university of groningen

university college groningen

Lifelong Machine Learning Under Concept Drift - Enforcing Forgetting by Noise Accumulation

Zhuoyun Kan

Faculty of Science and Engineering
University of Groningen

August 2019

Supervisors:

Prof. dr. Michael Biehl

Michiel Straat, Kerstin Bunte

Abstract

We consider a modelling framework for the investigation of prototype-based classifiers under concept drift processes. Specifically, We study the supervised learning model of a Learning Vector Quantization (LVQ) system in terms of two concept drift situations:virtual drift and real drift. In virtual drifts, the statistical properties of high-dimensional streaming data are time-dependent. In real drifts, however, the target task changes over time. Both class-balanced and class-imbalanced cases in training examples are considered under real drift. Previous studies demonstrate that weight decay as an explicit mechanism of forgetting has limitations since it imposes a significant restriction on the flexibility of learning systems. We present an alternative to the explicit weight decay: the application of additive noise in each training step, which imposes a forgetting effect without restricting the flexibility of the LVQ system. Our results show that the noise accumulation overcomes the limitations and can outperform weight decay under virtual drift. However, under real drift processes, the additive noise as an alternative mechanism of forgetting does not improve the performance, which therefore requires modification to the model in the future.

Contents

1 Introduction **1**

1.1 Incremental online learning 2

1.2 Concept drift 3

1.3 Outline 4

2 Related work **6**

2.1 Incremental learning methods 6

2.2 Concept drift handling techniques 7

2.3 Statistical mechanics 9

3 Models and theoretical analysis **10**

3.1 Learning vector quantization 10

3.1.1 Nearest prototype classification 11

3.1.2 Online LVQ1 algorithm 12

3.1.3 Clustered data 13

3.2 The dynamics of LVQ in stationary environment 13

3.2.1 Order parameters and recursions 14

3.2.2 Averages over the random examples 14

3.2.3 Self-averaging properties 15

3.2.4 Continuous learning time and ODE 16

3.2.5 Generalization error 16

3.2.6 Learning curves 17

3.3 LVQ learning dynamics under concept drift 17

3.3.1 Virtual drift 17

3.3.2 Real drift 19

3.4 Forgetting mechanisms 20
3.4.1 Weight decay 20
3.4.2 Noise accumulation 21

4 Experiments and results **25**
4.1 Virtual drift 26
4.1.1 Gradual drift 26
4.1.2 Sudden drift 28
4.1.3 Periodic drift 31
4.2 Real drift 33
4.2.1 Balanced case 33
4.2.2 Imbalanced case 34

5 Conclusion and discussion **36**

6 Outlook **38**

1 Introduction

Topics of learning in a data stream environment with concept drifts have attracted considerable attention recently in the machine learning community. Terms like *lifelong learning* or *incremental learning* have been frequently coined in contexts like [1–3].

In a standard machine learning set-up, the adaptation of the learning system is based on a given, stationary set of example data which is assumed to be representative of the considered task. However, in many practical contexts, the learning process continuously involves new data and statistical properties of the data and/or the actual task change while the system is trained. Hence, it is unwise and can be problematic to separate training phase and working phase, which is known as the classical approach of supervised learning [1, 4]. Consequently, designing appropriate methods of training in the non-stationary environments has become a challenging research area.

The development of modelling frameworks becomes instrumental in the course to gain general insights into various learning systems and achieve a sufficient theoretical understanding of relevant phenomenon. Analytic approaches that come from statistical

physics have played an important role in this field [5–9].

1.1 Incremental online learning

The ability to learn form a sequential data is essential to the nowadays machine learning. The traditional paradigm on machine learning is to learn from a given set of training data. Given a dataset $((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$ with $\vec{x}_i$ and y_i being the ith set of input and output, these data are trained in the batch mode where all sets of $(\vec{x}_i, y_i)$, for $1 \leq i \leq n$ are used at the same time without the consideration of the order. It however can be problematic when there is a new training set, as it leads to the problem so-called catastrophic interference [10, 11]. It is the situation where the previously learned information has been completely forgotten upon the newly learned information.

Given a data steam environment, the approach of online learning is becoming important in the community of machine learning. For some situations, for example the autonomous driving and robotics [12, 13], the autonomous reaction to the streaming data is even extremely relevant. In online learning, the temporal information of the dataset $(\vec{x}_i, \vec{y}_i)$ are also taken into account.

Incremental learning is a type of online learning in which it refers to the case with limited memory resources [14]. It involves the learning model to adapt the sequential data set without forgetting the existing knowledge. It is important because of the two main reasons:

- The properties of data are large-scale and dynamic, and thus the data mining cannot be done in one-time. The knowledge has to be built up through a successive learning over time just like the building-block.
- Incremental learning mimics the way of learning new knowledge of human intelligence. The biological intelligence is an example of incremental learner which has the ability to learn from a flow of data.

In short, the traditional learning approach is like the learning system of an irrational school, where students heavily focuses on the syllabus and their main goal is to pass the examination in a semester. In contrast, incremental learning requires the digestion of knowledge such that they becomes durable. Given this example, it is clear that incremental learning is more complicated than the traditional way since some additional algorithms are required for data processing, converting and prioritizing [14].

1.2 Concept drift

The importance of incremental learning for the efficient training performance leads further investigation of learning in non-stationary environments. In many practical applications, training examples are available in the form of data streams, such as stock forecast, weather prediction and recommendation systems [15]. Moreover, the distributions of training data from data streams is not always stable and the target classification may change over time. Such change in the underlying function is known as *concept drift*, which can attenuate the learning effect to a large degree. Consequently, learning from data streams with concept drift has become a challenging part of incremental learning.

According to the Bayesian Decision Theory [16], two major types of concept drifts have been frequently discussed in the literature: the *real drift* and the *virtual drift* [2, 17, 18]. In real drifts, the target classification is time-dependent, so that the true boundary between target classes changes with time. In other words, the previously learned model can become unreliable because of the drift. In virtual drifts, the statistic properties of the training examples are time-dependent, while the actual target task remains unaffected. Frequently, real drifts happen along with additional virtual drifts in real-world applications.

It is also common to classify different types of concept drifts based on the speed of the drift. The speeds occurs mainly in three main ways: sudden, gradual and oscillated. In this framework, we study drift scenarios with different speed types and discuss the relative effects with different learning strategies.

Moreover, concept drifts can result in *class imbalanced* problems in the course of training, which is another difficult but commonly seen problem in the community of machine learning. When class imbalance occurs together with drifting data streams, one class can be severely underrepresented in the dataset so that it becomes hard to achieve a plausible training performance. The problem is further exacerbated as the distributions of sources change with time, which may seriously impair the learning ability of the system. Some approaches dealing with class imbalanced learning with concept drift can be found in [17, 19].

1.3 Outline

In this project, we consider a modelling framework for the investigation of prototype-based classifiers in non-stationary environments. More specifically, we study the training of a Learning Vector Quantization (LVQ) system from a stream of high-dimensional, clustered data. The classifier is adapted in terms of a standard winner-takes-all prescription known as LVQ1 [20].

First, we consider the LVQ1 learning from high-dimensional streaming data with virtual drifts where statistical properties of the clustered input data are assumed to change on the time scale defined by the training process. We here focus on time-dependent class bias in the training data. Recent results demonstrate that while basic LVQ algorithms are suitable for the training in time-dependent environments, weight decay as an explicit and popular mechanism of forgetting does not improve their performance under the considered drift processes [21]. This is due to the fact that weight decay imposes a significant restriction on the flexibility of LVQ systems by enforcing small norms of the prototype vectors, which also confines decision boundaries between classes to remain close to the origin of feature pace.

In this modelling framework, we consider an alternative to the explicit weight decay: the application of additive noise to the prototype positions in each learning step. The accumulation of the noise imposes a *forgetting* effect without restricting the flexibility of the LVQ system in principle. Monte Carlo simulations of LVQ training in the described settings is performed, initially using artificial high-dimensional clustered data.

Second, we study the adaptive learning ability of LVQ systems to real drifts, where target tasks (cluster centers) change over time but the statistical properties of the clustered examples are unchanged. In the recent paper [22], results show that weight decay can improve the quasi-stationary performance for balanced classes. We here move emphasis on imbalanced classes for further investigation and compare results from LVQ weight decay and noise accumulation to see the potential learning abilities.

Finally, the structure of the remainder of this paper is the following: The literature review has been given in Chapter 2. We summarize relevant studies on the approaches used in incremental learning, concept drift handling and how the statistical mechanics methods being used in LVQ system. In Chapter 3 we revisit the models and the theoretical basis of LVQ training dynamics in stationary environment. Next, the proposed forgetting mechanism has been described in detail. In Chapter 4 we present our results from virtual drifts and real drifts. We conclude with a discussion and outlook in Chapter 5

2
Related work

In this chapter we briefly describe the relevant works on the incremental learning and concept drift. Next, we review the methods of Statistical Mechanics being used in LVQ learning systems.

2.1 Incremental learning methods

Many algorithms for incremental learning have been proposed in the literature. Most of them are built based on well-existing batch methods, however an additional forgetting mechanism has been incorporated on top of the batch methods. Some reviews have provided an overview of commonly used algorithms for incremental learning, for example [3].

One popular algorithm is Learn++, which aims for incremental training of supervised neural networks from Polikar *et al.* [23]. Then Huang *et al.* introduced Incremental Extreme Learning Machine, which has been applied in feedforward neural networks with random hidden nodes [24]. Other methods includes Naive Bayes [25], Online Random Forest [26], etc. In this study, we focus on the online learning implemented on the so-

called Learning Vector Quantization (LVQ) scheme. The concept of LVQ scheme will be introduced in detail in Chapter 3.

2.2 Concept drift handling techniques

Concept drift has been addressed as one prominent challenge in incremental learning due to the changing statistic properties in data. A large number of techniques and mechanisms have been put forward to overcome problems caused by concept drift. These approaches can be briefly grouped into following families [2, 27, 28]:

(A) Online learners

Online learners are able to dynamically adapt to the changing environment when new data is available, so that they continuously update the current model with the most recent examples. Additionally, they have the capability to process high-speed data stream by employing variants of the above mentioned incremental online learning methods. Such online algorithms include the Concept-Adapting Very Fast Decision Tree (CVFDT) algorithm [29], Just-in-time k-nearest neighbors (kNN) algorithm by Alippi and Roveri [30] and fuzzy ARTMAP (adaptive resonance theorymap) by Andrés-Andrés et al. with a pruning strategy [31].

(B) Adaptation methods

Adaptation approach is another popular strategy of dealing with concept drift. This approach manages adaptation of the predictive model by modifying the training set, which frequently incorporate *forgetting mechanisms* [27, 32, 33]. When dealing with incoming data generated from processes with unknown dynamics, a forgetting mechanism is commonly employed to forget or discard old knowledge which is no longer relevant to the learning system. The adaptation methods can be organized into three groups: sliding window-based methods, weighting schemes and random sampling techniques.

1. *Window-based method*, also known as *abrupt forgetting*, is one of the simplest approaches to deal with concept drift. At each time step, a predefined window is given containing the M latest input examples to retrain/update the classifier. Labeled examples out of the window are abruptly forgotten or discarded in the course of training. Consequently,

the main question is how to determine the exact window size M. If M is too small, the learning system is allowed to focus on only few emerging examples, which may not be representative to the long-term system and thus reduce the accuracy. A large window size, on the other hand, can result in a slow reaction to the drift and a mix of examples representing different context. In order to overcome the shortcomings of fixed-size windows, some extended strategies have been presented which dynamically adjust the window size. Reviews of these methods see, for example, [18, 34].

2. *Weighting scheme* is a full-memory strategy, which consider all available examples in the course of training. Instead of abruptly discarding training instances, in window-based methods no examples are completely discarded from the training system. Once new examples arrive, different weights are assigned to all the training examples according to their age or relevance. These methods gradually decrease the importance of old examples by setting a forgetting factor or decay factor. Depending on how the factor changes over time, the *gradual-forgetting* weighting mechanisms can have various decay functions ranging from linear decay by Koychev [35] to exponential decay by Klinkenberg [36].

Moreover, It is notable that R. Elwell and R. Polikar have proposed an advanced algorithm Learn++.NSE in [18], which allows the system to temporarily forget irrelevant knowledge and recall the information when it becomes relevant again.

3. *Random sampling techniques* provide an alternative to the above mentioned methods. These techniques aim to randomly select a subset of examples (without replacement) from a data stream such that the included examples are able to summarize the underlying characteristics of the data stream. One of the well-known algorithms is the reservoir sampling by Vitter based on a rejection-acceptance technique [37].

(C) Ensemble-based approaches

The last group of commonly used concept drift handling techniques is contributed by ensemble-based algorithms, which consist of a set of classifiers. Ensemble-based approaches present a nature fit to learn in the changing environments due to several advantages: high accuracy, high flexibility and nature forgetting mechanisms [2]. Some popular

examples have been introduced in the literature, such as Dynamic Weighted Majority (DWM) algorithm [38] and online non-stationary boosting algorithm (ONSBoost) [39].

2.3 Statistical mechanics

Instead of empirical studies of dealing with problems in non-stationary environments, we focus on a mathematical modelling framework which can abstract from various adaptive systems. In this study, we present a modelling framework for LVQ online learning under concept drift from high-dimensional examples with the help of statistical physics approaches.

Methods borrowed from statistical physics make possible to mathematically describe and analyze training dynamics and the corresponding behavior of a variety of learning systems. These methods are based on the assumption that a sequence of independent examples are randomly generated in N-dimensional feature space. Furthermore, considerations of the thermodynamic limit $N \to \infty$ successfully provide mathematical description of learning curves in terms of ordinary differential equations (ODE). The conceptual simplicity of the approach and its applicability facilitate better theoretical understanding of learning systems.

Successful work using methods of Statistical Mechanics with respect to supervised learning in perceptrons and multilayered neural networks can be found in [5–9]. Also, the dynamics of unsupervised learning has been studied, for example, the unsupervised prototype-based competitive learning in [40, 41]. Specifically, the dynamics of LVQ learning has been studied in [42–44]. Recently, an extended study presents the statistical mechanisms of online learning under concept drift for both classification and regression problems [22].

3

Models and theoretical analysis

In this chapter we introduce the concept of Learning Vector Quantization (LVQ) scheme for prototype-based classification problems with emphasis on the basic LVQ1 algorithm. Then the learning dynamics of LVQ systems in both stationary and non-stationary environment is included based on the literature from [21, 22, 43]. Moreover, two forgetting mechanisms are presented in the last part of this chapter. We first introduce the so-called *weight decay* as an element of explicit forgetting. Afterwards, we propose *noise accumulation* as an alternative approach of modelling forgetting memories.

3.1 Learning vector quantization

Learning vector quantization (LVQ) is a prototype-based classification algorithm which was first introduced by Kohonen [20]. The family of LVQ algorithms has been widely applied to many practical classification problems in different fields [45–47]. The popularity of this approach is due to a number of distinctive features: the LVQ procedure is intuitive, easy to implement and is a natural tool for classifying multi-class problems. Moreover, the family of LVQ algorithms takes advantages regarding robustness, storage memory and

computational effort compared to many other machine learning methods. It facilitates straightforward interpretations without the black-box character since prototype vectors are defined in the same feature space as input data and can directly parameterize classifiers [43, 48].

3.1.1 Nearest prototype classification

In LVQ approaches, classifiers are represented by a set of labelled prototypes, which indicate different classes in the same vector space of input examples. In general, multiple prototypes can be applied to represent each class. In this contribution, however, we investigate simple cases with only two prototype vectors $\vec{w}_i \in \mathbb{R}^N$ in binary classification problems, where N is the dimension of input vectors and $i \in \{1, 2\}$ indicates the class of data.

The modelling system is parameterized by a so-called *Nearest Prototype Classification* (NPC) scheme. Each prototype $\vec{w}_i$ with its class label represents a receptive field in the feature space. The class of an unlabelled example is determined by its nearest prototype based on a chosen distance measure $d(\vec{w}, \vec{\xi})$, frequently Euclidean distance. Therefore, given any input example, the class is assigned to 1 if $d(\vec{w}_1, \vec{\xi}) < d(\vec{w}_2, \vec{\xi})$, and to class 2 otherwise. The idea of assigning an input data to the class of the closest prototype is closely related to k-nearest neighborhood classification (k-NN) [49] where k equals to 1 in particular.

A variety of distance measures have been proposed in the literature of LVQ methods, aiming at enhancing the flexibility of the approach and optimizing the hypothesis margin [48, 50, 51]. These extensions include adaptive distance measures in *relevance learning*, which adopt metric adaptation techniques to learn discriminative distance measures from the training data. In this modelling framework, however, we restrict the following analysis to the most commonly-used squared Euclidean distance:

$$d(\vec{w}, \vec{\xi}) = (\vec{w} - \vec{\xi})^2. \tag{3.1}$$

3.1.2 Online LVQ1 algorithm

The original version of the learning vector quantization algorithm is known as LVQ1 [20, 45]. In this study, we consider that a sequence of single example data $\{\vec{\xi}^{\mu}, \sigma^{\mu}\}$ is introduced to the LVQ system in the online LVQ1 learning process [9]: At any given time step $\mu = 1, 2, ...$, the feature vector $\vec{\xi}^{\mu}$ is given with its class label $\sigma^{\mu} = \{1, 2\}$.

In the training phase, the NPC procedure follows a *Winner-Takes-All* (WTA) scheme so that only the closest prototype for the current example will be updated. For each input instance $\vec{\xi}^{\mu}$ with label σ^{μ}, the closest prototype with the same class label as $\vec{\xi}^{\mu}$ will be attracted by $\vec{\xi}^{\mu}$. On the contrary, if the closest prototype carries a different class label, it will be pushed away from $\vec{\xi}^{\mu}$. The objective of this learning procedure is to adapt prototypes automatically in the feature space so that the highest classification accuracy on novel data can be achieved.

The incremental LVQ1 updates in the online training process are of the general form [43, 44]

$$\vec{w}_i^{\mu} = \vec{w}_i^{\mu-1} + \Delta\vec{w}_i^{\mu} \quad \text{with} \quad \Delta\vec{w}_i^{\mu} = \frac{\eta_{\omega}}{N} f_i\left[d_1^{\mu}, d_2^{\mu}, \sigma^{\mu}\right](\vec{\xi}^{\mu} - \vec{w}_i^{\mu-1}) \tag{3.2}$$

where $d_i^{\mu} = d(\vec{w}_i^{\mu-1}, \vec{\xi}^{\mu})$ and η_{ω} denotes the constant learning rate scaled with the input dimension N. The function $f_i[...]$ is the *modulation function* which specifies the algorithm. It typically depends on the relevant distances of the input examples from the current prototypes and on the labels of the data points and the prototypes. More specific, the updating process for only two competing prototypes according to Eq.(3.2) is given formally by

$$\begin{aligned} f_i\left[d_1^{\mu}, d_2^{\mu}, \sigma^{\mu}\right] &= \Theta\big(d_{\widehat{i}}^{\mu} - d_i^{\mu}\big)\Psi(i, \sigma^{\mu}), \quad \text{where} \quad \widehat{i} = \begin{cases} 2, & \text{if } i = 1 \\ 1, & \text{if } i = 2, \end{cases} \\ \Theta(x) &= \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } else, \end{cases} \quad \text{and} \quad \Psi(i, \sigma) = \begin{cases} +1, & \text{if } i = \sigma \\ -1, & \text{if } else. \end{cases} \end{aligned} \tag{3.3}$$

The Heaviside function $\Theta(...)$ picks out the winning prototype and the factor $\Psi(i, \sigma^{\mu})$ identifies the direction (sign) of the update [21, 22].

3.1.3 Clustered data

In this modelling framework, we consider a stream of random input data points $\vec{\xi} \in \mathbb{R}^N$ defined by a given cluster structure. All the data are therefore generated independently according to a bi-modal distribution of the form [43]

$$P(\vec{\xi}) = \sum_{\sigma=1,2} p_\sigma \, P(\vec{\xi} \mid \sigma) \quad \text{with} \quad P(\vec{\xi} \mid \sigma) = \frac{1}{(2\pi\nu_\sigma)^{N/2}} \, \exp\left[- \frac{(\vec{\xi} - \lambda \vec{B}_\sigma)^2}{2\nu_\sigma} \right]. \quad (3.4)$$

The class-conditional densities $P(\vec{\xi}|\sigma = 1, 2)$ correspond to two isotropic, spherical Gaussian clusters with variance ν_σ and means $\lambda\vec{B}_\sigma$. Prior cluster weights or probabilities of the two clusters are denoted by p_σ satisfying $p_1 + p_2 = 1$. Besides, we assume that the vectors $\vec{B}_\sigma$ are normalized with $\vec{B}_1^2 = \vec{B}_2^2 = 1$ and orthogonal with $\vec{B}_1^2 \cdot \vec{B}_2^2 = 0$. Hence, the target classes are not linearly separable because of the overlap of the clusters.

We denote conditional averages over $P(\vec{\xi} \mid \sigma)$ as $\langle \cdots \rangle_\sigma$ and define mean values of the form $\langle \cdots \rangle = \sum_{\sigma=1,2} p_\sigma \langle \cdots \rangle_\sigma$ for the full density (3.4) in the following. For example, for a cluster σ with the corresponding variance ν_σ and input instance $\vec{\xi}$, the conditional and full averages are given as

$$\begin{aligned} &\langle \vec{\xi} \rangle_\sigma = \lambda \vec{B}_\sigma, \langle \vec{\xi}^2 \rangle_\sigma = \nu_\lambda N + \lambda^2, \\ &\langle \vec{\xi}^2 \rangle = (p_1\nu_1 + p_2\nu_2)N + \lambda^2. \end{aligned} \quad (3.5)$$

3.2 The dynamics of LVQ in stationary environment

In this section we follow the presentation in [43], which has successfully applied the theory of online learning to describe the learning dynamics and generalization behaviour of LVQ algorithms in stationary environments. We refer the reader to the original publication [43] for details. The extensions to non-stationary environments are discussed in the next section.

The following mathematical analysis is based on the consideration of the *thermodynamic limit* $N \to \infty$. We consider high dimensional data $\vec{\xi}$, adaptive prototypes $\vec{w}_{1,2}$ and cluster centers $\vec{B}_{1,2}$. One of the immediate simplifying consequences, for instance, is that the term λ^2 in Eq.(3.5) can be neglected. Furthermore, the limit $N \to \infty$ facilitates the following key steps with concepts from statistical physics to obtain the exact mathematical

description of the learning dynamics in terms of ordinary differential equations (ODE).

3.2.1 Order parameters and recursions

Given the large number of degrees of freedom, i.e., components of the prototype vectors, the learning system can be characterized in terms of only a few quantities. After a number μ of data points has been presented to the system, the system is characterized by prototypes $\vec{w}_1^{\mu}$, $\vec{w}_2^{\mu}$ and cluster centers $\vec{B}_1$, $\vec{B}_2$. Hence, we describe the particular mathematical structure of the model in the following set of order parameters:

$$R_{i\sigma}^{\mu} = \vec{w}_i^{\mu} \cdot \vec{B}_{\sigma} \quad \text{and} \quad Q_{ij}^{\mu} = \vec{w}_i^{\mu} \cdot \vec{w}_j^{\mu}, \tag{3.6}$$

where $\sigma, i, j \in \{1, 2\}$. The subscript μ denotes the μth data point. Apparently, self-overlaps Q_{11}^{μ}, Q_{22}^{μ} relate to the norm of the prototypes and mutual overlap $Q_{12}^{\mu} = Q_{21}^{\mu}$ tells the relative angels between them. Four quantities $R_{i\sigma}^{\mu}$ specify their projections into the subspace characterized by cluster centers $\{\vec{B}_1, \vec{B}_2\}$ respectively.

In order to obtain the change of the above order parameters upon each presentation of example $\vec{\xi}^{\mu}$, we can directly derive the recursion relations from the generic learning algorithm Eq.(3.2):

$$\begin{aligned}
\frac{R_{i\sigma}^{\mu} - R_{i\sigma}^{\mu-1}}{1/N} &= \eta_{\omega} f_i(\vec{B}_{\sigma} \cdot \vec{\xi}^{\mu} - R_{i\sigma}^{\mu-1}) \\
\frac{Q_{ij}^{\mu} - Q_{ij}^{\mu-1}}{1/N} &= \eta_{\omega} f_i(\vec{w}_j^{\mu-1} \cdot \vec{\xi}^{\mu} - Q_{ij}^{\mu-1}) + \eta_{\omega} f_j(\vec{w}_i^{\mu-1} \cdot \vec{\xi}^{\mu} - Q_{ij}^{\mu-1}) \\
&\quad + \eta_{\omega}^2 f_i f_j (\vec{\xi}^{\mu})^2 / N.
\end{aligned} \tag{3.7}$$

Note that the modulation functions f_i, f_j are used here as shorthand, omitting the arguments.

3.2.2 Averages over the random examples

we can perform averages over the independent random examples by employing the central limit theorem (CLT). Obviously, the current example $\vec{\xi}^{\mu}$ occurs on the right hand side of

Eq.(3.7) only through its length $|\vec{\xi}^{\mu}|^2$ and the projections

$$h_i^{\mu} = \vec{w_i}^{\mu-1} \cdot \vec{\xi}^{\mu} \quad \text{and} \quad b_{\sigma}^{\mu} = \vec{B}_{\sigma} \cdot \vec{\xi}^{\mu}. \tag{3.8}$$

Since the inner products are derived from the random sequence of independent examples, the CLT implies that the quantities in Eq.(3.8) are correlated to Gaussian quantities for large N. Their joint density $P(h_1^{\mu}, h_2^{\mu}, b_1^{\mu}, b_2^{\mu})$ is therefore defined by first and second moments. For an instance from the given cluster σ, the corresponding conditional moments read

$$\begin{aligned}
&\langle h_i^{\mu} \rangle_{\sigma} = \lambda R_{i\sigma}^{\mu-1}, \quad \langle b_m^{\mu} \rangle_{\sigma} = \lambda \delta_{im}, \\
&\langle h_i^{\mu} h_j^{\mu} \rangle_{\sigma} - \langle h_i^{\mu} \rangle_{\sigma} \langle h_j^{\mu} \rangle_{\sigma} = \nu_{\sigma} Q_{ij}^{\mu-1}, \\
&\langle h_i^{\mu} b_m^{\mu} \rangle_{\sigma} - \langle h_i^{\mu} \rangle_{\sigma} \langle b_m^{\mu} \rangle_{\sigma} = \nu_{\sigma} R_{im}^{\mu-1}, \\
&\langle b_n^{\mu} b_m^{\mu} \rangle_{\sigma} - \langle b_n^{\mu} \rangle_{\sigma} \langle b_m^{\mu} \rangle_{\sigma} = \nu_{\sigma} \delta_{nm}
\end{aligned} \tag{3.9}$$

where $i, j, m, n, \sigma \in \{1, 2\}$ and $\delta...$ is the Kronecker-Delta. Hence, the joint density of $h_{1,2}^{\mu}$ and $b_{1,2}^{\mu}$ is fully determined by the model parameters $\lambda, p_{1,2}, \nu_{1,2}$ and the above defined order parameters in the previous time step.

This important result makes it possible to perform an average of the recursion relations, Eq.(3.7), over the latest data point in terms of Gaussian integrals. Moreover, the result on the right hand sides can be written in closed form in $\{Q_{ij}^{\mu-1}, R_{im}^{\mu-1}\}$. The full form of the averaged recursions for several LVQ algorithms is given in [43].

3.2.3 Self-averaging properties

Self-averaging properties of the order parameters enable us to describe them in terms of their mean values due to the thermodynamic limit: Fluctuations of the stochastic dynamics decrease with the increasing N, which can be ignored in the limit $N \to \infty$.

This concept has been borrowed from the statistical physics of disordered materials and widely applied in the study of learning processes [5–7]. For a detailed mathematical discussion of the self-averaging in online learning processes, we refer the reader to the work in [52]. Accordingly, we can interpret the averaged recursion relations, Eq.(3.7), directly as deterministic recursions for the means of order parameters $\{Q_{ij}^{\mu}, R_{im}^{\mu}\}$. The values coincide with their actual values in the thermodynamic limit.

3.2.4 Continuous learning time and ODE

One important simplifying consequences of the thermodynamic limit is that, for $N \to \infty$, we can describe the scale on the left hand sides of Eq.(3.7) as derivatives with respect of the continuous learning time

$$\alpha = \mu/N. \tag{3.10}$$

This scaling adjustment of μ corresponds to the expectation that the number of the training examples required for successful learning should be proportional to the number of degrees of freedom in the learning system. Hence, the resulting set of coupled ODE obtained from Eq.(3.7) is of the following form:

$$\begin{aligned} \Big[\frac{dR_{im}}{d\alpha}\Big] &= \eta_\omega \left(\langle b_m f_i - R_{im}\langle f_i\rangle\right) \\ \Big[\frac{dQ_{ij}}{d\alpha}\Big] &= \eta_\omega \left(\langle h_i f_j + h_j f_i) - Q_{ij}\langle f_i + f_j\rangle\right) \\ &\quad + \eta_\omega^2 \sum_{\sigma=1,2} \nu_\sigma p_\sigma (\langle f_i f_j\rangle)_\sigma. \end{aligned} \tag{3.11}$$

Clearly, these ODE describe the learning from a density, which are specified by the model parameters ν_σ, p_σ and η_ω. For the detailed analysis of ODE for the specific modulation functions, we refer the reader to the original literature [43, 44].

3.2.5 Generalization error

After the training phase, the success of learning can be quantified in terms of generalization error. It measures the probability of misclassifying a novel, random input instance, which did not appear in the training phase.

In this model, we consider the generalization error for data from two clusters $\sigma = 1, 2$ in Eq.(3.4), which is the total misclassification rate, written as:

$$\varepsilon = p_1\, \varepsilon_1 + p_2\, \varepsilon_2 \quad \text{with} \quad \varepsilon_i = \left\langle\, \Theta\left(d_i - d_{\widehat{i}}\right) \right\rangle_i. \tag{3.12}$$

Here ε_i is the class-specific generalization error. It can be expressed as an explicit function

of the order parameters [43]:

$$\varepsilon_i = \Phi\left(\frac{Q_{ii} - Q_{\widehat{i}\widehat{i}} - 2\lambda(R_{ii} - R_{\widehat{i}\widehat{i}})}{2\sqrt{\sigma_i}\sqrt{Q_{11} - 2Q_{12} + Q_{22}}}\right) \quad \text{with} \quad \Phi(z) = \int_{-\infty}^{z} dx \frac{\exp\left(-x^2/2\right)}{\sqrt{2\pi}}. \quad (3.13)$$

3.2.6 Learning curves

Once working out the ODE for a modulation function, it can be numerically integrated. Hence, given a specific modulation function, model parameters $\{\eta_\omega, \nu, p_{1,2}, \lambda\}$ and initial conditions $\{R_{i\sigma}(0), Q_{ij}(0)\}$, the numerical integration of the ODE yields the temporal evolution of order parameters $\{R_{i\sigma}(\alpha), Q_{ij}(\alpha)\}$ in the course of training. After exploiting the self-averaging properties of order parameters, we can obtain the typical learning curves $\varepsilon(\alpha)$, i.e., the generalization error after online training with (αN) random input examples by inserting $\{R_{i\sigma}(\alpha), Q_{ij}(\alpha)\}$.

3.3 LVQ learning dynamics under concept drift

In the previous section, we have summarized the learning dynamics in a stationary environment, where cluster densities (weights) and target classes of the form (3.4) do not change in the course of training. In this part, we discuss the learning effects in non-stationary environments with concept drift. Both virtual drifts and real drifts are considered in this modelling framework, respectively.

3.3.1 Virtual drift

A virtual drift happens if the statistical property of the incoming data changes without affecting the posterior probability of classes, i.e., the actual target classification remains the same. Different interpretations of virtual drifts are widely discussed in the literature [17, 27, 53]. One virtual drift refers to changes in the incoming data distribution, which can be achieved in our framework by setting time-dependent parameters of the input density, e.g., $\nu_\sigma(\alpha)$ and $\lambda(\alpha)$ in Eq.(3.4).

In this study we focus on the virtual drift presented in the prior probabilities (weights) of the incoming examples. Given the prior weights of each cluster, $p_1 = 1 - p_2$, the priors are set to change over time in the course of training. Hence, a varying fraction of examples

represents each of the classes in the sequence of the incoming data. In practical use, the time-dependent class bias can significantly complicates the training phase and results in inferior performance.

(A) Drift in training data

Here we consider the case that virtual drifts happen only in the training data, thus the target classification can be characterized by a fixed *reference density* of data. For simplicity, we investigate equal prior weights, $p_1 = p_2 = 0.5$, in a symmetric reference density with the same cluster variance, i.e., $\nu_1 = \nu_2$ in Eq.(3.4). On the other hand, the probability of the observed training data $p_\sigma(\alpha)$ is assumed to change over time.

Give the order parameters of the learning system, the corresponding *reference generalization error* can be written as

$$\varepsilon_{ref}(\alpha) = \frac{1}{2}\left(\varepsilon_1 + \varepsilon_2\right) \tag{3.14}$$

by setting $p_1 = p_2 = 0.5$ in Eq.(3.12). The order parameters $R_{i\sigma}(\alpha)$ and $Q_{ij}(\alpha)$, however, are employed with time-dependent $p_1(\alpha) = 1 - p_2(\alpha)$ in the training phase.

(B) Drift in training and test data

In this interpretation, we make the assumption that the time-dependent $p_\sigma(\alpha)$ involve both training and test data, which are in fact identical. Hence, both training and test data share the same prior weights over time. The change of the statistical properties inevitably gives rise to the modification of the ODE system. We refer readers to [43] for more details on the modified ODE for the training dynamics.

The learning system is designed to track the drifting behavior in the data and the corresponding *tracking generalization error* is formally given by [21]

$$\varepsilon_{track}(\alpha) = p_1(\alpha)\,\varepsilon_1 + p_2(\alpha)\,\varepsilon_2. \tag{3.15}$$

Note that the *tracking error* is defined by Eq.(3.12) as well, but here we focus on the explicitly time-dependent $p_\sigma(\alpha)$ in both training and test data.

3.3.2 Real drift

Different from virtual drifts, a real drift refers to the term that the target classification changes over time. In the presented modelling framework, a real drift can be achieved by displacing the cluster centers $\vec{B}_\sigma$ in the N-dimensional feature space. Specifically, we consider to employ a random *diffusion* of the vectors $\vec{B}_\sigma(\mu)$ in N dimensions. Hence, the norms of the cluster centers remain unchanged and the centers are still orthogonal to each other. This ensures that the overlap between two clusters still exists after the drifting process.

The drift is applied before a new example is generated in the training step. At any given time step μ, the random vectors $\vec{B}_\sigma(\mu)$ are generated according to the following conditions [22]

$$\begin{aligned} \vec{B}_1(\mu)\cdot\vec{B}_1(\mu-1) = \vec{B}_2(\mu)\cdot\vec{B}_2(\mu-1) = (1-\frac{\delta}{N}) \\ \vec{B}_1(\mu)\cdot\vec{B}_2(\mu) = 0 \;\text{ and }\; |\vec{B}_1(\mu)|^2 = |\vec{B}_2(\mu)|^2 = 1, \end{aligned} \tag{3.16}$$

where δ controls the strength of the drift. Hence, the positions of the old and new cluster center are slightly different in each learning step and examples are generated from the new clusters. Furthermore, the centers tend to *lose memories* of their previous positions in terms of the defined continuous learning time $\alpha = \mu/N$ according to

$$\vec{B}_\sigma(\alpha_1)\cdot\vec{B}_\sigma(\alpha_0) = exp[-\delta\,(\alpha_1-\alpha_0)] \quad \text{with} \quad \alpha_1 > \alpha_0. \tag{3.17}$$

Therefore, the effect of such a real drift can be explained in a more straightforward way: For any given prototype vector $\vec{w}_i \in \mathbb{R}^N$, the overlaps between cluster centers and prototypes can be achieved by

$$\vec{w}_i\cdot\vec{B}_\sigma(\mu) = \big(1-\frac{\delta}{N}\big)\big[\vec{w}_i\cdot\vec{B}_\sigma(\mu-1)\big] \quad \text{where} \quad i,\sigma\in\{1,2\}. \tag{3.18}$$

Consequently, the overlaps continuously decrease during the training phase due to the drift process. Obviously, the learning effect tends to deteriorate compared with the stationary case with fixed cluster centers. The corresponding generalization error is given by Eq.(3.12).

3.4 Forgetting mechanisms

Forgetting mechanism is a popular method used to overcome the difficulties brought by concept drifts. It simply forgets or discards old information which is no longer relevant to the learning system by reducing the impacts of old instances. In the end, it potentially improves the performance of the trained system in the presence of concept drifts.

Also note that in this project, we adopt the idea of the given ODE and the corresponding analysis introduced in section 3.2 for noise accumulation to evaluate the learning behavior and the generalization abilities. The specific modified ODE for noise accumulation and its numerical integration will be studied in forthcoming projects.

3.4.1 Weight decay

The so-called *weight decay* is frequently employed in machine learning systems as a term of *regularization* in order to reduce the risk of *over-fitting* in regression or classification problems. It has also been studied as the modelling of *forgetful memories* in attractor neural networks such as [54] in which the authors solved analytically a large class of forgetful memories.

Here we concern weight decay as an explicit mechanism of *forgetting*. The aim is to improve the learning behavior of LVQ systems in the presence of concept drifts. To this end, we consider the multiplication of all adaptive prototype vectors $\vec{w}_i$ by a decay factor $(1-\gamma/N)$ before applying the generic learning step given by $\Delta\vec{w}_j^{\mu}$ in Eq.(3.2). Thus, the modified learning step can be written as

$$\vec{w}_i^{\mu} = (1-\frac{\gamma}{N})\ \vec{w}_i^{\mu-1} + \Delta\vec{w}_i^{\mu}. \tag{3.19}$$

Obviously, the multiplications with the decay factor $(1-\gamma/N)$ accumulate during the training phase. As a consequence, weight decay weakens the influence of old instances and imposes an increased effect of the most recent instances as compared to *earlier* instances.

Earlier study from [21] shows that the dominant effect of weight decay is the regularization of the LVQ system by reducing the norms Q_{11} and Q_{22} of the prototype vectors. As a result, it restricts the flexibility of learning systems, which makes the classifier less

flexible to reflect class bias.

3.4.2 Noise accumulation

In this modelling framework, we propose an alternative to the explicit weight decay: the use of additive noise in each learning step, which also imposes a forgetting effect by noise accumulation. The aim is to produce a forgetting effect without restricting the flexibility of the LVQ system.

As mentioned above, after applying the LVQ1 learning step in Eq.(3.2), the old prototypes are updated according to the rules. Then we introduce a random noise on these updated prototypes given as

$$\begin{aligned} \vec{w}_i^{\mu} &= \vec{w}_i^{\mu-1} + \Delta\vec{w}_i^{\mu} \\ \tilde{w}_i^{\mu} &= \vec{w}_i^{\mu} + \vec{n}_i^{\mu}. \end{aligned} \tag{3.20}$$

Here $\vec{w}_i$ represents the updated prototype after the generic learning step and $\tilde{w}_i$ is the latest prototype with additive noise $\vec{n}_i$. As a consequence, the prototypes shift a bit to a new position and lose memories on the previous position. This mechanism enforces the learning system to weaken the impacts of old examples by introducing the additive noise during the training phase. Hence it can have the same forgetting effect as weight decay, but overcomes the restriction on the flexibility of the system.

In order to constitute new prototypes properly, the random noise is added to the updated prototypes in the way that several conditions should be satisfied. First the prototypes before and after the noise step have the same magnitudes and the overlaps between them remain the same. Then the two prototypes have a bit shift from the previous position for the forgetting effect, in which the shift strength can be controlled as a model parameter.

(A) Constraints on prototypes

We obtain prototype vectors $\vec{w}_1$ and $\vec{w}_2$ after the generic learning step. Then we construct new prototype vectors $\tilde{w}_1$ and $\tilde{w}_2$ by adding random noise on the updated prototypes as in Eq.(3.20) in order to achieve the *forgetting* effect.

Here we define three matrics T, U, and V for illustrating the relations of prototypes and the corresponding constraints. T is the square matrix that characterizes the relations between prototypes $\vec{w}_1$ and $\vec{w}_2$. T(i,i) describes the magnitudes of two prototypes and T(i,j) for i $\neq$ j, indicates the in-between overlaps between them. T can be easily known from the trained prototypes after the generic LVQ1 learning step in Eq.(3.2).

$$T = \begin{bmatrix} \vec{w}_1^2 & \vec{w}_1\vec{w}_2 \\ \vec{w}_1\vec{w}_2 & \vec{w}_2^2 \end{bmatrix} \tag{3.21}$$

Next, we define anther square matrix U for the relations between new generated prototypes $\tilde{w}_1$ and $\tilde{w}_2$ after the noise step. U describes the magnitudes of the new prototypes and the corresponding in-between overlaps between them. In order to satisfy the order parameters, the magnitudes of two prototypes should remain the same and the in-between overlaps between them should be unchanged. Hence, U should be equal to T, which means that T(i,i) and T(i,j) should be unchanged after the noise step.

$$U = \begin{bmatrix} \tilde{w}_1^2 & \tilde{w}_1\tilde{w}_2 \\ \tilde{w}_1\tilde{w}_2 & \tilde{w}_2^2 \end{bmatrix} \tag{3.22}$$

The matrix V interprets the overlaps between old (before the noise step) and new prototypes. V(i,i) has a given value U(i,i) $\cdot(1 - \eta/N)$ for i=1,2 because of the given constraints that new prototypes have a bit shift from the original positions. The given η is the noise strength.

$$V = \begin{bmatrix} \tilde{w}_1\vec{w}_1 & \tilde{w}_1\vec{w}_2 \\ \tilde{w}_2\vec{w}_1 & \tilde{w}_2\vec{w}_2 \end{bmatrix} \tag{3.23}$$

As known from the above that V(i,i) depends on U(i,i). For the rest part V(i,j) with i $\neq$ j, we can safely assume that in high dimensions (large N), by the random rotation by an angle corresponding the $(1 - \eta/N)$, which is the change of the normalized overlap, the prototype vectors still satisfy

$$V(i,j) = U(i,j) \cdot (1 - \frac{\eta}{N}) \quad \text{where} \quad i \neq j. \tag{3.24}$$

Therefore, in total the list of conditions of R should read that for all V(i,j),

$$V(i,j) = U(i,j) \cdot (1 - \frac{\eta}{N}) \quad \text{where} \quad i,j \in \{1,2\}. \tag{3.25}$$

Note that here U(i,j) can be replaced by T(i,j) because they are not changed in the noise step.

(B) Random noise vectors

The following equations derived from the above constrains are all the conditions that the old and new prototypes should satisfy.

$$\begin{aligned}
|\tilde{w}_1|^2 = |\vec{w}_1|^2, \tilde{w}_1 \cdot \vec{w}_1 = |\vec{w}_1|^2 \left(1 - \frac{\eta}{N}\right) \\
|\tilde{w}_2|^2 = |\vec{w}_2|^2, \tilde{w}_2 \cdot \vec{w}_2 = |\vec{w}_2|^2 \left(1 - \frac{\eta}{N}\right) \\
\tilde{w}_1 \cdot \tilde{w}_2 = \vec{w}_1 \cdot \vec{w}_2 \\
\tilde{w}_1 \cdot \vec{w}_2 = \vec{w}_1 \cdot \vec{w}_2 \left(1 - \frac{\eta}{N}\right) \\
\tilde{w}_2 \cdot \vec{w}_1 = \vec{w}_1 \cdot \vec{w}_2 \left(1 - \frac{\eta}{N}\right)
\end{aligned} \tag{3.26}$$

In order to fulfill all the requirements in Eq.(3.26), we introduce three random vectors with the same dimensions as prototype vectors and input examples. Given two prototype vectors $\vec{w}_1$ and $\vec{w}_2$, we draw random vectors $\vec{r}_1, \vec{r}_2, \vec{r}_3$ satisfying $\vec{r}_1 \perp \vec{r}_2 \perp \vec{r}_3 \perp \vec{w}_1$ and $\vec{r}_1 \perp \vec{r}_2 \perp \vec{r}_3 \perp \vec{w}_2$, respectively. Then the random vectors are scaled to lengths

$$\begin{aligned}
|\vec{r}_3|^2 = |f_2||\vec{w}_1 \cdot \vec{w}_2| \\
|\vec{r}_1|^2 = f_2 \vec{w}_1^2 - |\vec{r}_3|^2 \\
|\vec{r}_2|^2 = f_2 \vec{w}_2^2 - |\vec{r}_3|^2,
\end{aligned} \tag{3.27}$$

where $f_2 = \frac{2\eta}{N} - \frac{\eta^2}{N^2}$. The term $\frac{\eta^2}{N^2}$ can be ignored in the high-dimensional environment for simplicity. Next we construct new prototypes $\tilde{w}_1$ and $\tilde{w}_2$ with the random vectors as below:

$$\begin{aligned}
\tilde{w}_1 = \vec{w}_1 \left(1 - \frac{\eta}{N}\right) + \vec{r}_1 + \vec{r}_3 \\
\tilde{w}_2 = \vec{w}_2 \left(1 - \frac{\eta}{N}\right) + \vec{r}_2 + sign(\vec{w}_1 \cdot \vec{w}_2) \cdot \vec{r}_3
\end{aligned} \tag{3.28}$$

After obtaining the new prototypes, we can confirm that the following equations hold:

$$
\begin{aligned}
|\tilde{w}_1|^2 &= |\vec{w}_1|^2 - f_2|\vec{w}_1|^2 + |\vec{r}_1|^2 + |\vec{r}_3|^2 = |\vec{w}_1|^2 \\
|\tilde{w}_2|^2 &= |\vec{w}_2|^2 - f_2|\vec{w}_2|^2 + |\vec{r}_2|^2 + |\vec{r}_3|^2 = |\vec{w}_2|^2 \\
\tilde{w}_1 \cdot \tilde{w}_2 &= \vec{w}_1 \cdot \vec{w}_2 - f_2\vec{w}_1 \cdot \vec{w}_2 + sign(\vec{w}_1 \cdot \vec{w}_2)|\vec{r}_3|^2 = \vec{w}_1 \cdot \vec{w}_2,
\end{aligned}
$$

and:

$$
\begin{aligned}
\tilde{w}_1 \cdot \vec{w}_1 &= |\vec{w}_1|^2 \left(1 - \frac{\eta}{N}\right) \\
\tilde{w}_2 \cdot \vec{w}_2 &= |\vec{w}_2|^2 \left(1 - \frac{\eta}{N}\right) \\
\tilde{w}_1 \cdot \vec{w}_2 &= \vec{w}_1 \cdot \vec{w}_2 \left(1 - \frac{\eta}{N}\right) \\
\tilde{w}_2 \cdot \vec{w}_1 &= \vec{w}_1 \cdot \vec{w}_2 \left(1 - \frac{\eta}{N}\right).
\end{aligned}
$$

4

Experiments and results

In this section we perform a number of experiments to investigate the learning behavior of LVQ systems with forgetting mechanisms under different types of concept drifts. In each drift scenario, we compare three learning schemes, namely the original LVQ1 scheme, the LVQ1 with weight decay and the LVQ1 with additive noise to see if the noise accumulation can have the same forgetting effect as weight decay, but without imposing the restriction on the flexibility of high dimensional systems.

We first do experiments under virtual drift in which the prior weights of input data are drifting over time. Specifically, three data stream cases are investigated: gradual drift, sudden drift and periodic change with time. Then we perform a number of experiments on the real drift, where the input data distribution is fixed but the classification targets drift in each learning step. Under real drift both balanced and imbalanced situations are studied respectively.

In all experiments the initial prototypes are normalized, random vectors generated independently without prior knowledge of the cluster structure:

$$Q_{ii}(0) = 1, Q_{ij}(0) = 0 \quad \text{for} \quad i, j \in \{1, 2\}. \tag{4.1}$$

It should be noted that throughout the following study we consider the same dimension $N = 500$ in all experiments. For a fair comparison, we set all the LVQ systems the same modelling parameters, i.e., learning rate, drift strength, and dimension in each experiment. The only variables are the decay factor γ and the noise strength η.

4.1 Virtual drift

Under virtual drift all the presented results are for the same dimension and constant learning rate $\eta_\omega = 1$. The cluster centers $\vec{B}_{1,2}$ are set as two independent, orthogonal random vectors with a unit length and fixed positions during the training phase, which can be given as

$$|\vec{B}_1| = |\vec{B}_2| = 1 \quad \text{and} \quad \vec{B}_1 \cdot \vec{B}_2 = 0. \tag{4.2}$$

4.1.1 Gradual drift

For the gradual change in the input data stream, we consider that the initial class weights start from 0.5 for both classes, and then the weights gradually increase(decrease) over time of the form

$$p_1(\alpha) = \begin{cases} 1/2 & \text{if } \alpha < \alpha_0 \\ 1/2 + \frac{(p_{max} - 1/2)(\alpha - \alpha_0)}{\alpha_{end} - \alpha_0} & \text{if } \alpha_0 \leq \alpha < \alpha_{end}, \end{cases} \tag{4.3}$$

where a_0 is the change point and p_{max} is the final probability of p_1 reached at learning time a_{end}. A linear increase (decrease) of p_1 (p_2) occurs after an initial phase of balanced class training.

Figure 4.1 shows time-dependent biases of two classes according to Eq.(4.3) with following parameters

$$a_0 = 50,\ a_{end} = 200,\ p_0 = 0.5,\ \text{and}\, p_{max} = 0.8. \tag{4.4}$$

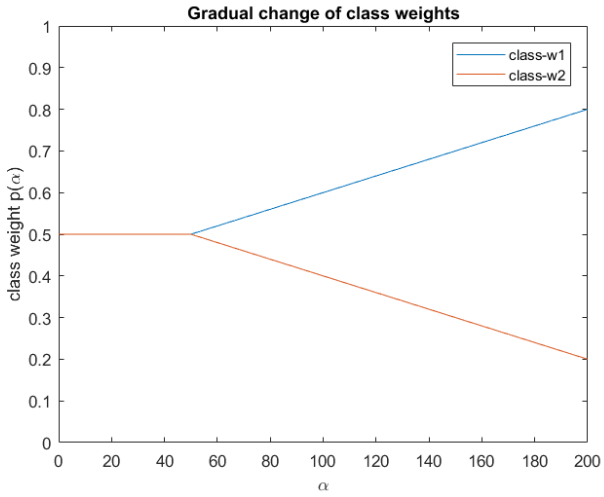
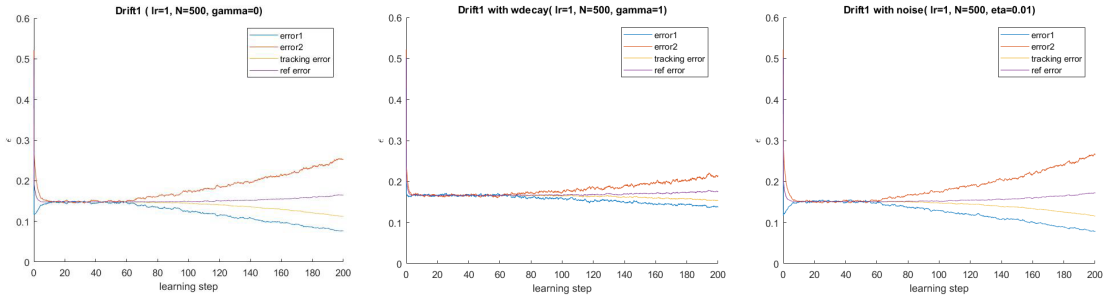


Figure 4.1: Gradual changes under virtual drift for both classes with parameters a_0 = 50, a_{end} = 200, p_0 = 0.5, and p_{max} = 0.8 for class 1. The blue line indicates the weight of class 1 growing linearly from 0.5 to 0.8, and the red one for class 2 decreasing linearly from 0.5 to 0.2.

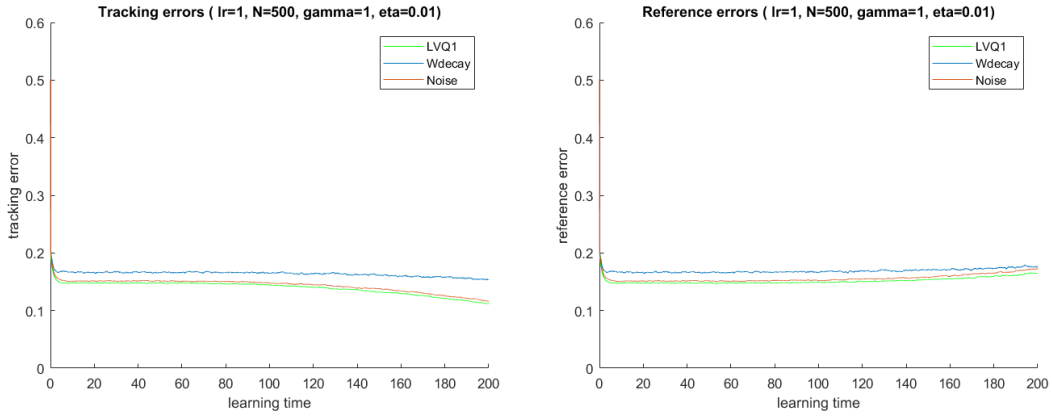


(a) LVQ1 (b) LVQ1 with weight decay (c) LVQ1 with additive noise

Figure 4.2: Learning curves of different LVQ1 systems under gradual virtual drift with settings as shown in Figure 4.1. Cluster parameters are set $\nu_1 = \nu_2 = 0.4$, $\lambda = 1$ in the density Eq.(3.4). Panel (a): original LVQ1 learning. Panel (b): LVQ1 learning with weight decay with $\gamma = 1$ in Eq.(3.19). Panel (c): LVQ1 learning with additive noise with $\eta = 0.01$ in Eq.(3.26). The simulated results for N=500 are shown: the red and blue lines indicate class-wise errors $\epsilon_1(\alpha)$ and $\epsilon_2(\alpha)$, respectively; the tracking error $\epsilon_{tra}(\alpha)$ is displayed as the yellow line and the reference error $\epsilon_{ref}(\alpha)$ marked in purple.

The corresponding learning curves are displayed in Figure 4.2. The class-wise error deceases (increases) with the increasing (decreasing) imbalance of the input data in all cases. In the presence of weight decay in panel (b), however, the class-wise errors are

much closer to each other because weight decay reduces the norms of prototypes during the training, which makes the learning system less sensitive to the class bias. The additive noise does not impose such restriction on the class-wise errors as shown in Figure 4.2 (panel (c)). The reference errors ϵ_{ref} do not show much difference for all the learning systems, but the tracking error ϵ_{tra} from additive noise is slightly lower than weight decay.



(a) Tracking errors (b) Reference errors

Figure 4.3: Tracking errors ϵ_{ref} (left panel) and reference errors ϵ_{tra} (right panel) for basic LVQ1 (green line), weight decay (blue line) and additive noise (red line).

In addition, we plot the weighted tracking error (ϵ_{tra}) and the reference error (ϵ_{ref}) for all three cases in Figure 4.3. Figure 4.3 (a) shows that the original LVQ1 and additive noise have a better tracking error compared with weight decay, while in Figure 4.3 (b) the weight decay has a little bit higher reference error at the beginning and tend to obtain the same value as other two in the end.

4.1.2 Sudden drift

Then we investigate a sudden change from high class bias $p_{max} > 0.5$ to low class bias $1 - p_{max}$ in the input data at change point a_0 of the form

$$p_1(\alpha) = \begin{cases} p_{max} & \text{if } \alpha < \alpha_0 \\ 1 - p_{max} & \text{else.} \end{cases} \tag{4.5}$$

As shown in Figure 4.4, we set $p_{max} = 0.75$ in default and the corresponding results of the LVQ learning system under this drift are shown in Figure 4.5.

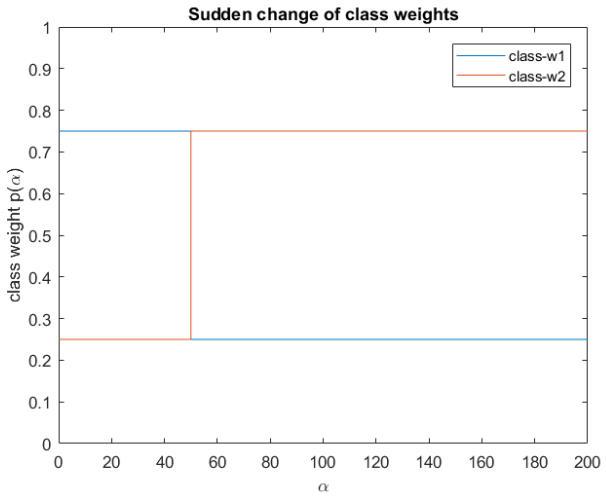
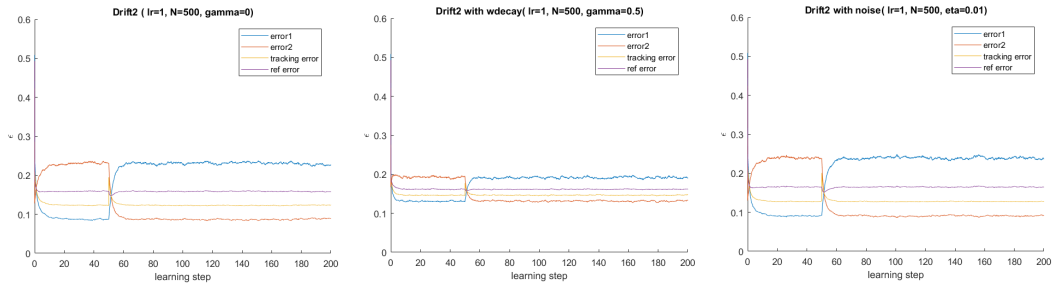


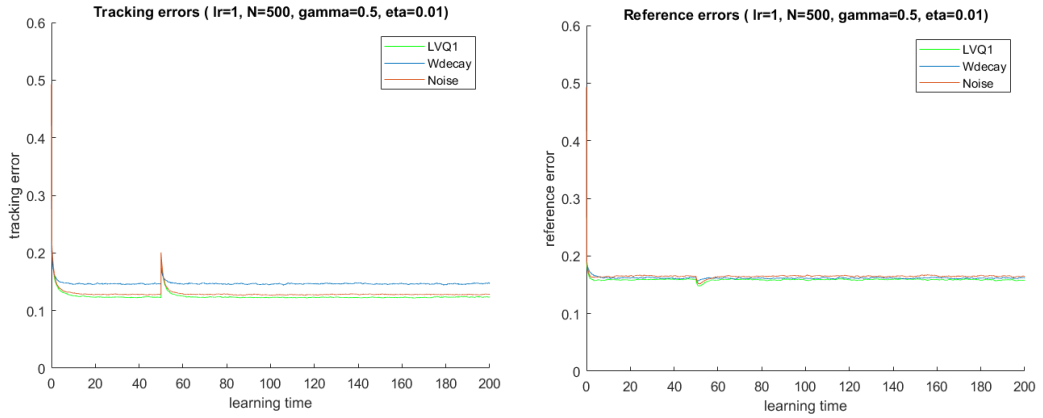
Figure 4.4: Sudden changes under virtual drift for both classes with parameters $p_{max} = 0.75$ for class 1. The blue line indicates that the weight of class 1 is 0.75 at the beginning and has an instantaneous drop from 0.75 to 0.25 at the learning time $a_0 = 50$. The red line is the corresponding weight for class 2.

From Figure 4.5 we can observe similar learning effects as under the gradual, linear drift: in all cases the system reacts rapidly to the sudden switch. The weight decay significantly reduces the influence of the class bias in Figure 4.5 (b) that the class-wise errors are more balanced compared with other two. It is clear that the noise accumulation overcomes the restriction on prototypes and can precisely reflect the class bias in Figure 4.5 (c). However, the additive noise does not seem to improve the learning performance based on the original LVQ1 system.



(a) LVQ1 (b) LVQ1 with weight decay (c) LVQ1 with additive noise

Figure 4.5: Learning curves of different LVQ1 systems under sudden virtual drift with settings as shown in Figure 4.4. Cluster parameters are set $\nu_1 = \nu_2 = 0.4$, $\lambda = 1$ in the density Eq.(3.4). Panel (a): original LVQ1 learning. Panel (b): LVQ1 learning with weight decay with $\gamma = 1$ in Eq.(3.19). Panel (c): LVQ1 learning with additive noise with $\eta = 0.01$ in Eq.(3.26). The simulated results for N=500 are shown: the red and blue lines indicate class-wise errors $\epsilon_1(\alpha)$ and $\epsilon_2(\alpha)$, respectively; the tracking error $\epsilon_{tra}(\alpha)$ is displayed as the yellow line and the reference error $\epsilon_{ref}(\alpha)$ marked in purple.



(a) Tracking errors (b) Reference errors

Figure 4.6: Tracking errors ϵ_{tra} (left panel) and reference errors ϵ_{ref} (right panel) in the presence of sudden change for original LVQ1 (green line), weight decay (blue line) and additive noise (red line).

4.1.3 Periodic drift

In the last part of virtual drift, we consider a periodic, oscillatory change of the class bias in the training phase given as

$$p_1(\alpha) = \frac{1}{2} + (p_{max} - \frac{1}{2})\ \cos\ (\frac{2\pi\alpha}{T}), \tag{4.6}$$

where T is the oscillation period on α-scale and $p_{max} < 1$ is the maximum amplitude of the time-dependence as shown in Figure 4.7. In the experiments we set $T = 50, p_{max} = 0.8$ for class 1 as shown in Figure 4.7. The corresponding learning performance can be observed in Figure 4.8.

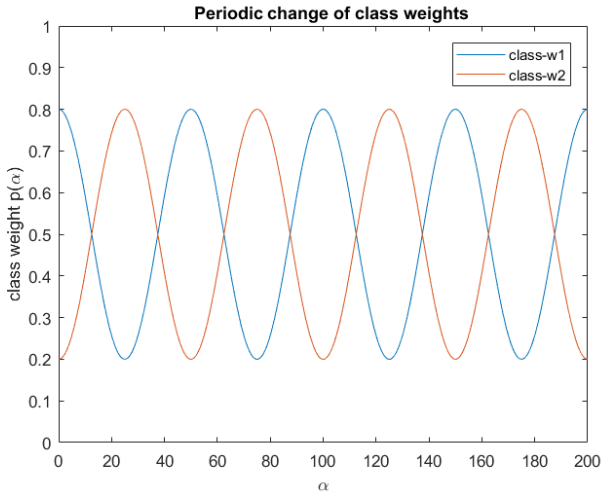
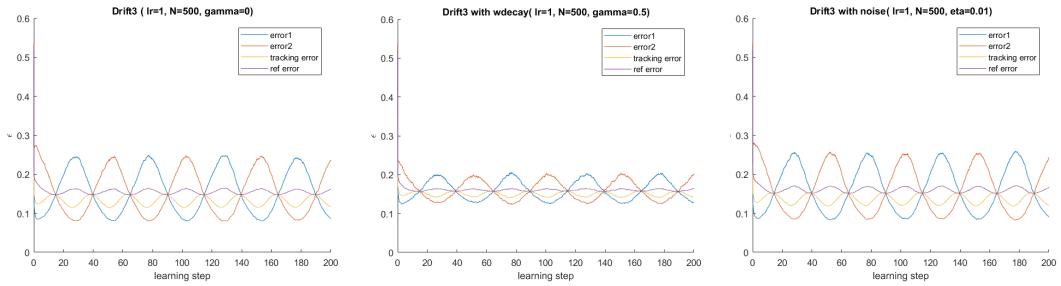


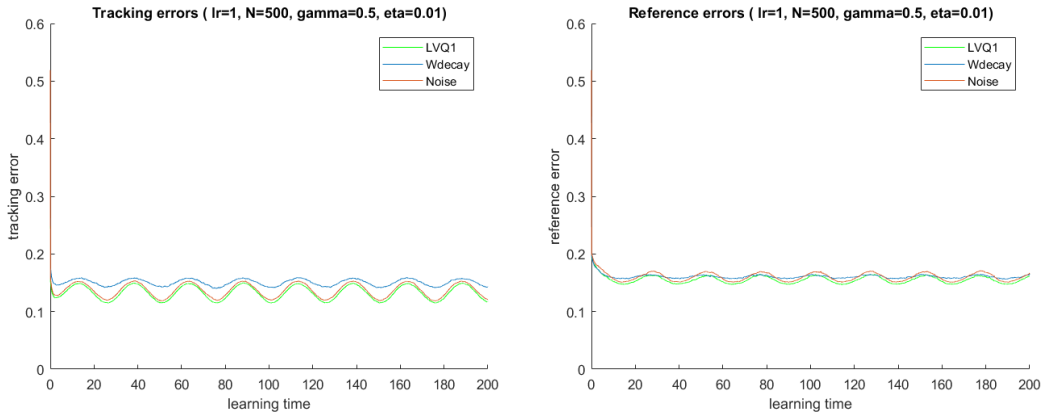
Figure 4.7: periodic changes under virtual drift for both classes with parameters $T = 50$ and $p_{max} = 80$.

The performance of different learning systems under periodic time dependence in Figure 4.8 confirms the above finding that weight decay imposes a limitation on the reaction of the learning system to class biases because of its regularization effect on the system, while the accumulation of additive noise does not have such restriction on the system. The noise accumulation is able to reflect the class bias and outperform weight decay in terms of tracking error in Figure 4.9.



(a) LVQ1 (b) LVQ1 with weight decay (c) LVQ1 with additive noise

Figure 4.8: Learning curves of different LVQ1 systems under oscillating virtual drift with settings as shown in Figure 4.7. Cluster parameters are set $\nu_1 = \nu_2 = 0.4$, $\lambda = 1$ in the density Eq.(3.4). Panel (a): original LVQ1 learning. Panel (b): LVQ1 learning with weight decay with $\gamma = 1$ in Eq.(3.19). Panel (c): LVQ1 learning with additive noise with $\eta = 0.01$ in Eq.(3.26). The simulated results for N=500 are shown: the red and blue lines indicate class-wise errors $\epsilon_1(\alpha)$ and $\epsilon_2(\alpha)$, respectively; the tracking error $\epsilon_{tra}(\alpha)$ is displayed as the yellow line and the reference error $\epsilon_{ref}(\alpha)$ marked in purple.



(a) Tracking errors (b) Reference errors

Figure 4.9: Tracking errors ϵ_{tra} (left panel) and reference errors ϵ_{ref} (right panel) in the presence of periodic class weights for original LVQ1 (green line), weight decay (blue line) and additive noise (red line).

4.2 Real drift

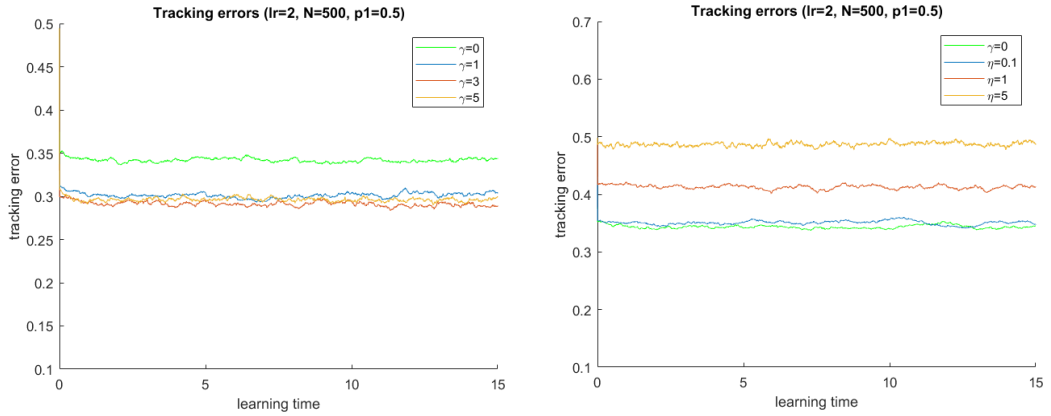
Different from the above-mentioned virtual drift, under real drift the input data distribution is stationary but the cluster center changes over time in this study. Throughout the following all the prototype vectors are initialized according to Eq.(4.1) and initial cluster centers are set as given in Eq.(4.2).

In the presented real drifts, the cluster centers are drifting over time independently according to Eq.(3.16) and all corresponding results are for the same drift strength $\delta = 1$ and learning rate $\eta_\omega = 2$.

We first investigate the balanced case where both class weights equal to 0.5 in the course of training. Next, the imbalanced case is studied with the class weight $p_1 = 0.7$ and $p_2 = 0.3$.

4.2.1 Balanced case

In the presence of real drifts, we first do a number of experiments for the balanced case where prior weights of both classes are the same ($p_1 = p_2 = 0.5$). The example learning curves $\epsilon(\alpha)$ are displayed in Figure 4.10 for different parameters. Model parameters are given in the caption in detail. Note that all the results are observed in 20 simulation runs for a high dimensional environment N = 500.



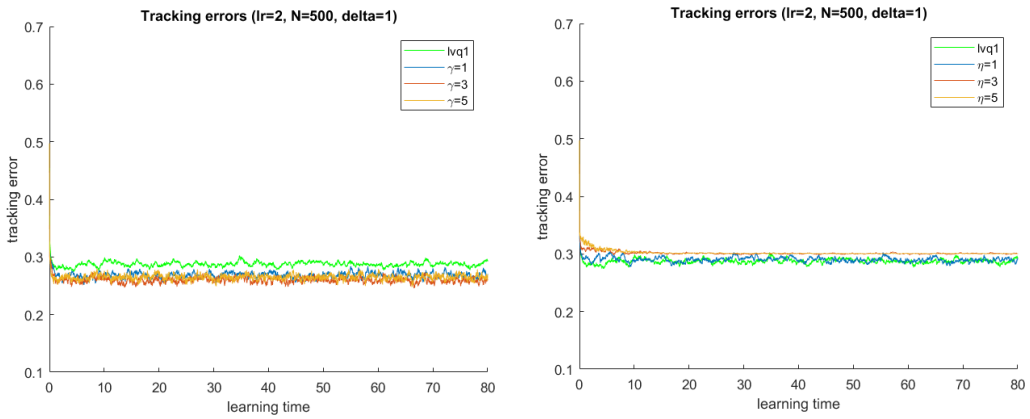
(a) Different weight decay parameters (b) Different noise strength parameters

Figure 4.10: Learning curves of different LVQ systems under real drifts in the balanced case: weight decay and noise accumulation, with model parameters $\delta = 1$ and $\eta_\omega = 2$. Left panel: learning curves $\epsilon(\alpha)$ in the presence of LVQ1 with weight decay and various learning rates γ. Right panel: learning curves $\epsilon(\alpha)$ in the presence of LVQ1 with additive noise and various learning rates η. The green lines indicate the original LVQ1 method in both cases.

Results in Figure 4.10 show that weight decay improves the learning behavior on the original LVQ1 (the green line) with different decay factors $\gamma = 1, 3, 5$. In the presence of additive noise, however, it is obvious that the noise accumulation deteriorates the learning effect with large noise strength parameters. For the small parameter $\eta = 0.1$, the learning behavior is about the same as original LVQ1. Besides, we can also observe from Figure 4.10 (b) that the generalization error $\epsilon(\alpha)$ tends to grow up with the increasing noise strength η. Hence, the noise does not improve the performance and therefore, the weight decay outperforms the noise accumulation in this balanced case.

4.2.2 Imbalanced case

In the second scenario of real drifts, we investigate the imbalanced case where class 1 is over-weighted ($p_1 = 0.7$) and class 2 is under-weighted ($p_1 = 0.3$). Results of the example learning curves $\epsilon(\alpha)$ are given in Figure 4.11 with the same model parameter setting as the balanced case.



(a) Different weight decay parameters (b) Different noise strength parameters

Figure 4.11: Learning curves of different LVQ systems under real drifts in the imbalanced case: weight decay and noise accumulation, with model parameters $\delta = 1$ and $\eta_\omega = 2$. Left panel: learning curves $\epsilon(\alpha)$ in the presence of LVQ1 with weight decay and various learning rates γ. Right panel: learning curves $\epsilon(\alpha)$ in the presence of LVQ1 with additive noise and various learning rates η. The green lines indicate the original LVQ1 method in both cases.

We observe the learning behavior in the presence of imbalanced class weights under real drift from Figure 4.11. Interestingly, both weight decay and additive noise do not have obvious improvements on the original LVQ1 system. The weigh decay in the left panel shows a slight decrease with respect to the tracking error ϵ_{tra} for all the parameters, while for the additive noise in the right panel, the learning performance are about the same as original LVQ1. Also, the larger noise parameters tend to gain a higher generalization error compared with the smaller parameters.

5
Conclusion and discussion

To summarize, this thesis focused on advancing the basic Learning Vector Quantization method, also known as LVQ1, to handle the prototype-based classifiers under concept drift processes. An important part in such an online learning procedure is the forgetting mechanism. In order to overcome the concept drift, the explicit weight decay is one of the standard forgetting mechanism that are frequently used. Previous studies have demonstrated that there is a significant restriction on the flexibility of the learning systems for using the explicit weight decay. Therefore, in this work, we have proposed a new forgetting mechanism which is the application of additive noise in each training step. We then base on the two paradigmatic concept draft situations, the virtual drift and the real drift, to compare their performance. For each of the them, we successfully simulated high dimensional clustered data streams for binary classification problems.

For virtual drift, it is the case where the prior weights of the input data are drifting with time. We tested three different kinds of temporal variations of the weighting, which are gradual drift, sudden drift and periodic drift. Both the class-wise error and the generalization errors are used to reflect the sensitivity to the class bias and the overall performance of

the method. From the class-wise error, our results suggest that the additive noise method do not impose restrictions on the sensitivity to the class bias compared to the explicit weight decay. Also, from the reference and tracking error, it suggests that the additive noise method can outperform the weight decay method.

For real drift, it is the case where the target classification changes with time. We studied both balanced and imbalanced cases in class weights during the training. It is found that when compared with the original LVQ1 and LVQ1 with weight decay, the noise accumulation does not seem to show a notable improvement with respect to the generalization error.

There are some possible reasons that lead to the experiment results. First, the noise introduced to the system in each learning step might be too strong for the effective training and the system is easily influenced by the additive noise. Second, the noise is randomly generated and brings in interference information, which can have a side effect in training. Also, the added noise on the prototypes could be in the wrong direction which weakens the learning effect while reserving the expected forgetting effect at the same time. Third, when we construct new prototypes $\tilde{\omega_i}$ with additive noise in Eq.(3.28), we multiply a factor $1 - \frac{\eta}{N}$ by $\vec{\omega}_i$. However, as in the general form of LVQ1 updates (Eq.(3.2)), $\vec{\omega}_i$ contains both the information of the prototype in the previous step and the recent update. Hence, both are weakened due to the noise step by the factor and it confirms that the additive noise step reduces the importance of learning effect in the training.

Another important underlying reason is that we have made use of the statistic mechanics approach in a rather limited way in this project. We borrow the methods from statistical physics and use the idea of so-called order parameters R^μ and Q^μ for subsequent mathematical descriptions, as well as apply the ordinary differential equations (ODE, Eq.(3.11)) and the generalization error (Eq.(3.13)) given in the original literature [43] throughout the experiments. We did not derive the ODE for the noise accumulation specifically in this project, which may cause offset of the results in absence of the (numerical) integration of the ODE for the given modulation function. The specific ODE and the integration for noise accumulation will be considered in forthcoming studies.

6
Outlook

In this section, we mention the topics for the future work:

- **Comparison with the theory that $N \to \infty$:**
  In all our experiments, we adopt the statistic mechanics approach in a relatively limited way and the dimension of the simulation environment is fixed at $N = 500$. However, the ODE of successful LVQ online learning is based on the theory from statistical mechanics which is derived for the situation of $N \to \infty$. Hence, in future work we would have to derive the ODE for noise accumulation and do the numerical integration, then compare the current results with the $N \to \infty$ theory to investigate whether the learning performance can be more solid or not.

- **Consideration of other types of drift**
  Besides the two concept drifts that we studied in this thesis, other kinds of drifts can be considered in the future work, such as adversarial drift and the combination of virtual drifts and real drifts. There are adversarial activities such as a malicious agent trying to evade the classification system, which leads to the behaviour called adversarial drift. In adversarial drift, the drift is aimed to degrade the system and

avoid detection by traditional drift detection. Besides, we can combine virtual and real drifts to compose a new type of drift in the learning environment.

- **Different speeds of noise accumulation**
  For now, the additive noise approach is to add noise in each learning step. This approach can be modified by introducing an additional control parameter, the frequency of adding noise, which is used to control the speed of noise accumulation. It also quantifies how strong the forgetting mechanisms is. We then can tune the control parameter to seek the optimal frequency of adding noise. In such a way, the learning performance of additive noise approach could be improved.

Bibliography

[1] A. R.R and D. P. R, "Methods for incremental learning : A survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, pp. 119–125, 07 2013.

[2] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, pp. 12–25, 2015.

[3] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, 2018.

[4] P. Joshi, "Incremental learning: Areas and methods a survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 2, pp. 43–51, 09 2012.

[5] H. S. Seung, H. Sompolinsky, and N. Tishby, "Statistical mechanics of learning from examples," *Physical review. A*, vol. 45, pp. 6056–6091, 05 1992.

[6] T. L. H. Watkin, A. Rau, and M. Biehl, "The statistical mechanics of learning a rule," *Rev. Mod. Phys.*, vol. 65, pp. 499–556, Apr 1993.

[7] A. Engel and C. Van den Broeck, *Statistical Mechanics of Learning*. Cambridge University Press, 2001.

[8] M. Biehl and N. Caticha, "Statistical mechanics of on-line learning and generalization," Handbook of Brain Theory and Neural Networks (2nd ed.) M.A. Arbib (ed.), MIT Press (2003), 2003.

[9] M. Biehl, N. Caticha, and P. Riegler, *Statistical Mechanics of On-line Learning*, pp. 1–22. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[10] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[11] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.," *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[12] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural computation*, vol. 1, no. 2, pp. 281–294, 1989.

[13] S. Thrun, "Toward robotic cars," *Communications of the ACM*, vol. 53, no. 4, pp. 99–106, 2010.

[14] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Symposium on Artificial Neural Networks (ESANN)*, (Bruges, Belgium), 2016.

[15] I. liobait, M. Pechenizkiy, and J. Gama, *An Overview of Concept Drift Applications*, vol. 16, pp. 91–114. 01 2016.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. New York, NY, USA: Wiley-Interscience, 2000.

[17] T. R. Hoens, R. Polikar, and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: an overview," *Progress in Artificial Intelligence*, vol. 1, pp. 89–101, Apr 2012.

[18] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, pp. 1517–1531, Oct 2011.

[19] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4802–4821, 2018.

[20] T. Kohonen, "Learning vector quantization for pattern recognition," Report TKK-F-A601, Laboratory of Computer and Information Science, Department of Technical Physics, Helsinki University of Technology, Helsinki, Finland, 1986.

[21] M. Biehl, F. Abadi, C. Göpfert, and B. Hammer, "Prototype-based classifiers in the presence of concept drift: A modelling framework," *CoRR*, vol. abs/1903.07273, 2019.

[22] M. Straat, F. Abadi, C. Göpfert, B. Hammer, and M. Biehl, "Statistical mechanics of on-line learning under concept drift," *Entropy*, vol. 20, p. 775, 2018.

[23] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[24] G.-B. Huang, L. Chen, C. K. Siew, *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[25] H. Zhang, "The optimality of naive bayes," *AA*, vol. 1, no. 2, p. 3, 2004.

[26] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *2009 ieee 12th international conference on computer vision workshops, iccv workshops*, pp. 1393–1400, IEEE, 2009.

[27] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, pp. 44:1–44:37, Mar. 2014.

[28] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4822–4832, Oct 2018.

[29] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, (New York, NY, USA), pp. 97–106, ACM, 2001.

[30] C. Alippi and M. Roveri, "Just-in-time adaptive classifierspart ii: Designing the classifier," *IEEE Transactions on Neural Networks*, vol. 19, pp. 2053–2064, Dec 2008.

[31] A. Andrés-Andrés, E. Gómez-Sánchez, and M. L. Bote-Lorenzo, "Incremental rule pruning for fuzzy artmap neural network," in *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005* (W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny, eds.), (Berlin, Heidelberg), pp. 655–660, Springer Berlin Heidelberg, 2005.

[32] B. Krawczyk and M. Woźniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift," *Soft Computing*, vol. 19, pp. 3387–3400, Dec 2015.

[33] P. R. L. Almeida, L. E. S. de Oliveira, A. S. Britto, and R. Sabourin, "Adapting dynamic classifier selection for concept drift," *Expert Syst. Appl.*, vol. 104, pp. 67–85, 2018.

[34] A. Witoelar, A. Ghosh, G.-J. De Vries, B. Hammer, and M. Biehl, "Window-based example selection in learning vector quantization," *Neural Computation*, vol. 22, pp. 2924–2961, 11 2010.

[35] I. Koychev, "Gradual forgetting for adaptation to concept drift," *ECAI*, 08 2004.

[36] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intell. Data Anal.*, vol. 8, pp. 281–300, Aug. 2004.

[37] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, pp. 37–57, Mar. 1985.

[38] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.

[39] A. Pocock, P. Yiapanis, J. Singer, M. Luján, and G. Brown, "Online non-stationary boosting," in *Multiple Classifier Systems* (N. El Gayar, J. Kittler, and F. Roli, eds.), (Berlin, Heidelberg), pp. 205–214, Springer Berlin Heidelberg, 2010.

[40] M. Biehl, A. Freking, and G. Reents, "Dynamics of on-line competitive learning," *EPL (Europhysics Letters)*, vol. 38, p. 73, 01 2007.

[41] M. Biehl, A. Freking, G. Reents, and E. Schlosser, "Specialization processes in on-line unsupervised learning," *Philosophical Magazine Part B*, vol. 77, 05 1997.

[42] M. Biehl, A. Ghosh, and B. Hammer, "The dynamics of learning vector quantization," in *ESANN*, 2005.

[43] M. Biehl, A. Ghosh, and B. Hammer, "Dynamics and generalization ability of lvq algorithms," *J. Mach. Learn. Res.*, vol. 8, pp. 323–360, May 2007.

[44] A. Ghosh, M. Biehl, and B. Hammer, "Performance analysis of lvq algorithms: A statistical physics approach," *Neural Networks*, vol. 19, pp. 817–829, 07 2006.

[45] T. Kohonen, M. R. Schroeder, and T. S. Huang, eds., *Self-Organizing Maps*. Berlin, Heidelberg: Springer-Verlag, 3rd ed., 2001.

[46] G. R. Lloyd, R. G. Brereton, R. Faria, and J. C. Duncan, "Learning vector quantization for multiclass classification: Application to characterization of plastics," *Journal of Chemical Information and Modeling*, vol. 47, no. 4, pp. 1553–1563, 2007.

[47] D. Nova and P. A. Estévez, "A review of learning vector quantization classifiers," *Neural Computing and Applications*, vol. 25, pp. 511–524, Sep 2014.

[48] M. Biehl, B. Hammer, and T. Villmann, "Prototype-based models in machine learning," *Wiley interdisciplinary reviews. Cognitive science*, vol. 7, 01 2016.

[49] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, January 1967.

[50] P. Schneider, *Advanced methods for prototype-based classification*. PhD thesis, 2010. Relation: https://www.rug.nl/ Rights: University of Groningen.

[51] M. Biehl, B. Hammer, and T. Villmann, "Distance measures for prototype based classification," in *Brain-Inspired Computing*, pp. 100–116, Springer International Publishing, 2014.

[52] G. Reents and R. Urbanczik, "Self-averaging and on-line learning," *Phys. Rev. Lett.*, vol. 80, pp. 5445–5448, Jun 1998.

[53] I. Zliobaite, "Learning under concept drift: an overview," *CoRR*, vol. abs/1010.4784, 2010.

[54] L. van Hemmen, G. Keller, and R. Khn, "Forgetful memories," *EPL (Europhysics Letters)*, vol. 5, p. 663, 07 2007.