# Supervised learning in the presence of concept drift: a modelling framework

M. Straat[1] · F. Abadi[2] · Z. Kan[1] · C. Göpfert[3] · B. Hammer[3] · M. Biehl[1]

## Abstract

We present a modelling framework for the investigation of supervised learning in non-stationary environments. Specifically, we model two example types of learning systems: prototype-based learning vector quantization (LVQ) for classification and shallow, layered neural networks for regression tasks. We investigate so-called student–teacher scenarios in which the systems are trained from a stream of high-dimensional, labeled data. Properties of the target task are considered to be non-stationary due to drift processes while the training is performed. Different types of concept drift are studied, which affect the density of example inputs only, the target rule itself, or both. By applying methods from statistical physics, we develop a modelling framework for the mathematical analysis of the training dynamics in non-stationary environments. Our results show that standard LVQ algorithms are already suitable for the training in non-stationary environments to a certain extent. However, the application of *weight decay* as an explicit mechanism of forgetting does not improve the performance under the considered drift processes. Furthermore, we investigate gradient-based training of layered neural networks with sigmoidal activation functions and compare with the use of rectified linear units. Our findings show that the sensitivity to concept drift and the effectiveness of weight decay differs significantly between the two types of activation function.

## 1 Introduction

The topic of efficiently learning from example data in the presence of *concept drift* has attracted significant interest in the machine learning community. Terms such as *lifelong learning* or *continual learning* have become popular keywords in this context [55].

Very often, machine learning processes [23] are realized according to a standard setup which distinguishes two main stages: In the first, the so-called *training phase,* parameters of the learning system are adapted in an optimization process which is guided by a given set of example data. In the following *working phase,* the obtained hypothesis, e.g., a classifier or regression system, can be applied to novel data. This workflow relies on the implicit assumption that the training data is indeed representative for the target task in the working phase. Statistical properties of the data and the target itself should not change during or after training.

However, in many practical tasks and relevant real-world scenarios, the assumed separation of training and working phase appears artificial and cannot be justified. Obviously, in most human or other biological learning processes [3], the assumption is unrealistic. Similarly, in many technical contexts, training data is available as a non-stationary stream of observations. In such settings, the separation of training and working phase is meaningless, see [1, 17, 27, 32, 55] for reviews.

✉ M. Biehl
  m.biehl@rug.nl

1  Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands

2  Institute of Engineering and Technology, Computing Science Department, Aksum University, Axum, Tigray, Ethiopia

3  CITEC, Machine Learning Group, Bielefeld University, 33594 Bielefeld, Germany

In the literature, two major types of non-stationary environments have been discussed: The term *virtual drift* refers to situations in which statistical properties of the training data are time-dependent, while the actual target task remains unchanged. Scenarios where the target classification or regression scheme itself changes with time are referred to as *real drift* processes. Frequently, both effects coincide and a clear distinction of the two cases becomes difficult.

The presence of drift requires some form of *forgetting* of dated information while the system is adapted to more recent observations. The design of useful, forgetful training schemes hinges on an adequate theoretical understanding of the relevant phenomena. To this end, the development of a suitable modelling framework is instrumental. An overview of earlier work and more recent developments in the context of non-stationary learning environments can be found in references like [1, 17, 27, 32, 55].

Methods developed in statistical physics can be applied in the mathematical description of the training dynamics to obtain typical *learning curves*. The statistical mechanics of on-line learning has helped to gain insights into the behavior of various learning systems; see, e.g., [5, 19, 43, 53] and references therein. Here, we apply these concepts to study the influence of concept drift and weight decay in two exemplary model situations: prototype-based binary classification and continuous regression with feedforward neural networks. We study standard training algorithms under concept drift and address, both, virtual and real drift processes.

This paper presents extensions of our contribution to the *Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering, and Visualization* (WSOM 2019) [48]. Consequently, parts of the text resemble or have been taken over literally from [14] without explicit notice. This concerns, for instance, parts of the introduction and the description of models and methodology in Sect. 2. Similarly, some of the results have also been presented in [14], which focused on the study of explicitly time-dependent densities in a stream of clustered data for LVQ training.

We complement our conference contribution [14] significantly by studying also the influence of drift on the training of regression type layered neural networks. First results concerning such systems with sigmoidal hidden unit activation function under concept drift have been published in [47], recently. Here, the scope of the analysis is extended to layered networks of rectified linear units (ReLU). We concentrate on the comparison of the latter, very popular activation function and its classical, sigmoidal counterpart with respect to the sensitivity to drift and the effect of weight decay.

We have selected LVQ for classification and layered neural networks for regression as representatives of important paradigms in machine learning. These systems provide a workshop in which to develop modelling techniques and analytical approaches that will facilitate the study of other setups in the future.

In the following section, we introduce the machine learning systems, the model setup including the assumed densities of data, the target rules as well as the mathematical framework of the statistical physics-based analysis. Our results concerning classification and regression systems in the presence of concept drift are presented and discussed in Sect. 3 before we conclude with a summary and outlook on forthcoming investigations.

## 2 Model and methods

In Sect. 2.1, we introduce learning vector quantization for classification tasks with emphasis on the well established LVQ1 training scheme. We also propose a model density of data which was previously investigated in the mathematical analysis of LVQ training in stationary and specific non-stationary environments. Here, we extend the approach to the presence of virtual concept drift and consider *weight decay* as an explicit mechanism of forgetting.

Thereafter, Sect. 2.2 presents a student–teacher scenario for the learning of a regression scheme with shallow, layered neural networks of the feedforward type. Emphasis is on the comparison of two important types of hidden unit activations; traditional sigmoidal transfer functions and the popular rectified linear unit (ReLU) activation. We consider gradient-based training in the presence of real concept drift and also introduce weight decay as a mechanism of forgetting.

A unified description of the theoretical approach to analyse the training dynamics in classification and regression systems is given in Sect. 2.3.

### 2.1 Learning vector quantization

The family of LVQ algorithms is widely used for practical classification problems [13, 29, 30, 39]. The popularity of LVQ is due to a number of attractive features: It is straightforward to implement, very flexible and intuitive. Moreover, it constitutes a natural tool for multi-class problems. The actual classification scheme is very often based on Euclidean metrics or other simple measures, which quantify the distance of inputs or feature vectors from the class-specific prototypes. Unlike many other methods, LVQ facilitates direct interpretation of the classifier because prototypes are defined in the same space as the data [13, 39]. The approach is based on the idea of representing classes by more or less typical representatives of the training instances. This suggests that LVQ

algorithms should also be capable of tracking changes in the density of samples, a hypothesis that has been studied for instance in [14, 25], recently.

### 2.1.1 Nearest prototype classifier

In general, several prototypes can be employed to represent each class. However, we restrict the analysis to the simple case of only one prototype per class in binary classification problems. Hence we consider two prototypes $\mathbf{w}_k \in \mathbb{R}^N$ each representing one of the classes $k \in \{1, 2\}$. Together with a distance measure $d(\mathbf{w}, \boldsymbol{\xi})$, the system parameterizes a Nearest Prototype Classification (NPC) scheme: Any given input $\boldsymbol{\xi} \in \mathbb{R}^N$ is assigned to the class $k = 1$ if $d(\mathbf{w}_1, \boldsymbol{\xi}) < d(\mathbf{w}_2, \boldsymbol{\xi})$ and to class 2, otherwise. In practice, ties can be broken arbitrarily.

A variety of distance measures have been used in LVQ, enhancing the flexibility of the approach even further [13, 39]. This includes the conceptually interesting use of adaptive metrics in *relevance learning,* see [13] and references therein. Here, we restrict our analysis to the simple (squared) Euclidean measure

$$d(\mathbf{w}, \boldsymbol{\xi}) = (\mathbf{w} - \boldsymbol{\xi})^2. \tag{1}$$

We assume that the training procedure provides a stream of single examples [5]: At time step $\mu = 1, 2, \ldots$, the vector $\boldsymbol{\xi}^\mu$ is presented, together with its given class label $\sigma^\mu = 1, 2$. Iterative on-line LVQ updates are of the general form [12, 20, 54]

$$
\begin{aligned}
\mathbf{w}_k^\mu &= \mathbf{w}_k^{\mu-1} + \frac{\eta}{N} \, \Delta \mathbf{w}_k^\mu \text{ with} \\
\Delta \mathbf{w}_k^\mu &= f_k \left[ d_1^\mu, d_2^\mu, \sigma^\mu, \ldots \right] \left( \boldsymbol{\xi}^\mu - \mathbf{w}_k^{\mu-1} \right)
\end{aligned}
\tag{2}
$$

where $d_i^\mu = d(\mathbf{w}_i^{\mu-1}, \boldsymbol{\xi}^\mu)$ and the learning rate $\eta$ is scaled with the input dimension $N$. The precise algorithm is specified by choice of the *modulation function* $f_k[\ldots]$, which depends typically on the Euclidean distances of the data point from the current prototype positions and on the labels $k, \sigma^\mu = 1, 2$ of the prototype and training example, respectively.

### 2.1.2 The LVQ1 training algorithm

A popular and intuitive LVQ training scheme was already suggested by Kohonen and is known as LVQ1 [29, 30]. Following the NPC concept, it updates only the currently closest prototype in a so-called *Winner-Takes-All* (WTA) scheme. Formally, the LVQ1 prescription for a system with two competing prototypes is given by Eq. (2) with

$$f_k[d_1^\mu, d_2^\mu, \sigma^\mu] = \Theta\left( d_{\widehat{k}}^\mu - d_k^\mu \right) \Psi(k, \sigma^\mu), \tag{3}$$

where

$$\widehat{k} = \begin{cases} 2 & \text{if } k = 1 \\ 1 & \text{if } k = 2, \end{cases} \text{ and } \Psi(k, \sigma) = \begin{cases} +1 & \text{if } k = \sigma \\ -1 & \text{else.} \end{cases}$$

Here, the Heaviside function $\Theta(\ldots)$ singles out the winning prototype and the factor $\Psi(k, \sigma^\mu)$ determines the sign of the update: The WTA update according to Eq. (3) moves the prototype towards the presented feature vector if it carries the same class label $k = \sigma^\mu$. On the contrary, if the prototype is meant to present a different class, its distance from the data point is increased even further. Note that LVQ1 cannot be interpreted as a gradient descent procedure of a suitable cost function in a straightforward way due to discontinuities at the class boundaries, see [12] for a discussion and references.

Numerous variants and modifications of LVQ have been presented in the literature, aiming at better convergence or classification performance, see [12, 13, 29, 39]. Most of these modifications, however, retain the basic idea of attraction and repulsion of the winning prototypes.

### 2.1.3 Clustered model data

LVQ algorithms are most suitable for classification schemes which reflect a given cluster structure in the data. In the modelling, we therefore consider a stream of random input vectors $\boldsymbol{\xi} \in \mathbb{R}^N$ which are generated independently according to a mixture of two Gaussians [12, 20, 54]:

$$
\begin{aligned}
P(\boldsymbol{\xi}) &= \sum_{m=1,2} p_m P(\boldsymbol{\xi} \mid m) \text{ with contributions} \\
P(\boldsymbol{\xi} \mid m) &= \frac{1}{(2\pi v_m)^{N/2}} \exp\left[ -\frac{1}{2 v_m} (\boldsymbol{\xi} - \lambda \mathbf{B}_m)^2 \right].
\end{aligned}
\tag{4}
$$

The target classification coincides with the cluster membership, i.e., $\sigma = m$ in Eq. (3). The class-conditional densities $P(\boldsymbol{\xi}|m = 1, 2)$ correspond to isotropic, spherical Gaussians with variance $v_m$ and mean $\lambda \mathbf{B}_m$. Prior weights of the clusters are denoted as $p_m$ and satisfy $p_1 + p_2 = 1$. We assume that the vectors $\mathbf{B}_m$ are orthonormal with $\mathbf{B}_1^2 = \mathbf{B}_2^2 = 1$ and $\mathbf{B}_1 \cdot \mathbf{B}_2 = 0$. Obviously, the classes $m = 1, 2$ are not perfectly separable due to the overlap of the clusters.

We denote conditional averages over $P(\boldsymbol{\xi} \mid m)$ by $\langle \cdots \rangle_m$, whereas mean values $\langle \cdots \rangle = \sum_{m=1,2} p_m \langle \cdots \rangle_m$ are defined with respect to the full density (4). One obtains, for instance, the conditional and full averages

$$
\begin{aligned}
\langle \boldsymbol{\xi} \rangle_m &= \lambda \mathbf{B}_m, \langle \boldsymbol{\xi}^2 \rangle_m = v_m N + \lambda^2 \text{ and} \\
\langle \boldsymbol{\xi}^2 \rangle &= (p_1 v_1 + p_2 v_2) N + \lambda^2.
\end{aligned}
\tag{5}
$$

Note that in the thermodynamic limit $N \to \infty$ considered later, $\lambda^2$ can be neglected in comparison to the terms of $\mathcal{O}(N)$ in Eq. (5).

Similar clustered densities have been studied in the context of unsupervised learning and supervised perceptron training; see, e.g., [4, 10, 35]. Also, online LVQ in stationary situations was analysed in, e.g., [12].

Here we focus on the question whether LVQ learning schemes are able to cope with drift in characteristic model situations and whether extensions like weight decay can improve the performance in such settings.

## 2.2 Layered neural networks

The term Soft Committee Machine (SCM) has been established for shallow feedforward neural networks with a single hidden layer and a linear output unit, see for instance [2, 8, 9, 11, 26, 42, 44, 45, 49]. Its structure resembles that of a *(crisp)* committee machine with binary threshold hidden units, where the network output is given by their *majority vote*, see [4, 19, 53] and references therein.

The output of an SCM with $K$ hidden units and fixed hidden-to-output weights is of the form

$$y(\boldsymbol{\xi}) = \sum_{k=1}^{K} g(\mathbf{w}_k \cdot \boldsymbol{\xi}) \text{ where } \mathbf{w}_k \in \mathbb{R}^N \qquad (6)$$

denotes the weight vector connecting the $N$-dimensional input layer with the $k$-th hidden unit. A non-linear transfer function $g(\cdots)$ defines the hidden unit states and the final output is given as their sum.

As specific examples we consider the sigmoidal

$$g(x) = \text{erf}\left(x/\sqrt{2}\right) \text{ with } g'(x) = \sqrt{2/\pi} \; e^{-x^2/2} \qquad (7)$$

and the popular *rectified linear unit* (ReLU):

$$g(x) = x\,\Theta(x) \text{ with } g'(x) = \Theta(x). \qquad (8)$$

The activation (7) resembles closely other sigmoidal functions, e.g., the more popular $\tanh(x)$, but it facilitates the analytical treatment in the mathematical analysis as exploited in [8], originally. In the following, we refer to an SCM with the above sigmoidal activation as Erf-SCM, for brevity.

Similarly, we use the term ReLU-SCM for networks with hidden unit states given by Eq. (8). The ReLU activation has recently gained significant popularity in the context of Deep Learning [22]. This is, among other reasons, due to its simplicity which offers computational ease and numerical stability. According to the literature, ReLU networks have displayed favorable training and generalization behavior in several practical applications and benchmark problems [18, 31, 34, 38, 40].

Note that an SCM, cf. Eq. (6), is not quite a *universal approximator*. However, this property could be achieved by introducing hidden-to-output weights and adaptive local thresholds $\vartheta_i \in \mathbb{R}$ in hidden unit activations of the form $g(\mathbf{w}_i \cdot \boldsymbol{\xi} - \vartheta_i)$, see [16]. Adaptive hidden-to-output weights have been studied in, for instance, [42] from a statistical physics perspective. However, we restrict ourselves to the simpler model defined above and focus on basic dynamical effects and potential differences of ReLU- versus Erf-SCM in the presence of concept drift.

### 2.2.1 Regression scheme and on-line learning

The training of a neural network with real-valued output $y(\boldsymbol{\xi})$ based on examples $\left\{ \boldsymbol{\xi}^\mu \in \mathbb{R}^N, \tau^\mu \in \mathbb{R} \right\}$ for a regression problem is frequently guided by the quadratic deviation of the network output from the target values [15, 22, 23] . It serves as a cost function which evaluates the network performance with respect to a single example as

$$e^\mu\left(\{\mathbf{w}_k\}_{k=1}^K\right) = \frac{1}{2}\left(y^\mu - \tau^\mu\right)^2 \text{ with } y^\mu = y(\xi^\mu). \qquad (9)$$

In stochastic or on-line gradient descent, updates of the weight vectors are based on the presentation of a single example at *time step* $\mu$

$$\mathbf{w}_k^\mu = \mathbf{w}_k^{\mu-1} + \frac{\eta}{N}\,\Delta\mathbf{w}_k^\mu \text{ with } \Delta\mathbf{w}_k^\mu = -\frac{\partial e^\mu}{\partial \mathbf{w}_k} \qquad (10)$$

where the gradient is evaluated in $\mathbf{w}_k^{\mu-1}$. For the SCM architecture specified in Eq. (6), $\partial y^\mu / \partial \mathbf{w}_k = g'\left(h_k^\mu\right)\xi^\mu$, and we obtain

$$\Delta\mathbf{w}_k^\mu = -\left(\sum_{i=1}^{K} g(h_i^\mu) - \tau^\mu\right) g'\left(h_k^\mu\right)\xi^\mu \qquad (11)$$

with the inner products $h_i^\mu = \mathbf{w}_i^{\mu-1} \cdot \boldsymbol{\xi}^\mu$ of the current weight vectors with the next example input in the stream. Note that the change of weight vectors is proportional to $\boldsymbol{\xi}^\mu$ and can be interpreted as a form of *Hebbian Learning* [15, 22, 23].

### 2.2.2 Student–teacher scenario and model data

In order to define and model meaningful learning situations, we resort to the consideration of student–teacher scenarios [4, 5, 19, 53].

We assume that the target can be defined in terms of an SCM with a number $M$ of hidden units and a specific set of weights $\left\{ \mathbf{B}_m \in \mathbb{R}^N \right\}_{m=1}^M$:

$$\tau(\boldsymbol{\xi}) = \sum_{m=1}^{M} g(\mathbf{B}_m \cdot \boldsymbol{\xi}) \quad \text{and} \quad \tau^\mu = \tau(\boldsymbol{\xi}^\mu) = \sum_{m=1}^{M} g(b_m^\mu) \qquad (12)$$

with $b_m^\mu = \mathbf{B}_m \cdot \boldsymbol{\xi}^\mu$ for one of the training examples. This so-called teacher network can be equipped with $M > K$ hidden units in order to model regression schemes which cannot be learnt by an SCM student of the form (6). On the contrary, $K > M$ would correspond to an *over-learnable* target or *over-sophisticated* student. For the discussion of these highly interesting cases in stationary environments, see for instance [8, 9, 42, 44, 45]. In a student–teacher scenario with $K$ and $M$ hidden units the update of the student weight vectors by on-line gradient descent is given by Eq. (11) with $\tau^\mu$ from Eq. (12).

In the following, we will restrict our analysis to perfectly matching student complexity with $K = M = 2$ only, which further simplifies Eq. (11). Extensions to more hidden units and settings with $K \neq M$ will be considered in forthcoming projects.

In contrast to the model for LVQ-based classification, the vectors $\mathbf{B}_m$ define the target outputs $\tau^\mu = \tau(\boldsymbol{\xi}^\mu)$ explicitly via the teacher network for any input vector. While clustered input densities of the form (4) can also be studied for feedforward networks as in [35, 36], we assume here that the actual input vectors are uncorrelated with the teacher vectors $\mathbf{B}_m$. Consequently, we can resort to a simpler model density and consider vectors $\boldsymbol{\xi}$ of independent, zero mean, unit variance components with

$$P(\boldsymbol{\xi}) = (2\pi)^{-N/2} \exp\left[-\boldsymbol{\xi}^2/2\right]. \qquad (13)$$

Note that the density (13) is recovered formally from Eq. (4) by setting $\lambda = 0$ and $v_1 = v_2 = 1$, for which both clusters in (4) coincide in the origin and the parameters $p_{1,2}$ become irrelevant.

Note that the student/teacher scenario considered here is different from concepts used in studies of *knowledge distillation*, see [51] and references therein. In the context of distillation, a teacher network is itself trained on a given data set to approximate the target function. Thereafter a student network, frequently of a simpler architecture, distills the knowledge in a subsequent training process. In our work, as in most statistical physics-based studies [4, 19, 53], the teacher network is taken to directly define the true target function. A particular architecture is chosen and, together with its fixed weights, it controls the complexity of the task. The teacher network provides correct target outputs to all input data that are generated according to the distribution in Eq. (13). In the actual training process, a sequence of such input vectors and teacher-generated labels is presented to the student network.

## 2.3 Mathematical analysis of the training dynamics

In the following we sketch the successful theory of on-line learning [4, 5, 19, 43, 53] as, for instance, applied to the dynamics of LVQ algorithms in [12, 20, 54] and to on-line gradient descent in SCM in [8, 9, 26, 42, 44, 45, 49]. We refer the reader to the original publications for details. The extensions to non-stationary situations with concept drifts are discussed in Sect. 2.4.

The mathematical analysis proceeds along the same generic steps in both settings. Our presentation follows closely the descriptions in [14, 47].

We consider adaptive vectors $\mathbf{w}_{1,2} \in \mathbb{R}^N$ (prototypes in LVQ, student weights in the SCM) while the characteristic vectors $\mathbf{B}_{1,2}$ specify the target task (cluster centers in LVQ training, SCM teacher vectors for regression).

The consideration of the *thermodynamic limit* $N \to \infty$ is instrumental for the theoretical treatment. The limit facilitates the following key steps which, eventually, yield an exact mathematical description of the training dynamics in terms of ordinary differential equations (ODE):

(a) *Order parameters*

The many degrees of freedom, i.e., the components of the adaptive vectors, can be characterized in terms of only very few quantities. The definition of these so-called *order parameters* follows naturally from the mathematical structure of the model. After presentation of a number $\mu$ of examples, as indicated by corresponding superscripts, we describe the system by the projections for $i, k, m \in \{1, 2\}$

$$R_{im}^\mu = \mathbf{w}_i^\mu \cdot \mathbf{B}_m \quad \text{and} \quad Q_{ik}^\mu = \mathbf{w}_i^\mu \cdot \mathbf{w}_k^\mu. \qquad (14)$$

Obviously, $Q_{11}^\mu, Q_{22}^\mu$ and $Q_{12}^\mu = Q_{21}^\mu$ relate to the norms and mutual overlap of the adaptive vectors, while the quantities $R_{im}$ specify their projections into the linear subspace defined by the characteristic vectors $\{\mathbf{B}_1, \mathbf{B}_2\}$, respectively.

(b) *Recursions*

Recursion relations for the order parameters (14) can be derived directly from the update steps, which are of the generic form $\mathbf{w}_k^\mu = \mathbf{w}_k^{\mu-1} + \eta/N \, \Delta\mathbf{w}_k^\mu$. The corresponding inner products yield

$$N(R_{im}^\mu - R_{im}^{\mu-1}) = \eta \, \Delta\mathbf{w}_i^\mu \cdot \mathbf{B}_m$$
$$N(Q_{ik}^\mu - Q_{ik}^{\mu-1}) = \eta \left( \mathbf{w}_i^{\mu-1} \cdot \Delta\mathbf{w}_k^\mu + \mathbf{w}_k^{\mu-1} \cdot \Delta\mathbf{w}_i^\mu \right) \qquad (15)$$
$$+ \eta^2/N \, \Delta\mathbf{w}_i^\mu \cdot \Delta\mathbf{w}_k^\mu.$$

Terms of order $\mathcal{O}(1/N)$ on the r.h.s. will be neglected in the following. Note however that $\Delta\mathbf{w}_i^\mu \cdot \Delta\mathbf{w}_k^\mu$ comprises contributions of order $|\boldsymbol{\xi}|^2 \propto N$ for the considered updates (2) and (10).

(c) *Averages over the model data*

Applying the central limit theorem (CLT) we can perform an average over the random sequence of independent examples.

Note that $\Delta \mathbf{w}_k^\mu \propto \boldsymbol{\xi}^\mu$ or $\Delta \mathbf{w}_k^\mu \propto \left( \boldsymbol{\xi}^\mu - \mathbf{w}_k^{\mu-1} \right)$ for the SCM and LVQ, respectively. Consequently, the current input $\boldsymbol{\xi}^\mu$ enters the r.h.s. of Eq. (15) only through its norm $| \xi |^2 = \mathcal{O}(N)$ and the quantities

$$h_i^\mu = \mathbf{w}_i^{\mu-1} \cdot \boldsymbol{\xi}^\mu \text{ and } b_m^\mu = \mathbf{B}_m \cdot \boldsymbol{\xi}^\mu. \tag{16}$$

Since these inner products correspond to sums of many independent random quantities in our model, the CLT implies that the projections in Eq. (16) are correlated Gaussian quantities for large $N$ and the joint density $P(h_1^\mu, h_2^\mu, b_1^\mu, b_2^\mu)$ is given completely by first and second moments.

*LVQ:* For the clustered density, cf. Eqs. (4), the conditional moments read

$$\begin{aligned}
\langle h_i^\mu \rangle_m &= \lambda R_{im}^{\mu-1}, \quad \langle b_m^\mu \rangle_n = \lambda \delta_{mn}, \\
\langle h_i^\mu h_k^\mu \rangle_m - \langle h_i^\mu \rangle_m \langle h_k^\mu \rangle_m &= v_m Q_{ik}^{\mu-1}, \\
\langle h_i^\mu b_n^\mu \rangle_m - \langle h_i^\mu \rangle_m \langle b_n^\mu \rangle_m &= v_m R_{in}^{\mu-1}, \\
\langle b_l^\mu b_n^\mu \rangle_m - \langle b_l^\mu \rangle_m \langle b_n^\mu \rangle_m &= v_m \delta_{ln},
\end{aligned} \tag{17}$$

with $i, k, l, m, n \in \{1, 2\}$ and the Kronecker-Delta $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ else.

*SCM:* In the simpler case of the isotropic, spherical density (13) with $\lambda = 0$ and $v_1 = v_2 = 1$ the moments reduce to

$$\begin{aligned}
\langle h_i^\mu \rangle &= 0, \langle b_m^\mu \rangle = 0, \langle h_i^\mu h_k^\mu \rangle - \langle h_i^\mu \rangle \langle h_k^\mu \rangle = Q_{ik}^{\mu-1} \\
\langle h_i^\mu b_n^\mu \rangle &- \langle h_i^\mu \rangle \langle b_n^\mu \rangle = R_{in}^{\mu-1}, \langle b_l^\mu b_n^\mu \rangle - \langle b_l^\mu \rangle \langle b_n^\mu \rangle = \delta_{ln}.
\end{aligned} \tag{18}$$

Hence, in both cases (LVQ and SCM) the four-dim. density of $h_{1,2}^\mu$ and $b_{1,2}^\mu$ is fully specified by the values of the order parameters in the previous time step and the parameters of the model density. This important result enables us to average the recursion relations (15) over the most recent training example by means of Gaussian integrals. The resulting r.h.s. can be expressed as functions of $\{R_{im}^{\mu-1}, Q_{ik}^{\mu-1}\}$. Obviously, the precise form depends on the details of the algorithm and model setup.

**(d) Self-Averaging Properties**

The self-averaging property of the order parameters allows us to describe the dynamics in terms of mean values: Fluctuations of the stochastic dynamics can be neglected in the limit $N \to \infty$. The concept relates to the statistical physics of disordered materials and has been transferred successfully to the study of neural network models and learning processes [4, 19, 53]. A detailed mathematical discussion in the context of sequential online learning dynamics is given in [41]. As a consequence,

we can interpret the averaged equations (15) directly as deterministic recursions for the actual values of $\{R_{im}^\mu, Q_{ik}^\mu\}$, which coincide with their disorder average in the thermodynamic limit.

**(e) *Continuous Time Limit***

In the thermodynamic limit $N \to \infty$, ratios of the form $(\ldots)/(1/N)$ on the left hand sides of Eq. (15) can be interpreted as derivatives with respect to a continuous learning time $\alpha$ defined by

$$\alpha = \mu/N \text{ with } d\alpha \sim 1/N. \tag{19}$$

This scaling corresponds to the natural assumption that the number of examples should be proportional to the number of adaptive quantities in the system.

Averages are performed over the joint density $P(h_1^\mu, h_2^\mu, b_1^\mu, b_2^\mu)$ corresponding to the latest, independently drawn input vector. For simplicity, we omit indices $\mu$ in the following. The resulting sets of coupled ODE is of the form

$$\left[ \frac{dR_{im}}{d\alpha} \right]_{\text{stat}} = \eta F_{im}; \quad \left[ \frac{dQ_{ik}}{d\alpha} \right]_{\text{stat}} = \eta\, G_{ik}^{(1)} + \eta^2 G_{ik}^{(2)}. \tag{20}$$

Here, the subscript stat indicates that the ODE describe learning from a stationary density, Eqs. (4) or (13).

*Limit of small learning rates*

The dynamics can also be studied in the limit of small learning rates $\eta \to 0$. In this case, the term $\eta^2 G_{ik}^{(2)}$ can be neglected in Eq. (20). In order to retain non-trivial performance, the small step size has to be compensated for by training with a large number of examples that diverges like $1/\eta$. Formally, we introduce the quantity $\tilde{\alpha}$ in the simultaneous limit

$$\tilde{\alpha} = \lim_{\eta \to 0} \lim_{\alpha \to \infty} (\eta \alpha), \tag{21}$$

which leads to a simplified system of ODE

$$\left[ \frac{dR_{im}}{d\tilde{\alpha}} \right]_{\text{stat}} = F_{im}; \quad \left[ \frac{dQ_{ik}}{d\tilde{\alpha}} \right]_{\text{stat}} = G_{ik}^{(1)} \tag{22}$$

in rescaled continuous time $\tilde{\alpha}$ for $\eta \to 0$.

*LVQ:* In the classification model we have to insert

$$\begin{aligned}
F_{\text{im}} &= (\langle b_m f_i \rangle - R_{\text{im}} \langle f_i \rangle), \\
G_{ik}^{(1)} &= \left( \langle h_i f_k + h_k f_i \rangle - Q_{ik} \langle f_i + f_k \rangle \right) \\
\text{and } G_{ik}^{(2)} &= \sum_{m=1,2} v_m p_m \langle f_i f_k \rangle_m
\end{aligned} \tag{23}$$

in Eqs. (20) or (22). The LVQ1 modulation functions $f_i$ is given in Eq. (3) and conditional averages $\langle \ldots \rangle_m$ are with respect to the density (4).

*SCM:* In the case of non-linear regression we obtain

$$\begin{aligned}
F_{\text{im}} &= \langle \rho_i b_m \rangle, \quad G_{ik}^{(1)} = \langle (\rho_i h_k + \rho_k h_i) \rangle, \\
&\text{and } G_{ik}^{(2)} = \langle \rho_i \rho_k \rangle \text{ with } \rho_k = -(y - \tau) g'(h_k).
\end{aligned} \tag{24}$$

Eventually, the r.h.s. of Eqs. (20) or (22) are expressed in terms of elementary functions of order parameters. For the straightforward, yet lengthy results we refer the reader to the original literature for LVQ [12, 20] and SCM [9, 42, 44, 45], respectively.

(f) *Generalization error*

After training, the success of learning is quantified in terms of the generalization error $\epsilon_g$, which is also given as a function of the macroscopic order parameters.

*LVQ:* In the case of the LVQ model, $\epsilon_g$ is given as the probability of misclassifying a novel, randomly drawn input vector. The class-specific errors corresponding to data from clusters $k = 1, 2$ in Eq. (4) can be considered separately:

$$\epsilon_g = p_1\,\epsilon_g^1 + p_2\,\epsilon_g^2 \quad \text{where} \quad \epsilon_g^k = \left\langle \Theta\left(d_k - d_{\widehat{k}}\right) \right\rangle_k \qquad (25)$$

is the class-specific misclassification rate, i.e., the probability for an example drawn from a cluster $k$ to be assigned to $\widehat{k} \neq k$ with $d_k > d_{\widehat{k}}$. For the derivation of the class-wise and total generalization error for systems with two prototypes as functions of the order parameters we also refer to [12]. One obtains

$$\epsilon_g^k = \Phi\left(\frac{Q_{kk} - Q_{\widehat{kk}} - 2\lambda(R_{kk} - R_{\widehat{kk}})}{2\sqrt{v_k}\sqrt{Q_{11} - 2Q_{12} + Q_{22}}}\right) \qquad (26)$$

with the function $\Phi(z) = \int_{-\infty}^{z} dx\, e^{-x^2/2}/\sqrt{2\pi}$.

*SCM:* In the regression scenario, the generalization error is defined as an average $\langle \cdots \rangle$ of the quadratic deviation between student and teacher output over the isotropic density, cf. Eq. (13):

$$\epsilon_g = \frac{1}{2}\left\langle \left[\sum_{k=1}^{K} g(h_k) - \sum_{m=1}^{M} g(b_m)\right]^2 \right\rangle. \qquad (27)$$

In the simplifying case of $K = M = 2$ we obtain for Erf-SCM:

$$\epsilon_g = \frac{1}{3} + \frac{1}{\pi}\sum_{i,k=1}^{2}\sin^{-1}\left(\frac{Q_{ik}}{\sqrt{1 + Q_{ii}}\sqrt{1 + Q_{kk}}}\right) \\ - \frac{2}{\pi}\sum_{i,m=1}^{2}\sin^{-1}\left(\frac{R_{im}}{\sqrt{2}\sqrt{1 + Q_{ii}}}\right) \qquad (28)$$

and for ReLU-SCM:

$$\epsilon_g = \sum_{i,j=1}^{2}\left[\frac{Q_{ij}}{8} + \frac{\sqrt{Q_{ii}Q_{jj} - Q_{ij}^2} + Q_{ij}\sin^{-1}\left(\frac{Q_{ij}}{\sqrt{Q_{ii}Q_{jj}}}\right)}{4\pi}\right] \\ - \sum_{i,j=1}^{2}\left[\frac{R_{ij}}{4} + \frac{\sqrt{Q_{ii} - R_{ij}^2} + R_{ij}\sin^{-1}\left(\frac{R_{ij}}{\sqrt{Q_{ii}}}\right)}{2\pi}\right] + \frac{\pi+1}{2\pi}. \qquad (29)$$

Both results are for orthonormal teacher vectors, extensions to general $\mathbf{B}_m \cdot \mathbf{B}_n = T_{mn}$ can be found in [45, 47].

(g) *Learning curves*

The (numerical) integration of the ODE for a given particular training algorithm, model density and specific initial conditions $\{R_{\mathrm{im}}(0), Q_{ik}(0)\}$ yields the temporal evolution of order parameters in the course of training.

Exploiting the self-averaging properties of order parameters once more, we can obtain the learning curves $\epsilon_g(\alpha) = \epsilon_g(\{R_{\mathrm{im}}(\alpha), Q_{ik}(\alpha)\})$ or the class-wise $\epsilon_g^k(\alpha)$, respectively. Hence, we determine the typical generalization error after on-line training with $(\alpha N)$ random examples.

## 2.4 The learning dynamics under concept drift

The analysis summarized in the previous section concerns learning in the presence of a stationary concept, i.e., for a density of the form (4) or (13) which does not change in the course of training. Here, we introduce the effect of concept drift to the modelling framework and consider weight decay as an example mechanism for explicit *forgetting*.

### 2.4.1 Virtual drift in classification

As defined above, virtual drifts affect statistical properties of the observed example data while the actual target function remains unchanged.

A variety of virtual drift processes can be addressed in our modelling framework. For example, time-varying *label noise* in regression or classification could be incorporated in a straightforward way [4, 19, 53]. Similarly, non-stationary cluster variances in the input density, cf. Eq. (4), can be introduced through explicitly time-dependent $v_\sigma(\alpha)$ into Eq. (20) for the LVQ system.

Here we focus on a particularly relevant case in classification, in which a varying fraction of examples represents each of the classes in the data stream. We consider non-stationary, $\alpha$-dependent prior probabilities $p_1(\alpha) = 1 - p_2(\alpha)$ in the mixture density (4). In practical situations, varying class bias can complicate the training significantly and lead to inferior performance [52]. Specifically, we distinguish the following scenarios:

(A) *Drift in the training data only*

Here we assume that the true target classification is defined by a fixed *reference density* of data. As a simple example we consider equal priors $p_1 = p_2 = 1/2$ in a symmetric reference density (4) with $v_1 = v_2$. On the contrary, the characteristics of the observed training data are assumed to be time-dependent. In particular, we study the effect of non-stationary $p_m(\alpha)$ and weight decay on the learning dynamics. Given the order parameters of the learning systems in the course of training, the corresponding *reference generalization error*

$$\epsilon_{\text{ref}}(\alpha) = \left(\epsilon_g^1 + \epsilon_g^2\right)/2 \tag{30}$$

is obtained by setting $p_1 = p_2 = 1/2$ in Eq. (25), but inserting $R_{\text{im}}(\alpha)$ and $Q_{ik}(\alpha)$ as obtained from the integration of the corresponding ODE with time dependent $p_1(\alpha) = 1 - p_2(\alpha)$ in the training process.

(B) *Drift in training and test data*

In the second interpretation we assume that the variation of $p_m(\alpha)$ affects training and test data in the same way. Hence, the change of the statistical properties of the data is inevitably accompanied by a modification of the target classification: For instance, the Bayes optimal classifier and its best linear approximation depend explicitly on the actual priors [12].

The learning system is supposed to track the actual drifting concept and we refer to the corresponding generalization error as the *tracking error*

$$\epsilon_{\text{track}} = p_1(\alpha)\,\epsilon_g^1 + p_2(\alpha)\,\epsilon_g^2. \tag{31}$$

In terms of modelling the training dynamics, both scenarios, (A) and (B), require the same straightforward modification of the ODE system: the explicit introduction of $\alpha$-dependent quantities $p_\sigma(\alpha)$ in Eq. (20). The obtained temporal evolution yields the reference error $\epsilon_{\text{ref}}(\alpha)$ for the case of drift in the training data (A) and $\epsilon_{\text{track}}(\alpha)$ in interpretation (B).

Note that in both interpretations, we consider the very same drift processes affecting the training data. However, the interpretation of the relevant performance measure is different. In (A) only the training data is subject to the drift, but the classifier is evaluated with respect to an idealized static situation representing a fixed target. On the contrary, the tracking error in (B) is thought to be computed with respect to test data available from the stream, at the given time. Alternatively, one could interpret (B) as an example of real drift with a non-stationary target, where $\epsilon_{\text{track}}$ represents the corresponding generalization error. However, we will refer to (A) and (B) as virtual drift throughout the following.

### 2.4.2 Real drift in regression

In the presented framework, a real drift can be modelled as a process which displaces the characteristic vectors $\mathbf{B}_{1,2}$, i.e., the cluster centers in LVQ or the teacher weight vectors in the SCM. Here we focus on the latter case and refer the reader to [47] for earlier results on LVQ training under real drift.

A variety of time dependences could be considered in the model. We restrict ourselves to the analysis of diffusion-like random displacements of vectors $\mathbf{B}_{1,2}(\mu)$ at each time step. Upon presentation of example $\mu$, we assume that random vectors $\mathbf{B}_{1,2}(\mu)$ are generated which satisfy the conditions

$$\begin{aligned} \mathbf{B}_1(\mu) \cdot \mathbf{B}_1(\mu-1) &= \mathbf{B}_2(\mu) \cdot \mathbf{B}_2(\mu-1) = (1-\delta/N) \\ \mathbf{B}_1(\mu) \cdot \mathbf{B}_2(\mu) &= 0 \text{ and } |\mathbf{B}_1(\mu)|^2 = |\mathbf{B}_2(\mu)|^2 = 1. \end{aligned} \tag{32}$$

Here $\delta$ quantifies the strength of the drift process. The displacement of the teacher vectors is very small in an individual training step. For simplicity we assume that the orthonormality of teacher vectors is preserved in the drift. In continuous time $\alpha = \mu/N$, the drift parameter defines a characteristic scale $1/\delta$ on which the overlap of the current teacher vectors with their initial positions decay: $\mathbf{B}_m(\mu) \cdot \mathbf{B}_m(0) = \exp[-\delta\,\mu/N]$.

The effect of such a drift process is easily taken into account in the formalism: For a particular student $\mathbf{w}_i \in \mathbb{R}^N$ we obtain [6, 7, 28, 50]

$$[\mathbf{w}_i \cdot \mathbf{B}_k(\mu)] = (1-\delta/N)\,[\mathbf{w}_i \cdot \mathbf{B}_k(\mu-1)]. \tag{33}$$

under the above specified random displacement. Hence, the drift tends to decrease the quantities $R_{ik}$ which clearly reduces the success of training compared with the case of stationary teachers. The corresponding ODE in the limit $N \to \infty$ in the drift process (32) become

$$\begin{aligned} [dR_{\text{im}}/d\alpha]_{\text{drift}} &= [dR_{\text{im}}/d\alpha]_{\text{stat}} - \delta R_{\text{im}} \text{ and} \\ [dQ_{ik}/d\alpha]_{\text{drift}} &= [dQ_{ik}/d\alpha]_{\text{stat}} \end{aligned} \tag{34}$$

with the terms $[\cdots]_{\text{stat}}$ for stationary environments taken from Eq. (20). Note that now order parameters $R_{\text{im}}(\alpha)$ correspond to the inner products $\mathbf{w}_i^\mu \cdot \mathbf{B}_m(\alpha)$, as the teacher vectors themselves are time-dependent.

### 2.4.3 Weight decay

Possible motivations for the introduction of so-called *weight decay* in machine learning systems range from *regularization* as to reduce the risk of *over-fitting* in regression and classification [15, 22, 23] to the modelling of *forgetful memories* in attractor neural networks [24, 37].

Here we include weight decay as to enforce *explicit forgetting* and to potentially improve the performance of

the systems in the presence of real concept drift. We consider the multiplication of all adaptive vectors by a factor $(1 - \gamma/N)$ before the generic learning step given by $\Delta \mathbf{w}_i^\mu$ in Eq. (2) or Eq. (10) is performed:

$$\mathbf{w}_i^\mu = (1 - \gamma/N)\,\mathbf{w}_i^{\mu-1} + \eta/N\,\Delta \mathbf{w}_i^\mu. \qquad (35)$$

Since the multiplications with $(1 - \gamma/N)$ accumulate in the course of training, weight decay enforces an increased influence of the most recent training data as compared to *earlier* examples. Note that analagous modifications of perceptron training under concept drift have been discussed in [6, 7, 28, 50].

In the thermodynamic limit $N \to \infty$, the modified ODE for training under real drift, cf. Eq. (32), and weight decay, Eq. (35), are obtained as

$$[dR_{im}/d\alpha]_{\text{decay}} = [dR_{im}/d\alpha]_{\text{stat}} - (\delta + \gamma)R_{im} \text{ and}$$
$$[dQ_{ik}/d\alpha]_{\text{decay}} = [dQ_{ik}/d\alpha]_{\text{stat}} - 2\,\gamma\,Q_{ik} \qquad (36)$$

where the terms for stationary environments in absence of weight decay are given in Eq. (20).

# 3 Results and discussion

Here we present and discuss our results obtained by integrating the systems of ODE with and without weight decay under different time-dependent drifts. For comparison, averaged learning curves obtained by means of Monte Carlo simulations are also shown. These simulations of the actual training process provide an independent confirmation of the ODE-based description and demonstrate the relevance of results obtained in the thermodynamic limit $N \to \infty$ for relatively small, finite systems.

## 3.1 Virtual drift in LVQ training

All results presented in the following are for constant learning rate $\eta = 1$ in the LVQ training. The results remain qualitatively the same for a range of learning rates. LVQ prototypes were initialized as normalized independent random vectors without prior knowledge:

$$Q_{11}(0) = Q_{22}(0) = 1,\ Q_{12}(0) = 0,\ \text{and}\ R_{ik}(0) = 0. \qquad (37)$$

We study three specific scenarios for the time dependence $p_1(\alpha) = 1 - p_2(\alpha)$ as detailed in the following.

### 3.1.1 Linear increase of the bias

Here we consider a time-dependent bias of the form $p_1(\alpha) = 1/2$ for $\alpha < \alpha_o$ and

$$p_1(\alpha) = \frac{1}{2} + \frac{(p_{\max} - 1/2)\,(\alpha - \alpha_o)}{(\alpha_{\text{end}} - \alpha_o)}\ \text{for}\ \alpha \geq \alpha_o. \qquad (38)$$

where the maximum class weight $p_1 = p_{\max}$ is reached at learning time $\alpha_{\text{end}}$. Figure 1 shows the learning curves as obtained by numerical integration of the ODE together with Monte Carlo simulation results for $(N = 100)$-dimensional inputs and prototype vectors. As an example we set the parameters to $\alpha_o = 25, p_{\max} = 0.8, \alpha_{\text{end}} = 200$. The learning curves are displayed for LVQ1 without weight decay (upper) and with $\gamma = 0.05$ (lower panel). Simulations show excellent agreement with the ODE results.

The system adapts to the increasing imbalance of the training data, as reflected by a decrease (increase) of the class-wise error for the over-represented (under-represented) class, respectively. The weighted overall error $\epsilon_{\text{track}}$ also decreases, i.e., the presence of class bias facilitates smaller total generalization error, see [12]. The performance with respect to unbiased reference data deteriorates slightly, i.e., $\epsilon_{\text{ref}}$ grows with increasing class bias as the training data represents the target less faithfully.

### 3.1.2 Sudden change of the class bias

Here we consider an instantaneous switch from low bias $p_1(\alpha) = 1 - p_{\max}$ for $\alpha \leq \alpha_o$ to high bias

$$p_1(\alpha) = \begin{cases} 1 - p_{\max} & \text{for}\ \alpha \leq \alpha_o. \\ p_{\max} > 1/2 & \text{for}\ \alpha > \alpha_o. \end{cases} \qquad (39)$$
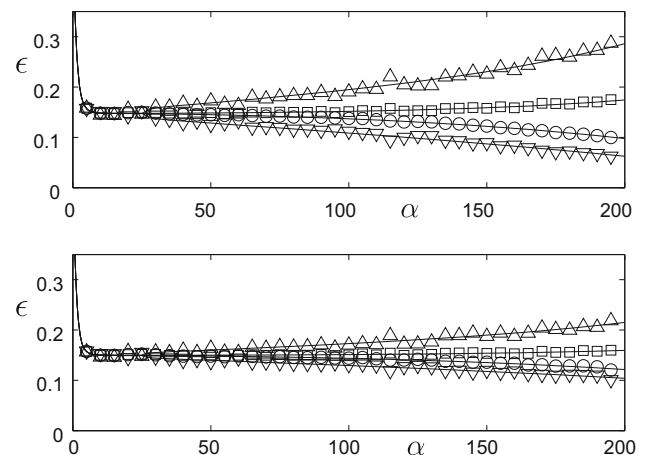


**Fig. 1** LVQ1 in the presence of a concept drift with linearly increasing $p_1(\alpha)$ given by $\alpha_o = 20$, $\alpha_{\text{end}} = 200$, $p_{\max} = 0.8$ in (38). Solid lines correspond to the integration of ODE with initialization as in Eq. (37). We set $v_{1,2} = 0.4$ and $\lambda = 1$ in the density (4). The upper graph corresponds to LVQ1 without weight decay, the lower graph displays results for $\gamma = 0.05$ in Eq. (35). In addition, Monte Carlo results for $N = 100$ are shown: class-wise errors $\epsilon^{1,2}(\alpha)$ are displayed as downward (upward) triangles, respectively; squares mark the reference error $\epsilon_{\text{ref}}(\alpha)$; circles correspond to $\epsilon_{\text{track}}(\alpha)$, cf. Eqs. (30, 31)

We consider $p_{max} = 0.75$ as an example, the corresponding results from the integration of ODE and Monte Carlo simulations are shown in Fig. 2 for training without weight decay (upper) and for $\gamma = 0.05$ (lower panel).

We observe similar effects as for the slow, linear time dependence: The system reacts rapidly with respect to the class-wise errors and the tracking error $\epsilon_{track}$ maintains a relatively low value. Also, the reference error $\epsilon_{ref}$ displays robustness with respect to the sudden change of $p_1$. Weight decay, as can be seen in the lower panel of Fig. 2 reduces the overall sensitivity to the bias and its change: Class-wise errors are more balanced and the weighted $\epsilon_{track}$ slightly increases compared to the setting with $\gamma = 0$.

### 3.1.3 Periodic time dependence

As a third scenario we consider oscillatory modulations of the class weights during training:

$$p_1(\alpha) = 1/2 + (p_{max} - 1/2) \cos(2\pi\alpha/T) \qquad (40)$$

with periodicity $T$ on $\alpha$-scale and maximum amplitude $p_{max} < 1$. Example results are shown in Fig. 3 for $T = 50$ and $p_{max} = 0.8$. Monte Carlo results for $N = 100$ are only displayed for the class-wise errors, for the sake of clarity. They show excellent agreement with the numerical integration of the ODE for training without weight decay (upper panel) and for $\gamma = 0.05$ (lower panel). These results confirm our findings for slow and sudden changes of the prior weights: Weight decay limits the flexibility of the LVQ system to react to the presence of a bias and its time dependence.
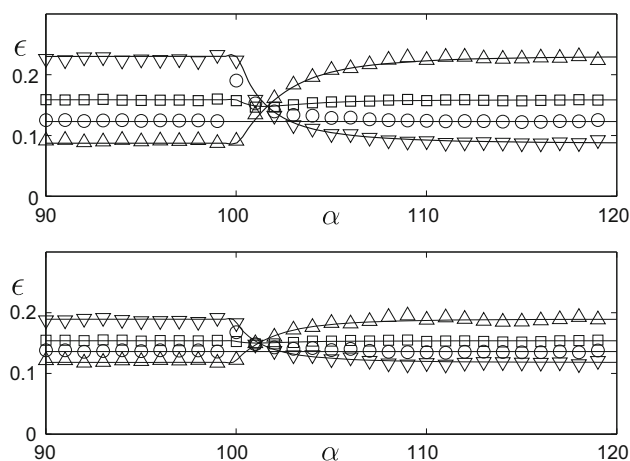


**Fig. 2** LVQ1 in the presence of a concept drift with a sudden change of class weights according to Eq. (39) with $\alpha_o = 100$ and $p_{max} = 0.75$. Only the $\alpha$-range close to $\alpha_o$ is shown. All other details are provided in Fig. 1
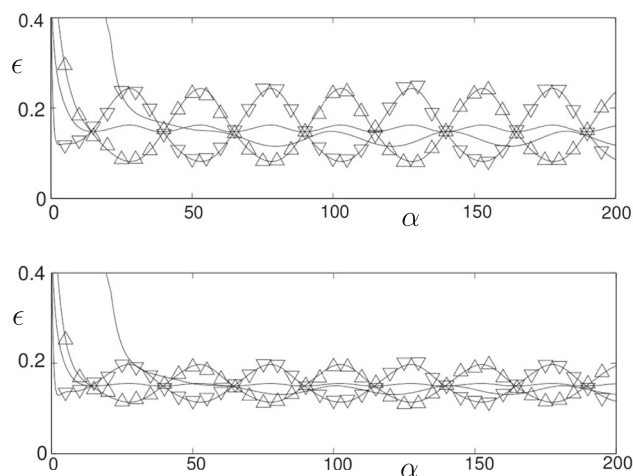
**Fig. 3** LVQ1 in the presence of oscillating class weights according to Eq. (40) with parameters $T = 50$ and $p_{max} = 0.8$, without weight decay $\gamma = 0$ (upper graph) and for $\gamma = 0.05$ (lower). For clarity, Monte Carlo results are only shown for the class-conditional errors $\epsilon^1$ (downward) and $\epsilon^2$ (upward triangles). All other details are given in Fig. 1

### 3.1.4 Discussion: LVQ under virtual drift

Our results for the different realizations of time-dependent class weights show that Learning Vector quantization can cope with this form of drift to a certain effect. By design, standard incremental updates like the classical LVQ1 allow the prototypes to adjust to the changing statistics of the data. This has been shown in [47] for the actual drift of the cluster centers in the model density. Here we show that LVQ1 can also cope with the virtual drift processes.

In analogy to our findings in [47], one might have expected improved performance when introducing weight decay as a mechanism of *forgetting*. As we demonstrate, however, weight decay does not have a very strong effect on the the system's reaction to changing prior class weights. Essentially, weight decay limits the prototype norms and hinders shifts of the decision boundary by prototype displacement. The overall influence of class bias and its time dependence is reduced in the presence of weight decay. Weight decay restricts the norm of the prototypes, i.e., the possible offset of the decision boundary from the origin. As a consequence, the tracking error slightly increases for $\gamma > 0$, in general. On the contrary, the error $\epsilon_{ref}$ with respect to the reference density decreases compared to the training without weight decay.

A clear beneficial effect of *forgetting* previous information in favor of the most recent examples cannot be confirmed. The reaction of the learning system to sudden (B) or oscillatory changes of the priors (C) remains also unchanged when introducing weight decay.

## 3.2 Results: SCM regression under real drift

Here we present the results concerning the SCM student–teacher scenario with $K = M = 2$ under real concept drift, i.e., random displacements of the teacher vectors as introduced in Sect. 2.4.2. Unlike LVQ for classification, gradient descent-based training of a regression system is expected to be much more sensitive to the choice of the learning rate. Here, we restricted the discussion to the well-defined limit of small learning rates, $\eta \to 0$ and $\alpha \to \infty$ with $\widetilde{\alpha} = \eta\alpha = \mathcal{O}(1)$, see the discussion before Eq. (21). In the corresponding Monte Carlo simulations, cf. Fig. 4a, b, we employed a small learning rate $\eta = 0.05$ which yielded very good agreement.

Already in the absence of concept drift, the model displays non-trivial effects as shown in, for instance [9, 44, 45]. Perhaps the most thoroughly studied phenomenon in the SCM training process is the existence of quasi-stationary plateaus in the evolution of the order parameters and the generalization error. In the most clear-cut cases, they correspond to approximately symmetric configurations of the student network with respect to the teacher network, i.e., $R_{im} \approx R$ for all $i$, $m$. In such a state, all student units have acquired the same, limited knowledge of the target rule. Hence, the generalization error in the plateau is sub-optimal. In terms of Eq. (20), plateaus correspond to weakly repulsive fixed points of the ODE system. One can show in case of orthonormal teacher units and for small learning rates that a symmetric fixed point with $R_{im} = R$ and the associated plateau state always exists; see, e.g., [45]. In order to achieve a further decrease of the generalization error, the symmetry of the student with respect to the teacher units has to be broken by *specialization*: Each student weight vector $\mathbf{w}_{1,2}$ has to represent a specific teacher unit and $R_{i1} \neq R_{i2}$ is required for successful learning.

Our recent comparison of Erf-SCM and ReLU-SCM revealed interesting differences even in absence of concept drift [46]. For instance, in the Erf-SCM, student vectors are nearly identical in the symmetric plateau with $Q_{ik} \approx Q$ for all $i, k \in \{1, 2\}$. On the contrary, in ReLU systems the student weights are not aligned in the quasi-stationary state: $Q_{ii} = Q$ and $Q_{12} < Q$ [46].

### 3.2.1 ODE and Monte Carlo simulations

Here, we investigate and compare the learning dynamics of networks with Erf- and ReLU-activation under concept drift and in the presence of weight decay. To this end we study the models by numerical integration of the corresponding ODE and, in addition, by Monte Carlo simulations.

We study training processes in absence of prior knowledge in the student. In the following we consider exemplary initial conditions with

$$\begin{aligned} R_{im}(0) &= 0, \\ Q_{11}(0) &= Q_{22}(0) = 0.5, \\ Q_{12}(0) &= 0.49 \end{aligned} \tag{41}$$

which correspond to almost identical $\mathbf{w}_1(0)$ and $\mathbf{w}_2(0)$, which are both orthogonal to the teacher vectors. Note that the initial norm of the student vectors and their mutual *overlap* $Q_{12}(0)$ can be set arbitrarily in practice.

For the networks with two hidden units we define the quantity $S_i(\alpha) = |R_{i1}(\alpha) - R_{i2}(\alpha)|$ as the specialization of student units $i = 1, 2$. In the plateau state, $S_i(\alpha) \approx 0$ for an extended amount of training time, while an increasing value of $S_i(\alpha)$ indicates the specialization of the unit. In practice, one expects that initially $R_{im}(0) \approx 0$ for all $i$, $m$ if no prior information is available about the target rule. Hence, the student specialization $S_i(0) = |R_{i1}(0) - R_{i2}(0)|$ is also small, initially.
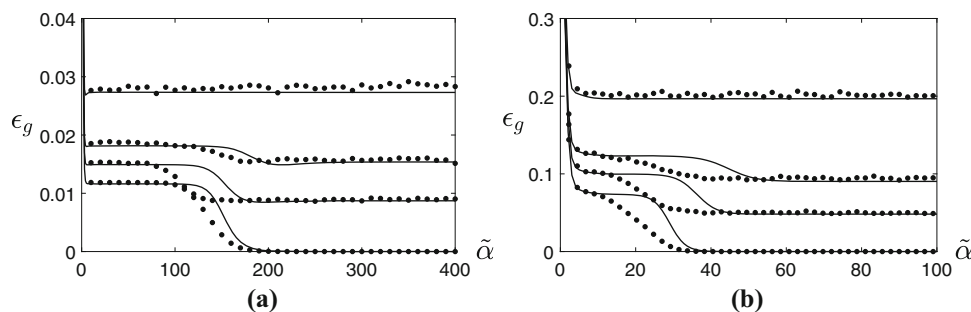


**Fig. 4** The learning performance under concept drift in terms of generalization error as a function of the learning time $\widetilde{\alpha}$. Dots correspond to 10 runs of Monte Carlo simulations with $N = 500$, $\eta = 0.05$ with initials conditions as in Eq. (41). Solid lines show ODE integrations. **a** Erf SCM. From bottom to top, the curves correspond to the levels of target drift $\widetilde{\delta} = \{0, 0.01, 0.02, 0.05\}$. **b** ReLU SCM. From bottom to top, the levels of target drift are: $\widetilde{\delta} = \{0, 0.05, 0.1, 0.3\}$

The unspecialized plateau can dominate the learning process and, consequently, its length is a quantity of significant interest. Quite generally, it is governed by the repulsive properties of the relevant fixed point of the ODE system and depends logarithmically on the the magnitude of the initial specialization $S_i(0)$, see [9] for a detailed discussion. In simulations for large $N$, a random initialization of student vectors would result in overlaps $R_{im}(0) = \mathcal{O}(1/\sqrt{N})$ with the teacher vectors which also implies that $S_i(0) = \mathcal{O}(1/\sqrt{N})$. The accurate extrapolation of simulation results for $N \to \infty$ is complicated by this interplay of finite size effects and initial specialization which governs the escape from the plateau states [9]. Due to fluctuations in a finite system, plateaus are typically left earlier than predicted by the theoretical prediction for $N \to \infty$. Here we focus on the performance achieved in the plateau states and resort to a simpler strategy: The values of the order parameters observed at $\widetilde{\alpha} = 0.05$ in the Monte Carlo simulation are used as initial values for the numerical integration of the ODE. This does not necessarily warrant a one-to-one correspondence of the precise shape and length of the plateau states. However, the comparison shows excellent qualitative agreement and allows for the quantitative comparison of the performance in the quasistationary and states.

We have studied the Erf-SCM and the ReLU-SCM under concept drift, Eq. (32), and weight decay, Eq. (35), in the limit of small learning rates $\eta \to 0$. We resorted to this simplifying limit as the term $G_{ik}^{(2)}$ in Eq. (24) could not be obtained analytically for the ReLU-SCM. However, non-trivial results can be achieved in terms of the rescaled training time $\widetilde{\alpha}$ in the limit (21). Hence we integrate the ODE provided in Eq. (22), combined with the drift and weight decay terms from Eqs. (34, 36) that also have to be scaled with $\eta$ in this case: $\widetilde{\delta} = \eta\delta$, $\widetilde{\gamma} = \eta\gamma$. In addition to the numerical integration we have performed and averaged over 10 independent runs of Monte Carlo simulations with system size $N = 500$ and small but finite learning rate $\eta = 0.05$.

### 3.2.2 Learning curves under concept drift

Figure 4 shows the learning curves $\epsilon_g(\widetilde{\alpha})$ as results of the averaged Monte Carlo simulations and the ODE integration for different strengths $\widetilde{\delta}$ of concept drift with no weight decay ($\widetilde{\gamma} = 0$). The left and right panel corresponds to Erf- and ReLU-SCM, respectively.

Apart from deviations in terms of the plateau lengths, simulations and the numerical integration of the ODE show very good agreement. In particular, the generalization error in the plateau and final states nearly coincides. As outlined in Sect. 3.2.1, the actual length of plateaus in simulations

depends on subtle details [9] which were not addressed here.

Note also that a direct, quantitative comparison of Erf- and ReLU-SCM in terms of training times $\widetilde{\alpha}$ is not meaningful. For instance, it seems tempting to conclude that the ReLU-SCM exhibit shorter plateau states for the same network size and training conditions. However, one has to take into account that the activation functions influence the complexity of the input output relation of the network in a non-trivial way.

From the behavior of the learning curves for increasing strengths $\widetilde{\delta}$, several impeding effects of the drift can be identified: The generalization error in the unspecialized plateau and in the final state for large $\widetilde{\alpha}$ increase with $\widetilde{\delta}$. At the same time, the plateau lengths increase. These effects are observed for both types of activation function. More specifically, the behavior for small $\widetilde{\delta}$ is close to the stationary setting with $\widetilde{\delta} = 0$: A rapid initial decrease of the generalization error is followed by the quasi-stationary plateau state that persists for a relatively long training time. Eventually, the system escapes from the plateau and improved generalization performance becomes possible. Despite the matching complexity of student and teacher, perfect generalization cannot be achieved in the presence of on-going concept drift.

We note that the stronger the drift, the smaller is the difference between the performance in the plateau and the final state. For very large values of $\widetilde{\delta}$ both versions of the SCM cannot escape the plateau state anymore as it corresponds to a stable fixed point of the ODE.

In the following we discuss for both activation functions the effect of concept drift on the plateau- and final generalization error in greater detail. The influence of weight decay on the dynamics is also presented.

*Erf-SCM under drift and weight decay*

Figure 5a displays the effect of the drift strength $\widetilde{\delta}$ on the generalization error in the unspecialized plateau state and in the final state for $\widetilde{\alpha} \to \infty$, i.e., $\epsilon_g^p(\widetilde{\delta})$ and $\epsilon_g^\infty(\widetilde{\delta})$, respectively. As mentioned above, weak drifts still allow for student specialization with improved performance in the final state for large $\widetilde{\alpha}$. However, increasing the drift strength results in a decrease of the difference $|\epsilon_g^\infty(\widetilde{\delta}) - \epsilon_g^p(\widetilde{\delta})|$. We have marked the value of $\widetilde{\delta}$, above which the plateau becomes the stable final state for $\widetilde{\alpha} \to \infty$ in the figure and refer to it as $\widetilde{\delta}_c$.

Interestingly, in a small range of the drift parameter, $0.036 < \widetilde{\delta} < 0.061$, the final performance is actually worse than in the plateau with $\epsilon_g^\infty(\widetilde{\delta}) > \epsilon_g^p(\widetilde{\delta})$. Since $\epsilon_g$ depends explicitly also on the $Q_{ik}$, it is possible for an unspecialized state with $R_{im} = R$ to generalize better than a slightly
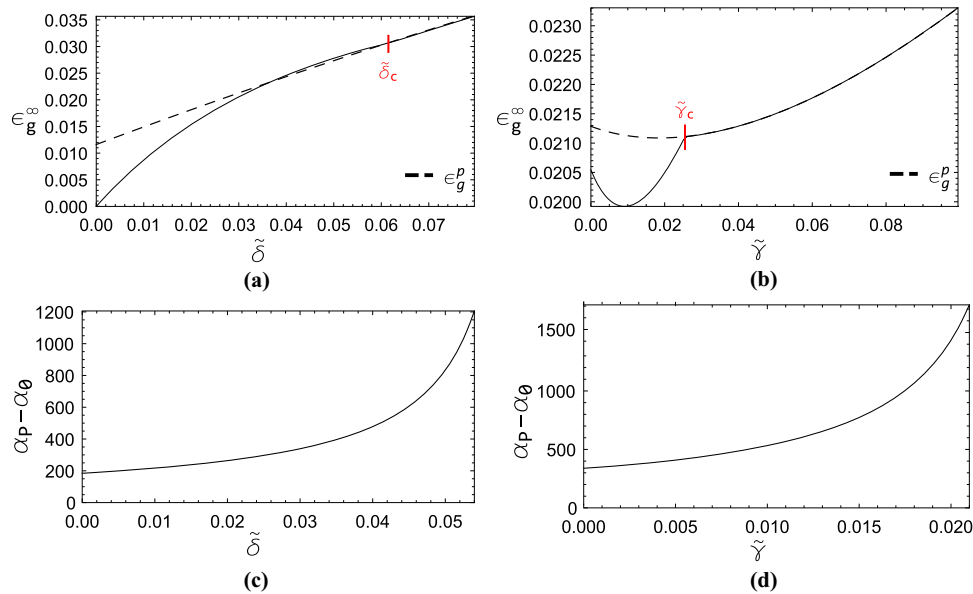
**Fig. 5** Erf-SCM: Generalization error under concept drift in unspecialized plateau states (dashed lines) and final states (solid) of the learning process. **a** Plateau- and final generalization error for an increasing strength $\widetilde{\delta}$ of the target drift. Here, weight decay is not applied: $\widetilde{\gamma} = 0$. For $\widetilde{\delta} > \widetilde{\delta}_c$ as marked by the vertical line, the curves merge. **b** The plateau- and final generalization error as a function of the weight decay parameter $\widetilde{\gamma}$ for a fixed level of real target drift, here: $\widetilde{\delta} = 0.03$. The curves merge for $\widetilde{\gamma} > \widetilde{\gamma}_c$, as marked by the vertical line. The lower panels show the observed plateau lengths as a function of $\widetilde{\delta}$ for $\widetilde{\gamma} = 0$ (**c**) and as a function of $\widetilde{\gamma}$ for fixed $\widetilde{\delta} = 0.03$ (**d**), respectively

specialized configuration with unfavorable values of the student norms and mutual overlaps.

Figure 5c shows the effect of the drift on the plateau length. The start and end of the plateau are defined as

$$
\begin{aligned}
\widetilde{\alpha}_0 &= \min\{\widetilde{\alpha} \mid \epsilon_g^p - 10^{-4} < \epsilon_g(\widetilde{\alpha}) < \epsilon_g^p + 10^{-4}\} \\
\widetilde{\alpha}_P &= \min\{\widetilde{\alpha} \mid S_i(\widetilde{\alpha}) \geq 0.2\, S_i(\widetilde{\alpha} \to \infty)\}\,.
\end{aligned}
\tag{42}
$$

Here, $S_i(\widetilde{\alpha} \to \infty)$ represents the final specialization that is achieved by the system for large training times. $(\alpha_P - \alpha_0)$ is used as a meaure of the plateau length.

In the weak drift regime, the plateau length increases slowly with $\widetilde{\delta}$ as shown in panel (c) for $\widetilde{\gamma} = 0$. It eventually diverges as $\widetilde{\delta}$ approaches $\widetilde{\delta}_c$ from Fig. 5a.

The dependence of $\epsilon_g^p$ and $\epsilon_g^\infty$ on the weight decay parameter $\widetilde{\gamma}$ is shown in Fig. 5b. We observe improved performance for a small amount weight decay compared to absence of weight decay ($\widetilde{\gamma} = 0$). However, the system is quite sensitive to the actual setting of $\widetilde{\gamma}$: Values slightly larger than the optimum quickly deteriorate the ability for improvement from the plateau generalization error. The value of $\widetilde{\gamma}$, above which the plateau- and final generalization error coincide has been marked in the figure and we refer to it as $\widetilde{\gamma}_c$.

Figure 5d shows the effect of the weight decay on the plateau length in the same setting as in Fig. 5b. Introducing a weight decay always extends the plateau length. For small $\widetilde{\gamma}$ the plateau length grows slowly and diverges as $\widetilde{\gamma}$ approaches $\widetilde{\gamma}_c$ from Fig. 5b.

*ReLU-SCM under drift and weight decay*

The effect of the strength of the drift on the generalization error in the unspecialized plateau state and in the final state is displayed in Fig. 6a. The picture is similar to the Erf-SCM: an increase in the drift strength causes an increase in the plateau- and final generalization error. We have marked in the figure the drift strength $\widetilde{\delta}_c$ at which there is no further change in performance from the plateau. In contrast to the Erf-SCM, there is no range of $\widetilde{\gamma}$ for which the ReLU-SCM generalization error increases after leaving the plateau.

Figure 6c shows the effect of the strength of the drift on the plateau length. Here too, a similar dependence is observed as for the Erf-SCM: For the range of weaker drifts, the plateau length grows slowly and diverges for strong drifts up to the drift strength $\widetilde{\delta}_c$ from Fig. 6a.

Figure 6b shows the effect of the amount of weight decay on the plateau- and final generalization error in a concept drift situation. A small amount of weight decay can improve the generalization error compared to no weight decay ($\widetilde{\gamma} = 0$). The effect weight decay has on the ReLU-SCM shows a much greater robustness compared to the Erf-SCM in terms of the ability to improve from the plateau value: For high amounts of weight decay, an escape from the plateau to better performance can still be

**Fig. 6** ReLU-SCM: Generalization error under concept drift in unspecialized plateau states (dashed lines) and final states (solid), as a function of the drift strength (**a**) and weight decay (**b**). In (**b**), $\widetilde{\delta} = 0.2$. The drift strength $\widetilde{\delta}_c$ above which the curves merge is marked in (**a**) and similar for weight decay $\widetilde{\gamma}_c$ in (**b**). The lower panels show the observed plateau lengths as a function of $\widetilde{\delta}$ for $\widetilde{\gamma} = 0$ (**c**) and as a function of $\widetilde{\gamma}$ for fixed $\widetilde{\delta} = 0.2$ (**d**), respectively
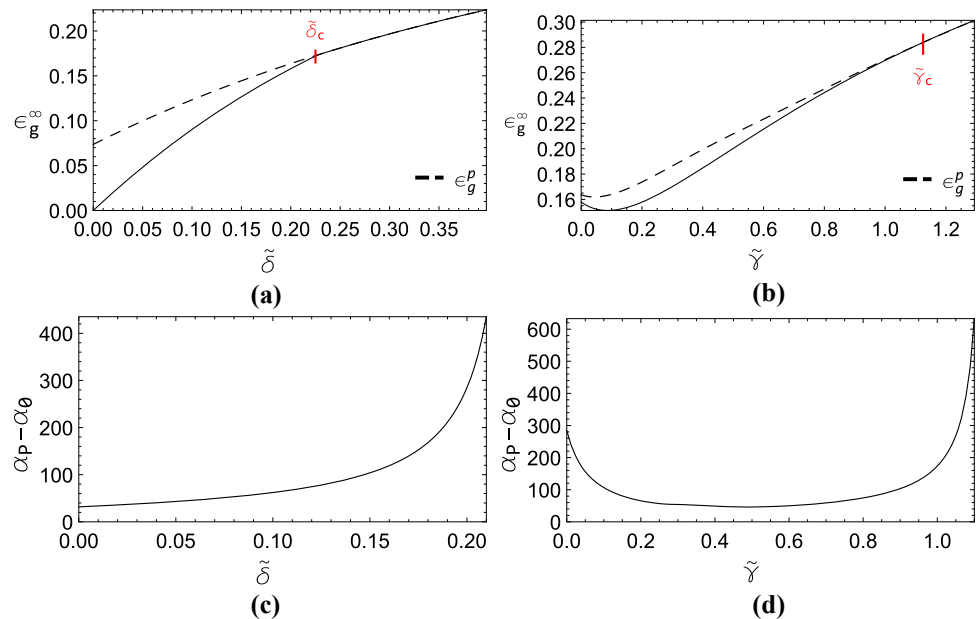


observed. The value $\widetilde{\gamma}_c$, above which the plateau- and final generalization error coincide has been marked in the figure.

Figure 6d shows the effect of the amount of weight decay on the plateau length in the same concept drift situation as in Fig. 6b. It shows that the plateau is shortened significantly in the smaller range of weight decay, the same range that also improves the final generalization error as observed in Fig. 6b. The plateau length increases again for very high levels of weight decay and diverges as $\widetilde{\gamma}$ approaches the $\widetilde{\gamma}_c$ from Fig. 6b.

### 3.3 Discussion: SCM regression under real drift

As was already discussed, the symmetric plateau corresponds to states where the student units have all learned the same limited and general knowledge about the teacher units, i.e., $R_{ij} \approx R$ and therefore the specialization of each student unit $i$ is small: $S_i(\widetilde{\alpha}) = |R_{i1}(\widetilde{\alpha}) - R_{i2}(\widetilde{\alpha})| \approx 0$. Eventually, the symmetry is broken by the start of specialization, when $S_i(\widetilde{\alpha})$ increases for each student unit $i$. For stationary learnable situations with $K = M$, throughout learning the students units will acquire a full overlap to the teacher units: $S_i = 1$ for all student units $i$. In this configuration, the target rule has been fully learned and therefore the generalization error is zero. In our modelled concept drift, the teacher vectors are changing continuously. This reduces the overlaps the student units can achieve with the teacher units, which increases the generalization error in the plateau state and the final state.

Identifying the specific teacher vectors is more difficult than learning the general structure of the teacher: Hence, increasing the drift causes the final generalization error to deteriorate faster than the plateau generalization error. For very strong target drift, the teacher vectors are changing too fast for specialization to be possible. We have identified the strength of the drift above which any kind of specialization is impossible for both SCM by studying the properties of the fixed point in the ODE. In stationary situations, one eigenvalue of the linearized dynamics near the fixed point is positive and causes the repulsion away from the fixed point to specialization. We refer to this positive eigenvalue as $\lambda_s$. The eigenvalue decreases linearly with the drift strength: For small $\widetilde{\delta}$, $\lambda_s$ is still positive and therefore an escape from the plateau is observed. However, $\lambda_s$ is negative for $\widetilde{\delta} > \widetilde{\delta}_c$, the symmetric fixed point is stable and specialization becomes impossible. For the Erf-SCM, $\widetilde{\delta}_c \approx 0.0615$ and for the ReLU-SCM $\widetilde{\delta}_c \approx 0.225$. The weaker repulsion of the fixed point for stronger drift causes the plateau length to grow for $\widetilde{\delta} \to \widetilde{\delta}_c$. In practice, this implies that higher training effort is necessary the stronger the concept drift is.

In the $\widetilde{\alpha} \to \infty$ final state, the student tracks the drifting target rule. For $\widetilde{\delta} \ll \widetilde{\delta}_c$, the student can achieve highly specialized states while tracking the teacher. The closer the drift strength is to $\widetilde{\delta}_c$, the weaker is the specialization that can be achieved by the student while following the rapidly moving teacher vectors. For $\widetilde{\delta} > \widetilde{\delta}_c$, the unspecialized student can only track the rule in terms of a simple approximation.

In the results of the Erf-SCM, a range of drift strength $0.036 < \widetilde{\delta} < \widetilde{\delta}_c$ was observed for which the final generalization error in the tracking state is worse than the plateau generalization error. Upon further inspection, this is due to

the large values of $Q_{11}$ and $Q_{22}$ of the student vectors in the specialized regime. Hence, the effect can be prevented by introducing an appropriate weight decay.

### 3.3.1 Erf SCM versus ReLU SCM: weight decay in concept drift situations

Our results show that weight decay can improve the final generalization error in the specialized tracking state for both SCM. The suppression of the contributions of older and thus less representative data shows benefits in both systems.

However, from the result in Fig. 5b, we find that it is particularly important to tune the weight decay parameter for the Erf-SCM, since the specialization ability deteriorates quickly for values slightly off the optimum, as shown in the figure by the rapid increase in $\epsilon_g^\infty$. This reflects a steep decrease of the largest eigenvalue $\lambda_s$ in the ODE for the Erf-SCM with $\widetilde{\gamma}$, which also causes the increase of the plateau length as observed in Fig. 5d. Already from $\widetilde{\gamma}_c \approx 0.0255$, the eigenvalue $\lambda_s$ becomes negative, and therefore the fixed point becomes an attractor.

We found a very different effect of weight decay on the performance of the ReLU-SCM. Not only is it able to improve the final generalization error in the tracking state as shown in Fig. 6b, but it also significantly reduces the plateau length in the lower range of weight decay. This reflects the increase of $\lambda_s$ with the weight decay parameter in the fixed point of the ODE, which increases the repulsion from the unspecialized fixed point. Clearly, suppressing the contribution of older data is beneficial for the specialization ability of the ReLU-SCM. For larger values of $\widetilde{\gamma}$, the plateau length increases, reflecting a decrease of $\lambda_s$. However, specialization remains possible up to a rather high value of weight decay $\widetilde{\gamma}_c \approx 1.125$. The greater robustness to weight decay with respect to specialization as shown in Fig. 6b is likely related to our previous findings in [46], which show that the ReLU student–teacher setup needs fewer examples to reach specialization. We hypothesize that the simple linear nature of the function makes it easier for the student to learn features of the target rule. Hence a relatively small window of recent examples can already facilitate a degree of specialization.

## 4 Summary and outlook

We have presented a mathematical framework in which to study the influence of concept drift systematically in model scenarios. We exemplified the use of the versatile approach in terms of models for the training of prototype-based

classifiers (LVQ) and shallow neural networks for regression, respectively.

*LVQ for classification under drift and weight decay*

In all specific drift scenarios considered here, we observe that simple LVQ training can track the time-varying class bias to a non-trivial extent: In the interpretation of the results in terms of real drift, the class-conditional performance and the tracking error $\epsilon_{\text{track}}(\alpha)$ clearly reflect the time dependence of the prior weights. In general, the reference error $\epsilon_{\text{ref}}(\alpha)$ with respect to class-balanced test data, displays only little deterioration due to the drift in the training data. The main effect of introducing weight decay is a reduced overall sensitivity to bias in the training data: Figures 1, 2 and 3 display a decreased difference between the class-wise errors $\epsilon^{1,2}$ for $\gamma > 0$. Naïvely, one might have expected an improved tracking of the drift due to the imposed *forgetting*, resulting in, for instance, a more rapid reaction to the sudden change of bias in Eq. (39). However, such an improvement cannot be confirmed. This finding is in contrast to a recent study [47], in which we observe increased performance by weight decay for a particular real drift, i.e., the randomized displacement of cluster centers.

The precise influence of weight decay clearly depends on the geometry and relative position of the clusters. Its dominant effect, however, is the regularization of the LVQ system by reducing the norms of the prototype vectors. Consequently, the NPC classifier is less flexible to reflect class bias which would require significant offset of the prototypes and decision boundary from the origin. This mildens the influence of the bias and its time dependence, and it results in a more robust behavior of the employed error measures.

*SCM for regression under drift and weight decay*

On-line gradient descent learning in the SCM has proven able to cope with drifting concepts in regression: For weak drifts, the SCM still achieves significant specialization with respect to the drifting teacher vectors, although the required learning time increases with the strength of the drift. In practice, this results in higher training effort to reach beneficial states in the network. The drift constantly reduces the overlaps with the teacher vectors which deteriorates the performance. After reaching a specialized state, the network efficiently tracks the drifting target. However, in the presence of very strong drift, both versions of the SCM (with Erf- and ReLU-activation) lose their ability to specialize and as a consequence their generalization behavior remains poor.

We have shown that weight decay can improve the performance in the plateau and in the final tracking state. For the Erf-SCM, we found that there is a small range in which weight decay yields favorable performance while

the network quickly loses the specialization ability for values outside this range. Therefore, in practice a careful tuning of the weight decay parameter would be required. The ReLU network showed greater robustness to the magnitude of the weight decay parameter and displayed a stronger tendency to specialize. Weight decay also reduced the plateau length significantly in the training of ReLU SCM. Hence, weight decay could speed up the training of ReLU networks in practical concept drift situations, achieving favorable weight configurations more efficiently. Also, the network performs well with a larger range of the weight decay parameter and does not require the careful tuning necessary for the Erf-SCM.

### Outlook

The presented modelling framework offers the possibility to extend the scope of our studies in several relevant directions. For instance, the formalism facilitates the consideration of more complex model scenarios. Greater values of $K$ and $M$ should be studied in, both, classification and regression. While we expect key results to carry over from $K = M = 2$, the greater complexity of the systems should result in richer dynamical behavior in detail. We will study if and how a mismatched number of prototypes further impedes the ability of LVQ systems to react appropriately to the presence of concept drift. The training of an SCM with $K \neq M$ should be of considerable interest and will also be addressed in forthcoming studies. One might speculate that concept drift could enhance overfitting effects in over-sophisticated SCM with $K > M$ hidden units. Ultimately, the characteristic robustness of the ReLU activation function to weight decay that was found should be studied in practical situations. Qualitative results are likely to carry over to similarly shaped activation functions, which will be verified in future work.

In a sense, the considered sigmoidal and ReLU activation functions are prototypical representatives of the most popular choices in machine learning practice. The extension to various modifications or significantly different transfer functions [18, 22] should provide additional valuable insights of practical relevance. Exact solutions to the averages that are necessary for the formulation of the learning dynamics in the thermodynamic limit may not be available for all activation functions. In such cases we can resort to approximations schemes and simulations.

The consideration of more complex input densities will also shed light on the practical relevance of our theoretical investigations. Recent work [21, 33] shows that the statistical physics-based investigation of machine learning processes can take into account realistic input densities, bridging the gap between the theoretical models and practical applications.

Our modeling framework can also be applied in the analysis of other types of drift or combinations thereof.

Several virtual processes could readily be implemented in the model of LVQ training: time-dependent characteristics of the input density could include the variances of the clusters or their relative position in feature space. A number of extensions is also possible in the regression model. For instance, teacher networks with time-dependent complexity could be studied by varying the mutual teacher overlaps $\mathbf{B}_m \cdot \mathbf{B}_n$ in the course of training.

Alternative mechanisms of *forgetting* beyond weight decay should be considered, which do not limit the flexibility of the trained systems as drastically. As one example strategy we intend to investigate the accumulation of additive noise in the training processes. We will also explore the parameter space of the model density in greater depth and study the influence of the learning rate systematically.

One of the major challenges in the field is the reliable detection of concept drift in a stream of data. Learning systems should be able to discriminate drift from static noise in the data and infer also the type of drift, e.g., virtual versus real. Moreover, the strength of the drift has to be estimated reliably in order to adjust the training prescription accordingly. It could be highly beneficial to extend our framework towards efficient drift detection and estimation procedures.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Ade R, Desmukh P (2013) Methods for incremental learning—a survey. Int J Data Min Knowl Manag Process 3(4):119–125

2. Ahr M, Biehl M, Urbanczik R (1999) Statistical physics and practical training of soft-committee machines. Eur Phys J B 10:583–588

3. Amunts K, Grandinetti L, Lippert T, Petkov N (eds) (2014) Brain-inspired computing, second international workshop brain-Comp 2015. LNCS, vol 10087. Springer, Berlin

4. Barkai N, Seung H, Sompolinsky H (1993) Scaling laws in learning of classification tasks. Phys Rev Lett 70(20):L97–L103

5. Biehl M, Caticha N (2003) The statistical mechanics of on-line learning and generalization. In: Arbib M (ed) The handbook of brain theory and neural networks. MIT Press, London, pp 1095–1098

6. Biehl M, Schwarze H (1992) On-line learning of a time-dependent rule. Europhys Lett 20:733–738

7. Biehl M, Schwarze H (1993) Learning drifting concepts with neural networks. J Phys A Math Gen 26:2651–2665

8. Biehl M, Schwarze H (1995) Learning by on-line gradient descent. J Phys A Math Gen 28:643–656

9. Biehl M, Riegler P, Wöhler C (1996) Transient dynamics of on-line learning in two-layered neural networks. J Phys A Math Gen 29:4769–4780

10. Biehl M, Freking A, Reents G (1997) Dynamics of on-line competitive learning. Europhys Lett 38:73–78

11. Biehl M, Schlösser E, Ahr M (1998) Phase transitions in soft-committee machines. Europhys Lett 44:261–266

12. Biehl M, Ghosh A, Hammer B (2007) Dynamics and generalization ability of LVQ algorithms. J Mach Learn Res 8:323–360

13. Biehl M, Hammer B, Villmann T (2016) Prototype-based models in machine learning. Wiley Interdiscipl Rev Cogn Sci 7(2):92–111. https://doi.org/10.1002/wcs.1378

14. Biehl M, Abadi F, Göpfert C, Hammer B (2020) Prototype-based classifiers in the presence of concept drift: a modelling framework. In: Vellido A, Gibert K, Angulo C, Martin Guerrero J (ed) 13th workshop on self-organizing maps and learning vector quantization, clustering and data visualization (WSOM 2019). Springer, Cham, Switzerland, Advances in Intelligent Systems and Computing, vol 976, pp 210–221

15. Bishop C (2006) Pattern recognition and machine learning. Springer, Berlin

16. Cybenko G, z1Ss (1989) QApproximations by superpositions of sigmoidal functions. Math Control Signals Syst 2(4):303–314

17. Ditzler G, Roveri M, Alippi C, Polikar R (2015) Learning in nonstationary environment: a survey. Comput Intell Mag 10(4):12–25

18. Eger S, Youssef P, Gurevych I (2018) Is it Time to Swish? Comparing deep learning activation functions across NLP tasks. In: Proceedings of the 2018 conference on empirical methods in natural language processing, association for computational linguistics, Brussels, Belgium, pp 4415–4424

19. Engel A, van den Broeck C (2001) The statistical mechanics of learning. Cambridge University Press, Cambridge

20. Ghosh A, Biehl M, Hammer B (2006) Performance analysis of LVQ algorithms: a statistical physics approach. Neural Netw 19(6–7):817–829

21. Goldt S, Mézard M, Krzakala F, Zdeborová L (2020) Modeling the influence of data structure on learning in neural networks: the hidden manifold model. Phys Rev X 10(4):041044. https://doi.org/10.1103/PhysRevX.10.041044

22. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge

23. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning: data mining, inference, and prediction. Springer, Berlin

24. van Hemmen J, Keller G, Kühn R (1987) Forgetful memories. Europhys Lett 5(7):663–668

25. Heusinger M, Raab C, Schleif FM (2020) Passive concept drift handling via momentum based robust soft learning vector quantization. In: Vellido, A and Gibert, K and Angulo, C and Martin Guerrero, J (ed) 13th workshop on self-organizing maps and learning vector quantization, clustering and data visualization (WSOM 2019), Springer, Cham, Advances in Intelligent Systems and Computing, vol 976, pp 200–209

26. Inoue M, Park H, Okada M (2003) On-line learning theory of soft committee machines with correlated hidden units—steepest gradient descent and natural gradient descent. J Phys Soc Jpn 72(4):805–810

27. Joshi J, Kulkarni P (2012) Incremental learning: areas and methods—a survey. Int J Data Min Knowl Manag Process 2(5):43–51

28. Kinouchi O, Caticha N (1993) Lower bounds on generalization errors for drifting rules. J Phys A Math Gen 26(22):6161–6172

29. Kohonen T (2001) Self-Organizing Maps. Springer Series in Information Sciences, vol 30, 2nd edn. Springer, Berlin

30. Kohonen T, Barna G, Chrisley R (1988) Statistical pattern recognition with neural network: benchmarking studies. In: Proceedings of the IEEE 2nd international conference on neural networks, San Diego, pp 61–68. IEEE

31. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th international conference on neural information processing systems (NIPS) - vol 1. Curran Association Inc., USA, pp 1097–1105

32. Losing V, Hammer B, Wersing H (2017) Incremental on-line learning: a review and of state of the art algorithms. Neurocomputing 275:1261–1274

33. Loureiro B, Gerbelot C, Cui H, Goldt S, Krzakala F, Mézard M, Zdeborová L (2021) Capturing the learning curves of generic features maps for realistic data sets with a teacher–student model. arxiv:2102.08127

34. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th ICML workshop on deep learning for audio, speech and language processing

35. Marangi C, Biehl M, Solla SA (1995) Supervised learning from clustered input examples. Euro Phys Lett 30:117–122

36. Meir R (1995) Empirical risk minimization versus maximum-likelihood estimation: a case study. Neural Comput 7(1):144–157

37. Mezard M, Nadal J, Toulouse G (1986) Solvable models of working memories. J Phys (Paris) 47(9):1457–1462

38. Nair V, Hinton G (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of 27th international conference on machine learning (ICML). Omni Press, USA, pp 807–814

39. Nova D, Estevez P (2014) A review of learning vector quantization classifiers. Neural Comput Appl 25(3–4):511–524

40. Ramachandran P, Zoph B, Le QV (2017) Searching for activation functions. ArXiv abs/1710.05941, Presented at sixth international conference on learning representations. ICLR 2018

41. Reents G, Urbanczik R (1998) Self-averaging and on-line learning. Phys Rev Lett 80(24):5445–5448

42. Riegler P, Biehl M (1995) On-line backpropagation in two-layered neural networks. J Phys A Math Gen 28:L507–L513

43. Saad D (ed) (1999) On-line learning in neural networks. Cambridge University Press, Cambridge

44. Saad D, Solla S (1995a) Exact solution for on-line learning in multilayer neural networks. Phys Rev Lett 74:4337–4340

45. Saad D, Solla S (1995b) On-Line learning in soft committee machines. Phys Rev E 52:4225–4243

46. Straat M, Biehl M (2019) On-line learning dynamics of RELU neural networks using statistical physics techniques. In:

Verleysen M (ed) 27th European symposium on artificial neural networks (ESANN 2019), Ciaco-i6doc.com, p 6

47. Straat M, Abadi F, Göpfert C, Hammer B, Biehl M (2018) Statistical mechanics of on-line learning under concept drift. Entropy 20(10), art. No. 775

48. Vellido A, Gibert K, Angulo C, Martin Guerrero J (eds) (2019) 13th workshop on self-organizing maps and learning vector quantization, clustering and data visualization (WSOM 2019), Advances in intelligent systems and computing, vol 976. Springer, Cham

49. Vicente R, Caticha N (1997) Functional optimization of online algorithms in multilayer neural networks. J Phys A Math Gen 30:L599–L605

50. Vicente R, Caticha N (1998) Statistical mechanics of on-line learning of drifting concepts: a variational approach. Mach Learn 32(2):179–201

51. Wang L, Yoon KJ (2021) Knowledge distillation and student–teacher learning for visual intelligence: a review and new outlooks. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2021.3055564, early access

52. Wang S, Minku LL, Yao X (2017) A systematic study of online class imbalance learning with concept drift. CoRR abs/1703.06683. arxiv:1703.06683

53. Watkin T, Rau A, Biehl M (1993) The statistical mechanics of learning a rule. Rev Mod Phys 65(2):499–556

54. Witoelar A, Biehl M, Hammer B (2007) Learning vector quantization: generalization ability and dynamics of competing prototypes. In: Proceedings of 6th international workshop on self-organizing-maps (WSOM 2007), Univ. Bielefeld, Germany

55. Zliobaite I, Pechenizkiy M, Gama J (2016) An overview of concept drift applications. In: Big data analysis: new algorithms for a new society. Springer, Berlin