Incremental learning with self-organizing maps

Alexander Gepperth
University of Applied Sciences Fulda
Department of Applied Computer Science
Leipzigerstr. 123, 36036 Fulda, Germany
Email: alexander.gepperth@cs.hs-fulda.de

Cem Karaoguz
ENSTA ParisTech
UIIS Lab
Blvd des Marechaux 128, 91762 Palaiseau, France
Email: cem.karaoguz@ensta.fr

***Abstract*—We present a novel use for self-organizing maps (SOMs) as an essential building block for incremental learning algorithms. SOMs are very well suited for this purpose because they are inherently online learning algorithms, because their weight updates are localized around the best-matching unit, which inherently protects them against catastrophic forgetting, and last but not least because they have fixed model complexity limiting execution time and memory requirements for processing streaming data. However, in order to perform incremental learning which is usually supervised in nature, SOMs need to be complemented by a readout layer as well as a self-referential control mechanism for prototype updates in order to be protected against negative consequences of concept drift. We present the PROPRE architecture which implements these functions, thus realizing incremental learning with SOMs in very high-dimensional data domains, and show its capacity for incremental learning on several known and new classification problems. In particular, we discuss the required control of SOM parameters in detail and validate our choices by experimental results.**

I. Introduction

This article is in the context of research on incremental learning algorithms, and elucidates how self-organizing maps can be an integral and necessary part of such algorithms.

Incremental learning [1] is a form of learning that is quite different from the more well-established algorithms like neural networks (including deep learning), support vector machines or bagged decision trees, as it does not impose a separation of training and test phases. Incremental models can be updated with new samples at any time, and thus re-trained as often as desired which is not (or not fully) possible with the aforementioned models.

The precise definition of incremental learning is not a subject of universal agreement. Although attempts at formal definition ([2], [1]) conflict in minor points, they seem to agree that incremental learning algorithms

- take their training samples one by one, a capacity which shall be termed here *online learning*)
- do not know the total number of samples in advance
- can deal with changes in the statistics underlying sample generation

When one thinks about it, the last two properties are really equivalent: *any* finite dataset that is split into two parts will almost surely exhibit different statistics in each part (see [3] on how this fact affects cross-validation methods in machine learning). Since data in incremental learning scenarios can always be divided into past (already seen) and future (as yet unseen) samples, statistics in those two parts will in general be different. For algorithms not suited for incremental learning, this leads to the so-called *catastrophic forgetting* effect [4], [5], [6], [7], [8] which was mainly coined for supervised learning with neural networks.

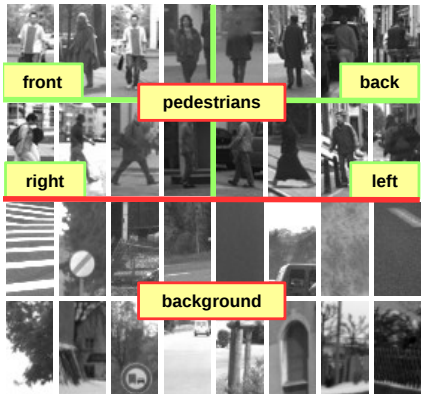

Fig. 1. Representative samples from the real-world pedestrian detection and classification datasets used in this study, in addition to MNIST. There are two classification problems: pose classification into the four classes left/right/front/back (boxes with green borders), and pedestrian detection, i.e. classification into a pedestrian and a background class (boxes with red borders).

Generally, changes in data statistics as encountered in incremental learning are denoted somewhat generally as *concept drift* [9], [10], which can be gradual or abrupt. In the latter case one often uses the term *concept shift*. When data statistics do not change globally but only in a specific region of data space, sometimes the term *local concept drift* is used [10]. A prominent example is the addition of a new, visually dissimilar class to a classification problem, which is the kind of concept drift mainly treated in this article. Local concept drift in particular is an important use case for incremental learning algorithms as there is, a priori, no reason why new statistics in localized regions of data space should disrupt learned models elsewhere. Another and much more problematic case is local concept drift/shift with conflict (also treated here), for example when a new but similar class appears in the data: this will in any event have an impact on classification performance until the model can be locally re-adapted to separate the old from the new class.

978-1-5090-6638-4/17/$31.00

A. Focus and contributions of this article

This article makes a strong point for self-organizing maps (SOMs) as an essential building for incremental learning. This is mainly due to the following properties of SOMs:

- online learning: traditionally, SOMs are fed their training samples one by one although batch SOM models exist. While the justification of this online learning rule remains unsatisfactory in the original formulation, there are modified SOM models that explain it as a stochastic gradient descent on an energy function[11].
- robustness against changing statistics: all SOM-like models update prototypes exclusively in the vicinity of the best-matching unit (BMU). Due to the topological organization of prototypes, concept drift in one part of data space will not affect prototypes for another, distant part of data space, thus avoiding catastrophic forgetting (see Sec. III-D).
- constant model complexity: except for growing neural gas models, the model complexity of SOMs is fixed by the number of units. For incremental learning, this is an advantage as it is often conducted under real-world/real-time conditions where guaranteed time and memory complexity is of high importance (e.g., for treating streaming data or in robotics applications). Comparable incremental learning algorithms es listed in Sec. I-C often exhibit variable model complexity which can effectively render them unusable depending on the chosen application.

However, the SOM model is not directly usable for incremental learning as it is not a supervised algorithm, for one thing. In addition, there are several additional mechanisms that need to be implemented before incremental learning becomes feasible in practice:

- supervised read-out layer for SOM
- a SOM-based concept drift detection mechanism
- protection of the SOM against sampling bias by selectively controlling prototype updates
- adaptive suppression of sub-leading SOM activity to protect read-out layer against concept drift
- adaptive control of SOM parameters to maintain topological ordering in the face of concept drift

These mechanisms have been realized in the PROPRE ("PROjection-PREdiction") architecture [12], [13] which was shown to be capable of incremental learning in high-dimensional ($>$ 1000 dimensions) spaces due to its internal SOM layer. In this article, these mechanisms are described in detail, and an in-depth analysis of their behavior in a typical incremental learning scenario is conducted, namely the abrupt addition/presentation of a new, previously unseen class. In this way, best practices are identified and solid justifications are obtained for mechanisms that can make SOMs an integral building block of incremental learning algorithms. We pay particular attention to validate our results on a representative set of real-world learning problems.

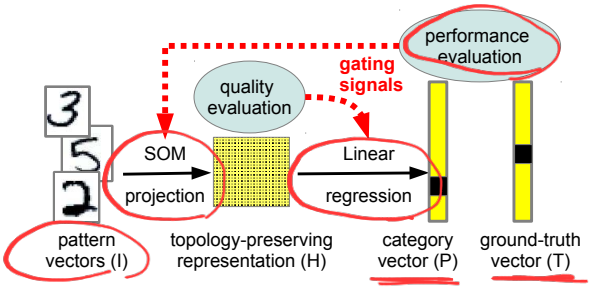

Fig. 2. Block diagram of the PROPRE architecture for incremental learning.

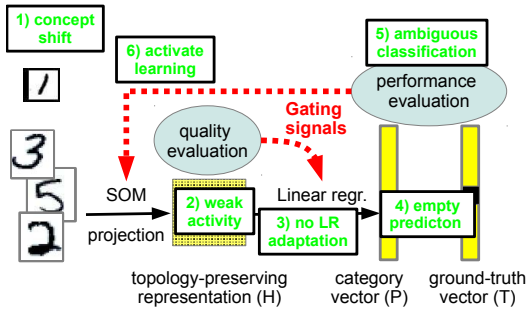

Fig. 3. Sequence of events as a reaction to concept shift in the form of a new visual class, to be regarded together with Fig. 2.

B. The PROPRE architecture for incremental learning

The PROPRE architecture for incremental supervised learning was proposed in [12], [13] It combines generative, SOM-based learning of an internal representation with discriminative read-out of classification or regression outputs, see Fig. 2. From the latter, a task-related error signal is derived which adapts the internal SOM representation in case of mismatch or classification ambiguity. This ensures that prototype density increases in regions of the input space that are difficult to classify, or in which concept drift is occurring. Prototype adaptation is stably self-terminating when no more errors are made, or when concept drift subsides.

As a typical SOM model, the internal representation is topologically organized, and prototype adaptation modifies weights only locally. It is above all this property that allows for incremental learning of prototypes: adaptation of a single prototype changes just its neighbours, which are close in data space as ensured by the topological organization of selectivities. A read-out mechanism between hidden and output layer maps local input space regions (i.e., sets of prototypes) to class memberships using simple linear regression learning based on SOM input-prototype distances that have been converted into bounded activation values. The mapping from hidden to output layer is adapted only when there is sufficient activation in the internal representation. If this is not the case, e.g., when concept drift is occurring, adaptation is suspended, because random weak activations due to unknown inputs can severely disrupt existing read-out weights.

C. Overview of other incremental learning algorithms

There are a number of approaches for incremental learning with support vector machines (see [14] for an overview), but

in the light of the given definitions, these approaches are closer to online learning and will run into trouble under concept drift. Furthermore, there are ensemble learning algorithms [15], [14] that achieve incremental learning simply by training new classifiers for new batches of data, and combining all existing classifiers for decision making. While this indeed achieves incremental learning under some conditions, it makes the implicit hypothesis that concept drift coincides with new data batches, whereas a *detection* of concept drift is not addressed at all. As the problem of catastrophic forgetting was first remarked for multilayer perceptron (MLP) models [4], [5], it is hardly surprising that there is significant work on the subject of how catastrophic forgetting could be avoided[16], [17], [18], [19], [20], [21], [22], [23] although none of these proposals is completely free of problems and of limitations. To perform incremental learning as defined earlier in this section, most modern approaches perform an explicit local partitioning of the input space and train a separate classification/regression model for each partition [24], [25], [26], [27], [28]. The manner of performing this partitioning is very diverse, ranging from kd-trees [28] to genetic algorithms [27] and adaptive Gaussian receptive fields [24]. Equally, the choice of local models varies between linear models [24], Gaussian mixture regression [28] or Gaussian Processes [25]. Since this article is concerned with high-dimensional perceptual problems, it can be stated for all cited approaches that it is really the partitioning of the input space that is costly in terms of memory. Most notably, covariance matrices used in [24] are quadratic in the number of input dimensions which makes their use prohibitive for high data dimensionalities.

II. Methods

PROPRE is as a three-layer neural network architecture depicted in Figure 2. The size of the input layer is given by the dimension of the data vectors, whereas the size of the output layer is determined by the number of outputs a particular task requires. In this article we are only dealing with classification tasks with population-coded class labels, therefore the size of the output layer corresponds to the number of classes in a task. A free parameter is the size of the hidden layer H which we denote by $n^H \times n^H$. To train the hidden layer H, we use a SOM-based learning scheme (with slight modifications) which is described in Section II-A, whereas the readout from hidden to output layer O is performed by linear regression (LR) explained in Section II-B. A particular point are the *modulation* influences within the architecture, which control and restrict learning in hidden and output layers, see Section II-C. The reaction of PROPRE to the kind of concept drift/shift considered in this article is shown in Fig. 3.

We denote a neural activity vector in a 2D representation X by $z^X(\vec{y},t)$, and weight matrices for SOM and LR, represented by their line vectors attached to target position $y=(a,b)$, by $w^{\text{SOM}}_{\vec{y}}$. For reasons of readability, we often skip the dependencies on space and time and include them only where ambiguity would otherwise occur. Thus we write z^X instead of $z^X(\vec{y},t)$ and w^{SOM} instead of $w^{\text{SOM}}_{\vec{y}}(t)$.

A. Activity generation and prototype training in the hidden layer

The hidden layer H is not intended to reduce the dimensionality of the input but rather to re-encode it in a way that enables incremental learning. Therefore, instead of reducing the output of the hidden layer to the best-matching unit (as it is usually done for the SOM model), we the calculate (graded) activations of *all* hidden layer units for performing the following linear regression. Hidden layer activations z^H are meant to measure a sparse similarity between input and the prototype associated to a particular unit and are normalized in the $[0,1]$ interval. They are obtained by first passing all input-prototype distances $\bar{z}^H$ through a Gaussian function with standard deviation κ, and then applying a transfer function $\text{TF}(\cdot)$ that sparsifies these similarities. Here, there is a technical point to be observed: since there is no way of knowing a priori the typical input-prototype distances, κ must be adapted to the data during the learning process, making it a dynamic, time-dependent quantity:$\kappa \equiv \kappa(t)$.

In order not to trigger the adaptation of too many linear regression weights, the transfer function thresholds these similarities, thus ensuring that only those units whose prototypes are very close to the input are active and take part in linear regression training.

Prototype adaptation is performed using the conventional SOM update step except that it takes into account a control signal $\lambda(t)$ coming from the output level of the hierarchy, which will be described in Section II-C. In precise terms, the equations for activity generation in the hidden layer in response to input activity z^I are as follows:

$$\bar{z}^H(\vec{y}) = ||w^{\text{SOM}}_{\vec{y}} - z^I|| \tag{1}$$

$$\tilde{z}^H = g_\kappa\left(\bar{z}^H\right) \tag{2}$$

$$z^H = \text{TF}_{\theta,p}\left(\tilde{z}^H\right) \tag{3}$$

The adaptation steps for the weights and the distance metric $\kappa(t)$ are as follows:

$$w^{\text{SOM}}_{\vec{y}}(t+1) = w^{\text{SOM}}_{\vec{y}} + \lambda(t)\epsilon^{\text{SOM}} g_\sigma(||\vec{y}-\vec{y}^*||)(z^I - w^{\text{SOM}}_{\vec{y}}) \tag{4}$$

$$\kappa(t+1) = (1-\tau)\ \kappa(t) + \tau\ \max_{\vec{y}} \bar{z}^H(\vec{y}) \tag{5}$$

where $g_\sigma(x)$ is a zero-mean Gaussian function with standard deviation σ, and $\vec{y}^*$ denotes the position of the best-matching unit (i.e., the one with the highest similarity-to-input) in H. In accordance with standard SOM training practices, the SOM learning rate and radius, ϵ^{SOM} and σ, are maintained at ϵ_0, σ_0 for $t < T_1$ and are exponentially decreased afterwards in order to attain their long-term values $\epsilon_\infty, \sigma_\infty$ at $t = T_{\text{conv}}$. TF represents a monotonous non-linear transfer function, $\text{TF}: [0,1] \rightarrow [0,1]$ which we model as follows with the goal of maintaining the BMU value unchanged while non-linearly suppressing smaller values:

$$\text{TF}_{\theta,p}(\tilde{z}^H) = \begin{cases} \tilde{\text{TF}}_p(\tilde{z}^H) & \text{if } \tilde{\text{TF}}(\tilde{z}^H) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where

$$\tilde{\text{TF}}_p(\tilde{z}^H) = \frac{\left(\tilde{z}^H\right)^p}{\max_{\vec{y}}\left(\tilde{z}^H(\vec{y},t)\right)^{p-1}} \tag{7}$$

range	meaning
$\in [-1, -0.5]$	certain, incorrect
$\in [-0.5, -0.1]$	uncertain, incorrect
$\in [0.1, 0.5]$	uncertain, correct
$\in [0.5, 1]$	certain, correct

TABLE I. SIGNIFICANCE OF VARIOUS VALUE RANGES OF THE CONFIDENCE MEASURE $m(z^P)$.

B. Decision making and learning in the output layer

Generation of output layer activities is performed by a simple linear transformation of thresholded hidden layer activities z^H, using the linear regression weights w^{LR}. Learning is subsequently modifying these weights to optimize the mapping of hidden layer activities z^H to the target representation z^T containing the "true" class of a sample.

$$z^P(\vec{y}) = w^{\text{LR}}_{\vec{y}} \cdot z^H \tag{8}$$

$$w^{\text{LR}}_{\vec{y}}(t+1) = w^{\text{LR}}_{\vec{y}}(t) + 2\frac{\epsilon^{\text{LR}} z^H}{(n^H)^2}\left(z^P(\vec{y}) - z^T(\vec{y})\right) \tag{9}$$

In contrast to the hidden layer learning rate, the learning rate of linear regression, ϵ^{LR} remains constant at all times although we normalize it by the size of the hidden layer, so that it need not be changed by hand when the hidden layer size is changed between experimental runs.

C. Feedback control of learning and detection of novelty

Hidden layer and output layer neurons do not adapt their weights all the time, only when it is deemed appropriate. Here, there are two distinct cases to be distinguished: first, when the hidden layer has low overall activity, maybe because the input belongs to a newly added class, then linear regression should not adapt its weights because hidden layer activity is probably not meaningful and it will impair performance for already represented classes. This is achieved automatically by thresholding hidden layer activities in the transfer function $TF_{\theta,p}$ by θ as specified in eqn. (3), leading to zero activity if activity is too low. In this case, linear regression weights will not be updated as this requires non-zero activities in the hidden layer H. Secondly, hidden layer weights w^{SOM} will only be updated when the current estimate of class membership, i.e. the output layer activities z^P, is either uncertain or wrong. To measure uncertainty, we first define an uncertainty measure based on the output layer activities, whose basic idea is that a certain estimate of class membership has a clear activity maximum, so a good measure is just to use the bounded difference between first and second maximum:

$$u(z^P) = \max_{\vec{y}} z^P - \max 2_{\vec{y}} z^P \tag{10}$$

This measure, $u(z^P)$, can be combined with the fact whether the activity maximum of z^P is in the right place, i.e., in accordance with ground-truth information z^T:

$$m(z^P, z^T) = \begin{cases} u(z^P), & \text{if } \arg\max_{\vec{y}} z^P = \arg\max_{\vec{y}} z^T \\ -u(z^P), & \text{otherwise} \end{cases} \tag{11}$$

Finally, we obtain the modulation measure $\lambda(t)$ that decides whether hidden layer weights should be trained, by thresholding the confidence measure $m(z^P, z^T)$:

$$\lambda(t) = \begin{cases} 0, & m(z^P, z^T) > \theta_m \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

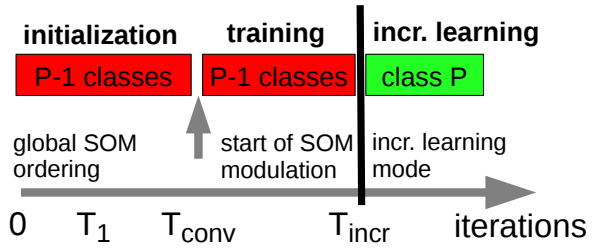

Fig. 4. Type of incremental learning problems considered in this article. For a classification problem with P classes, the PROPRE architecture is trained with $P-1$ classes until iteration T_{incr}. From this time onward, only the left-out class is presented to the architecture, and performance is subsequently evaluated.

Table I gives an overview over the intuitive meaning of various values of the confidence measure m. By thresholding $m(z^P)$ we thus allow hidden layer weights to be trained only when the current class estimate is of less-than-perfect quality, either outright incorrect (for $\theta_m \leq 0$) or correct but of significant uncertainty (for $\theta_m > 0$).

III. EXPERIMENTS

In this section, we will first of all describe the datasets that we are going to use to conduct all experiments in this article. Subsequently, we will give detailed parameter settings for the two incremental learning algorithms that we will employ for experiments. Afterwards, experiments will be conducted for each of the point indicated in Sec. I-A: initially, Sec. III-F will show the basic feasibility of a readout layer for SOM based on simple linear regression. In Sec. III-G, it will be demonstrated that the SOM layer of the PROPRE architecture can detect concept drift/shift by comparing the activation of the BMU to long-term average values. In Sec. III-H, it will be demonstrated that concept drift detection is absolutely necessary for SOM-based incremental learning, as diffuse SOM activations due to concept drift would otherwise break the readout layer's performance.

We wish so emphasize that this article investigates incremental learning faced with a particular, very application-relevant type of concept drift: the addition of a new class of samples starting at a certain time T_{incr}, see Fig. 4.

A. Datasets

In this article, we use three different classification benchmarks[1] in order to investigate incremental learning with SOMs. First of all, we make use of the MNIST dataset of handwritten digits [29] which is a standard benchmark in machine learning. In addition, we use two datasets obtained from a visual pedestrian classification benchmark, the Daimler Pedestrian Detection Benchmark[30]. The first dataset if about pedestrian detection, i.e., the distinction of pedestrian images from background/non-pedestrian samples, whereas the second task is about pedestrian pose classification, assigning one of the four classes "front/back/left/right" to pedestrian samples (no background samples in this task) according to their visually perceived orientation. Please see Fig. 1 for a visualization of the pedestrian-related tasks, whereas we refer to [29] for an in-depth description of the MNIST benchmark.

[1] Available under www.gepperth.net/alexander/downloads/dataWSOM.tar.gz

TABLE II. IMPORTANT PROPERTIES OF THE THREE DATASETS USED IN THIS ARTICLE.

Dataset	#train	# test	dimensions	preprocessing	# classes
MNIST	60.000	10.000	28x28=784	raw	10
Ped.Pose	12.684	2.516	756	HOG	4
Ped.Det.	10.000	19.148	756	HOG	2

TABLE III. PARAMETERS USED IN PROPRE EXPERIMENTS. PLEASE REFER TO SEC. II FOR AN EXPLANATION OF THE SYMBOLS.

n^H	30	θ	0.7	p	10
τ	0.005	ϵ_0	0.1	σ_0	$0.3n^H$
T_1	10000	T_{conv}	80000	ϵ_∞	0.001
σ_∞	0.01	ϵ^{LR}	0.1	θ^m	0.7
T_{incr}	600000	ΔT_{incr}	15000		

The MNIST dataset is a relatively "clean" problem (very little noise and overall variance) where excellent recognition rates can be achieved even on the raw image data. In contrast, the two pedestrian datasets are difficult real-world problems which require a more sophisticated preprocessing in order to be solved with any degree of precision. The applied preprocessing method is termed HOG (histogram of oriented gradients, see [31]) and represents a standard method in real-world visual object recognition. Using the terms of [31], the parameters of the HOG transform applied to cropped images downsampled to a size of 32×64 are: block size 16×16, cell size 8×8, 2×2 cells per block, 9 orientations bins and normalization enabled.

Important global facts about all three datasets are given in Tab. II.

B. Evaluation measures

In the experiments presented later in this section, we present several different measures to evaluate and visualize results. For comparing the topological ordering in a given SOM layer, we resort to the so-called topographic error e_{top} [32]

$$e(t) \equiv \begin{cases} 1 & \text{if } \vec{y}^* \text{ is adjacent to } \vec{y}_2^* \\ 0 & \text{else} \end{cases}$$
$$e_{\text{top}} \equiv 1 - \left\langle e(t) \right\rangle_t, \tag{13}$$

where $\vec{y}_2^*$ is the position of the unit with the second-highest activation after the BMU which resides at $\vec{y}^*$.

Lastly, we consider to classification rate E in order to assess the success of supervised learning in assigning the proper classes to input vectors. Since we assume that the ground-truth vector T and therefore the output category vector P (see Fig. 2) are population-coded, the classification rate is defined as

$$E(t) \equiv \begin{cases} 1 & \text{if } \arg\max_i z_i^P = \arg\max_i z_i^T \\ 0 & \text{else} \end{cases}$$
$$E = \left\langle E(t) \right\rangle_t \tag{14}$$

C. Parametrization of algorithms and organization of experiments

All PROPRE experiments begin with a convergence phase where SOM parameters vary over time (high constant values until iteration T_1 and exponential decrease to asymptotic values until T_{conv}. Subsequently, normal PROPRE training is conducted with modulatory influenced turned on until T_{incr}. Subsequently, either performance is evaluated or an incremental learning step is conducted where typically a new, previously unseen class is presented to the network. The incremental learning step, presenting exclusively examples of a previously unseen class, lasts for I iterations. In order to maintain a topological SOM organization, these new samples must be smoothly integrated into the existing map, which is ensured by increasing the neighbourhood radius of SOM learning to σ_{incr} at $t = T_{\text{incr}}$, and exponentially reducing it to its asymptotic value until $t = T_{\text{incr}} + \Delta T_{\text{incr}}$. The adaptable parameters of the PROPRE algorithm are given in Tab. III. All experiments will use these values unless explicitly stated otherwise.

D. Preliminary experiment: basic feasibility of SOMs for incremental learning

In this experiment, we wish to demonstrate the basic feature of the SOM model that makes it an excellent choice for incremental learning: its purely local update behavior. Using the parameter settings from Sec. III-C but using a hidden layer size of $n^H = 10$ and an asymptotic neighbourhood radius of $\sigma_\infty = 1.0$, we train the PROPRE architecture on the MNIST dataset where the class "0" has been removed from the training data. At $t = T_{\text{incr}}$, we introduce strong concept drift by presenting only the hitherto excluded class 0 and the evolution of SOM prototypes is observed at various times $t > T_{\text{incr}}$, here 500, 1500, 5000 and 9000. In order to obtain a purely SOM-like behavior without any control of updating, we set the predictability threshold to $\theta^m = 1.1$, which results in a guaranteed SOM update for every sample as in the original SOM algorithm, see eqn. (12). The learning rate for the linear read-out layer is set to $\eta = 10^{-1}$. The SOM activation threshold is kept at $\theta = 0.7$ although it is irrelevant here as we wish to investigate the behavior of the SOM only. The results can be observed in Fig. 5.

E. Protecting the SOM layer against sampling bias

As the experiments of the last section (see Fig. 5) show, SOM prototype adaptation in the hidden layer H is strongly impacted by the time this class is presented. The longer it is presented, the more prototypes lean towards the new class, and consequently prototypes describing old classes are forgotten. This is a variation of the sampling bias problem often encountered in machine learning: the frequency of classes during training is not correlated at all to their frequency during application. In our case, this is a highly undesirable effect as it is not at all the time of presenting a new class that should control forgetting, but classification performance! PROPRE therefore adapts prototypes only as long as classification performance for the new class, as measured by the generalized confidence measure of eqn. (11), is unsatisfactory. This effectively protects the SOM layer and makes prototype adaptation totally independent of the time of presentation. To demonstrate this, we repeat the experiments of the previous section but this time using a parameter of $\theta^m = 0.7$, meaning that once the generalized correctness measure for a sample exceeds 0.7, no SOM adaptation is carried out any longer. By varying this parameter one can effectively control forgetting in the SOM layer H. The results are shown in Fig. 6.

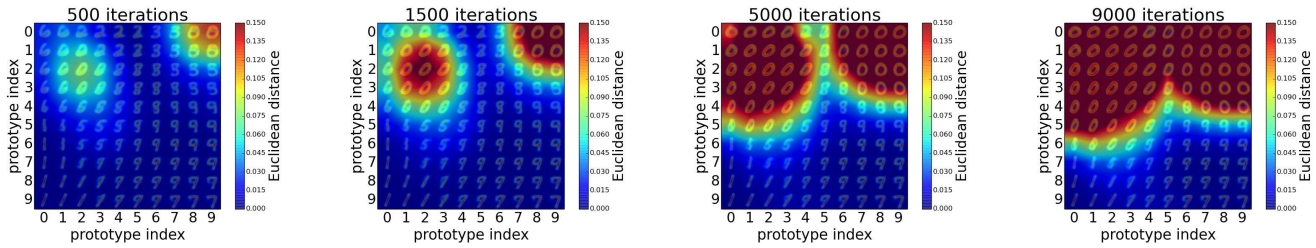

Fig. 5. Demonstration of the purely local update behavior of the SOM model, shown exemplarily for the MNIST handwritten digit dataset. Shown by color coding in each diagram is the euclidean distance between prototypes at times t and T_{incr}, for each prototype with index x/y. For better understanding, the color-coded distance map has been augmented by linear interpolation, and a visualization of prototypes at time $t > T_{\text{incr}}$ is overlaid over each distance image. As t increases, the insertion of the new class affects more of the SOM prototypes but it can be observed that the change is gradual and strictly local.

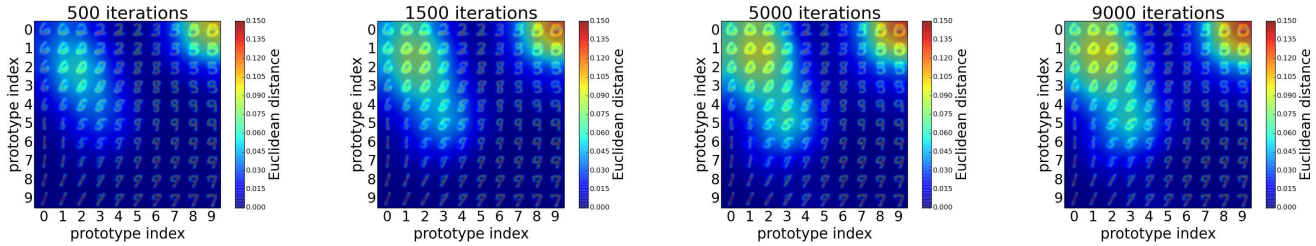

Fig. 6. Protecting the SOM layer against sampling bias, shown exemplarily for the MNIST handwritten digit dataset. Color coding in each diagram indicates the euclidean distance between prototypes at times t and T_{incr}, for each prototype with index x/y. For better understanding, the color-coded distance map has been augmented by linear interpolation, and a visualization of prototypes at time $t > T_{\text{incr}}$ is overlaid over each distance image. As t increases, SOM prototypes are updated and subsequently protected against unnecessary adaptation as new class is sufficiently well recognized by the read-out mechanism. This figure should be compared to Fig. 5 where SOM protection is disabled.

F. Supervised read-out layer for SOM

TABLE IV. CLASSIFICATION ACCURACY E AS A FUNCTION OF HIDDEN LAYER SIZE n^H. AS MAY BE EXPECTED, INCREASING H INCREASES PERFORMANCE.

Task	E_{10x10}	E_{30x30}	E_{50x50}
MNIST	91.3%	96.7%	98.2%
PedDetect	95.1%	97.0 %	99.2%
PedPoses	70.2%	73.8%	74.5%

This experiment is conducted with the settings of Sec.III-C but only until T_{incr}, that is, without incremental learning. We simply wish to establish that a read-out by simple linear regression is indeed possible, and that satisfying results can be achieved in this way. No comparison to deep learning results on MNIST or other datasets is intended, although the classification rates obtained here (see Tab. IV) are close to state-of-the-art results. We find that σ_∞ has a strong impact on classification accuracy and should be chosen very small. We also find that keeping $\theta = 0$ during training increases classification accuracy moderately. To a much lesser extent, the learning rates ϵ^{LR} and ϵ_∞ have an impact in final classification accuracy, although this is very task-dependent.

G. Concept drift detection

For this experiment, we ask the question whether the SOM layer is able to detect the presence of a new class in the training data, or, more precisely, the percentage of samples from this class that can be recognized as "outliers". Using the setting from Sec. III-C except for a hidden layer size of $n^H = 10$, we train the architecture on all three datasets described in Sec. III-A until $t = T_{\text{incr}}$. For each dataset, one class (class 0 for MNIST, the "background" class for pedestrian detection and the "left" class for pose classification) is excluded from training. At $t > T_{\text{incr}}$, the previously excluded class is exclusively presented whereas adaptation in the network is disabled by setting $\epsilon_\infty = 0$, $\epsilon^{\text{LR}} = 0$ and $\tau = 0$. It is assumed that outlier samples generate less BMU activity that inliers, therefore a threshold θ^{out} is applied to BMU activities and a sample is labelled as an outlier $z_{\vec{y}^*} < \theta^{\text{out}}$. By varying θ^{out} in the $[0, 1]$ interval, graphs reminiscent of ROCs can be obtained that indicate how effective this strategy is at detecting outliers while letting pass inliers. These results are depicted, for the three datasets, in Fig. 7.

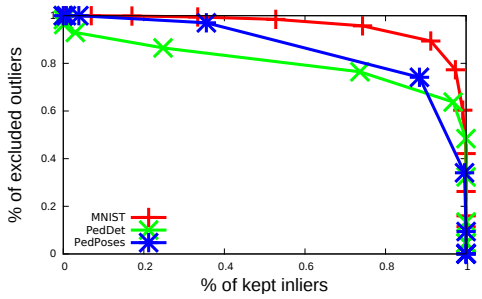

Fig. 7. Elimination of outliers/concept drift using activities in the SOM layer only, shown for the three datasets used in this study. The goal is to exclude the class that was not used during the training phase, thus modeling the effects of abrupt concept drift. As can be expected, MNIST as the cleanest dataset performs best here.

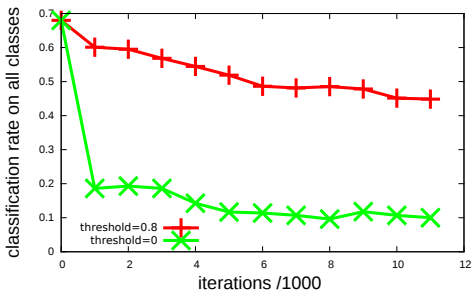

Fig. 8. Effect of the threshold θ during incremental learning. Shown is classification performance on the test set of the pose classification task during incremental learning of the left-out "left" class with $\theta = 0.8$ (red curve) and $\theta = 0$ (green curve). Results clearly show that thresholding "protects" the readout layer against concept drift effects.

H. Concept drift detection for controlling readout

An interesting finding is that the readout layer can be strongly "damaged" by the advent of concept drift of the hidden SOM layer H is not managed appropriately. The addition of a class the SOM layer was not trained on will, in general, cause diffuse, low-level activity in the SOM. The readout mechanism will however try to map this essentially random activity to the new class label, therefore destroying readout weights all over the internal SOM layer. This is the reason behind the introduction of the threshold $theta$ in eqn.(6): it will suppress random activities until such a time as the SOM has partially converged on the new class, which is indicated by super-threshold SOM activity. We show this effect exemplarily for the pedestrian pose classification task (the behavior is exactly the same for all three datasets) in Fig. 8. Using the settings of Sec. III-C except a smaller hidden layer size of $n^H = 10$, we train the PROPRe architecture on all classes but the "left" class until T_{incr}. Subsequently, we add the "left" class, one time with $\theta = 0.8$ and another time using $\theta = 0$. In parallel, we monitor the classification performance on the test (in which all classes are present) set every 1000 iterations. Results can be observed in Fig. 8.

I. Adaptive control of SOM parameters

As the focus of this article is on incremental learning scenarios where a new class is abruptly introduced, it is important to ensure the continued topological organization of the internal SOM layer H. As explained in Sec. II-A, the SOM neighbourhood radius is reduced from initially large values to an asymptotic value. As seen in Sec. III-F, this value needs to be small in order to ensure good classification performance of the whole architecture. Therefore, adding a new class using a very small neighbourhood radius may conceivably introduce strong topological defects, breaking the assumption that close-by prototypes are close in data space as well. As this is the fundamental principle on which we build incremental learning on top of the SOM model, the formation of topological defects must be avoided by appropriate mechanism. A simple way is to simply increase the neighbourhood radius temporarily when a new class is presented, and smoothly reducing it to its asymptotic value over time. sec:exp:detect In order to test whether this approach is necessary and, if so, feasible, we conduct a simple experiment using the MNIST benchmark: training is conducted normally (see Sec. III-C) with a hidden SOM layer size of $n^H = 10$ using all classes except the class 0. In contrast to Sec. III-C, we use parameter values of $\theta^m = 0.1$ to accelerate incremental learning and $T_{\text{incr}} = 200000$ to speed up the experiments. At $t = T_{\text{incr}}$, the class 0 is presented exclusively for 15.000 iterations. The experiment is conducted twice, one time with an initially high neighbourhood radius of $\sigma = 1.5$ that is exponentially decayed to its asymptotic value of $\sigma_\infty = 0.01$, and one time with $\sigma = \sigma_\infty$ throughout the remaining experiment. The development of the topographic error e_{top} and a visualization of the prototype modifications are given in Fig. 9.

IV. Conclusion

The basic message of this article is that SOMs are an extremely useful tool for performing incremental learning if appropriately managed. We presented the PROPRE architecture for incremental learning in very high dimensions which draws its incremental learning capacity from its internal SOM layer, and showed that PROPRE's basic classification ability (without incremental learning) is close to the state-of-the-art on several challenging real-world tasks. The internal SOM layer exhibits localized and gradual update behavior as demonstrated in Sec. III-D which avoids catastrophic forgetting, but can also detect concept drift by BMU analysis (see Sec. III-G. Based on this ability, we presented PROPRE's mechanisms of controlling its SOM layer in order maintain topological organization (see Sec. III-I) and stability in the face of sampling bias (see Sec. III-E), and verified that they are necessary ingredients when performing incremental learning with SOMs.

It is clear that incremental learning as performed here (first train with $P-1$ classes, then train with remaining class only) would need to be complemented by a re-training phase where all P classes are presented together in order to compensate for SOM units that now respond to the newly added class, and indeed it has been shown in [12] that this is a perfectly viable strategy. Other incremental learning methods such as LWPR who have adaptive model complexity can avoid such steps as the addition of a new class does not impair the representation of the "old" ones as long as there is no overlap. Since we wish, for reasons of real-world processing capacity, to have constant model complexity, incremental learning becomes slightly more complicated but, on the other hand, a fixed upper bound on computational resources is guaranteed with our approach.

Future work will, first of all, to automate the incremental learning step: for now, parameters are set appropriately (high θ, namely) because the onset of concept drift is known in advance. A mechanism will have to be devised that uses the concept drift detection mechanism shown in Sec. III-G and sets internal parameters automatically, thereby removing the need to tune these parameters by hand. Furthermore, incremental learning scenarios which are less radical than the one considered here are of interest, e.g., when an unknown, new class comes interspersed with samples of known classes. In this case, concept drift can neither be detected not being adapted to because misclassifications will occur only rarely. The introduction of a short-term storage system for misclassified samples as introduced in [12] might be a fruitful approach here which will be pursued in future works.

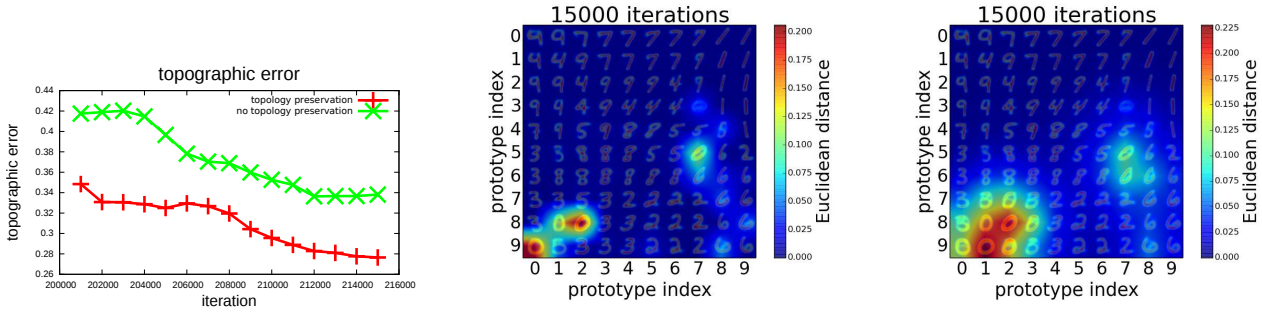

Fig. 9. Left: topographic error e_{top} over time when adding a new class, depending on whether SOM topology is "protected" or not. Middle: prototype changes at $t = 215.000$ w.r.t. $t = 200.000$, no topology preservation. Right: same diagram only this time with topology preservation. We observe that transition in the latter case is smoother and no structural defects are introduced, which is further corroborated by the consistently lower e_{top}. The price is of course the "overwriting" of a larger area of the original SOM layer.

REFERENCES

[1] A Gepperth and B Hammer. Incremental learning algorithms and applications. In *European Sympoisum on Artificial Neural Networks (ESANN)*, 2016.

[2] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005.

[3] R.J. May, H.R. Maier, and G.C. Dandy. Data splitting for artificial neural networks using som-based stratified sampling. *Neural Networks*, 23(2):283 – 294, 2010.

[4] M. McCloskey and N. Cohen. Catastrophic interference in connectionist networks: the sequential learning problem. In G. H. Bower, editor, *The psychology of learning and motivation*, volume 24. 1989.

[5] R. Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97, 1990.

[6] RM French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connect. Sci.*, 4, 1992.

[7] RM French. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychol Rev.*, 97(2), 1990.

[8] McCloskey M and Cohen NJ. Catastrophic interference in connectionist networks: the sequential learning problem. *Psychol. Learn. Motiv.*, 24, 1989.

[9] Pallavi Kulkarni and Roshani Ade. Incremental learning from unbalanced data with concept class, concept drift and missing features: a review. *International Journal of Data Mining and Knowledge Management Process*, 4(6), 2014.

[10] Alexey Tsymbal. The problem of concept drift: definitions and related work. Technical report, Computer Science Department, Trinity College Dublin, 2004.

[11] Tom M Heskes and Bert Kappen. Error potentials for self-organization. In *Neural Networks, 1993., IEEE International Conference on*, pages 1219–1223. IEEE, 1993.

[12] A Gepperth and C Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, pages 1–11, 2016.

[13] A Gepperth and M Lefort. Biologically inspired incremental learning for high-dimensional spaces. In *IEEE International Conference on Development and Learning (ICDL)*, 2015.

[14] Y. M. Wen and B. L. Lu. Incremental learning of support vector machines by classifier combining. In *Proc. of 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, volume 4426 of *LNCS*, 2007.

[15] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31(4):497–508, 2001.

[16] N. Sharkey and A. Sharkey. An analysis of catastrophic interference. *Connection Science*, 7(3-4), 1995.

[17] R. M. French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*. 1994.

[18] J. Murre. The effects of pattern presentation on interference in backpropagation networks. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*. 1992.

[19] R. M. French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 4, 1992.

[20] C. Kortge. Episodic memory in connectionist networks. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. 1990.

[21] Ian J Goodfellow, Mehdi Mirza, Xia Da, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

[22] J. Krushke. ALCOVE: An exemplar-based model of category learning. *Psychological Review*, 99, 1992.

[23] S. Sloman and D. Rumelhart. Reducing interference in distributed memories through episodic gating. In A. Healy and S. Kosslynand R. Shiffrin, editors, *Essays in Honor of W. K. Estes*. 1992.

[24] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high-dimensional spaces. In *International Conference on Machine Learning*, 2000.

[25] D. Nguyen-Tuong and J. Peters. Local gaussian processes regression for real-time model-based robot control. In *IEEE/RSJ International Conference on Intelligent Robot Systems*, 2008.

[26] O. Sigaud, C. Sagan, and V. Padois. On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems*, 2011.

[27] M. Butz, D. Goldberg, and P. Lanzi. Computational complexity of the xcs classifier system. *Foundations of Learning Classifier Systems*, 51, 2005.

[28] T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. 2010.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In S. Haykin and B. Kosko, editors, *Intelligent Signal Processing*, pages 306–351. IEEE Press.

[30] M. Enzweiler and D.M. Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, 2009.

[31] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[32] Daniel Polani. Measures for the organization of self-organizing maps. In *Self-Organizing neural networks*, pages 13–44. Springer, 2002.