field. Instead, each cluster's sub-field is considered to be the mode (i.e., the sub-field with the highest frequency) among the samples in its receptive field, and the cluster accuracy is the ratio of the mode (the most frequent value) to the size of the receptive field. Then, clustering accuracy in each experiment is computed as the mean of the accuracy values for each cluster.

In each experiment, a network is trained to cluster the data, with networks of three types: simple competitive learning, self-organizing map, and growing neural gas. All networks have 25 neurons (the self-organizing map is configured as a 5×5 grid) to make results more directly comparable.

As described in Section 3, four different functions (linear, quadratic, inverse, constant) are used to determine the learning rate, all with one parameter, a. As a form of sensibility analysis, experiments are performed for each one of these functions, and in each case, with various values for a:

- For the first three functions, 40 values, logarithmically spaced between 0.001 and 5.
- For the constant function, 30 values, logarithmically spaced between 0.001 and 1.

For the first three functions, experiments are also performed for three different sizes for the time window used to compute $C(t)$: $N = 10$, $N = 50$, and $N = 100$. As described in Section 2, the Growing Neural Gas is a special case since it has two learning rates, ϵ_b, and ϵ_n. In the seminal GNG article [69], the default values were $\epsilon_b = 0.2$ and $\epsilon_n = 0.06$. To keep a feasible computational cost and a consistent framework (using a single parameter a for all network architectures), the secondary learning rate ϵ_n is fixed to 3% of the primary learning rate ϵ_b, keeping the ratio between both learning rates the same as when default values are used.

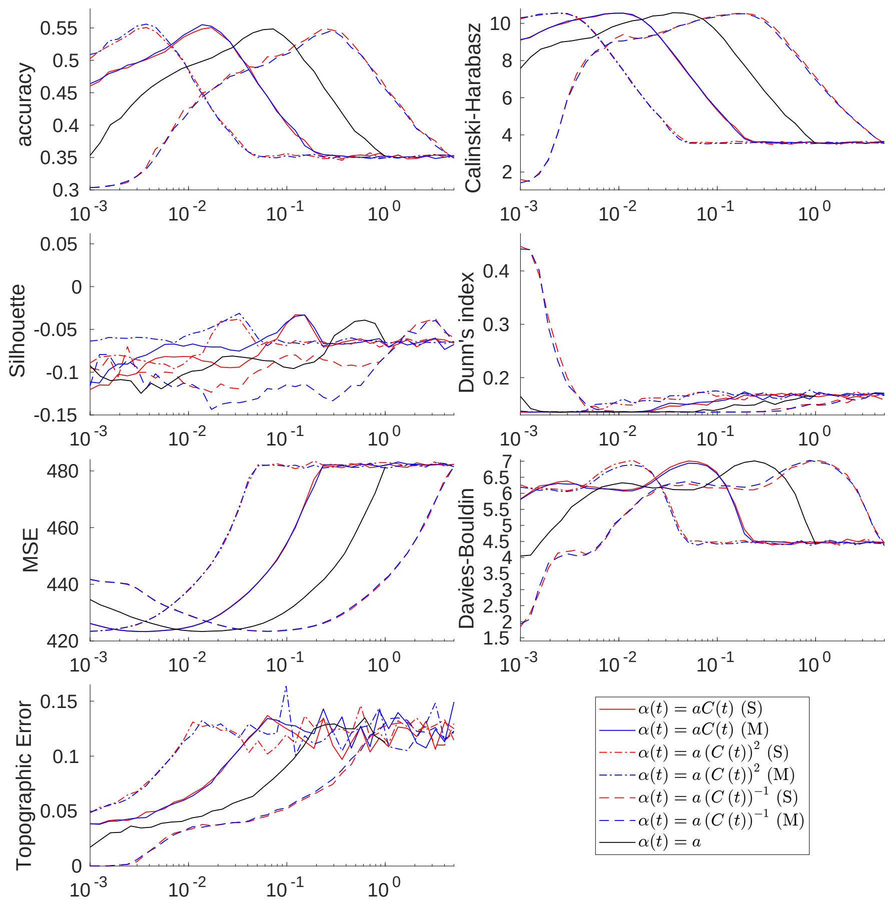

Fig. 3. Results for growing neural gas, $N = 10$, regular training (no batches). X axes: parameter a in each $\alpha(t)$. Y axes: mean value of the corresponding performance metric. (S) stands for single learning rate, (M) for multiple learning rates. See text for details.

Two different regimes are tested for the learning rate:

- A single learning rate for all neurons: the $e(t)$ distances are included in a single time window for determining $C(t)$.
- A different learning rate for each neuron: each neuron has its own time window, and consequently its own $C(t)$ and learning rate $\alpha(t)$. Each $e(t)$ distance goes to the time window of the neuron selected at training step t.

Finally, two different training regimes are tested:

- A classical (regular) training regime, with the network training on the whole dataset during four epochs, each epoch corresponding to. Clustering performance metrics are computed after training, considering the whole dataset to compute the metrics.
- A continual learning regime, with the dataset randomly divided into ten separate batches of approximately 208 samples each, training the network in sequence with each batch during four mini-epochs. After each of the four mini-epochs, clustering performance metrics are computed considering just the samples for the current batch (not the whole dataset), and the training for the next batch starts without resetting the neurons. This setup simulates an environment with continual learning, where the network is concurrently trained and used to predict the clustering of incoming samples.

All combinations of the previously described options and parameter values are tested (see Table 1 for a summary of these parameters) to make sure we carry out an unbiased search over the parameter space. In particular, three different types of competitive learning networks are tested, with three well-known unsupervised learning models, in order to cover a wide range of architectural concepts in the underlying competi-

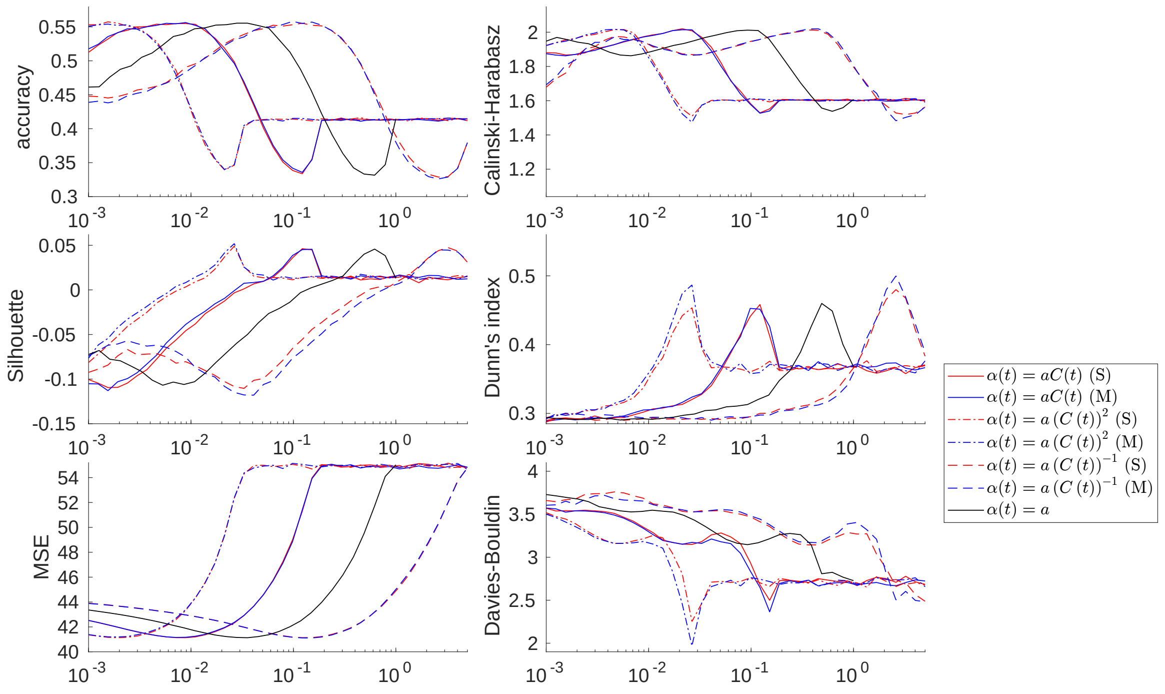

Fig. 4. Results for Competitive learning network, $N = 10$, batch training. X axes: parameter a in each $\alpha(t)$. Y axes: mean value of the corresponding performance metric. (S) stands for single learning rate, (M) for multiple learning rates. See text for details.

tive models: competitive networks as an example of a model with no topology and a fixed number of neurons, SOMs as an example of a model with a fixed topology and a fixed number of neurons, and GNGs as an example of a model with dynamic topology and a variable number of neurons. A total of 100 runs are performed for every possible configuration, resulting in a total of $100 \cdot 2 \cdot 2 \cdot 3 \cdot (40 \cdot 3 + 30) = 180000$ different experiments for each possible network type (competitive, SOM, GNG). For each configuration, the results are obtained by averaging the values for each performance metric across the 100 runs with that configuration, in order to minimize the possibility that results represent a statistical fluke.

4.3. Results

Figures from 1 to 6 show the curves for the mean values of each performance metric as a function of the parameter a. For each specific value of a, the value of each performance metric is computed as the mean of the corresponding values across each set of 100 runs with that precise configuration. Each plot shows results for a specific metric for window size $N = 10$, a specific network architecture (competitive, self-organizing map or growing neural gas), and a specific training mode

Table 2

Summarized performance results by learning rate configuration for regular training experiments (without batches). Each row summarizes results across all performance metrics for a specific configuration of the learning rate function and learning rate mode. Within each row, each column labeled from A. to T.E. represents results for each performance metric: how many times (for all possible window sizes and network architectures) that specific learning rate configuration beats other learning rate configurations for that performance metric. The last column (*all*) is the sum of the previous columns. Please see the text for a more detailed description.

regular training (no batches)									
mode	function	A.	C.-H.	S.	D.	MSE	D.-B.	T.E.	all
single	$\alpha(t) = a \cdot C(t)$	0	0	2	1	1	1	1	6
	$\alpha(t) = a \cdot C(t)^2$	1	1	0	1	3	0	0	6
	$\alpha(t) = a \cdot C(t)^{-1}$	0	0	1	2	0	2	3	8
multi	$\alpha(t) = a \cdot C(t)$	3	1	0	3	3	1	0	11
	$\alpha(t) = a \cdot C(t)^2$	5	0	6	1	2	2	2	18
	$\alpha(t) = a \cdot C(t)^{-1}$	0	6	0	1	0	3	0	10
	$\alpha(t) = a$	0	1	0	0	0	0	0	1

(regular training or batch training) for the three functions (linear, quadratic and inverse), both with a single learning rate (S) and a different learning rate for each neuron (M), as well as the results for the constant function baseline. Please note that results for regular training and batch training are not directly comparable for

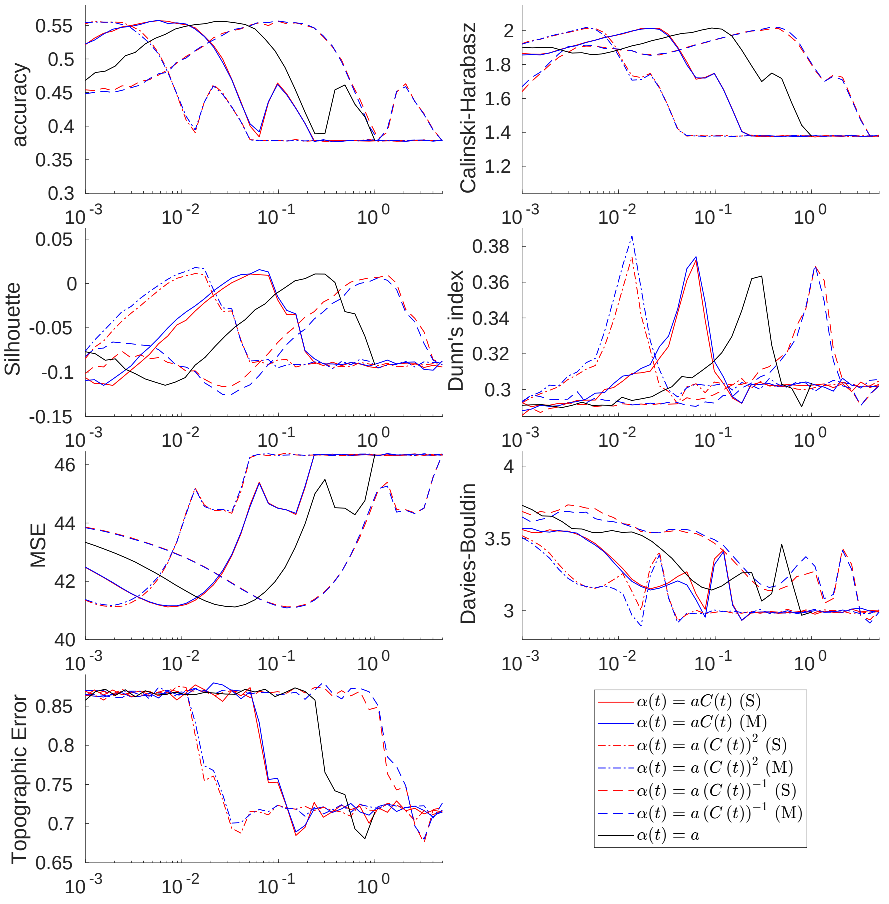

Fig. 5. Results for self-organizing map, $N = 10$, batch training. X axes: parameter a in each $\alpha(t)$. Y axes: mean value of the corresponding performance metric. (S) stands for single learning rate, (M) for multiple learning rates. See text for details.

Table 3

Summarized performance results by learning rate configuration for batch training experiments. Please compare with Table 2.

batch training									
mode	function	A.	C.-H.	S.	D.	MSE	D.-B.	T.E.	all
single	$\alpha(t) = a \cdot C(t)$	1	1	0	0	0	0	0	2
	$\alpha(t) = a \cdot C(t)^2$	1	0	0	0	3	0	0	4
	$\alpha(t) = a \cdot C(t)^{-1}$	0	0	3	2	0	1	3	9
multi	$\alpha(t) = a \cdot C(t)$	4	3	0	0	0	0	0	7
	$\alpha(t) = a \cdot C(t)^2$	0	2	5	3	0	5	0	15
	$\alpha(t) = a \cdot C(t)^{-1}$	3	3	1	4	6	3	3	23
	$\alpha(t) = a$	0	0	0	0	0	0	0	0

all metrics since, in the first case, metrics are computed for the whole dataset (2708 samples), while in the second case, they are computed for each batch (approx. 208 samples).

Note also that the figures do not represent all experiments, only those with window size $N = 10$. No figures are shown for experiments with window sizes $N = 50$ and $N = 100$, but the curves are broadly similar to the $N = 10$ case. In each figure, the first two rows show metrics where larger values are better: accuracy, Calinski-Harabasz criterion, Silhouette coefficient, and Dunn's index. The other rows show metrics where lower values are better: mean squared error, Davies-Bouldin index, and topographic error.

For experiments without batches (Figures 1-3), best a values tend to roughly coincide for accuracy and Calinski-Harabasz criterion, while for mean squared error, best a values are consistently shifted to lower values with respect the first two. For example, in Figure 1, the best accuracy value for $\alpha(t) = ax^2$ with a single learning rate is at $a = 0.0057$, and for Calinski-Harabasz is at $a = 0.03$, while the best value for mean squared error is at $a = 0.001$. These three metrics present relatively smooth and consistent curves for all network types (each figure presents results for a network type), but Silhouette, Dunn's index, Davies-

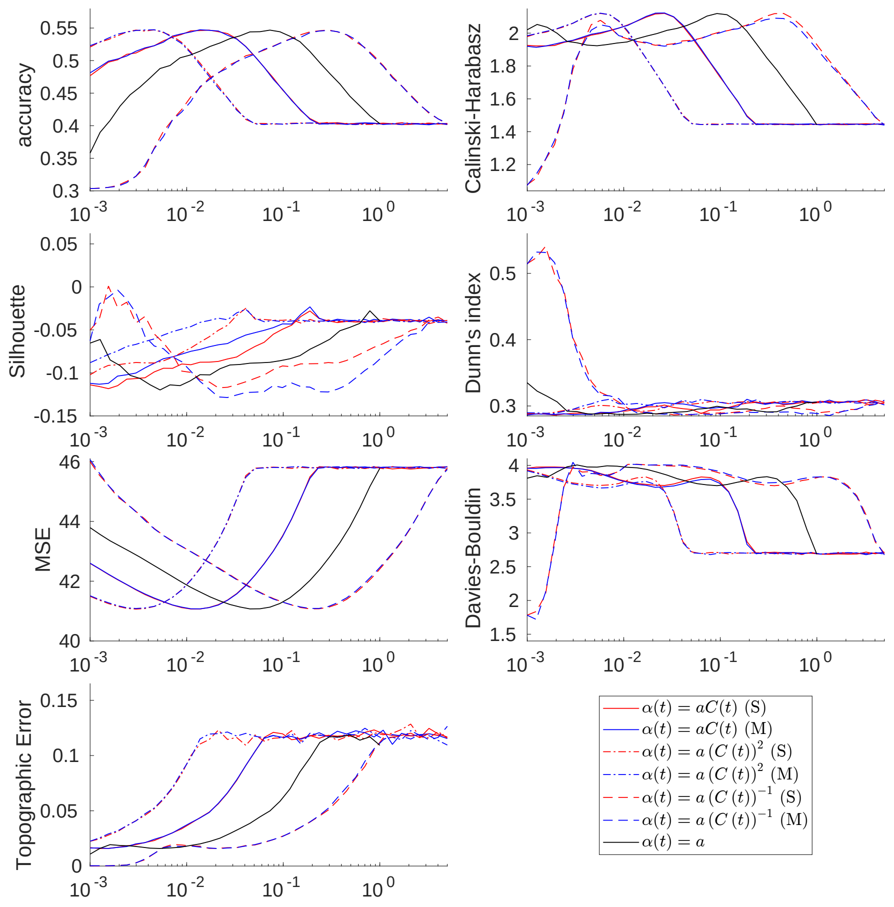

Fig. 6. Results for growing neural gas, $N = 10$, batch training. X axes: parameter a in each $\alpha(t)$. Y axes: mean value of the corresponding performance metric. (S) stands for single learning rate, (M) for multiple learning rates. See text for details.

Boudin and topographic error present curves that are noisier and also have different shapes depending on the type of network, so that no general rules can be drawn like for the other three ones, except for Silhouette, with best a values larger in general than for accuracy and Calinski-Harabasz. In the case of competitive networks (Figure 1) and SOM (Figure 2), Dunn's index and Davies-Bouldin have similar profiles, with best a values larger in general than for accuracy and Calinski-Harabasz. However, for GNG (Figure 3), these two metrics present very different profiles, with best a values in general at $a = 0.001$ (except for linear functions), and inverse functions (dashed lines) significantly outperforming the others. In general, for all network types, the best values attained in each metric by the constant function (black line) are either significantly worse than for the other functions (dependent on error rate) or roughly similar.

For experiments with batch training (Figures 4-6), in contrast, it is best a values for accuracy and mean squared error those tending to coincide, while Calinski-Harabasz's are consistently shifted towards larger a values. For example, in Figure 4, the best accuracy value for $\alpha(t) = ax^2$ with a single learning rate is at $a = 0.0015$, and for mean squared error is at $a = 0.0019$, while the best value for Calinski-Harabasz is at $a = 0.0057$. The shapes of the curves for all metrics are, in general, either as smooth or smoother (i.e., less noisy) than for experiments with regular training (without batches, Figures 1-3), but shapes and tendencies for the curves are in general similar to experiments without batches). Particularly, for GNG, inverse functions are also significantly better than others for Dunn's index and Davies-Bouldin, with best a values at $a = 0.001$.

Tables 2 and 3 distill the information in the plots of the previously discussed figures, summarized to highlight the relative performance of each learning rate function (linear, quadratic, inverse) with different learning rate modes (either a single learning rate or multiple ones) with respect to using a constant learning rate. Please note that each metric's mean value for each configuration (with a specific value for parameter a) is computed by averaging the results over the 100 runs with that specific configuration. The **best** mean value for each metric is found searching along all configurations with different values of the parameter a for each possible combination of:

- Learning rate mode (either single or multi).
- Learning rate function (either linear, quadratic, inverse, or constant, but note that constant is independent of learning rate mode).
- Window size ($N \in \{10, 50, 100\}$).
- Network architecture (either competitive, self-organizing map, or growing neural gas).

To contextualize the significance of these best mean values: in the plots in Figures 1 to 6, the best mean values correspond to the top of each curve's range along the Y axis for the first four metrics, and the bottom of the range for the other metrics (but the plots are only for $N = 10$, while we are considering best values also for configurations with $N = 50$ and $N = 100$).

To build tables 2 and 3, we add up all the times a configuration with a specific combination of learning rate mode and function wins over configurations with other combinations. Please note that 100 experiments are run for each configuration, and we use the mean value across the 100 runs to measure the number of times each combination wins over others. Table 2 summarizes results for regular training mode (without batches), and Table 3 does the same for batch training mode. There are nine different combinations of window size and network architecture, so for each metric, there are nine possible "wins" to be distributed among the seven possible combinations of learning rate mode and function (rows in Tables 2 and 3). Consequently, each column in these tables adds up to 9, except in the case of the topographic error: it only applies to the self-organizing map and growing neural gas, so each column adds up to 6.

Two conclusions can be gleaned from these tables:

- Varying the learning rate $\alpha(t)$ as a function of the training error $C(t)$ is better than using a constant learning rate in almost all cases. While none of the different functions dominates clearly above others for all performance metrics, quadratic and inverse functions seem to win more times.
- Using a different learning rate for each neuron seems to be better than using a single learning rate for all neurons, as measured by accuracy, Calinski-Harabasz criterion, Silhouette coefficient, Dunn's index, mean squared error, and Davies-Bouldin index. For all these metrics, configurations with multiple learning rates win (having the best mean value) more times than configurations with a single learning rate. In the case of topographic error, there is a draw in the number of wins.

Additionally, Table 4 presents a comparison between the results of one of the configurations tested in the experiments and various other clustering techniques, always with 25 clusters, since the number of clusters is 25 for the experiments with competitive networks and SOM, and at most 25 with GNG: three classical clustering techniques (K-means, K-medoids, DBSCAN) and the three competitive learning networks tested in the experiments, but with classical learning rate decay. The learning rate decays linearly from an initial value of 0.4 for the first half of the training; then, it is set at a low value (0.01) for the second half. A total of 100 experiments are run for each clustering technique: the values in the table are the mean and standard deviation over these 100 runs for each metric. Please note that for the first four performance results, larger values are better, and *vice versa* for the last three. While many configurations of our method perform better than the other six techniques in some metric (in isolation), we compare them against just one configuration. To select this configuration, we first filter the ones that are better than the other techniques in accuracy and MSE (since we consider these to be the most relevant metrics). This results in a still sizeable number of configurations. From these, we select the configurations with a minimal sum of ranks, where the configuration's rank for each metric is its ordinal when all competing values in the same value are sorted from better to worse, considering directly the configurations with globally minimal rank results in configurations that do not prioritize accuracy and MSE. From these, we select for the table the one with the best accuracy: GNG network, with history size $N = 10$, and quadratic function with scaling parameter $a = 0.0019$. This configuration is better than the other six clustering techniques not only for accuracy and MSE but also Calinski-Harabasz and topographic error, but it is not well ranked in the other three metrics (Silhouette, Dunn's index and Davies-Bouldin).

As a graphical example of the accuracy of the clustering for experiments, we also show in Figure 7 specific examples of two experiments, one with a self-organizing map and the other with a growing neural

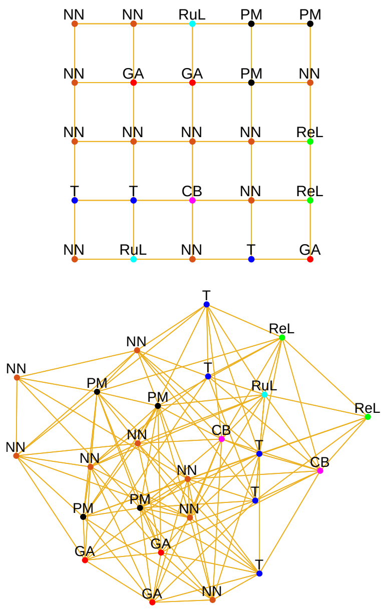

Fig. 7. Representation of two specific experiments, one with a self--organizing map (up) and the other with growing neural gas (down). In both cases, the topology of the network is depicted, with each neuron showing the mode of the sample sub-field for all samples in each neuron's receptive field. If the neural network has learned a good clustering of the dataset, adjacent neurons will tend to have similar modes. Sub-field acronyms are GA (Genetic Algorithms), ReL (Reinforcement Learning), T (Theory), RuL (Rule Learning), CB (Case-Based), PM (Probabilistic Methods), and NN (Neural Networks).

gas. They are plotted to show how well the topology of the network maps to sample classes (defining the class of each sample as the sub-field of the underlying article). For each experiment, the resulting neural network is represented, showing for each neuron the mode of the sample class for the samples in the neuron's receptive field. As can be seen, there are clear clusters of sample class modes, for example, with the growing neural gas.

5. Discussion

To adapt competitive learning models to continual learning, we have defined the learning rate as a function of the error rate during training. In this way, if the model has converged for a particular probability distribution, but the incoming samples drift to another one, then the learning rate can change dynamically to address this situation.

Thus, adapting the learning rate to the amount of training error has been shown to produce better clusterings. Furthermore, in the context of continual learning, for many of the metrics used to measure clustering performance, Table 3 shows that using multiple learning rates (i.e., a different learning rate adapted to the particular history of training errors for each neuron) can produce better results than using a single learning rate for the whole model. Table 2 shows that multiple learning rates are also better in the context of regular training regimes, even if the effect seems less intense. However, these results should be put into context: as it can be seen from Figures 1-6, it is apparent that, for most metrics, the differences in average peak performance between different learning rates and different functions $\alpha(t)$ are relatively small, so the best configurations are winning, in most cases, by relatively small margins, with one instance (in Table 2) of the constant function winning once over the others for the Calinski-Harabasz metric. It should also be noted that, while we can find configurations whose average peak performance beats other clustering methods for any given metric, there is no configuration that dominates for all metrics over other clustering methods, and the example we provide in Table 4 is noticeably worse than other clustering methods for the Silhouette, Davies-Bouldin and Dunn's indexes.

Looking at Figures 1-6 for instances of configurations with multiple learning rates outperforming their counterparts with a single learning rate, this happens more frequently for the metrics Silhouette, Dunn's index and Davies-Bouldin, but not for Calinski-Harabasz. All these four metrics measure intra-cluster compactness and inter-cluster separation, but in different ways. In particular, the first three compute the values finding maxima and minima of various terms (the fourth doesn't), so their values depend more on samples being extreme in some respect. For example, Dunn's index just measures the ratio between the smallest inter-cluster distance and the largest intra-cluster distances. This is evidence that using multiple learning rates can lead to clusterings with better-behaving clusters at the extreme range of various ways to measure intra- and inter-cluster distances.

It is also remarkable that the inverse function (i.e., defining the learning rate as an inverse function of training errors) can be better for some performance metrics, most notably the Calinski-Harabasz criterion when using multiple learning rates in Table 2, and more broadly for multiple metrics in Table 3. This seemingly counterintuitive result can be explained by outliers. In the presence of outliers in the input distribution, an inverse correspondence keeps the units

Table 4

For each performance metric, the mean and standard deviation values from 100 different runs of several clustering strategies. Three classical clustering techniques (K-means clustering, K-medoids clustering, and DBSCAN), three competitive learning techniques with linearly decreasing learning rate (competitive network, SOM, GNG), and one configuration of our online method (GNG with $N = 50$, *multi* learning rate, and quadratic function) outperforming other competitors on four metrics: accuracy, Calinski-Harabasz, MSE and topographic error. Please note that larger values are better for the first four performance metrics, and *vice versa* for the last three ones.

performance metric	other clustering methods						competitive learning							
							regular (not online)						Ours (online): GNG, $N = 10$, multi L.R.,	
	K-means		K-medoids		DBSCAN		comp.		SOM		GNG		$\alpha(t) = a \cdot C(t)^2$, $a = 0.0019$	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
accuracy	0.352	0.056	0.425	0.001	0.307	0.001	0.478	0.035	0.517	0.023	0.471	0.021	**0.533**	0.017
Calinski-Harabasz	4.121	1.894	8.109	0.008	1.002	0.045	7.516	0.491	9.089	0.287	7.513	0.248	**10.509**	0.183
Silhouette	-0.0724	0.045	-0.032	0.001	-0.225	0.060	**0.0084**	0.012	-0.030	0.014	-0.085	0.030	-0.060	0.017
Dunn's Index	**0.206**	0.093	0.136	0	0.097	0.099	0.160	0.039	0.162	0.041	0.148	0.023	0.136	0.001
Mean Squared Error	431.4	3.363	440.2	0	438.9	0.084	427.9	1.009	425.4	0.606	439.3	0.745	**424.308**	0.312
Davies-Bouldin	**2.107**	0.813	4.959	0.005	2.996	0.051	3.886	0.237	5.664	0.159	6.985	0.167	6.127	0.164
topographic error	-	-	-	-	-	-	-	-	0.716	0.056	0.122	0.025	**0.057**	0.011

(neurons) that represent the outliers relatively static because such units have large training errors, as the outliers are far away from the other samples. Consequently, these units do not migrate to the large central clusters of the distribution, thereby keeping the outliers well-represented. As demonstrated in the experiments, this mechanism can greatly enhance the overall performance of the studied clustering algorithms.

6. Conclusions

In this work, a novel framework for continual learning has been proposed. It is aimed at dealing with non-stationary inputs for unsupervised learning models. A dynamic learning rate is designed, which depends on the current reconstruction error. It can be applied either per neuron or per network. This method can be employed for any application of unsupervised neural networks. In particular, we have chosen a scientific article unsupervised clustering problem. For this real-life problem, our approach with multiple learning rates dependent on individual neuron error rates has demonstrated that it outperforms the classic constant learning rate scheme according to a wide range of unsupervised clustering performance measures. Moreover, the number of recent samples taken into account for computing the error rates seem not to be too relevant. In fact, the best configuration found in our experiments was a GNG network with a small history size $N = 10$ and quadratic function learning rate, which outperformed other configurations of the proposed model, as well as six non-competitive clustering techniques.

While this work has focused on applying our proposed approach to online competitive learning to document classification, there are other active research areas where our proposal can also work, such as online clustering of images or image patches [55–58], online clustering of sound samples [59], or the more generic multi-armed bandit problem [60]. Regarding future work on the model itself, there are a number of lines of future work amenable to our results. Taking advantage of the relatively inexpensive computational cost of competitive learning, negative correlation learning in dynamic ensembles [70] can provide a performance boost by tuning a clustering model with multiple sub-networks whose results are combined using a negative correlation learning rule. Also, incoming samples can be mapped into another multidimensional space where they are easier to classify, with a suitable procedure such as Neural Dynamic Classification [71]. Another possibility is to keep our proposed approach to online competitive learning as an initial unsupervised step used to feed a supervised learning approach in order to model the resulting partitions of the input space in a more effective way using a Finite Element Machine [72] or to avoid catastrophic forgetting in the latter supervised step [73].

Acknowledgements

This work is partially supported by the Ministry of Science, Innovation and Universities of Spain under grant RTI2018-094645-B-I00, project name Automated detection with low-cost hardware of unusual activities in video sequences. It is also partially supported by the Autonomous Government of Andalusia (Spain) under project UMA18-FEDERJA-084, project name

Detection of anomalous behavior agents by deep learning in low-cost video surveillance intelligent systems. It is also partially supported by the Autonomous Government of Andalusia (Spain) under project UMA20-FEDERJA-108, project name Detection, characterization and prognosis value of the non-obstructive coronary disease with deep learning. All of them include funds from the European Regional Development Fund (ERDF). It is also partially supported by the University of Malaga (Spain) under grants B1-2019_01, project name Anomaly detection on roads by moving cameras, and B1-2019_02, project name Self-Organizing Neural Systems for Non-Stationary Environments. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga. They also gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan X GPUs. The authors also thankfully acknowledge the grant of the Universidad de Málaga and the Instituto de Investigación Biomédica de Málaga - IBIMA.

References

[1] Gil-Gala, F.J., Mencía, C., Sierra, M.R. & Varela, R. (2021). Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. *Integrated Computer-Aided Engineering*, **28**(1), 65–80. doi:10.3233/ICA-200634.

[2] Rafiei, M.H. & Adeli, H. (2017). NEEWS: A Novel Earthquake Early Warning System Using Neural Dynamic Classification and Neural Dynamic Optimization Model. *Soil Dynamics and Earthquake Engineering*, **100**, 417–427.

[3] Rafiei, M.H., Khushefati, W.H., Demirboga, R. & Adeli, H. (2017). Supervised Deep Restricted Boltzmann Machine for Estimation of Concrete Compressive Strength. *ACI Materials Journal*, **114**(2), 237–244.

[4] Jeong, J.H. & Jo, H. (2021). Deep reinforcement learning for automated design of reinforced concrete structures. *Computer-Aided Civil and Infrastructure Engineering*, **36**(12), 1508–1529. doi:10.1111/mice.12773.

[5] Zhang, Y. & Yuen, K.V. (2021). Crack detection using fusion features-based broad learning system and image processing. *Computer-Aided Civil and Infrastructure Engineering*, **36**(12), 1568–1584. doi:10.1111/mice.12753.

[6] Wu, Y., Qin, Y., Qian, Y., Guo, F., Wang, Z. & Jia, L. (2022). Hybrid deep learning architecture for rail surface segmentation and surface defect detection. *Computer-Aided Civil and Infrastructure Engineering*, **37**(2), 227–244. doi:10.1111/mice.12710.

[7] Gao, T., Li, Z., Gao, Y., Schonfeld, P., Feng, X., Wang, Q. & He, Q. (2022). A deep reinforcement learning approach to mountain railway alignment optimization. *Computer-Aided Civil and Infrastructure Engineering*, **37**(1), 73–92. doi:10.1111/mice.12694.

[8] Lin, Y.Z., Nie, Z.H. & Ma, H.W. (2022). Dynamics-based cross-domain structural damage detection through deep transfer learning. *Computer-Aided Civil and Infrastructure Engineering*, **37**(1), 24–54. doi:10.1111/mice.12692.

[9] Nogay, H.S. & Adeli, H. (December,2020). Machine Learning (ML) for the Diagnosis of Autism Spectrum Disorder (ASD) Using Brain Imaging. *Reviews in the Neurosciences*, **31**(8), 825–841.

[10] Nogay, H.S. & Adeli, H. (2020). Detection of Epileptic Seizure Using Pre-trained Deep Convolutional Neural Network and Transfer Learning. *European Neurology*, **83**(6), 602–614.

[11] Hassanpour, A., Moradikia, M., Adeli, H., Khayami, S.R. & Shamsinejadbabaki, P. (December, 2019). A novel end-to-end deep learning scheme for classifying multi-class motor imagery electroencephalography signals. *Expert Systems*, **36**(6), e12494 1–21. doi:10.1111/exsy.12494.

[12] Martins, G.B., Papa, J.P. & Adeli, H. (2020). Deep learning techniques for recommender systems based on collaborative filtering. *Expert Systems*, **37**(6), e12647 1–21. doi:10.1111/exsy.12647.

[13] Rafiei, M.H. & Adeli, H. (2018). Novel Machine-Learning Model for Estimating Construction Costs Considering Economic Variables and Indexes. *Journal of Construction Engineering and Management*, **144**(12), e04018106 1–9. doi:10.1061/(ASCE)CO.1943-7862.0001570.

[14] Rafiei, M.H. & Adeli, H. (2016). A Novel Machine Learning Model for Estimation of Sale Prices of Real Estate Units. *Construction Engineering and Management*, **142**(2), e04015066 1–10. doi:10.1061/(ASCE)CO.1943-7862.0001047.

[15] Shen, J., Yan, W., Li, P. & Xiong, X. (2021). Deep learning-based object identification with instance segmentation and pseudo-LiDAR point cloud for work zone safety. *Computer-Aided Civil and Infrastructure Engineering*, **36**(12), 1549–1567. doi:10.1111/mice.12749.

[16] Gasienica-Jozkowy, J., Knapik, M. & Cyganek, B. (2021). An ensemble deep learning method with optimized weights for drone-based water rescue and surveillance. *Integrated Computer-Aided Engineering*, **28**(3), 221–235. doi:10.3233/ICA-210649.

[17] Macias-Garcia, E., Galeana-Perez, D., Medrano-Hermosillo, J. & Bayro-Corrochano, E. (2021). Multi-stage deep learning perception system for mobile robots. *Integrated Computer-Aided Engineering*, **28**(2), 191–205. doi:10.3233/ICA-200640.

[18] Rao, D., Visin, F., Rusu, A.A., Teh, Y.W., Pascanu, R. & Hadsell, R. (2019). Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems* (Vol. 32). Neural information processing systems foundation.

[19] Beyer, O. & Cimiano, P. (2012). Online semi-supervised growing neural gas. *International journal of neural systems*, **22**(05), 1250023.

[20] McCloskey, M. & Cohen, N.J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, **24**(C), 109–165. doi:10.1016/S0079-7421(08)60536-8.

[21] Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A. & Bengio, Y. (2014). An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *2nd International Conference on Learning Representations, ICLR 2014 - Confer-*

ence Track Proceedings. International Conference on Learning Representations, ICLR.

[22] Mermillod, M., Bugaiska, A. & Bonin, P. (2013). The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, **4**, 504. doi:10.3389/fpsyg.2013.00504.

[23] Shin, H., Lee, J.K., Kim, J. & Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems* (Vol. 2017-Decem, pp. 2991–3000). Neural information processing systems foundation.

[24] Zenke, F., Poole, B. & Ganguli, S. (2017). Continual learning through synaptic intelligence. In *34th International Conference on Machine Learning, ICML 2017* (Vol. 8, pp. 6072–6082). International Machine Learning Society (IMLS).

[25] Nguyen, C.V., Li, Y., Bui, T.D. & Turner, R.E. (2018). Variational continual learning. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

[26] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. & Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, **114**(13), 3521–3526. doi:10.1073/pnas.1611835114.

[27] Jain, A.K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, **31**(8), 651–666. doi:https://doi.org/10.1016/j.patrec.2009.09.011.

[28] Chen, M.-S., Han, J. & Yu, P.S. (1996). Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, **8**(6), 866–883. doi:10.1109/69.553155.

[29] T., J.I. & Jorge, C. (2016). Principal component analysis: a review and recent developments. *Phil. Trans. R. Soc. A., 374:20150202*.

[30] Ahalt, S.C., Krishnamurthy, A.K., Chen, P. & Melton, D.E. (1990). Competitive learning algorithms for vector quantization. *Neural Networks*, **3**(3), 277–290. doi:https://doi.org/10.1016/0893-6080(90)90071-R.

[31] Uchiyama, T. & Arbib, M.A. (1994). Color Image Segmentation Using Competitive Learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, **16**(12), 1197–1206–. doi:10.1109/34.387488.

[32] García-Lamont, F., Cervantes, J. & López-Chau, A.e.a. (2020). Automatic computing of number of clusters for color image segmentation employing fuzzy c-means by extracting chromaticity features of colors. *Pattern Anal. Applic.*, **23**, 59–84. doi:https://doi.org/10.1007/s10044-018-0729-9.

[33] Yuan, H., Duo, C.-H. & Niu, W.-h. (2016). A Human Behavior Recognition Method Based on Latent Semantic Analysis. *J. Inf. Hiding Multim. Signal Process.*, **7**, 489–498.

[34] Xu, L., Krzyzak, A. & Oja, E. (1993). Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Transactions on Neural Networks*, **4**(4), 636–649. doi:10.1109/72.238318.

[35] López-Rubio, E., Luque-Baena, R.M. & Domínguez, E. (2011). Foreground detection in video sequences with probabilistic self-organizing maps. *International Journal of Neural Systems*, **21**(03), 225–246.

[36] Hsu, W.-Y. (2012). Application of competitive Hopfield neural network to brain-computer interface systems. *International journal of neural systems*, **22**(01), 51–62.

[37] Banković, Z., Moya, J.M., Araujo, Á., Fraga, D., Vallejo, J.C. & de Goyeneche, J.-M. (2010). Distributed intrusion detection system for wireless sensor networks based on a reputation system coupled with kernel self-organizing maps. *Integrated Computer-Aided Engineering*, **17**(2), 87–102.

[38] Alhoniemi, E., Hollmén, J., Simula, O. & Vesanto, J. (1999). Process monitoring and modeling using the self-organizing map. *Integrated Computer-Aided Engineering*, **6**(1), 3–14.

[39] Allen, M.J., Marin, F., García-Lagos, F., Gough, N.E. & Mehdi, Q. (2003). Fuzzy processing for active vision. *Integrated Computer-Aided Engineering*, **10**(3), 267–285.

[40] Cao, X. & Suganthan, P.N. (2002). Neural network based temporal video segmentation. *International Journal of Neural Systems*, **12**(03n04), 263–269.

[41] Palomo, E.J. & López-Rubio, E. (2016). Learning topologies with the growing neural forest. *International journal of neural systems*, **26**(04), 1650019.

[42] Angelopoulou, A., Psarrou, A., Rodríguez, J.G. & Revett, K. (2005). Automatic landmarking of 2D medical shapes using the growing neural gas network. In *International Workshop on Computer Vision for Biomedical Image Applications* (pp. 210–219). Springer.

[43] Benito-Picazo, J., Palomo, E.J., Domínguez, E. & Ramos, A.D. (2020). Image clustering using a growing neural gas with forbidden regions. In *2020 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–7). IEEE.

[44] Toda, Y., Wada, A., Miyase, H., Ozasa, K., Matsuno, T. & Minami, M. (2022). Growing Neural Gas with Different Topologies for 3D Space Perception. *Applied Sciences*, **12**(3), 1705.

[45] Molina-Cabello, M.A., Luque-Baena, R.M., López-Rubio, E., Ortiz-de-Lazcano-Lobato, J.M., Domínguez, E. & Pérez, J.M. (2017). Vehicle classification in traffic environments using the growing neural gas. In *International Work-Conference on Artificial Neural Networks* (pp. 225–234). Springer.

[46] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B. & Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, **29**(3), 93–93.

[47] Rakaraddi, A., Siew Kei, L., Pratama, M. & De Carvalho, M. (2022). Reinforced Continual Learning for Graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (pp. 1666–1674).

[48] Zhang, X., Song, D. & Tao, D. (2022). Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[49] Wang, J., Song, G., Wu, Y. & Wang, L. (2020). Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (pp. 1515–1524).

[50] Zhang, J., Ghahramani, Z. & Yang, Y. (2004). A probabilistic model for online document clustering with application to novelty detection. *Advances in neural information processing systems*, *17*.

[51] Khy, S., Ishikawa, Y. & Kitagawa, H. (2008). A novelty-based clustering method for on-line documents. *World Wide Web*, **11**(1), 1–37.

[52] Zhong, S. (2005). Efficient online spherical k-means clustering. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 5, pp. 3180–3185). IEEE.

[53] Borgelt, C. & Nürnberger, A. (2004). Fast fuzzy clustering of web page collections. In *Proc. of PKDD Workshop on Statistical Approaches for Web Mining (SAWM).*

[54] Cottrell, M., Hammer, B., Hasenfuß, A. & Villmann, T. (2006). Batch and median neural gas. *Neural Networks*, **19**(6-7), 762–771.

[55] Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y.W. & Hadsell, R. (2019). Continual unsupervised representation learning. *Advances in Neural Information Processing Systems*, *32*.

[56] Zheng, K., Liu, W., He, L., Mei, T., Luo, J. & Zha, Z.-J. (2021). Group-aware label transfer for domain adaptive person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5310–5319).

[57] He, J. & Zhu, F. (2022). Unsupervised continual learning via pseudo labels. In *International Workshop on Continual Semi-Supervised Learning* (pp. 15–32). Springer.

[58] Taufique, A.M.N., Jahan, C.S. & Savakis, A. (2022). Unsupervised Continual Learning for Gradually Varying Domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3740–3750).

[59] Marxer, R. & Purwins, H. (2016). Unsupervised incremental online learning and prediction of musical audio signals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **24**(5), 863–874.

[60] Lin, B., Bouneffouf, D., Cecchi, G.A. & Rish, I. (2018). Contextual bandit with adaptive feature extraction. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 937–944). IEEE.

[61] Davies, D.L. & Bouldin, D.W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1**(2), 224–227.

[62] Farsadnia, F., Kamrood, M.R., Nia, A.M., Modarres, R., Bray, M.T., Han, D. & Sadatinejad, J. (2014). Identification of homogeneous regions for regionalization of watersheds by two-level self-organizing feature maps. *Journal of Hydrology*, **509**, 387–397.

[63] Higuera, C., Pajares, G., Tamames, J. & Morán, F. (2013). Expert system for clustering prokaryotic species by their metabolic features. *Expert Systems with Applications*, **40**(15), 6185–6194.

[64] Dunn, J.C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, **3**(3), 32–57.

[65] Bezdek, J.C. & Pal, N.R. (1998). Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **28**(3), 301–315.

[66] Caliński, T. & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, **3**(1), 1–27. doi:10.1080/03610927408827101.

[67] Huneiti, A.M. (2012). Interpreting web usage patterns generated using a hybrid SOM-based clustering technique. *International Review on Computers and Software*, **7**(3), 1078–1088.

[68] Xu, R., Xu, J. & Wunsch, D.C. (2012). A comparison study of validity indices on swarm-intelligence-based clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **42**(4), 1243–1256.

[69] Fritzke, B. (1994). A growing neural gas network learns topologies. *Advances in neural information processing systems*, *7*.

[70] Alam, K.M., Siddique, N., Adeli, H., et al. (2020). A dynamic ensemble learning algorithm for neural networks. *Neural Computing and Applications*, **32**(12), 8675–8690.

[71] Rafiei, M.H. & Adeli, H. (2017). A new neural dynamic classification algorithm. *IEEE transactions on neural networks and learning systems*, **28**(12), 3074–3083.

[72] Pereira, D.R., Piteri, M.A., Souza, A.N., Papa, J.P. & Adeli, H. (2020). FEMa: A finite element machine for fast learning. *Neural Computing and Applications*, **32**(10), 6393–6404.

[73] Tao, X., Hong, X., Chang, X., Dong, S., Wei, X. & Gong, Y. (2020). Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12183–12192).