OSNN: An Online Semisupervised Neural Network for Nonstationary Data Streams

Rodrigo G. F. Soares, Leandro L. Minku, *Senior Member, IEEE,*

***Abstract*—Learning from data streams that emerge from nonstationary environments has many real-world applications and poses various challenges. A key characteristic of such a task is the varying nature of target functions and data distributions over time (concept drifts). Most existing work relies solely on labeled data to adapt to concept drifts in classification problems. However, labeling all instances in a potentially life-long data stream is frequently prohibitively expensive, hindering such approaches. Therefore, we propose a novel algorithm to exploit unlabeled instances, which are typically plentiful and easily obtained. The algorithm is an Online Semisupervised Radial Basis Function Neural Network (OSNN) with manifold-based training to exploit unlabeled data while tackling concept drifts in classification problems. OSNN employs a novel Semisupervised Learning Vector Quantization (SLVQ) to train network centers and learn meaningful data representations that change over time. It uses manifold learning on dynamic graphs to adjust the network weights. Our experiments confirm that OSNN can effectively use unlabeled data to elucidate underlying structures of data streams while its dynamic topology learning provides robustness to concept drifts.**

***Index Terms*—Data stream learning, Nonstationary environments, Online Semisupervised learning, manifold learning, neural networks**

I. Introduction

Tackling concept drift is a challenging problem, and many approaches have been proposed for that [1]. Most existing work relies on incoming labeled training examples to react and adapt to concept drift in classification or regression problems. However, labeling all instances in a potentially infinite data stream is usually impractical. This is because, despite unlabeled data being typically abundant and relatively cheap to acquire through automated mechanisms, labeling such instances is frequently a manual task that relies on human resources, being potentially very costly and time consuming. For example, in software effort estimation [2], an entire team may need to keep track of the effort that they are spending on a given project, throughout the course of the project, to label a single software project instance. In contrast, input attributes describing the software project are more readily available [3].

To deal with this problem, a potential approach would be to make use of the abundant unlabeled incoming instances together with a small number of labeled incoming points through online semisupervised learning. In this sense, the otherwise discarded unlabeled instances could, in fact, help data stream applications to react and adapt to concept drift and find better predictive models. However, the use of the distribution of unlabeled data to aid the training of semisupervised classification algorithms often requires the entire unlabeled dataset at disposal beforehand [4], which is not possible in data stream scenarios. Therefore, the majority of the semisupervised classification methods are not designed or easily extended to online learning. Very few studies investigated Online SemiSupervised Classification (OSSC), and most are unable to deal with concept drift [5], [6]. The only approaches [7], [8] able to perform online learning based on mixed labeled and unlabeled data that have considered concept drift are limited to specific types of concept drift and have been proposed in the context of active learning, potentially requiring a human annotator to label specifically chosen instances.

Therefore, our work aims at developing a more flexible OSSC approach for classification problems with nonstationary data streams where any incoming instance may or may not have a label and any type of concept drift may occur. Each instance (labeled or unlabeled) is learned as soon as it arrives. Overall, we answer the following Research Questions (RQs):

RQ1: *How can a neural network benefit from unlabeled data in online data stream learning of nonstationary classification problems?* Radial Basis Function Network (RBFN) have the ability to learn compact data representations through the training of their hidden layer [9], [10], whilst adjusting a decision boundary through the weights of the output layer. The training of the hidden layer can be performed with both labeled and unlabeled data, meaning that RBFNs lend themselves to semisupervised learning. Therefore, we answer RQ1 by proposing the first RBFN that is able to perform OSSC in nonstationary data streams, called Online Semisupervised Radial Basis Function Neural Network (OSNN).

RQ2: *Is the proposed approach able to benefit from unlabeled data to improve predictive performance over its supervised counterpart when we have nonuniform labeling distributions? And when we have uniform labeling distribution?* We consider stream learning scenarios where the probability of labeling an instance is *uniform* across the input space. This analysis is based on stream applications where the presence of labels is not a systematic event, that is, the knowledge to label instances is available entirely at random, independent from any underlying process. We also study streams with *nonuniform*

Rodrigo G. F. Soares is with the Department of Statistics and Informatics of the Federal Rural University of Pernambuco, Recife, Brazil. rodrigo.gfsoares@ufrpe.br.

Leandro L. Minku is with the School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK. L.L.Minku@cs.bham.ac.uk.

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grant Number EP/R006660/2 and the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186.

labeling distribution, where the probability of labeling an instance differs depending on the value of the input attributes. Such a scenario simulates stream learning applications where the knowledge to label instances is only available at certain regions of the input space within a given concept.

RQ3: *How does the proposed approach perform compared to existing data stream learning approaches in the uniform and nonuniform labeling distribution cases?* To establish the effectiveness and robustness of OSNN to the different factors involved in OSSC, we answer RQ3 by performing a comparative study with OSNN and state-of-the-art stream learning algorithms. We will investigate these approaches with different amounts of labels, types of drift and streams.

Experiments have been performed to answer the RQs, where we validate our approach, compare OSNN to its fully supervised counterpart and analyze its performance versus several state-of-the-art methods. The results showed that OSNN can exploit unlabeled data to improve generalization; was effective in the vast majority of the comparisons with existing approaches; and is robust to different types of concept drift and numbers of labels.

Our novel contributions are i) a Semisupervised Learning Vector Quantization (SLVQ) training algorithm that continuously adjusts basis functions to represent data and to implicitly handle concept drifts; ii) an online manifold regularization mechanism that employs the graph produced by SLVQ to train the network weights; iii) the first efficient and robust OSSC RBFN algorithm that is able to use unlabeled data to improve predictive performance in nonstationary environments compared to supervised online RBFNs.

The remainder of this paper is organized as follows. Section II presents the definitions of OSSC. Section III presents the state-of-the-art of stream learning. Section IV introduces our approach to OSSC. The experimental studies to answer RQ1, RQ2 and RQ3 are in Sections V, VI and VII, respectively. And Section VIII presents our conclusions and future work.

II. Problem formulation

Online learning algorithms update predictive models whenever a new training example becomes available, possibly also making use of a limited number of past examples stored in memory [11]. In particular, sliding window approaches store a buffer containing the N most recent training examples. This buffer can be referred to as a sliding window, minibatch or chunk of size N. These methods iterate over chunks to produce good decision boundaries for noisy or difficult classification.

A data stream is a sequence of instances $\big(s^{(1)}, \ldots, s^{(t)}, \ldots\big)$, where $s^{(t)} = (\mathbf{x}^{(t)}, y^{(t)})$ if $s^{(t)}$ is labeled or $s^{(t)} = \mathbf{x}^{(t)}$ if $s^{(t)}$ is unlabeled; $\mathbf{x}^{(t)} \in \mathbb{R}^D$ and $y^{(t)} \in \{y_1, \ldots, y_{CL}\}$ denote the labels. In this work, $CL = 2$, i.e., we focus on binary classification problems. Each value of t is referred to as a time step. A classifier may store a limited amount of the latest $N > 0$ instances in a minibatch $B^{(t)} = \{s^{(t-N+1)}, \ldots, s^{(t)}\}$. In online learning, the classifier is updated whenever a new $s^{(t)}$ becomes available by using the minibatch. We also define the sets $B_l^{(t)} = \{s \mid s \in B^{(t)} \wedge s \text{ is labeled}\}$ and $B_u^{(t)} = \{s \mid s \in B^{(t)} \wedge s \text{ is unlabeled}\}$ of size L and U, respectively. We assume that both labeled and unlabeled data are received over time, and not only in the initial portion of the data stream as in extreme verification latency [12] scenarios. Online supervised algorithms use only $B_l^{(t)}$ for learning, whereas online semisupervised methods use both $B_l^{(t)}$ and $B_u^{(t)}$.

We assume labeled instances are drawn from an unknown distribution $\mathcal{P}^{(t)}$ and unlabeled points follow the marginal distribution $p^{(t)}(\mathbf{x})$ of $\mathcal{P}^{(t)}$. The fundamental assumption of SemiSupervised Classification (SSC) and, more specifically, OSSC is that $p^{(t)}(\mathbf{x})$ and the conditional $p^{(t)}(y|\mathbf{x})$ have a relationship that, if found, can be exploited for better function learning [13]. In other words, SSC methods attempt to induce $p^{(t)}(\mathbf{x})$ via unlabeled data to improve generalization over methods that use solely labeled data. When the context is clear, we will omit the time step superscript to simplify notation.

A concept drift is a change in the joint distribution, that is, $p^{(t+1)}(\mathbf{x}, y)$ might differ from $p^{(t)}(\mathbf{x}, y)$ within the length of a stream. Such changes can be categorized according to the rate at which a new concept becomes the true target. There are two types of drift: abrupt, in which the change occurs instantly (in one time step); and gradual, where the new concept takes place slowly over time.

To exploit unlabeled data, SSC algorithms follow assumptions over $p^{(t)}(\mathbf{x})$ [4]. Many effective SSC methods assume that $p^{(t)}(\mathbf{x})$ follows cluster structures [4] (cluster assumption). They often implement clustering procedures [10] to estimate $p^{(t)}(\mathbf{x})$, which incurs in a computational overhead and in the tuning of clustering hyperparameters. In contrast, the *manifold assumption* states that a meaningful structure of the data lies in a low-dimensional nonlinear manifold embedded in the high dimensional input space [4].

III. Related Work

Several RBFNs have been employed to handle online learning [14], [15], [16]. Huang et al. proposed the approaches Growing and Pruning RBFN (GAP-RBF) [14] and Generalized GAP-RBF (GGAP-RBF) [15] to dynamically adjust the architecture of the network via growing and pruning strategies. Later, Zhang et al. [16] proposed Fast GAP-RBF (FGAP-RBF), which extended GAP-RBF with improved growing and pruning techniques for better predictive performance. However, these approaches are unable to use unlabeled data and are not prepared to tackle concept drift. In [5], the authors proposed an efficient algorithm that performs OSSC under severe time and memory constraints. Goldberg et al. [17] proposed to use either a buffer to save a small amount of data points or online random projection trees to represent the manifold structure in an online manifold update procedure via label propagation. Shen et al. [6] proposed an LVQ approach to OSSC that handles labeled and unlabeled data separately. They proposed a conditional log-likelihood mechanism to adjust prototypes with labeled data and used Gaussian Mixture Model and Neural Gas to include unlabeled data. Except for [17], all these techniques cannot handle nonstationary streams and may degrade their generalization accuracy when a new concept emerges. Despite being prepared for nonstationary streams, the technique in [17] was shown to be outperformed by methods that cannot handle concept drifts [6].

A few approaches to OSSC within different application contexts exist. Shen et al. [8] use a self-organizing neural network to represent the data topology and to divide nodes into different clusters. OASIS [7] is a Bayesian model that introduces a semisupervised likelihood function along with an efficient online Bayesian updating rule. Semi-Supervised Adaptive Novel Class Detection and Classification over Data Stream (SAND) [18] is a semi-supervised ensemble that estimates classifier confidence in predicting instances from evolving data stream. These approaches may require a human annotator to label specific queried instances, which is not the scenario tackled by this paper. In addition, Shen et al. [8]'s approach only contains mechanisms to efficiently deal with concept drifts affecting $p(y|\mathbf{x})$ when such drifts are associated to changes in $p(\mathbf{x})$. And, even though the acceptance probability used to move particles in OASIS [7] is prepared for concept drifts, the posterior of each particle is updated based on a rule that is unprepared for concept drifts.

The COMPOSE framework [12] and the Stream Classification Algorithm Guided by Clustering (SCARGC) [19] also use unlabeled data in data streams. However, they consider an extreme verification latency scenario where the approaches cannot benefit from labeled data that may arrive after an initial learning stage, which is not the scenario tackled in this paper. COMPOSE also assumes that the concept drifts are gradual, i.e., take several time steps to complete. Our proposed algorithm is applicable not only to gradual, but also abrupt drifts, i.e., drifts that happen suddenly.

Data stream learning algorithms for nonstationary environments are typically categorized into explicit – which use explicit methods to detect concept drifts and activate an adaptation procedure – or implicit – where the algorithm is continuously updated over time without explicit drift detection. Explicit algorithms are often the most effective techniques for abrupt drifts. Implicit approaches often deliver better generalization for gradual drifts [1].

State-of-the-art online classifiers for nonstationary data streams employ ensemble learning. They require larger computational resources and are not capable of using unlabeled data. Since adapting to concept drifts is crucial to nonstationary data stream applications, these algorithms are, in fact, more closely related to our study than the aforementioned semisupervised techniques. Diversity for Dealing with Drift (DDD) [20], Diversity Pool (DP) [21] and Concept Drift Handling Based on Clustering in the Model Space (CDCMS) [22] are explicit ensemble approaches that employ diversity to select and train base learners. Recurring Concept Drift (RCD) [23] is an explicit ensemble approach that employs a statistical test and memory management to identify recurring concepts. Online Accuracy Update Ensemble (OAUE) [24] is an implicit ensemble approach that maintains a weighted majority vote ensemble with a pruning mechanism. A base learner frequently adopted in data stream learning ensembles is Hoeffding Tree (HT) [25], which is an incremental decision tree algorithm able to handle large-scale data. While these ensembles produce high generalization accuracy, they have the computational burden of training multiple base learners which is aggravated by the amount of data that must be processed in typical data stream applications.

Some approaches have also been proposed to enable tackling unstructured data such as images based on deep learning. The Parsimonious Network (ParsNet) [26] is an explicit self-evolving deep neural network designed for the weakly-supervised learning scenario. It uses self-labeling with the hedge method to address the accumulation of mistakes in generating pseudo-labeled instances. ParsNet depends on a drift detection mechanism to add and prune hidden neurons. Li et al. [27] proposed an incremental semi-supervised learning algorithm formed of a generative network, a discriminant structure and a bridge. Being composed of deep neural networks, these approaches may also lead to a computational time overhead on data stream learning.

IV. A NEURAL NETWORK FOR ONLINE SEMISUPERVISED LEARNING

In this section, we present our novel RBFN OSSC training algorithm for nonstationary environments.

A. *OSNN loss function*

In this section, we introduce our novel loss function. By employing relevant manifolds instead of the original input, classifiers may improve generalization [4]. Typically, such methods perform manifold learning by maximizing smoothness of their output over graphs [13]. To use the induced $p^{(t)}(\mathbf{x})$ to improve generalization, we employ Manifold Regularization (MR) [13]. It assumes that one can approximate $p^{(t)}(\mathbf{x})$ with a compact manifold used in a regularizer to learn a smooth function along the manifold, that is, the regularization minimizes the function complexity along the manifold. The classifier smoothness can be interpreted as the sum of the local variations of the classifier function among nearby instances [28].

OSNN assumes that network centers $C^{(t)} = \{\mathbf{c}_1, \ldots, \mathbf{c}_H\}$ and incoming instances $B^{(t)}$ can be interpreted as vertices $V^{(t)} = C^{(t)} \cup B^{(t)}$ and the similarity between them forms edges in a graph with similarity (adjacency) matrix $\mathbf{S}^{(t)}$. Any similarity measure can be used. See Section IV-C for details. OSNN outputs posterior class probabilities, $f_i = f^{(t)}(\mathbf{x}_i)$, where $0 \leq f_i \leq 1$ and $\mathbf{x}_i \in B^{(t)}$. We estimate the smoothness of the classifier function f using estimates (pseudolabels) u_i of the desired target according to the classifier outputs obtained in the vicinity of points $\mathbf{x}_i$. Each $0 \leq u_i \leq 1$ of each vertex i is

$$u_i = \frac{\sum_{j \in V^{(t)}} S_{ij} \hat{y}_j}{\sum_{j \in V^{(t)}} S_{ij}}, \tag{1}$$

where $\hat{y}_j = y_j$ if $\mathbf{x}_j$ is labeled and $\hat{y}_j = f_j$ otherwise.

To encourage smoothness along nearby points, we use these estimates to penalize the classifier if f changes abruptly for adjacent points. Since we are focusing on binary classification, we employ cross entropy to define such a regularization (second term) and to determine the supervised loss (first term) in our loss function:

$$\begin{aligned}\mathcal{L}(B^{(t)}, \mathbf{w}) = &- \frac{1}{L} \sum_{i \in B_l^{(t)}} [y_i \ln(f_i) + (1 - y_i) \ln(1 - f_i)] \\ &- \frac{\lambda}{U} \sum_{i \in B_u^{(t)}} [u_i \ln(f_i) + (1 - u_i) \ln(1 - f_i)] \\ &+ \frac{\alpha}{2N} ||\mathbf{w}||^2,\end{aligned} \tag{2}$$

where the first term is the supervised cross-entropy; the second term is the manifold regularization, which captures the smoothness of the classifier outputs; the third term $||\mathbf{w}||^2$ is the l_2 regularization on the network weights $\mathbf{w} = [w_1, \ldots, w_H]^\top$; and the hyperparameters λ and α control the degrees of the manifold and l_2 regularizations, respectively.

The manifold assumption fits our purpose as we can use a dynamic graph to represent ever-changing concepts. Since data stream scenarios only provide one instance at a time, our approach is based on manifold regularization that incrementally keeps track of the current data distribution while being able to continuously adapt to sudden or gradual concept drifts.

MR requires all data to build a graph, which is not possible with data stream applications. A sparse representation accounts for a meaningful portion of information of a signal within a linear combination of elementary signals (basis). In this context, RBFN would fit an online manifold-based method well, as the centers in the hidden layer, along with the current labeled and unlabeled data at each time step can be interpreted as a sparse estimate of the current manifold and could be continuously optimized so that new concepts are learned; and the output layer is a classifier that could be regularized according to the smoothness of the learned function over the current manifold. Next sections introduce the proposed RBFN learning algorithm, which is based on this idea.

B. Semisupervised learning vector quantization

In this section, we present our novel SLVQ. RBFN consists of three feedforward layers. The input layer corresponds to the input vector $\mathbf{x}_i \in \mathbb{R}^D$. The hidden layer is formed of H nodes with basis functions $\phi_{ij} = \phi(\mathbf{x}_i, \mathbf{c}_j, \sigma_j) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{c}_j||^2}{2\sigma_j^2}\right)$, where $\mathbf{c}_j$ is the center and σ_j is the width of the radial basis function ϕ_{ij} of node j. In this study, we are considering binary classification, hence the output layer has a single node with activation $f_i = \frac{1}{1+\exp(-z_i)}$, where $z_i = \sum_{j=1}^H w_j \phi_{ij}$.

Our approach to MR uses the basis function centers $\mathbf{c} \in \mathbb{R}^D$, the instances $B^{(t)}$ available at a time t, and a similarity matrix to form a graph and estimate a manifold. Centers can be trained by gradient descent [29]. However, such a method incurs problems, such as choosing learning rates and avoiding local minima. There is also a tendency of widths to grow large, producing nonlocalized basis functions [9]. Learning Vector Quantization [30] has shown good predictive performance for training RBFN centers in the offline supervised setting [29]. Since only one new instance becomes available at a time, we consider these centers as a smoothed sparse representation of data. Such centers should incrementally and sparsely approximate the geometry of a potential manifold with each incoming instance. Since MR requires the minimization of local variations of f, we propose the Semisupervised Learning Vector Quantization (SLVQ) to use centers and unlabeled data to approximate $p^{(t)}(\mathbf{x})$. We introduce a closed-form solution that does not present the issues of gradient descent.

Our method extends the LVQ variation in [31], which has more stable convergence than the original LVQ. As suggested in [31], to enforce each center to summarize the local geometry of the manifold, we use 1-Nearest-Neighbor (1NN) as the quantizer for SLVQ. A Vector Quantization (VQ) of dimension D and size H is the mapping from an input space in $\mathbb{R}^D$ into a set of centers (codebook) $C^{(t)}$. Each center has a region of influence in which corresponding input points are mapped to it according to 1NN. Each region is defined as

$$R_i = \left\{ \mathbf{x} \mid d(\mathbf{x}, \mathbf{c}_i) = \min_{j=1,\ldots,H} d(\mathbf{x}, \mathbf{c}_j) \right\},$$

where $d(\mathbf{x}, \mathbf{c}_i)$ is a distance measure. The VQ mapping $q(\mathbf{x})$ is denoted by $q(\mathbf{x}) = \sum_{i=1}^H \mathbb{1}(\mathbf{x} \in R_i)\mathbf{c}_i$, where $\mathbb{1}(\cdot)$ is the indicator function.

A modified version of the Bayesian expected loss was proposed in [31]. This approach employed a modified probability density function $g_i(\mathbf{x})$ that returns zero values on Bayesian borders instead of typical class-conditional probability density function. Our proposed SLVQ introduces a second term to account for unlabeled data to the loss functional proposed in [31], so that it encourages the codebook to follow the marginal distribution $p^{(t)}(\mathbf{x})$ induced by unlabeled data. It minimizes the quantization loss $d(\mathbf{x}, q(\mathbf{x}))$ along $p^{(t)}(\mathbf{x})$ to represent the data geometry. Hence, SLVQ consists of finding a set of centers $C^{(t)}$ that minimizes the average quantization error defined by the functional

$$\begin{aligned}\mathcal{I}_1[q] = &\sum_{i=1}^{CL} \int_{Q_i} d(\mathbf{x}, q(\mathbf{x})) g_i(\mathbf{x}) d\mathbf{x} + \\ &\int_{-\infty}^{\infty} d(\mathbf{x}, q(\mathbf{x})) p(\mathbf{x}) d\mathbf{x},\end{aligned} \tag{3}$$

where Q_i is the Bayes region for class y_i, CL is the number of classes (in this work, $CL = 2$) and $d(\mathbf{x}, q(\mathbf{x})) = \sum_{i=1}^H \mathbb{1}(\mathbf{x} \in R_i) d(\mathbf{x}, \mathbf{c}_i)$ denotes the loss associated to the VQ $q(\mathbf{x})$. The modified density function is

$$g_i(\mathbf{x}) = \frac{p(\mathbf{x}|y_i')p(y_i') - p(\mathbf{x}|y_i'')p(y_i'')}{K_i},$$

where y_i' and y_i'' are the classes with highest and lowest posteriors in the Bayes region Q_i, respectively. The constant K_i is a scaling factor that ensures that g_i is a probability density function. Assuming there exists a Bayes region Q_{ij} for every R_j that satisfies $Q_{ij} \cap R_j \approx R_j$ and substituting

$d(\mathbf{x}, q(\mathbf{x}))$, Equation 3 can be approximated by

$$\begin{aligned}\mathcal{I}_2[q] =& \sum_{j=1}^{H} \int_{R_j} d(\mathbf{x}, \mathbf{c}_j) g'_j(\mathbf{x}) d\mathbf{x} + \\ & \sum_{j=1}^{H} \int_{R_j} d(\mathbf{x}, \mathbf{c}_j) p(\mathbf{x}|R_j) d\mathbf{x},\end{aligned} \tag{4}$$

where

$$g'_j = \frac{p(\mathbf{x}|y'_j, R_j)p(y'_j|R_j) - p(\mathbf{x}|y''_j, R_j)p(y''_j|R_j)}{K'_j}$$

is the 1NN class-conditional density for the winner class y'_j of the region R_j and K'_j is a constant that ensures that g'_j is a probability density function.

As the class-conditional probability functions are unknown, it is not possible to solve Equation 4. However, by using empirical risk minimization, $\mathcal{I}_2[q]$ can be approximated by

$$\begin{aligned}\mathcal{I}_{\text{emp}}[q] =& \frac{1}{L}\Bigg[\sum_{i \in C^{(t)}} \sum_{j \in B_l^{(t)}} \mathbb{1}(\mathbf{x}_j \in R_i) \mathbb{1}(y_j = y'_i) d(\mathbf{x}_j, \mathbf{c}_i) - \\ & \sum_{i \in C^{(t)}} \sum_{j \in B_l^{(t)}} \mathbb{1}(\mathbf{x}_j \in R_i) \mathbb{1}(y_j = y''_i) d(\mathbf{x}_j, \mathbf{c}_i)\Bigg] + \\ & \frac{1}{U} \sum_{i \in C^{(t)}} \sum_{j \in B_u^{(t)}} \mathbb{1}(\mathbf{x}_j \in R_i) d(\mathbf{x}_j, \mathbf{c}_i)\end{aligned} \tag{5}$$

Since only the instances in region R_i affect its center, Equation 5 only evaluates locally-defined distances between instances and centers, which corresponds to the manifold assumption [4]. Using the Euclidean distance $d(\mathbf{x}_j, \mathbf{c}_i) = ||\mathbf{x}_j - \mathbf{c}_i||$ and solving $\frac{\partial \mathcal{I}_{\text{emp}}[q]}{\partial \mathbf{c}_i} = 0$ for each $\mathbf{c}_i$, we have

$$\begin{aligned}\frac{\partial \mathcal{I}_{\text{emp}}[q]}{\partial \mathbf{c}_i} =& \frac{1}{L}\left[\sum_{j \in B'_{li}} (\mathbf{c}_i - \mathbf{x}_j) - \sum_{j \in B''_{li}} (\mathbf{c}_i - \mathbf{x}_j)\right] + \\ & \frac{1}{U} \sum_{j \in B_{ui}} (\mathbf{c}_i - \mathbf{x}_j) = 0,\end{aligned}$$

where B'_{li} and B''_{li} contain the labeled instances of the majority and minority classes of region R_i that belong to $B_l^{(t)}$, respectively, as a consequence of employing 1NN quantizer to minimize Equation 3. The set B_{ui} has the unlabeled instances that belong to region R_i. We expand this equation and obtain

$$\begin{aligned}&\frac{1}{L}\left(|B'_{li}|\mathbf{c}_i - |B''_{li}|\mathbf{c}_i\right) + \frac{1}{U}|B_{ui}|\mathbf{c}_i = \\ &\frac{1}{L}\left(\sum_{j \in B'_{li}} \mathbf{x}_j - \sum_{j \in B''_{li}} \mathbf{x}_j\right) + \frac{1}{U} \sum_{j \in B_{ui}} \mathbf{x}_j,\end{aligned}$$

which can be rewritten as

$$\mathbf{c}_i = \frac{\frac{1}{L}\left(\sum_{j \in B'_{li}} \mathbf{x}_j - \sum_{j \in B''_{li}} \mathbf{x}_j\right) + \frac{1}{U}\sum_{j \in B_{ui}} \mathbf{x}_j}{\frac{1}{L}\left(|B'_{li}| - |B''_{li}|\right) + \frac{1}{U}|B_{ui}|} \tag{6}$$

Equation 6 trains each center $\mathbf{c}_i$ to minimize its distances to the majority class in its region R_i and to avert instances from the minority class, while moving towards a position of higher density of unlabeled points. In contrast to [6], SLVQ considers the relationship between labeled and unlabeled points. Centers are only affected by incoming instances that fall into their region. Only the appropriate centers are updated to represent such incoming data, whilst the other centers still represent other regions of previously learned data. This enables one-pass learning if desired, as the knowledge stored by the set of centers is not reset when $N = 1$. The codebook $C^{(t)}$ is then optimized to follow the intrinsic geometry of both labeled and unlabeled input data.

The width σ_i of basis function with center $\mathbf{c}_i$ is set to a proportion β of the mean of the Euclidean distances to the other centers [29] as

$$\sigma_i = \frac{\beta}{H} \sum_{j \in C^{(t)}} ||\mathbf{c}_i - \mathbf{c}_j|| \tag{7}$$

The hyperparameter β controls the radius of influence of each basis function. Both the centers $\mathbf{c}_i$ and the widths σ_i are calculated at each time step, so that the basis functions are rescaled to handle concept drifts, otherwise these functions would be left unlocalized.

C. Semisupervised weight training

Our novel weight training method consists of minimizing Equation 2. Since N and H can be relatively small, we employ the Iterative Reweighted Least Squares (IRLS) with a Newton-Raphson update rule [9] as it delivers high predictive performance despite the time complexity of inverting the Hessian matrix. One might use a first order optimization method to improve time complexity at the cost of generalization ability.

To obtain the similarity matrix $\mathbf{S}^{(t)}$ used to calculate each pseudolabels u_i, we augment the set of centers with the current minibatch $V^{(t)} = C^{(t)} \cup B^{(t)}$. We define $\mathbf{S}^{(t)} = \{S_{ij}\}_{i,j=1}^{H+N}$ for each $\mathbf{v}_i, \mathbf{v}_j \in V^{(t)}$ based on an RBF kernel:

$$S_{ij} = \exp\left(\frac{-||\mathbf{v}_i - \mathbf{v}_j||^2}{2\sigma_i^2}\right), \tag{8}$$

where the width $\sigma_i = \gamma \min_{j \neq i} ||\mathbf{v}_i - \mathbf{v}_j||$ is a proportion γ of the minimum Euclidean distance between v_i and the other vertices in $V^{(t)}$. The hyperparameter γ regulates the influence of neighboring instances on the pseudolabels u_i.

We use this scheme to leverage the centers $C^{(t)}$ – which were trained to contain information about the current manifold – and the available instances in the current batch $B^{(t)}$ – which also stores knowledge about the current manifold. By using both sets of points, OSNN can learn a more accurate representation of the current data distribution.

Since the classifier outputs posterior class probabilities, we use logistic sigmoid $f_i = 1/(1 + \exp(-z_i))$ as the activation function for the output node. The activation and loss functions of OSNN are a natural pair [9]. We optimize the smoothness of the output function f over a graph by evaluating the classifier f at each vertex $\mathbf{v} \in V^{(t)}$ and calculating the pseudolabels $\mathbf{u}$ using Equation 1.

To obtain the corresponding IRLS algorithm for OSNN, we define the Newton-Raphson update, in which the weight vector $\mathbf{w}$ is trained according to

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \Delta\mathbf{w}_i, \tag{9}$$

where

$$\Delta\mathbf{w}_i = \eta\mathbf{H}^{-1}\nabla_{\mathbf{w}}\mathcal{L},$$

and η is a learning rate trained with a backtracking line search [32] to obtain better convergence and to disregard η as another hyperparameter to be tuned. We used the Cholesky decomposition for matrix inversion [33] and employed l_2 regularization [34], [9] and manifold regularization [13] to avoid problems with oscillations and slow convergence when the gradient is small. Since the loss is calculated based on the most recent minibatch $B^{(t)}$, the learning rate will be automatically adjusted to deal with concept drift based on the minibatch. The smaller the minibatch, the faster OSNN will adapt to drifts and more sensitive it will become to noise. The bigger the minibatch, the better OSNN will learn the current concept and the more robust it will become to noise. This adaptive learning rate helps to deal with different types of concept drift, as it increases when larger steps for quicker adaptation are necessary (e.g. abrupt concept drifts) or decreases when the rate of the drift is slower. This mechanism allows the forgetting rate to be automatically adjusted according to the speed of the drift. The stopping criterion $0 < \epsilon \ll 1$ for the line search can be fixed in hyperparameter tuning. The gradient $\nabla_{\mathbf{w}}\mathcal{L}$ is

$$\nabla_{\mathbf{w}}\mathcal{L} = \frac{1}{L}\sum_{i\in B_l^{(t)}}(f_i - y_i)\boldsymbol{\phi}_i + \frac{\lambda}{U}\sum_{i\in B_u^{(t)}}(f_i - u_i)\boldsymbol{\phi}_i + \frac{\alpha}{N}\mathbf{w},$$

and the Hessian matrix $\mathbf{H} = \nabla\nabla_{\mathbf{w}}\mathcal{L}$ is defined as

$$\mathbf{H} = \frac{1}{L}\sum_{i\in B_l^{(t)}} f_i(1-f_i)\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top + \frac{1}{U}\sum_{i\in B_u^{(t)}} f_i(1-f_i)\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top + \frac{\alpha}{N}$$

We initialize OSNN by assigning the first H instances as centers and small normally distributed random values drawn from $\mathcal{N}(0, 0.01)$ as weights. OSNN can use $N = 1$, as its weights are not reset and only the centers corresponding to the regions of influence that contain the new minibatch are updated. Hence, our approach becomes one-pass learning if required, e.g. when the rate of incoming instances is very high. Moreover, the adaptive learning rate η also helps our approach to be more robust to different choices of N.

OSNN is summarized in Algorithm 1. Each iteration receives a new data point and the chunk of data $B^{(t)}$ is updated based on it (lines 3-5). If the amount of data received so far is less than the number of hidden nodes H (line 6), then the new point is added to the codebook $C^{(t)}$ (line 7) and a new network weight is randomly added (line 8). When OSNN receives enough data to complete the codebook with size H, it starts training the centers with SLVQ in line 10. Then, the widths of the RBF kernel with centers $C^{(t)}$ are updated in line 12. OSNN now proceeds to the training of the RBFN weights (lines 13-23). It uses a backtracking line search to adjust the learning rate η and train the weights (lines 13-14). It calculates the pseudolabels for the unlabeled data (line 15) and then computes the new weights of the output layer of the neural network. Following the line search, line 17 checks whether the new weights with the current η represent a significant decrease in our loss function. If so, the weights are updated with that specific η; otherwise η is halved until a tolerance ϵ is met. This algorithm yields a trained OSNN model at time step t and is capable of predicting the next instance s^{t+1}.

Algorithm 1 OSNN algorithm

Inputs: Stream $\mathcal{D}$, chunk size N, number of hidden neurons H, amount of manifold regularization λ, amount of l_2 regularization α, manifold RBF width β, RBFN width γ

Outputs: Trained OSNN and predictions on $\mathcal{D}$

```
 1: $t \leftarrow 1$, $C^{(0)} \leftarrow \varnothing$
 2: while $t <$ end of stream do
 3:     $B^{(t)} \leftarrow s^{(t-N+1)}, \dots, s^{(t)} \in \mathcal{D}$
 4:     $B_l^{(t)} \leftarrow \{s \mid s \in B^{(t)} \wedge s$ is labeled$\}$,
 5:     $B_u^{(t)} \leftarrow \{s \mid s \in B^{(t)} \wedge s$ is unlabeled$\}$
 6:     if $t < H$ then
 7:         $C^{(t)} \leftarrow C^{(t-1)} \cup s^{(t)}$
 8:         $\mathbf{w} \leftarrow [w_1, \dots, w_t, \mathcal{N}(0, 0.01)]^\top$
 9:     else
10:         Train centers $C^{(t)}$ as in Equation 6 using $B^{(t)}$
11:     end if
12:     Update widths $\sigma$ as in Equation 7 using $\beta$, $\lambda$ and $C^{(t)}$
13:     $\eta \leftarrow 1$
14:     while $\eta > \epsilon$ do
15:         Calculate pseudolabels as in Eq. 1 with $\alpha$ and $\gamma$
16:         $\mathbf{w}_{\text{new}} \leftarrow$ New weights as in Equation 9
17:         if $\mathcal{L}(B^{(t)}, \mathbf{w}_{\text{new}}) < \mathcal{L}(B^{(t)}, \mathbf{w})$ then
18:             $\mathbf{w} \leftarrow \mathbf{w}_{\text{new}}$
19:             Exit While
20:         else
21:             $\eta \leftarrow \frac{\eta}{2}$
22:         end if
23:     end while
24:     Predict $s^{(t+1)}$
25:     Yield trained OSNN (i.e., $C^{(t)}$, $\sigma$, $\mathbf{w}$) and the prediction $s^{(t+1)}$ as outputs and then continue to the next iteration of the loop with $t \leftarrow t + 1$
26: end while
```

D. Time complexity

SLVQ consists of adjusting centers and widths according to equations 6 and 7, respectively, which are calculated across all pairs of centers and instances in the current minibatch. Hence, SLVQ is performed in $\mathcal{O}(DHN)$. If implemented with the Newton-Raphson method, the weight update evaluates the pseudolabels, $\nabla_{\mathbf{w}}f$, $\mathbf{H}$, $\mathbf{H}^{-1}$ and performs a line search. Such a procedure is in $\mathcal{O}(HN(D + H) + D(H^2 + N^2) + H^3)$. Hence, OSNN is in $\mathcal{O}(HN(D + H) + D(H^2 + N^2) + H^3)$. Alternatively, the weight training can be implemented with a first-order method, e.g. stochastic gradient descent. The time complexity can be reduced to $\mathcal{O}(DHN + D(H^2 + N^2))$. However, first-order optimizers do not have the same convergence properties of second-order methods as they do not exploit curvature information in the update process. In the presence of abrupt concept drifts, smaller H and N are intuitively more effective in forgetting the old concept, while the fast convergence of Newton-Raphson technique is useful for a reliable adaptation to the new concept with small H and N. In contrast, first-order methods converge generally slower, which causes a slower adaptation to the new concept. These differences might cause a lower predictive performance for first-order optimizers when all the stream data is considered, despite the lower time complexity. In fact, in our preliminary experiments, OSNN with Newton-Raphson required smaller H and N when compared to gradient descent. With this trade-off, the time complexity of inverting $\mathbf{H}$ is mitigated.

It is difficult to assess the time complexity of state-of-the-art online classifier ensembles as it heavily depends on the choice of base learners and on drift detection mechanisms (if present). For example, let t_{base} be the computational time of a base learner and H_{ens} be the number of base learners, then OzaBag and DDD take $\mathcal{O}(H_{\text{ens}} t_{\text{base}})$. Nevertheless, the time complexities of such base learners are dependent on the minibatch size. The main difference in computational time of these base learners to OSNN is the choice of the model complexity H. Our experiments show that, with a relatively small H, OSNN can produce comparable generalization to classifier ensembles.

V. RQ1 – Validating the proposed OSNN

Section IV answers RQ1 by proposing the OSNN approach. The current section complements the answer to RQ1 by checking whether OSNN behaves as expected when the manifold assumption does or does not hold. In particular, OSNN should be able to benefit from unlabeled data (which would be discarded by supervised learning algorithms) to improve predictive performance when the manifold assumption holds. However, when this assumption does not hold, the algorithm is expected to fail to improve predictive performance due to the learning of irrelevant structures. To examine the influence of the presence of manifold structures on OSNN, we assess OSNN's predictive performance in comparison with its supervised counterpart, which does not use the unlabeled portions of the data streams for training.

A. Data streams

We used two datasets from [4, Ch.21] as data streams: g241d and Digit1. Both datasets have 1500 instances, 241 dimensions and two classes. G241d possesses a potentially misleading cluster structure and no manifold structure. Semisupervised classifiers are expected to have lower generalization performance in such a dataset as the fundamental SSC assumption – marginal data distribution helps to elucidate posterior class probabilities – does not hold, which tend to degenerate decision boundaries. In contrast, the data in Digit1 contains images of the digit "1" that lie close to a five-dimensional manifold embedded in a high-dimensional space and does not show a meaningful cluster structure. Manifold-based methods are expected to improve generalization on supervised learning.

TABLE I: Mean and standard deviation of the prequential accuracy, and p-value of the comparison between OSNN and its supervised counterpart.

Stream/Classifier	OSNN (Supervised)	OSNN	p-value
g241d	54.74±3.12	51.26±3.36	3.56e-233
Digit1	54.95±3.82	59.44±4.34	1.38e-165

Highlighted values denote significant superior performance.

B. Experimental setup

The supervised version of our approach is trained exactly as OSNN is, except that Equation 6 is calculated without the terms for unlabeled data ($B_u = \varnothing$) and $\lambda = 0$ for the weight training. We followed the experimental setting in [4, Ch.21], except we used the first set of 10 labeled points provided in [4, Ch.21] for each dataset to form the validation streams for hyperparameter tuning and the 12th set of 10 labeled points to compose the test stream. We employed the Wilcoxon signed-rank test with 5% of significance level to compare the approaches. The hyperparameter tuning was performed via randomized search on the validation streams [35]. For both versions of OSNN, N and H were uniformly selected from $[1, 1000]$. The trade-off values, λ and α, were uniformly drawn from $[0, 1]$. We set $\lambda = 0$ and $B_u^{(t)} = \varnothing$ for the supervised approach. The width proportions β and γ were randomly drawn from exponential distributions with mean $\mu = 1$ and $\mu = 2$, respectively. And ϵ was fixed at 2^{-8}.[1]

To replicate real-world learning scenarios in this and in the next experimental setups, classifiers have access to a data stream only once and are trained and evaluated on that stream at that single opportunity. The training and evaluation of each method are performed prequentialy [36], that is, classifiers are tested and then trained with each incoming instance. In supervised methods, unlabeled instances are not used for training. We used a prequential accuracy estimator with a fading factor of 0.999 [36]. We assess the average prequential accuracy over the entire length of each stream.

C. Results of the experiments

Table I shows the mean and standard deviation of each method on each stream. As expected, the supervised version of our approach presented significant superior performance for g241d, which indicates that the manifold regularization is in fact degrading the decision boundary produced by OSNN. It is learning a structure that has no relation to the actual target function. Moreover, the training of the centers might have been misled and produced irrelevant basis functions in suboptimal regions. On the other hand, OSNN delivered statistically superior generalization accuracy for Digit1. Such a fact suggests that the lack of labels (only 10 labeled points) was mitigated by the learning of the digit "1" as a manifold, which is crucial to the learning of the correct decision boundary. This result indicates that SLVQ can effectively train centers to represent the underlying structure of the image; and the weight training can exploit such a manifold to regularize and improve its decision boundary. Therefore, the use of unlabeled data in

[1] An analysis on OSNN's sensitivity to N and H is in the Supplementary Material due to space restrictions.

the learning of informative manifolds was beneficial to the predictive performance of OSNN.

> We answered RQ1 by proposing OSNN, a technique that can exploit unlabeled data to improve generalization on data streams. We complemented this answer by showing that OSNN indeed outperforms its supervised counterpart when the manifold assumption holds.

VI. RQ2 – The impact of the labeling distribution on OSNN

In this section, we answer RQ2 by studying the influence of the distribution of labels on OSNN. We compare OSNN with its supervised counterpart over artificial and real-world streams with uniform and nonuniform labeling distributions.

A. Data streams

We employed four different data stream generators to simulate streams with different types of concept drift to assess the classifiers' ability in adapting to new concepts with various amounts of labeled data. The Sine generator [37] produces various forms of the sine function. The Agrawal generator [38] contains categorical and ordinal attributes to predict groupings of individuals. The SEA generator [39] has only numerical attributes and its decision boundary is a hyperplane. STAGGER [40] contains categorical attributes – size, color and shape – for the prediction of a simple binary function of a set of objects. Each generator has a set of functions $\{r_i\}$ that produce a sequence of different concepts. Each concept has a length of 4000 time steps. Abrupt and gradual drifts take one and 400 time steps to complete, respectively. To assess the adaptation mechanisms of the classifiers, we produced multiple versions of these streams with different types of concept drift, including abrupt, gradual, recurrent, non-recurrent and different levels of severity. The data streams are summarized in Table II. Section VII of the Supplementary Material also shows the severities. We generated 10 artificial data streams with abrupt and 10 with gradual concept drifts. Both versions follow the settings in Table II. Each version of data stream stemmed into three versions with 5%, 10% and 20% of unlabeled data. We used a total of $2*3*10 = 60$ synthetic data streams in this study. We preprocessed these streams by transforming ordinal attributes into real-valued variables and categorical attributes into binary vectors. All attributes were scaled into $[-1, 1]$ and all streams have two classes.

We also used real-world data streams to evaluate classifiers. The Electricity Pricing Data (Elec) [41] stream contains data from the electricity market. The NOAA stream [42] has many decades of weather data. It contains daily measurements of temperature, pressure, visibility, and wind speed. The classification task is to predict whether or not rained in a particular day. The Power Supply stream [43] contains three years of hourly measurements of electricity supply. The task consists of identifying at which hour of the day a given power supply has been recorded. To perform binary classification, we grouped these 24 classes (hours) into two classes: day and night. The Sensor stream [43] contains temperature, humidity, light and sensor voltage collected from sensors in a research laboratory. The classifier aims to identify the sensor that provided a given measurement. We only used the two largest classes in this stream: sensors 29 and 31. These real-world streams, summarized in Table II, also have three versions with 5%, 10% and 20% of labeled data, totaling 12 real-world streams. In the next sections, we will consider two versions of these 72 streams: with uniform and nonuniform labeling distributions.

TABLE II: Summary of data streams.

Stream	Concept sequences	Number of inst.	Dim.
Artificial data streams			
Sine1	$r_3 \rightarrow r_4 \rightarrow r_3$	12000	4
Sine2	$r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_1$	20000	4
Agrawal1	$r_1 \rightarrow r_3 \rightarrow r_4 \rightarrow r_7 \rightarrow r_{10}$	20000	36
Agrawal2	$r_7 \rightarrow r_4 \rightarrow r_6 \rightarrow r_5 \rightarrow r_2 \rightarrow r_9$	24000	36
Agrawal3	$r_4 \rightarrow r_2 \rightarrow r_1 \rightarrow r_3 \rightarrow r_4$	20000	36
Agrawal4	$r_1 \rightarrow r_3 \rightarrow r_6 \rightarrow r_5 \rightarrow r_4$	20000	36
SEA1	$r_4 \rightarrow r_3 \rightarrow r_1 \rightarrow r_2 \rightarrow r_4$	20000	3
SEA2	$r_4 \rightarrow r_1 \rightarrow r_4 \rightarrow r_3 \rightarrow r_2$	20000	3
STAGGER1	$r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_2$	16000	7
STAGGER2	$r_2 \rightarrow r_3 \rightarrow r_1 \rightarrow r_2$	16000	7
Real-world data streams			
Elec	–	27549	7
NOAA	–	18159	8
Power S.	–	29928	2
Sensor	–	130073	5

To implement uniform labeling distribution in this and in the next experimental setup, we uniformly sample the appropriate amount of instances (5%, 10% or 20% of points) that will have labels. To implement nonuniform labeling distribution, for each stream, we adopt the following procedure: for each concept and for each class, we run k-means with $k = 5$ and reveal the labels of instances in each cluster until the appropriate number of instances is reached. For real-world streams, the entire stream is clustered at once. Hence, the probability of labeling an instance varies depending on $\mathbf{x}$.

B. Experimental setup

Due to the nature of stream learning, we tuned the hyperparameters of each algorithm with additional (validation) streams. For artificial streams, we generated one validation stream for each generator (Sine, Agrawal, SEA and STAGGER). These validation streams have random sequences of concepts, length and central point of each concept drift. For real-world streams, we extracted the first 10% of the instances of each stream to compose the respective validation stream.

The hyperparameter tuning follows Subsection V-B. We employed the Wilcoxon signed-rank test with 5% of significance level to compare both versions of our approach across data streams, following the recommendations in [44].

C. Results with uniform labeling probability

The supervised version of our technique was statistically superior to OSNN (p-value = 5.4e-8). Such a result, along with the results from Section V, indicate that the majority of streams does not possess meaningful manifold structures and this labeling procedure preserves the original data distributions, favoring the supervised learner.[2]

D. *Results with nonuniform labeling probability*

In contrast to the previous result, when streams have nonuniform labeling probability, OSNN could significantly outperform its supervised counterpart (p-value = 1.4e-6). In the scenario where the labeling process is only capable of labeling instances in particular regions of the space, the supervised method may be misguided by the lack of labels in structures that are important to the learning of the correct decision boundary, despite these structures being now formed by unlabeled data. Such a limitation might degrade its generalization. Whereas OSNN could still learn the instances that formed those meaningful structures via unlabeled data and produce better predictive performance than the supervised method.

> RQ2 – OSNN can improve generalization in comparison to its supervised counterpart when labels of meaningful structures in the input space become unavailable to the supervised learner. OSNN is able to induce such structures using unlabeled data and learn better decision boundaries.

VII. RQ3 – Comparison of OSNN and the State-of-the-Art

In this section, we answer RQ3 by comparing OSNN to existing algorithms, analyzing how the amount of labels, type of drift and streams affect generalization.

A. *Experimental setup*

This section uses the same streams as in Subsection VI-A and experimental setup as in Subsection VI-B. To evaluate how OSNN performs compared to other data stream learning approaches, we include six existing online algorithms in our study. Hoeffding Tree with Naive Bayesian Learning (HTNB) is employed as a benchmark for a single learner. It does not have the ability to handle concept drifts. OzaBag is a popular online bagging ensemble classifier that does not possess any mechanism to tackle concept drift. It will clarify OSNN's ability to adapt to new concepts. We investigate OAUE as it is an implicit method as OSNN. OAUE, RCD, DDD and DP are state-of-the-art algorithms that have the extra computational complexity of ensemble learning with drift detection mechanisms. All ensemble classifiers have HTNB as base learner. They are expected to deliver higher prediction accuracy among the compared methods. We aim to investigate how OSNN, a single learner that uses unlabeled data, compares to them.

We used the Scott-Knott multiple comparison procedure to evaluate statistical differences in prequential accuracy. This test was recommended in [45], as it can separate statistically different methods into non-overlapping groups and contains strategies to reduce the number of comparisons that need to be performed to create such groups, thus gaining power.

TABLE III: Statistical ranking of prequential accuracy on streams grouped by factors with uniformly distributed labels.

Groups	HTNB	OZABAG	OAUE	RCD	DDD	DP	OSNN
Grouped by amount of labeled data							
5%	2	2	2	1	1	1	1
10%	2	2	2	1	1	1	1
20%	2	2	2	1	1	1	1
Grouped by type of concept drift							
Abrupt	2	2	2	1	1	1	1
Gradual	2	2	2	1	1	1	1
Real-world	1	1	1	1	1	1	1
Grouped by streams							
Sine	3	3	3	1	1	1	2
Agrawal	2	2	2	2	2	2	1
SEA	1	1	2	1	1	1	1
STAG.	4	4	4	3	1	1	2
Elec	1	1	1	1	1	1	1
NOAA	2	2	2	2	1	2	1
Power S.	1	1	2	1	2	1	1
Sensor	2	1	1	2	1	1	1
All streams	2	2	2	1	1	1	1

Highlighted ranks denote significant superior performance.

Explicit approaches were optimized with DDM, EDDM and ADWIN change detectors [21]. The batch and period-related hyperparameters were tuned in $[1, 1000]$; ensemble sizes were optimized in $[1, 20]$ [21]. The p-value in RCD was optimized in $[0.01, 0.05]$ [23]. The diversity measure in DP was either Entropy or Q-Statistics [21]. The diversity level of DDD was optimized in $[0.0005, 0.5]$; p_l and p_h were tuned in $[5{*}10^{-4}, 5{*}10^{-1}]$ [20]. For OSNN, hyperparameter tuning was based on randomized search [35] as in Subsection V-B.

Different data stream learning approaches tend to be more or less adequate for different types of concept drift. In addition, different types of models (e.g., RBFNs, decision trees) tend to perform best for different problems. Therefore, to provide a more detailed comparison of the data stream learning approaches under uniform and nonuniform labeling distributions, we group their 72 streams according to type of drift, amount of labels and data stream in this analysis.

B. *Results with uniform labeling probability*

The Scott-Knott test was performed for each group and their rankings are shown in Table III. Best-performing methods are successively assigned ranks $1, 2, \ldots, 7$. The first grouping in Table III shows that OSNN consistently outperformed the single learner HTNB and two classifier ensembles: OzaBag and OAUE, both are implicit classifiers as OSNN. Although more labeled data was revealed, these approaches could not use the extra information to compensate for the learning mechanisms employed in OSNN. Our approach delivered statistically similar performance to RCD, DDD and DP, which are explicit ensemble algorithms that depend on mechanisms with extra complexity to detect concept drifts. Even with 20% of labels, their performance remained similar to OSNN's. This fact indicates that OSNN could, in general, exploit unlabeled data to balance the lack of label information across most streams.

We now analyze these methods according to the type of concept drift: abrupt or gradual. Drifts in real-world streams are unknown. OSNN could handle abrupt and gradual concept drifts as effectively as explicit approaches (RCD, DDD and

[2] Means and standard deviations of prequential accuracy from the analyses in Subsections VI-C, VI-D, VII-B and VII-C are in the Supplementary Material due to space restrictions.

DP), despite their additional burden of explicitly evaluating potential drifts. OSNN could also outperform HTNB, OzaBag and OAUE. Such a result shows that SLVQ is able to adapt the basis function centers to new concepts, regardless of their speed, while the weight training can produce relatively up-to-date decision boundaries. In real-world streams, all methods delivered similar performance. This fact might be due to the selected instances to have labels being sufficient to reconstruct the necessary structures for the learning of good decision boundaries. Nevertheless, OSNN had similar generalization to state-of-the-art methods.

Table III also shows the ranks grouped by stream generators (for artificial streams) and real-world streams. Sine and STAGGER are relatively simple functions that might not present meaningful manifold structures. In this context, RCD, DDD and DP could find better decision boundaries, whereas the use of unlabeled data might have been harmful for OSNN. Nevertheless, OSNN could significantly outperform HTNB, OzaBag and OAUE. Moreover, HTNB might be a better base learner than RBFN for these streams, so as long as there is a mechanism at ensemble level to deal with concept drift, ensembles with HTNB as base learner might generalize better than RBFN. All algorithms had similar performance with the SEA generator streams. In contrast, OSNN was significantly superior to all other algorithms on Agrawal streams. Such a result might denote the presence of relevant manifolds that OSNN could find and use to learn improved decision boundaries. Explicit methods generally produce good generalization with Elec [20]. However, the lack of labels might have incurred a worse performance in those algorithms and all classifiers obtained similar performance. OSNN could outperform HTNB, OzaBag, OAUE, RCD and DP for the NOAA stream. Such a result might indicate the presence of informative manifolds embedded in weather data. In Sensor streams, OSNN was able to outperform HTNB and RCD and to produce similar generalization to the other ensembles. The overall performance across all streams was also assessed. OSNN regularly outperformed HTNB, OzaBag, OAUE and produced similar generalization to RCD, DDD and DP. Independent of the factors of our analysis, OSNN is consistently among the highest ranked approaches. OSNN's ability to adapt and to exploit unlabeled data could compensate for the use of ensembles in existing methods when very few labels are available.

In Figure 1, we plot the prequential accuracy for Agrawal and Power Supply with uniform labeling distribution. We show the accuracy of the highest and lowest scoring ensemble, and of HTNB and OSNN (single learners). Figure 1a shows that, for Agrawal1 with abrupt drifts, OSNN maintained relatively stable performance in the presence of concept drifts, and was able to learn and adapt more quickly than other methods in each drift. These figures show that OSNN is able to recover from each sudden drifts more rapidly than the other classifiers for the Agrawal1 data stream.

In terms of gradual drifts, in Figures 1b, OSNN also outperformed state-of-the-art algorithms in the majority of concepts. In fact, from the third concept, the performance of OSNN increases drastically, which could indicate that the incoming concepts have more informative manifolds. The other methods do not present such a gain in accuracy.

TABLE IV: Statistical ranking of prequential accuracy on streams with nonuniformly distributed labels.

Groups	HTNB	OZABAG	OAUE	RCD	DDD	DP	OSNN
Grouped by amount of labels							
5%	2	2	2	2	**1**	2	**1**
10%	**1**	**1**	**1**	**1**	**1**	**1**	**1**
20%	2	2	2	2	**1**	2	**1**
Grouped by type of concept drift							
Abrupt	2	2	2	2	**1**	2	**1**
Gradual	2	2	2	2	**1**	2	**1**
Real-world	**1**	**1**	**1**	**1**	2	2	**1**
Grouped by stream							
Sine	3	3	3	3	2	3	**1**
Agrawal	2	2	2	2	2	2	**1**
SEA	**1**	**1**	2	**1**	**1**	2	**1**
STAGGER	2	**1**	2	2	**1**	**1**	**1**
Elec	**1**	**1**	**1**	**1**	**1**	**1**	**1**
NOAA	2	2	2	2	2	**1**	**1**
Power S.	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Sensor	**1**	**1**	2	2	3	3	2
All streams	2	2	2	2	2	2	**1**

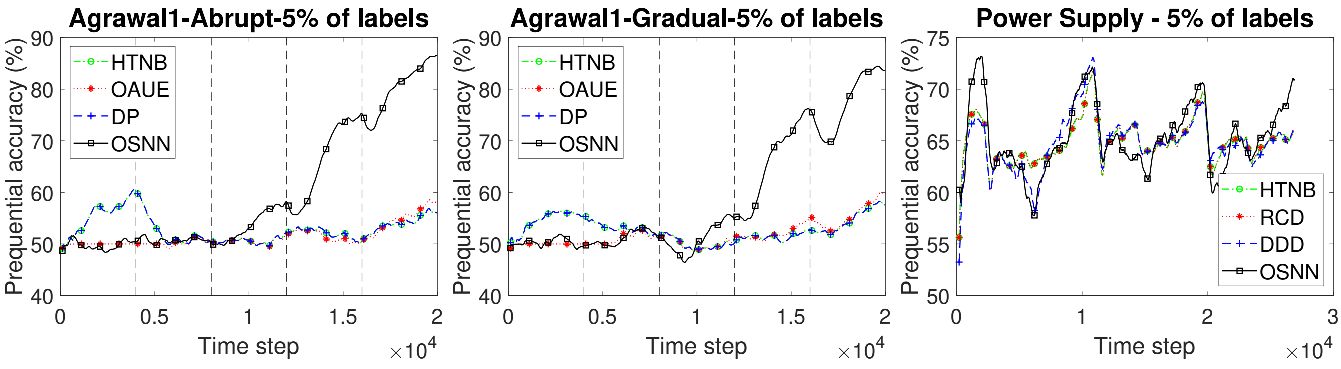
Highlighted ranks denote significant superiority.

For Power Supply (Figure 1c), OSNN could produce higher accuracy than state-of-the-art methods for most of the length of the streams, independent of the amount of labeled data. The results in Figure 1 indicate that OSNN is more robust to sudden drifts than state-of-the-art methods. The SLVQ algorithm was able to naturally and rapidly update the manifold representation (basis functions) of the current data and the MR technique for weight update could also adapt the weights quickly and effectively, whereas OAUE could not handle the lack of label information and the other ensemble methods in this study depend on drift detection mechanisms to start the model adaptation.

C. *Results with nonuniform labeling probability*

Table IV shows the Scott-Knott rankings for streams with nonuniform labels grouped according to the analysed factors – amount of labels, type of concept drift and streams. When labels are severely scarce (5% of labels is available), the true target function becomes more difficult to be learned as only very few regions of the input space have labels. OSNN outperforms HTNB, OzaBag, OAUE, RCD and DP in such a scenario, which indicates that it could successfully exploit unlabeled data to identify relevant structures despite their hidden labels. When the amount of labels was doubled, more informative labels were revealed and all methods delivered similar performance. DDD is a stacked ensemble that uses diversity to select the optimal subensemble to be trained and evaluated to handle concept drifts. Nevertheless, with 20% of labels, OSNN obtained similar performance to DDD and superior generalization to the other methods. Such a result indicates that SLVQ and the manifold regularization in OSNN could effectively find important structures to improve its decision boundaries.

For groups by type of drift, OSNN delivered greater accuracy than HTNB, OzaBag, OAUE, RCD and DP, and similar performance to DDD for both abrupt and gradual drifts. However, for real-world streams, DP and DDD were



(a) (b) (c)

Fig. 1: Prequential accuracy on data streams with uniform labeling distribution. Dashed vertical lines denote time steps where known abrupt (gradual) concept drifts happen (are centered). Plots for other label percentages are in the Supplement.

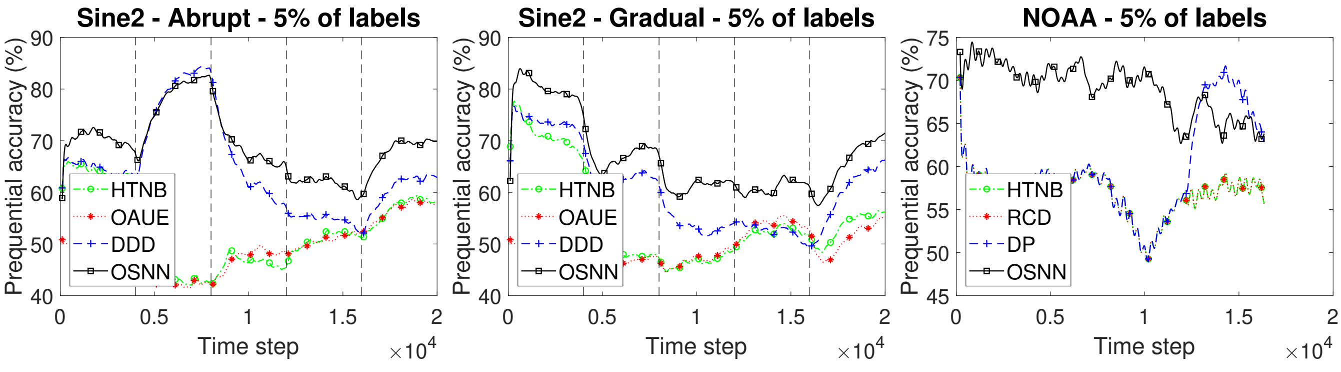
outperformed by the other methods, whereas OSNN continued to be the highest ranked approach. In this sense, SLVQ could adapt OSNN's basis functions to every drift by providing informative up-to-date vector quantization to the weight training without generating irrelevant basis functions. This adaptation demonstrates OSNN's robustness in tackling drifts of different types, including unknown drifts in real-world streams.

For stream groups, Table IV presents somewhat contrasting results to Table III. OSNN was the highest ranked approach for all artificial streams. In fact, OSNN was superior to all other methods for Sine and Agrawal streams and was not inferior to any of the other techniques. When label arrival depends on the region of input space instead of time, the advantages of the data representation and regularization mechanisms in OSNN over single and ensemble learners become more evident. For Elec and Power Supply, all methods had similar performance. OSNN and DP were the superior techniques for NOAA. However, for Sensor, OSNN and DP degraded to rank 2 and 3, respectively. Decision trees seem to be more adequate to Sensor with nonuniform labeling distributions. Nevertheless, OSNN was the approach with highest performance in the vast majority of cases. In fact, when we grouped all streams, OSNN produced superior generalization compared to all other algorithms. Such a result demonstrates that OSNN is the most robust classifier to scarce labels, different types of concept drift and diverse data from different environments, with uniformly and nonuniformly distributed labels.

According to Figure 2a, OSNN is the least affected by abrupt drifts, whilst being the technique with highest accuracy for most new concepts. For gradual drifts – Figure 2b – OSNN delivered the highest prequential accuracy, while being the least affected by the drifts. Its recovery was the fastest among these methods. OAUE is an implicit ensemble that was affected by the lack of labels. Even though DDD has a drift detection mechanism, its performance was more degraded during each drift than OSNN's. For real-world stream NOAA (Figure 2c), OSNN consistently produced superior generalization than HTNB, RCD and DP across all amounts of labeled data. OSNN's ability to recover from drifts shows the robustness of the SLVQ and MR weight update methods. According to Figure 2a, OSNN is the least affected by abrupt drifts, whilst is the technique with highest accuracy for most new concepts. For gradual drifts – Figure 2b – OSNN delivered the highest prequential accuracy, while being the least affected by the drifts. Its recovery was the fastest among these methods. OAUE is an implicit ensemble that was affected by the lack of labels. Although DDD uses a drift detection mechanism, its performance decayed after each drift more severely than OSNN's. For real-world stream NOAA (Figure 2c), OSNN consistently produced superior generalization than HTNB, RCD and DP across all amounts of labeled data. OSNN's ability to recover from drifts shows the robustness of the SLVQ and MR weight update methods.

In Figure 3a, we show the training of C and η throughout the Sine1 stream with abrupt concept drifts as an example. We plot η and the function $\Delta C = \sum_i ||\mathbf{c}_i^{(t)} - \mathbf{c}_i^{(t-1)}||$ as a measure of the adaptation of C at each time step. OSNN automatically adjusts η so that it takes larger steps when a concept drift occurs and smaller steps within a stable concept. From the figure we can see that OSNN was able to increase η following the drifts in time steps 4000 and 8000 for greater exploration, plasticity and faster adaptation, while encouraging more stability (smaller η) for more stable input data. SLVQ also helps OSNN adapting to changing concepts. In particular, note that the peaks in codebook changes in Figure 3a coincide with the peaks of η. A similar behavior is observed for the gradual drifts in Figure 3b, though in this case the η varied more smoothly and had a lower peak, and the changes in the codebook over time were smaller, reflecting the fact that the drift was gradual. This demonstrates OSNN's success in automatically adjusting the learning rate and adapting the codebook to both abrupt and gradual concept drifts, enabling quick recovery and proper learning of the incoming concept.

RQ3 – Besides being able to adapt to new concepts, OSNN could make use of both labeled and unlabeled data to maintain or improve generalization over state-of-the-art single and ensemble methods for the majority of incoming concepts. OSNN was particularly effective with nonuniform labeling distribution due to its ability to compensate for the lack of labels in important regions by using unlabeled data to learn meaningful structures in such regions.

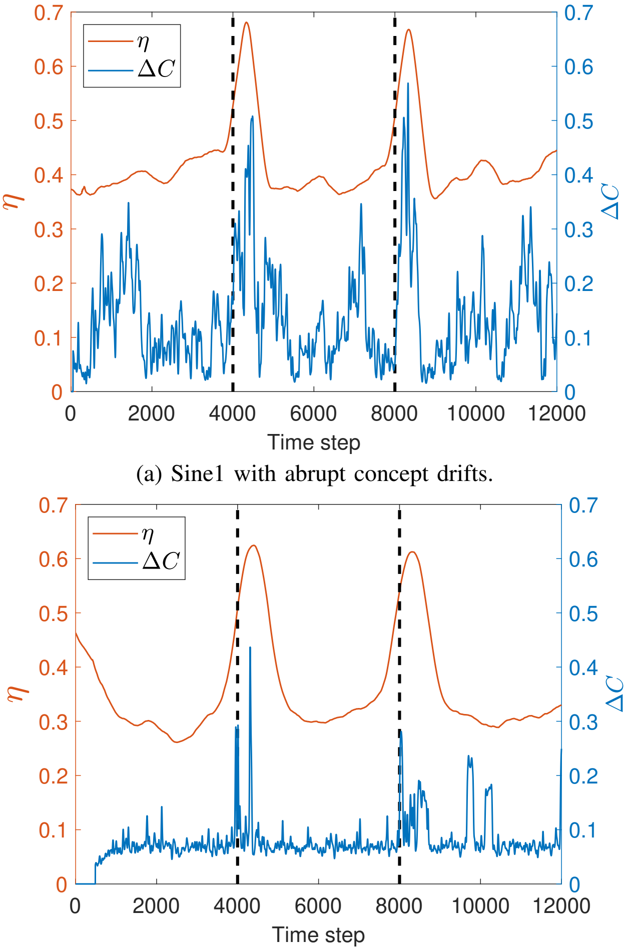


(a)

(b)

(c)

Fig. 2: Prequential accuracy on data streams with nonuniform labeling distribution. Dashed vertical lines denote time steps where known abrupt (gradual) concept drifts happen (are centred). Plots for other label percentages are in the Supplement.



(a) Sine1 with abrupt concept drifts.

(b) Sine1 with gradual concept drifts.

Fig. 3: Adaptive C and η for Sine1 with uniform labeling distribution. Vertical dashed lines mark the center of the drifts.

VIII. Conclusions

In this work, we studied data stream learning in nonstationary environments in which instances may arrive with or without labels and how to exploit the incoming unlabeled data to improve classification accuracy in the presence of concept drifts. We proposed OSNN, a RBF neural network that is able to continuously adapt to different concepts and to use unlabeled data to learn underlying manifolds in data to improve generalization when labels are scarce. The basis function training in OSNN is performed by SLVQ, a novel training algorithm that adjusts basis function centers in order to represent structures of the current concept of a stream based on labeled and unlabeled data. We also proposed a novel training algorithm for the RBF weights, which uses pseudolabels to estimate targets that optimize the smoothness of OSNN output over a manifold identified by basis functions training. Such estimates were used in a weight regularization technique to learn decision boundaries improved by the manifold assumption.

We performed a comprehensive empirical study to provide evidences of OSNN's effectiveness with both synthetic and real-world streams. Such a study investigated diverse learning scenarios with multiple streams, different types of concept drift and various amounts of labeled data. OSNN was compared to state-of-the-art stream learning algorithms and delivered consistently top predictive performance in the majority of scenarios, especially when labels were nonuniformly distributed.

The performance of SSC algorithms depends on the validity of their assumptions. Further investigations on the interactions of SSC assumptions in OSSC remain important in clarifying how the cluster assumption could be used in this context and how to use and balance the demand for multiple assumptions that might emerge as new concepts. We also intend to study OSNN-based ensembles and how different OSNN configurations can be automatically selected for each new concept. A new approach to self-tune hyperparameters such as H could also be proposed.

References

[1] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE CIM*, vol. 10, no. 4, pp. 12–25, 2015.

[2] R. G. F. Soares, "Effort estimation via text classification and autoencoders," in *IJCNN*, pp. 01–08, 2018.

[3] E. Kocaguneli, B. Cukic, T. Menzies, and H. Lu, "Building a second opinion: learning cross-company data," in *Proc. of PROMISE*, pp. 12.1–12.10, 2013.

[4] O. Chapelle, B. Schölkopf, and A. Zien, eds., *Semi-Supervised Learning*. MIT Press, 2006.

[5] T. Wagner, S. Guha, S. Kasiviswanathan, and N. Mishra, "Semi-supervised learning on data streams via temporal label propagation," in *ICML*, vol. 80, pp. 5095–5104, PMLR, July 2018.

[6] Y.-Y. Shen, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Online semi-supervised learning with learning vector quantization," *Neurocomputing*, 2020.

[7] A. Goldberg, X. Zhu, A. Furger, and J.-M. Xu, "Oasis: Online active semi-supervised learning," in *AAAI*, vol. 1, (San Francisco, California, USA), pp. 7–11, August 2011.

[8] F. Shen, H. Yu, K. Sakurai, and O. Hasegawa, "An incremental online semi-supervised active learning algorithm based on self-organizing incremental neural network," *Neural Comput. Appl.*, vol. 20, no. 7, pp. 1433–3058, 2011.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] R. G. F. Soares, H. Chen, and X. Yao, "Efficient cluster-based boosting for semisupervised classification," *IEEE TNNLS*, vol. 29, no. 11, pp. 5667–5680, 2018.

[11] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, 2014.

[12] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE TNNLS*, vol. 25, no. 1, pp. 12–26, 2014.

[13] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *JMLR*, vol. 7, pp. 2399—2434, dec 2006.

[14] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 34, no. 6, pp. 2284–2292, 2004.

[15] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 16, no. 1, pp. 57–67, 2005.

[16] R. Zhang, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Improved GAP-RBF network for classification problems," *Neurocomputing*, vol. 70, no. 16, pp. 3011–3018, 2007.

[17] A. B. Goldberg, M. Li, and X. Zhu, "Online manifold regularization: A new learning setting and empirical study," in *Mach. Learn. Knowl. Disc. Data.*, (Berlin, Heidelberg), pp. 393–407, 2008.

[18] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. of the 13th AAAI Conf. Artif. Intell.*, pp. 1652—-1658, AAAI Press, 2016.

[19] V. M. A. Souza, D. F. Silva, J. ao Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proc. of the 2015 SIAM Int. Conf. Data Min.*, pp. 873–881, 2015.

[20] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE TKDE*, vol. 24, no. 4, pp. 619–633, 2012.

[21] C. W. Chiu and L. L. Minku, "Diversity-based pool of models for dealing with recurring concepts," in *IJCNN*, pp. 1–8, 2018.

[22] C. W. Chiu and L. L. Minku, "A diversity framework for dealing with multiple types of concept drift based on clustering in the model space," *IEEE TNNLS*, pp. 1–11, 2020.

[23] P. M. Gonçalves Jr and R. S. M. D. Barros, "RCD: A recurring concept drift framework," *Pattern Recognit. Lett.*, vol. 34, pp. 1018—1025, july 2013.

[24] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Inf. Sci.*, vol. 265, pp. 50–67, 2014.

[25] P. Domingos and G. Hulten, "Mining high-speed data streams," in *KDD*, (New York, NY, USA), pp. 71—-80, Association for Computing Machinery, 2000.

[26] M. Pratama, A. Ashfahani, and A. Hady, "Weakly supervised deep learning approach in streaming environments," in *Proc. IEEE Int. Conf. Big Data*, (Los Alamitos, CA, USA), pp. 1195–1202, IEEE Computer Society, dec 2019.

[27] Y. Li, Y. Wang, Q. Liu, C. Bi, X. Jiang, and S. Sun, "Incremental semi-supervised learning on streaming data," *Pattern Recognit.*, vol. 88, pp. 383–396, 2019.

[28] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*, (Cambridge, MA, USA), pp. 321—328, MIT Press, 2003.

[29] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, no. 4, pp. 439–458, 2001.

[30] T. Kohonen, "Improved versions of learning vector quantization," in *IJCNN*, vol. 1, pp. 545–550, 1990.

[31] S. Bermejo and J. Cabestany, "A batch learning vector quantization algorithm for nearest neighbour classification," *Neural Process. Lett.*, vol. 11, pp. 173–184, Jun 2000.

[32] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives.," *Pacific J. Math.*, vol. 16, no. 1, pp. 1–3, 1966.

[33] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," in *Proc. of the Signal Process. - Algorithms Archit. Arrange. Appl. Conf. Proc. SPA*, pp. 70–72, 2013.

[34] I. Nabney, "Efficient training of rbf networks for classification," in *Ninth International Conference on Artificial Neural Networks. ICANN 99.*, vol. 1, pp. 210–215, 1999.

[35] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, vol. 13, pp. 281—-305, 2012.

[36] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, pp. 317—-346, Mar. 2013.

[37] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *SBIA*, (Berlin, Heidelberg), pp. 286–295, Springer, 2004.

[38] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: a performance perspective," *IEEE TKDE*, vol. 5, no. 6, pp. 914–925, 1993.

[39] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *KDD*, (New York, NY, USA), pp. 377—-382, Association for Computing Machinery, 2001.

[40] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Mach. Learn.*, vol. 1, pp. 317—-354, March 1986.

[41] M. Harries, "SPLICE-2 comparative evaluation: Electricity pricing," tech. rep., Uni. of New South Wales, Sch. Comp. Sci. and Eng., 1999.

[42] NOAA, "Fed. climate complex global surface summary of day data - version 7 - usaf datsav3 station n. 725540," 2012. [Online; accessed 2020-02-01].

[43] X. Zhu, "Stream data mining repository," 2010. [Online; accessed 2020-02-01].

[44] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, vol. 7, pp. 1–30, December 2006.

[45] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, "Negative results for software effort estimation," *EMSE*, vol. 22, no. 5, pp. 2658–2683, 2017.

Rodrigo G. F. Soares is a Lecturer at the Federal Rural University of Pernambuco, Recife, Brazil. He received the B.Sc. degree in Computer Engineering from the Federal University of Rio Grande do Norte (UFRN), Brazil, in 2005. And the M.Sc. degree in Computer Science from the Federal University of Pernambuco (UFPE), Brazil, in 2008. He has been the recipient of Brazilian Council for Scientific and Technological Development (CNPq) scholarships in both degrees. He received the Ph.D. degree in Computer Science at the University of Birmingham, UK, in 2014 with a scholarship from the CAPES Foundation, Brazil. His research interests include data stream learning, semi-supervised learning, neural networks, clustering and evolutionary computation.

Leandro L. Minku is an Associate Professor at the University of Birmingham (UK). He received the PhD degree in Computer Science from the University of Birmingham (UK) in 2010. Among other roles, Dr. Minku is Associate Editor-in-Chief for Neurocomputing, and Associate Editor for IEEE Transactions on Neural Networks and Learning Systems, Empirical Software Engineering Journal, and Journal of Systems and Software. Dr. Minku's main research interests are machine learning in non-stationary environments / data stream learning, online class imbalance learning, ensembles of learning machines and computational intelligence for software engineering.