RICARDO DE ALMEIDA

ENSEMBLE BASED ON RANDOMISED NEURAL NETWORKS FOR ONLINE DATA STREAM REGRESSION IN PRESENCE OF CONCEPT DRIFT

DOCTOR OF ENGINEERING

INDUSTRIAL AND SYSTEMS ENGINEERING GRADUATE PROGRAM
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

DOCTOR OF PHILOSOPHY

WOLFSON SCHOOL OF MECHANICAL, ELECTRICAL AND MANUFACTURING ENGINEERING
LOUGHBOROUGH UNIVERSITY

CURITIBA
2019

RICARDO DE ALMEIDA

Doctoral thesis submitted in partial fulfilment of the requirements for the award of

Doctor of Engineering Degree of Pontifícia Universidade Católica do Paraná
in Industrial and Systems Engineering

and

Doctor of Philosophy Degree of Loughborough University
in Mechanical and Manufacturing Engineering

ENSEMBLE BASED ON RANDOMISED NEURAL NETWORKS FOR ONLINE DATA STREAM REGRESSION IN PRESENCE OF CONCEPT DRIFT

Supervisors: Dr. Maria Teresinha Arns Steiner (PPGEPS - PUCPR)
Dr. Mey Y. Goh (Loughborough University)
Dr. Radmehr P. Monfared (Loughborough University)

Examiners: Dr. Osíris Canciglieri Junior (PPGEPS - PUCPR)
Dr. Júlio César Nievola (PPGIA - PUCPR)
Dr. Pedro José Steiner Neto (Universidade Positivo)
Dr. Nei Yoshihiro Soma (Instituto Tecnológico de Aeronáutica)
Dr. Diana Segura-Velandia (Loughborough University)
Dr. Weidong Li (Coventry University)

Industrial and Systems Engineering Graduate Program

Wolfson School of Mechanical, Electrical and Manufacturing Engineering

Abstract

The big data paradigm has posed new challenges for the Machine Learning algorithms, such as analysing continuous flows of data, in the form of data streams, and dealing with the evolving nature of the data, which cause a phenomenon often referred to in the literature as concept drift. Concept drift is caused by inconsistencies between the optimal hypotheses in two subsequent chunks of data, whereby the concept underlying a given process evolves over time, which can happen due to several factors including change in consumer preference, economic dynamics, or environmental conditions. This thesis explores the problem of data stream regression with the presence of concept drift. This problem requires computationally efficient algorithms that are able to adapt to the various types of drift that may affect the data. The development of effective algorithms for data streams with concept drift requires several steps that are discussed in this research. The first one is related to the datasets required to assess the algorithms. In general, it is not possible to determine the occurrence of concept drift on real-world datasets; therefore, synthetic datasets where the various types of concept drift can be simulated are required. The second issue is related to the choice of the algorithm. The ensemble algorithms show many advantages to deal with concept drifting data streams, which include flexibility, computational efficiency and high accuracy. For the design of an effective ensemble, this research analyses the use of randomised Neural Networks as base models, along with their optimisation. The optimisation of the randomised Neural Networks involves design and tuning hyperparameters which may substantially affect its performance. The optimisation of the base models is an important aspect to build highly accurate and computationally efficient ensembles. To cope with the concept drift, the existing methods either require setting fixed updating points, which may result in unnecessary computations or slow reaction to concept drift, or rely on drifting detection mechanism, which may be ineffective due to the difficulty to detect drift in real applications. Therefore, the research contributions of this thesis include the development of a new approach for synthetic dataset generation, development of a new hyperparameter optimisation algorithm that reduces the search effort and the need of prior assumptions compared to existing methods, the analysis of the effects of randomised Neural Networks hyperparameters, and the development of a new ensemble algorithm based on bagging meta-model that reduces the computational effort over existing methods and uses an innovative updating mechanism to cope with concept drift. The algorithms have been tested on synthetic datasets and validated on four real-world datasets from various application domains.

Keywords: Machine Learning, Ensemble Learning, Data Streams, Regression Problems, Concept Drift, Hyperparameter Optimisation.

Acknowledgements

I'm sincerely thankful to my supervisors, prof. Maria Teresinha Arns Steiner, prof. Mey Y. Goh and prof. Radmehr P. Monfared for the achievements of this research project. Teresinha has been a key person not only in my professional development but also in shaping my character. She has been restlessly supporting, guiding and helping me since I was an undergraduate. Besides sharing all her expertise, Teresinha has also being a friend and a role model. Mey's sharp mind, energy and professionalism inspired me to seek constant improvement of my research and to develop as a professional. Radmehr's wisdom also played an important part in my development, he always inspired me to question and therefore improve my research and ideas.

I thank the Industrial and Systems Engineering Post-Graduate Program of Pontifícia Universidade Católica do Paraná and all its staff for offering the infrastructure that allowed me to build my background. Among the staff, I'm especially grateful to Denise and prof. Osiris. Denise was always available to help with all the bureaucracy and documentation. Osiris, along with Teresinha, was responsible for making the double degree possible, which allowed me to work with two amazing people, prof. Mey and prof. Radmehr. I also would like to express my gratitude to all the professors of PPGEPS for sharing their expertise.

I'm thankful for all the financial support provided by the Brazilian government through the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and also the scholarship provided by Loughborough University and intermediated by prof. Mey.

To all my friends and colleagues that contributed to making this challenging journey even more enjoyable.

I'm also deeply thankful to my parents, João e Sueli, that always helped and supported my decisions and regardless of our limited resources, made everything possible to give me a good education and a good character. I can't forget to mention my brother, Fabiano, who was always there when an emergency appeared.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AddExp…….. Additive Expert Ensemble

ANOVA……...……..…….. Analysis of Variance

ASHT……..…….Adaptive-Size Hoeffding Tree

BP……..……...Back-propagation

B-NNRW……..……Bagging Neural Networks with Random Weights

CART……..Classification and Regression Tree

CEC……..……..…… Congress on Evolutionary Computation

CNN……..……...….Convolutional Neural Networks

DBN……..…… Deep Belief Networks

DNN……..…….Deep Neural Networks

DNNE……..……..…….Decorrelated Neural Network Ensemble

DOE……..Design of Experiments

DOER…….……..Dynamic and Online Ensemble Regression

DT……...……....Decision Tree

EA……..…….. Evolutionary Algorithms

ESMSE……...……Exponentially Smoothed Mean Squared Error

ELM……...…….……..Extreme Learning Machine

GA……..……..……Genetic Algorithm

ILLSA…….….Incremental Local Learning Soft Sensing Algorithm

k-NN……..……..k-Nearest Neighbors

LS……...……..….Least Squares

MAPE……..……...…….Mean Absolute Percentage Error

ML……..……..Machine Learning

MSE……..…Mean Squared Error

NCL……..……..……Negative Correlation Learning

NN……..……Neural Networks

NNRW…….……Neural Networks with Random Weights

NSGA-II……...……...…Non-dominated Sorting Genetic Algorithm II

Obag……..……...Online bagging

O-DNNE……..…….……Online Decorrelated Neural Network Ensemble

OPF……..…….Optimum-Path Forest

ORF…….……...Online Random Forest

O-SLFNN……..……Online Single-hidden Layer Feedforward Neural Networks

OWE…….Online Weighted Ensemble

PSO……..……..Particle Swarm Optimisation

RBF……..……..Radial Basis Function

RPLS……..……..……..Recursive Partial Least Squares

RS……..…….Random Search

RVFL…….……Random Vector Functional Link

SEA……..……...…….Streaming Ensemble Algorithm

SLFNN……..…….……Single-hidden Layer Feedforward Neural Networks

SMBO……..……...……..Sequential Model-Based Optimisation

SSHT……Sum of Squares Hyperparameter Tuning

SVM……..……Support Vector Machine

1. INTRODUCTION

The field of machine learning (ML) has been developing rapidly and proved useful in modelling complex real-life applications. The capacity of ML models to extract knowledge from massive amounts of data has increased the ML popularity and supported innovation and business growth in various industries. The development of information technologies has allowed that massive amounts of data are produced at a rapid rate, which imposes new challenges for data analysis techniques (Yaqoob et al., 2016). In many application domains, such as social networks, financial industries, and engineering systems, data are generated in continuous flows in the form of data streams. Such data format requires the ML algorithms to work in an online mode, i.e. analysing the data in real-time and evolving accordingly. Examples of data streams include network event logs, telephone call records, credit card transactional flows, sensing and surveillance video streams, financial applications, monitoring patient health, and many others (Wang et al., 2003; Fan, 2004; Zhang et al., 2012; Krawczyk et al., 2017).

The ML algorithms are mainly divided into 3 types: *supervised learning*, *unsupervised learning* and *reinforcement learning*. In the supervised learning, the training data include the input vectors and their corresponding target vector. Supervised learning tasks include *classification*, where the aim is to assign a class for each input vector, and *regression*, where the target for each input vector is a continuous variable. In the unsupervised learning, the target vector is not present and the aim is to determine similar groups within the data. In the reinforcement learning, the algorithm interacts with the environment in order to find appropriate actions that maximise the reward (Bishop, 2006).

Traditionally, the supervised learning approaches work on an offline mode where a fixed amount of data is collected and used to train and validate the predictive models. This leads to the assumption that the data probability distribution does not change between training data and the application data (González-Castro et al., 2013). This typically means that data used to train the predictive models can reflect the probability distribution of the problem, however, this assumption is often violated in real-world applications (Gállego et al., 2017; Ren et al., 2018).

For many reasons, the data distribution in real-world applications is often not stable and tends to change with time (Tsymbal, 2004; Zliobaite et al., 2016). This is due to the evolving nature of the processes, which causes a phenomenon frequently referred to in the literature as concept drift. The presence of concept drift is likely to cause a decrease in the accuracy of the models as time passes, since the training data used to build the models may be carrying out-of-date concepts. This has led to increasing research on data stream mining applications. Besides the evolving nature of data, other properties that make the prediction task in data streams challenging include infinite length, high dimensionality, orderliness, non-repetitiveness, high-speed, and time-varying (Masud et al., 2008; Farid et al., 2013).

A promising research direction in modelling data streams is the ensemble learning methods (Krawczyk et al., 2017). Ensemble approaches, also known as committees or multiple classifiers, can be characterised as a set of classification or regression models, whose outputs are combined to predict the output of a new instance. Single models usually require complex operations to modify the internal structure of the model and may perform poorly in the presence of concept drift (Masud et al., 2008). Ensemble approaches are proven to be effective to overcome common limitations of single models, such as accuracy and stability (Yin et al., 2015). Additionally, they are able to maintain information of different concepts, and new models can be easily trained to cope with new concepts that may appear; hence, they can effectively deal with evolving data streams and achieve superior accuracy compared to single models.

The diagram shown in Figure 1.1 contextualises this research within the field of ML, highlighting the scope in terms of the type of learning, task and environment this research aims to address, as well as ML technique applied.

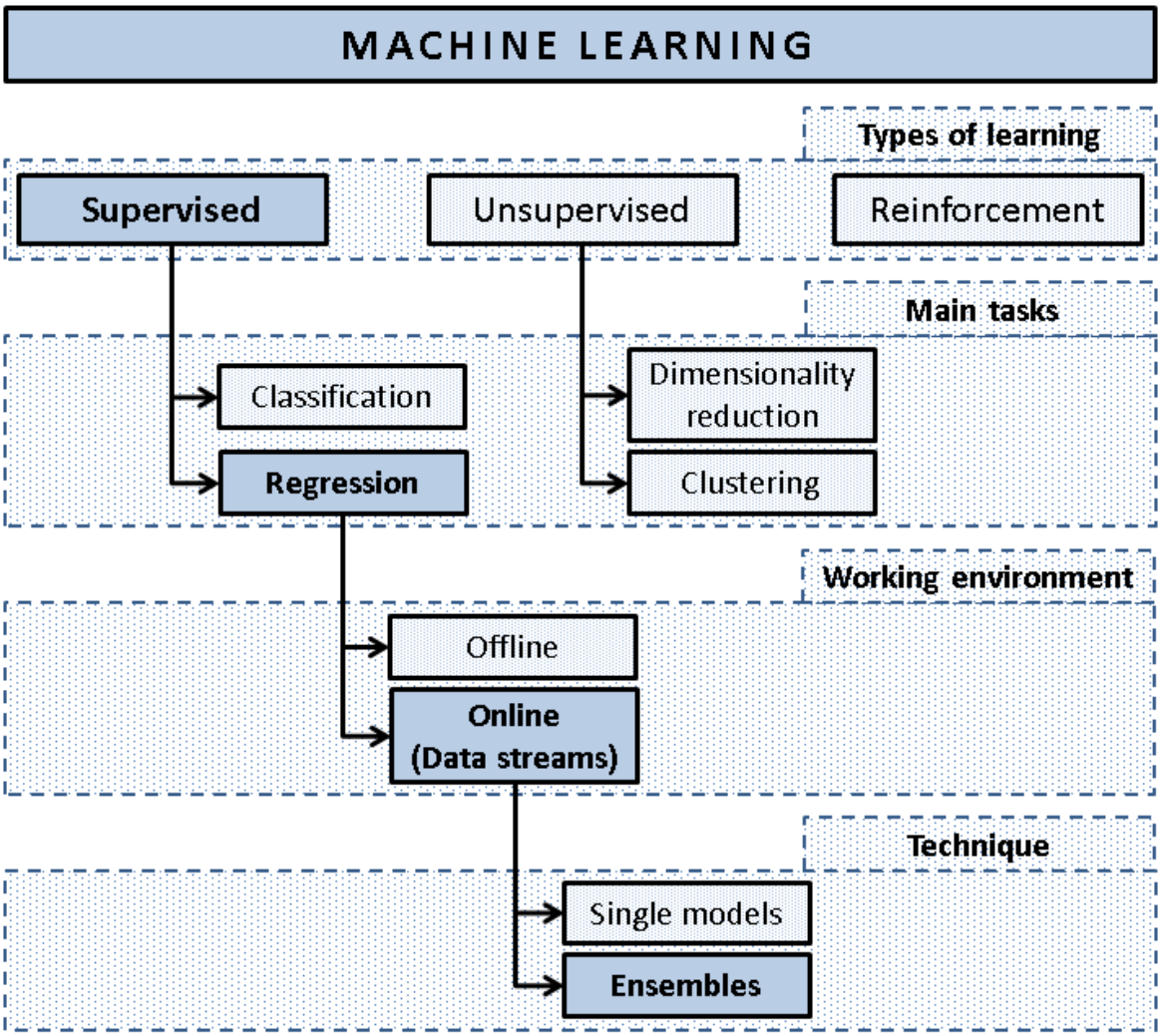

Figure 1.1: Research scope.

1.1. RESEARCH PROBLEM

Nowadays, large amounts of data at very high rates are generated by organizations. In this context, new ML techniques are required to address the new challenges imposed by the big data paradigm (Yaqoob et al., 2016). One such challenge is how to build predictive models that can work on an online mode and adapt to possible changes in the underlying process generating the data.

An increasing research effort has been made in the recent years towards the data stream mining, however, mainly focused on supervised classification problems (Ikonomovska et al., 2015; Krawczyk et al., 2017). Regression is an important field of study with many practical applications, which include quality control, process

monitoring, financial forecasting, weather prediction, among others. For instance, in regression tasks, the aim is to model the relationship between the input vector and the output variables, given a data sample composed of a feature vector x_i and a scalar output variable y_i. Therefore, ML approaches approximate a function f that transforms an input vector x_i into an output y_i, given by $y_i = f(x_i) + e_i$, where e_i is an approximately normally distributed noise with zero mean. The development of data stream regression algorithms has the potential to benefit many industries by solving practical problems in a continuous manner with high accuracy and adapting to cope with constantly changing environments.

Ensemble learning algorithms appear as a promising technique to deal with data streams with concept drift, mainly due to the high level of accuracy and computational efficiency that is possible to achieve using ensemble learning techniques. The concept drift may occur in several ways and is difficult to detect (Farid et al., 2013). The main approaches to deal with concept drift include active and passive approaches. The former updates the model without assuming the presence of drift, which may lead to unnecessary computation when no changes in data properties are observed. On the other hand, passive approaches wait until the drift is detected to update the model and some drawbacks include false alarms, inability to detect some types of drift and poor performance in case of data insufficiency.

An important decision in ensemble design is the choice of base models. Effective data stream ensembles require computationally efficient base models with a good level of accuracy. An algorithm that met those requirements is randomised Neural Networks (NN), also known as Neural Networks with Random Weights (NNRW) (Cao et al., 2018). NNRW was introduced in (Pao and Takefuji, 1992), which proposed the Random Vector Functional Link (RVFL). Later, Huang (2004) introduced a similar algorithm, the Extreme Learning Machine (ELM) that boosted the NNRW popularity in various applications. The main idea of such models is to randomly initialise the weights between input and hidden layers, which are kept fixed during the optimization process and optimize the weights between the hidden and output layers. This process not only reduces the number of parameters but also converts the non-convex optimisation into a convex optimisation, reducing the training complexity compared to the traditional backpropagation algorithms and therefore resulting in higher computational efficiency.

In general, the ensemble algorithms ignore the base model optimisation through the hyperparameter adjustment. Despite the popularity of ML algorithms it is common to find applications where the hyperparameters are simply set to default values or adjusted by trial and error approach. There is limited research exploring the hyperparameter optimisation; however, relying on a systematic way to optimise the hyperparameters could not only improve the algorithm accuracy and computational performance but also help understand how the hyperparameter setting affects the model.

In order to effectively assess data stream algorithms, data generators that can effectively simulate the various types of drift are required. The existing methods for data generation are not only limited in terms of dimensions, i.e. only a few predictive attributes can be simulated, but also does not allow checking the effect of simulated drift on the data.

Following a comprehensive review of the current literature and methods, a number of research areas within the current approaches that require improvements are identified. These include:

- Need for the development of fast algorithms for data stream regression problems.
- Need for updating mechanisms that avoid the drawbacks of active and passive approaches to deal with concept drift.
- Advance the research on hyperparameter optimisation and therefore improve the effectiveness of ML algorithms.
- Need for effective ways to simulate regression problems and the various types of concept drift.

1.2. RESEARCH AIM AND OBJECTIVES

The main aim of this research is to **develop an ensemble learning method based on NNRW algorithms for data stream regression problems.** The algorithm must be capable of effectively predicting continuous variables and adapt continuously to possible changes in the underlying process that generates the data in order to keep the model updated and avoid loss of accuracy.

Several key points were identified as the research objectives for the achievement of the aim of this research, these include:

- Develop a robust methodology for generating synthetic data streams and simulating concept drift.
- Analysis of the NNRW approaches and their main differences.
- Development of a new hyperparameter optimisation algorithm.
- Development of effective updating mechanisms to cope with concept drift.
- Test and validate the proposed approaches using synthetic and benchmark datasets and comparing them with existing methods from the literature.

1.3. THESIS OUTLINE

The remainder of this thesis is organized as follows. Besides the introduction chapter, where the research problem and research objectives are presented, a literature review is covered in Chapter 2. The literature review outlines the main challenges involving data streams and concept drift and also presents the main aspects involving ensemble design. The state-of-the-art approaches for concept-drifting data streams are also shown in the literature review. The structure of this research is presented in Chapter 3 and shows how the research was organised to achieve the established goals. The datasets used to assess and validate the proposed algorithms are detailed in Chapter 4. An analysis of the base models

(NNRW) is carried out in Chapter 5 and its optimisation is tackled in Chapter 6. Chapter 7 details the development of the new ensemble algorithm for data stream regression and Chapter 8 closes this thesis, outlining the main conclusions and opportunities for future research.

2. LITERATURE REVIEW

In this chapter, the existing ensemble approaches for concept drifting data stream regression and classification are discussed. Before that, an overview of the main challenges involving data streams and concept drift is presented. This is followed by a brief review of ensemble methods, the effect of diversity and pruning on the ensemble's performance and how the base models can be effectively combined. Bullet point remarks on the literature review close this chapter.

2.1. DATA STREAMS AND CONCEPT DRIFT

In many applications data are generated in continuous flows. Examples of data streams include network event logs, telephone call records, credit card transactional flows (Wang et al., 2003; Fan, 2004), sensing and surveillance video streams, financial applications, monitoring patient health, and many others. Several challenges are imposed on ML algorithms due to the characteristics of data streams, which include infinite length, the evolving nature of data, high dimensionality, orderliness, non-repetitiveness, high-speed, and time-varying properties (Masud et al., 2008; Farid et al., 2013, Krawczyk et al., 2017).

It is usually impractical to store all data generated by data streams and mine them to discover patterns or hypothesis. Different from traditional knowledge discovery tools that assume a volume of data that can be stored in memory and non-strict limitation of processing time, data stream models have space and time restrictions (Bifet et al., 2009). A common practice is to mine a subset of data, however, as mentioned by Fan (2004), this approach could be ineffective due to oversimplified models as a result of sub-sampling or to the dynamically and unpredictable evolving nature of the data. Bifet et al. (2009) highlight some properties desired for an ML algorithm for data streams: high accuracy and fast adaptation to change; low computational cost in both space and time; theoretical performance guarantees; and a minimal number of parameters.

It is expected in many practical applications that the concept underlying a given process evolves over time, which can happen due to several factors including change in consumer preference, economic dynamics, or environmental conditions. The evolving nature of data has presented an important challenge for data stream learning algorithms. This phenomenon is commonly referred to as concept drift, in the context of machine learning, data mining and predictive analytics; covariate shift or dataset shift, in the context of pattern recognition; and non-stationarity in the context of signal processing (Zliobaite et al., 2016). Fan (2004) describes concept drift as inconsistencies between the optimal hypotheses in two subsequent chunks of data. Yin et al. (2015) define concept drift as a change in data distribution that occurred in dynamic environments, where non-stationary data are observed, that results in a change of the concept of class definitions.

Some authors worked on formally defining the concept drift. Given posterior distribution $P(x,y) = P(y|x)*P(x)$, where x is the input vector and y is the target value, Gao et al. (2008) defined three possible sources of concept drift:

- *Features change*: P(x) changes but P(y|x) does not.
- *Conditional change*: P(x) remains unchanged but P(y|x) changes.
- *Dual change*: Both P(x) and P(y|x) change.

This typology corresponds to the sources of concept drift reported by (Masud et al., 2008), which states that the data may evolve through a change in prior distribution, change in posterior distribution or both (feature change, conditional change and dual change, respectively).

Zhu et al. (2010) also defined concept drift in classification problems as a change in the posterior probability of a given class due to possible changes in conditional probability and/or priori probability. They further decomposed drifting concepts as:

- *Priori probability drifting*: drifting is triggered by class priori probability.
- *Conditional probability drifting*: drifting is triggered by class conditional probability.

- *Conjunct probability drifting*: both conditional and priori probability constantly changes across the data stream.

More recently, Gállego et al. (2017) have characterized the evolving nature of data as *dataset shift*. Considering an instance x, a class value y and a joint probability $P(x|y)$, they identified three types of dataset shift:

- *Covariate shift*: where $P(x)$ changes but $P(y|x)$ does not.
- *Prior probability shift*: where $P(y)$ changes but $P(x|y)$ remains constant.
- *Concept drift*: where either $P(y|x)$ changes and $P(x)$ does not or $P(x|y)$ changes but $P(y)$ remains constant.

Krawczyk et al. (2017) point out a distinction between real drift and virtual drift. They define the real drift as a change in $P(y|x)$, which may happen without changes in $P(x)$ and therefore may not be detected by drift detection mechanism based on input attributes. The virtual drift is related to changes in $P(x)$ and $P(y)$.

Further concept drift categorisation is related to how it occurs. Mainly, they can be distinguished between sudden and gradual drifts (Tsymbal, 2004). Krawczyk et al. (2017) extend the categorization, including incremental drift and recurrent drift, as shown in Figure 2.1.

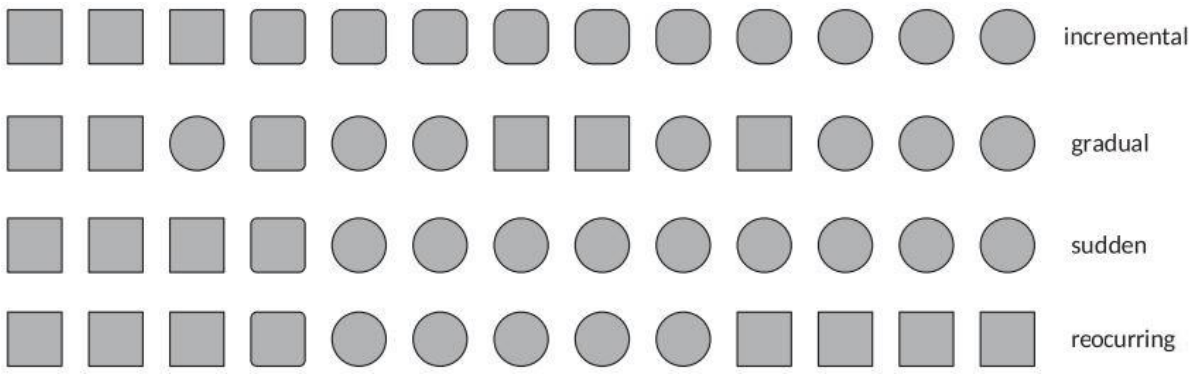

Figure 2.1: Types of drift (Krawczyk et al. 2017).

When any type of concept drift occurs, it is likely that the accuracy of the model decreases since that training data used to build the model may be carrying

out-of-date concepts. One challenge in learning concepts from data streams in presence of concept drift is how to identify the data in the training set that are no longer consistent with the current concept (Wang et al., 2003).

Tsai et al. (2009) defined three main categories of algorithms for concept drift: window-based approaches, weight-based approaches, and ensemble classifiers. Elwell and Polikar (2011) further classified algorithms for concept drift as:

- *Online versus batch algorithms*: Online algorithms learn one instance at a time while batch algorithms learn chunks of instances.
- *Single model versus ensemble-based approach*: The former refers to a single learning algorithm used for prediction while the latter refers to multiple learning algorithms (not necessarily of the same type) that are combined to increase the prediction performance.
- *Active versus passive approaches*: Active approaches rely on drift detector mechanism while passive approaches assume constant drift and update the model continuously.

Although online algorithms may be better to learn new concepts, they suffer from poor stability and are sensitive to noise. On the other hand, batch algorithms may be ineffective if the data chunk size is not properly adjusted and/or the chunk contains multiple concepts (Elwell and Polikar, 2011).

Single models can be based on window-based approaches, where models are built selecting instances within a fixed or dynamic sliding window, and weight-based approaches, where weights are attributed to instances and outdated instances are conveniently discarded (Farid et al., 2013). Usually, single models require complex operations to modify the internal structure of the model and may perform poorly in the presence of concept drift (Masud et al., 2008).

Although the main types of concept drift can be classified, drift detection mechanisms may be inaccurate and yield false reports, particularly in noisy datasets (Fan, 2004; Elwell and Polikar, 2011). In general, drift detection mechanisms are based on monitoring some indicators over time, such as performance measures or data properties (Gama et al., 2004). Additionally, Farid et al. (2013) point out that the

statistical properties of the target class, in case of classification problems, change over time in unforeseen ways. Another risk for active approaches, especially for detection mechanism based on error, is data insufficiency (Fan, 2004), where the data used for training the models do not represent the learning hypothesis adequately. Gao et al. (2008) state that it is optimal to always update the model according to the most recent data, regardless of how concepts evolve. Elwell and Polikar (2011) argue that the knowledge present in the models should be categorized based on its relevance to the current environment, represented by the most recent data, and should be dynamically updated as new data are generated.

2.2. ENSEMBLES

Ensemble approaches have been successfully applied in both classification and regression problems. They are inspired by the decision process based on different opinions from experts (Rokach, 2010; Mousavi and Eftekhari, 2015), and are proved both theoretically and empirically to outperform single classifiers in various tasks (Wang et al., 2003; Brown et al., 2005). Opinions from different sources reduce the risk of low performance of a single agent; furthermore, the ensemble tends to reduce the variance of its base classifiers. In the human decision-making process, a set of opinions, notably when a high degree of diversity is presented, is richer than an isolated opinion.

The classical ensemble approaches include Boosting (Schapire, 1990), Staking (Wolpert, 1992), Bagging (Breiman, 1996), and Random Forests (Breiman, 2001), and many variants that can be found in the literature for solving a wide variety of tasks. The ensemble learning represents an important research direction in solving concept-drifting data streams (Yin et al., 2015) and has been successfully applied in classification and regression problems. Some advantages of ensemble approaches, compared to single models, include the suitability for dynamic updates and integration with drift detection mechanisms (Gomes et al., 2017). Moreover, they are easy to scale and parallelise, the under-performing parts can be pruned to adapt to changes, and usually generate more accurate concept description, compared to single models (Bifet et al., 2009).

Wozniak et al. (2014) highlighted three main issues with ensemble learning methods:

- *System topology*: how to interconnect individual classifiers.
- *Ensemble design*: how to drive the generation and selection of a pool of valuable classifiers.
- *Fuser design*: how to build a decision combination function (fuser) which can exploit the strengths of selected classifiers and combine them optimally.

The ensembles can be divided into two categories: fixed ensemble, where base predictors are trained in advance and are updated online; and growing ensembles, where component learners are added and/or removed, and voting weights are updated according to the incoming data.

2.2.1. The importance of ensemble diversity

Breiman (1996) argues that an effective ensemble requires that the singular predictors must be unstable. Instability means that changes in training data or initialization produce diversity among the learners, i.e. differences in their output in response to a given input. The use of stable models could potentially produce biased ensembles and their use requires a mechanism to cause instability. One example is presented by López et al. (2015), which introduced diversity in a Support Vector Machine (SVM) ensemble by weakening the base models through a data sampling mechanism based on boosting. The models that compose the ensemble are built sequentially and for each new model, a combination of high emphasised data points (Data points difficult to classify by previous models) and data points that can be easily classified are sampled to train new models. The proposed procedure aims at building a diversified and compact ensemble.

In terms of diversity, ensemble approaches can be categorised as *explicit diversity methods*, when the information about diversity may be taken into account. An example is the Boosting algorithm, where the data distribution is manipulated to

ensure diversity. In contrast, *implicit diversity methods* do not take diversity measure into account. An example of this category is the Bagging algorithm, where the dataset is randomly sampled to create different training sets for each classifier (Brown et al., 2005). Similarly, Rokach (2010) distinguish ensembles as dependent frameworks, where the output of a classifier is used in the construction of next classifiers, and independent frameworks, where each classifier is built independently. Tumer and Ghosh (1996) showed that combining procedures are more effective when the base models are negatively correlated, moderately effective when the experts are uncorrelated and slightly effective when they are positively correlated.

The diversity among members of the ensemble is a widely discussed issue and the meaning of diversity may still be a controversial concept. Brown et al. (2005) reviewed attempts to provide a formal definition of error diversity. The concept of diversity can be well explained in the case of regression problems by *ambiguity decomposition*, which shows that the error of the convex-combined ensemble is lower than or equal to the average error of the individuals; and *Bias-variance-covariance decomposition*, which takes into account the possible distribution over different training sets (Brown et al., 2005). This also can be extended to classification problems by converting the class outputs to ordinal values, i.e. probability estimates. The diversity measure for non-ordinal values is a more complex issue and could be approximated using methods such as a heuristic approach proposed by Sharkey and Sharkey (1997).

One approach to measuring diversity is kappa-statistic (Margineantu and Dietterich, 1997; Bifet et al., 2009). Additionally, some diversity measures have been developed to capture the degree of disagreement among base classifiers; Kuncheva and Whitaker (2003) studied four pairwise methods (Q-statistic, Correlation coefficient, Disagreement measure and Double-fault measure); and six non-pairwise ones (Kohavi-Wolpert variance, Interrater agreement, Entropy measure, Measure of difficulty, Generalized diversity and Coincident failure diversity). They showed that there exists a strong correlation between each other. Besides the fact that there is no consensus on what a good diversity measure should be, Bhardwaj et al. (2016) found evidence that diversity measures may not be effective when considered for ensemble pruning methods.

Brown et al. (2005) categorised some techniques for ensemble diversity induction as follows:

- *Starting point in hypothesis space*: A common example is random initial weights of NNs, which increases the probability of convergence in different trajectories. Besides it is widely used, it is also accepted as the least effective method.

- *Set of accessible hypotheses*: There are two ways to manipulate the accessible hypothesis: Manipulation of training data, also referred to as resampling methods, where each learner can be trained using different training patterns or different feature subsets; and changing the architecture of the learner.

- *Traversal of hypothesis space*: Rely on the path the algorithm uses to traverse the hypothesis space in search of the best hypothesis, to generate diversity.

The ensemble diversity helps to avoid the issue of overfitting (López et al., 2015) since the disagreement between ensemble members cancel out the effect of overfitted models. Overfitted models reduce the bias component of error while the ensemble is responsible for reducing the variance (Brown et al., 2005). According to the Ambiguity decomposition (Brown et al., 2005), the increased individual variability, as a consequence of higher diversity, has an effect on the individual's accuracy and the right balance between individual error and diversity must be taken into consideration to achieve the lowest overall ensemble error. Some research has been done to address the trade-off between diversity and accuracy using multi-objective optimisation approaches, such as Mousavi and Eftekhari (2015), which proposed a combination of Static and Dynamic Ensemble Selection based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II).

2.2.2. Combination rules and pruning approaches

The combination of outputs from different classifiers is an issue that can highly influence the ensemble results. The simplest procedures are averaging, in case of regression problems, and majority vote, in case of classification problems. Omari and Vidal (2015) highlighted two main approaches for output aggregation: training of the learner first and then aggregating their outputs; and training different learners and training an aggregation unit using all examples, in which case the ensemble should be of moderate size. Kittler et al. (1998) proposed five ensemble rules for combining multiple classification results, which include Max Rule, Min Rule, Product Rule, Majority Vote Rule and Sum Rule; and are based on the probabilities with each classifier predicts an instance. Sun et al. (2015) enhanced those rules by considering the relationship between new data and training data through a measure of similarity based on a distance weighting mechanism.

Omari and Vidal (2015), refer to a concept of post-aggregation to improve the performance of massive ensembles, i.e., ensembles with a high number of learners. They proposed a fusion procedure that includes two steps: first, a traditional non-trainable aggregation unit for classifiers' output is used; then, a soft version of previous aggregation (average or voting, for example) is introduced as input for a complementary learning machine that also reads the observations. Omari and Vidal (2015) found out that their post-aggregation method can improve the ensemble's accuracy, except for very high-quality ensembles, however at an additional computational cost.

Basic linear combination strategies include simple average, trimmed mean, Winsorized mean and median (Jose and Winkler, 2008). An error-based approach is presented by Armstrong (2001) where the model's weights are inversely proportional to their error. Elwell and Polikar (2011) applied dynamically weight updating based on time-adjusted errors. Additionally, their approach temporarily disables classifiers that do not match the current environment. Ordinary Least Square is another popular method (Granger and Ramanathan, 1984; Aksu and Gunter, 1992; Lemke and Gabrys, 2010). An outperformance approach that applies a Bayesian framework and assigns weights based on past forecasting trials is proposed by Bunn (1975). A

variance-based pooling that relies on *k*-Means to form clusters of constituent forecasts was developed by Aiolfi and Timmermann (2006).

Pruning approaches intend to selectively choose the members of the ensemble in order to eliminate inaccurate and redundant learners, which may reduce both diversity and accuracy of the ensemble. Selective ensembles are believed to be more effective than single learners and traditional ensembles, as a result of smaller ensembles with potentially better generalization ability (Guo et al., 2015; Yin et al., 2015). Ensemble pruning has been an active area of research and numerous pruning algorithms have been proposed (Bhardwaj et al., 2016). They can be categorized as Search-based, Ranking-based, Optimization-based, and Statistics based approaches.

Wang et al. (2003) apply a pruning mechanism that excludes base models when their accuracy becomes worse than a random classifier. Bhardwaj et al. (2016) highlighted the importance of the size of ensembles, due to computational speed and storage issues, and proposed a metric that considers not only the accuracy but also the size of the ensemble. They argue that in some applications, the size is important and shorter models may be desirable at the expense of accuracy. The developed metric evaluates the cost-effectiveness of an ensemble biased to accuracy and also allows that more importance is given for the size of the ensemble when required.

2.3. ENSEMBLE APPROACHES FOR CONCEPT-DRIFTING DATA STREAMS

Wang et al. (2003) introduced a weighted ensemble classifier to address data stream mining and concept drift. They emphasise the advantage of their approach compared to single classifiers in terms of accuracy, efficiency and ease of use. The classifiers are trained sequentially from chunks of data. The criterion to discard data is not based on time of arrival, i.e. old models are replaced, but base on the class distributions that better represent the current concept. In the approach developed by Fan (2004), the new models are built based on the last chunk of data and a combination of new data and old data. The old data are composed of a selection of examples from past chunks. Fan (2004) also highlighted the problem of data

insufficiency, where the use of additional data from previous chunks improves the model accuracy when concept drift is not present.

An approach developed by Gao et al. (2008) trains a new classifier at each new chunk of data. Besides keeping the model up to date with the latest concept, a sampling mechanism allows the model to deal with unbalanced datasets, where the number of data points that belongs to one class is much larger than the number of data points in other classes. Another method that trains a new model for every new chunk of data to cope with data evolution is presented by Masud et al. (2008). The classification is performed using k-NN (k-Nearest Neighbours) as base models and is designed to be effective in problems with a limited amount of labelled data. Furthermore, this approach also incorporates a novel class detection mechanism based on clustering. In both algorithms, the new model is incorporated into the ensemble based on its accuracy in modelling the current concept.

Two variants of Bagging were introduced by Bifet et al. (2009), ADWIN Bagging and Adaptive-Size Hoeffding Tree (ASHT) Bagging. While both algorithms deal with classification tasks, the first one adapts the concept drift using a drift detector, and the latter takes advantage of the incremental property of Hoeffding Trees to restart the trees according to its size and keep the ensemble updated. Elwell and Polikar (2011) developed an incremental learning algorithm to solve classification problems in nonstationary environments. The algorithm trains a new classifier for each new chunk of data and uses a dynamically weighted majority voting scheme in order to cope with concept drift. An adaptive ensemble that is not only able to deal with concept drift but also is capable of detect new classes is presented by Farid et al. (2013). The authors trained three Decision Trees (DT) in a boosting manner, i.e. creating subsets of the training data based on instance weighting. A new DT is trained for each new data chunk, and this new tree can replace one of the existing trees based on accuracy criterion. The novel class detection is performed by a clustering mechanism in the tree leaves.

An ensemble of ensembles is proposed by Yin et al. (2015). They argue that while in the traditional batch growing ensemble methods all the previous ensembles are discarded, their approach takes advantage of them for the final decision. Since the previous ensembles are composed of the same classifiers minus the last trained

classifiers, the combination of ensembles is performed through the weights of previous ensembles. Ren et al. (2018) aggregated the operators of online ensembles and chunk-based ensembles to develop an ensemble classifier that is able to manage different types of drift and a limited number of labelled data. Iwashita et al. (2019) tackled classification in drifting data streams using ensembles of Optimum-Path Forest (OPF) with different approaches for training and updating the OPFs, i.e. full-memory, no-memory and window-of-fixed-size. The base models are combined using three voting mechanisms: Combined, Weighted and Major.

In the context of data stream regression learning, only a few research papers have been published in the literature (Ding et al., 2017). Despite the success of batch growing ensembles achieved in data stream classification, in general, regression ensemble algorithms use iterative strategies. The Additive Expert Ensemble (AddExp) was developed to deal with online classification tasks with concept drift (Kolter and Maloof, 2005). However, the authors argue that this approach can be further extended to also deal with regression problems. AddExp relies on incremental algorithms, i.e. algorithms that adapt to every new instance. In the case of regression tasks, an online version of least squares regression is adopted as base learner. In order to control the size of the ensemble, two pruning strategies were evaluated, i.e., oldest first (the oldest model is excluded) and weakest first (the weakest model is excluded). The latter proves a better pruning choice. This approach works under the assumption that there is no change in the output distribution, since it is designed to make predictions in the interval [0, 1], and this assumption would be easily violated in practical applications. The AddExp also relies on a threshold parameter that determines when new experts should be added to the ensemble, which may be especially difficult to adjust in noisy datasets.

Kadlec and Gabrys (2011) developed an algorithmic soft sensor, i.e. simulating the sensor's output, based on iterative Recursive Partial Least Squares (RPLS) model, called ILLSA (Incremental Local Learning Soft Sensing Algorithm). The ensemble is built using partitions of historical data. In order to cope with concept drift, the ensemble is updated in two levels. At the local level, the RPLSs are updated using the new data, and at the global level, the model's weights are updated according to its performance. Another incremental online ensemble algorithm for regression based on Partial Least Squares, the OWE (Online Weighted Ensemble)

algorithm, was proposed by Soares and Araújo (2015a). It updates the ensemble weights at the arrival of each new data sample based on the error on a sliding window of data. The training of new models considers the error of the ensemble in each sample of the current data window using a boosting strategy. It also retains information about old data windows in the hope that this information could be useful in case of recurrent concept drift.

Soares and Araújo (2015b) also developed another sliding window-based ensemble, the Dynamic and Online Ensemble Regression (DOER). DOER uses OS-ELM (Liang et al., 2006), which is a type of NNRW, as base models. The updating approach is based on an overlapping sliding window, and at each new data sample, all the base models are re-trained and the weights of each model are updated. The approach also considers a mechanism that replaces under-performing models when the accuracy of the ensemble decreases.

Two algorithms based on online Hoeffding-based regression trees (Ikonomovska et al., 2011b), namely OBag (Online Bagging) of Hoeffding-based trees for regression and ORF (Online Random Forest) for any-time regression are presented by Ikonomovska et al. (2015). The models are constructed using online bagging meta-algorithm and learn in an incremental fashion. The adaptation to concept drift is performed by replacing the less accurate models when a significant increase in error is detected.

The main problem with iterative approaches is the fact that, in general, all new samples are presented to the base models, which could result in a higher correlation between the base models and consequently lower diversity of the ensemble. The diversity among the models is responsible for uncorrelated predictions that lead to improved accuracy. Several authors have highlighted the importance of ensemble diversity (Tumer and Ghosh, 1996; Liu and Yao, 1999; Brown et al., 2005; Rokach, 2010; Alhamdoosh and Wang, 2014; Ding et al., 2017).

More recently, regression of sequential data stream is addressed by Ding et al. (2017), who proposed the O-DNNE (Online Decorrelated Neural Network Ensemble). Their algorithm is an online version of the DNNE (Decorrelated Neural Network Ensemble) (Alhamdoosh and Wang, 2014), which is based on a decorrelation strategy (Bruce, 1996) and the negative correlation learning (Liu and

Yao, 1999). DNNE is an ensemble of NNRWs that trains all base models simultaneously and considers the correlation among them in the optimisation process. This method allows that fewer models are required to build the ensemble since redundant models are avoided; however, the training and updating process may become computationally cumbersome, especially when a large number of models and/or a large number of hidden nodes are required, as shown in section 3.3. Additionally, base models with convergence problems due to the choice of the random weights are kept in the ensemble since no pruning mechanism is provided.

A summary of the ensembles approaches for data stream classification and regression in the presence of concept drift is presented in Table 1, in chronological order.

Table 2.1: Ensemble approaches developed to deal with data streams in the presence of concept drift.

Authors (year)	Task	Strategy
Wang et al. (2003)	Classification	Batch growing ensemble using each chunk of data to build a new model
Fan (2004)	Classification	Batch growing ensemble using selected past data to build new models
Kolter and Maloof (2005)	Classification	Ensemble-based on incremental algorithms to adapt to every new instance. New models are added according to a threshold parameter and excluded based on age or accuracy.
Gao et al. (2008)	Classification	Batch growing ensemble and sampling mechanism to deal with unbalanced datasets
Masud et al. (2008)	Classification	Batch growing ensemble designed to deal with limited labelled data and novel class detection
Bifet et al. (2009)	Classification	Fixed ensemble that uses drift detector and restarting Trees to update the model.
Elwell and Polikar (2011)	Classification	Batch growing ensemble that updates using a dynamically weighted majority voting scheme
Kadlec and Gabrys (2011)	Regression	Fixed ensemble based on PLS with local and global updating.
Farid et al. (2013)	Classification	Fixed ensemble that trains new models based on optimised data selection and detects new classes based on clustering.
Ikonomovska et al. (2015)	Regression	Incremental Hoeffding-based regression trees built based on bagging and low performing models are excluded.
Soares and Araújo (2015a)	Regression	PLS models are updated at every new instance. Each model is weighted according to its accuracy on a sliding window
Soares and Araújo (2015b)	Regression	The models (ELM variant) are updated at every instance, and the weights are updated based on accuracy on a sliding window
Yin et al. (2015)	Classification	Combination of ensembles that builds a new ensemble at each new chunk of data
Ding et al. (2017)	Regression	NNRW models trained using decorrelation learning that can be updated at each instance or by chunk
Ren et al. (2018)	Classification	Bach growing ensemble that incorporates drift detection mechanisms and applies online and chunk based updating mechanisms to cope with various types of drift
Iwashita et al. (2019)	Classification	Bach growing ensemble using OPF base classifiers that consider approaches to training the new models (full-memory, no-memory and window-of-fixed-size)

2.4. REMARKS

- Data streams impose several challenges for ML algorithms, such as time and memory restrictions, which requires computationally effective algorithms; and the evolving nature of data, which requires algorithms that can be effectively updated.

- The ensemble approaches have been successfully applied in solving data stream regression and classification problems. They offer a number of possibilities that allow them to effectively adapt to the evolving nature of the data.

- Passive updating approaches generally rely on drift detection mechanism that may generate false reports and prevent the model to improve its accuracy due to data insufficiency.

- Active updating approaches tend to be more effective; however, the updating frequency may have an important impact on the algorithm's accuracy.

- There is a lack of research toward stream regression problems.

- Effective ensembles require computationally effective base models that offer some degree of instability to produce diversity. An algorithm for regression that meets these requirements is the NNRW, which will be covered more deeply in Chapter 5.

- The base model optimisation is not discussed in the existing literature on ensembles. The model optimisation could not only enhance the ensemble's accuracy but also reduce its size and therefore improve its computational efficiency. This issue is discussed in chapter 6.

In the next Chapter (3), the methodology describing how the research gaps are going to be addressed are presented.

3. THESIS DEVELOPMENT METHODOLOGY

The literature review showed that there is a gap in the development of algorithms for regression data streams that can be computationally efficient to deal with high dimensional datasets, achieve state-of-the-art accuracy, and effectively deal with the various types of concept drift. To this end, this research aims at developing an ensemble of NNRWs to cope with this task. Before the development of the ensemble approach, some issues need to be addressed.

First, it is required sets of data where the approach can be effectively evaluated and the assumptions about the capabilities of the algorithm can be assessed. Many authors had discussed the assumption that real-world data are in general not stable and evolve over time (Hofer and Krempl, 2013; Yaqoob et al., 2016, Gomes et al., 2017). However, it is not possible to assure in real-world datasets when concept drift is happening and which type(s) of concept drift is(are) taking place. This requires synthetic datasets where the various types of concept drift can be simulated to allow the evaluation of the algorithm's responsiveness to concept drift. Additionally, there is a need for synthetic data generated on high dimensional spaces to assess the computational efficiency of the proposed methods.

To cope with this, in Chapter 4, a study on synthetic datasets is carried out. The existing methods for synthetic data generation are discussed. A novel data generation approach is proposed based on existing functions for evaluation of optimisation algorithms. This approach allows not only generating data on potentially unbounded dimension spaces but also simulating the various types of drifts reported in the literature. The proposed algorithms are also assessed on real datasets, which are also outlined in Chapter 4.

Second, an accurate and fast learning base model is needed to cope with the proposed task. It is also required that the base model falls into the class of weak models in order to achieve good ensemble diversity. The use of NNRW's meets these requirements; however, a fundamental question arises: How to build an effective NNRW? The answer to this question is elaborated in chapters 5 and 6. In Chapter 5, the structure of the main representatives of NNRW is discussed and the design and tuning questions are pointed out. In chapter 6, the NNRW optimisation is

carried out, not only in terms of structural decisions but also in terms of tuning hyperparameters. For NNRW optimisation, after a review of the existing hyperparameter tuning approaches, a new hyperparameter tuning algorithm based on sums of squares is proposed. The effectiveness of the proposed algorithm is compared to a metaheuristic algorithm, the Genetic Algorithm (GA).

The core of this research, the development of the ensemble for data stream regression in the presence of concept drift, is developed in Chapter 7. It uses as base models the NNRWs studied in Chapter 5, which were optimised using the hyperparameter tuning algorithm developed in Chapter 6. The resulting algorithm is validated using the data sets and data generation approach discussed in Chapter 4. Figure 3.1 summarises the main steps for the development of this research.

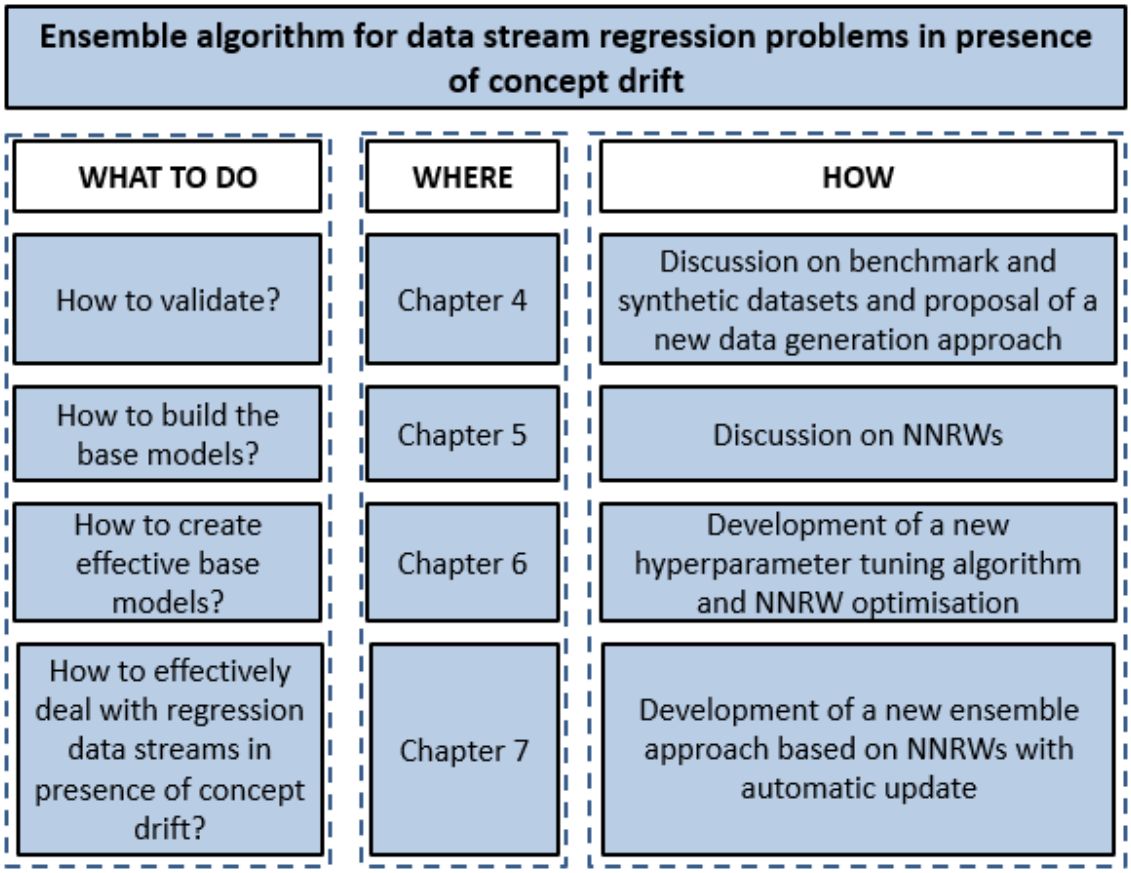

Figure 3.1: Research steps representing the methodology adopted in this research.

4. DATASETS

In this chapter, the datasets used for validating the algorithms developed in this thesis are detailed. The datasets used in this research are mainly divided into two categories: benchmark datasets and synthetic datasets. The benchmark datasets refer to real-world datasets from practical applications in various domains, found in public data repositories. Given the nature of this research and the fact that it is not possible to make assumptions about the data from practical applications, the use of synthetic data is necessary. It allows for the simulation of the various types of drifts reported in the literature.

After a review of the main approaches for synthetic data generation, a need for an approach to generate high dimensional regression data that can effectively simulate various types of drifts was identified. In the remainder of this chapter, a literature review with the work developed for synthetic datasets is presented, followed by a new methodology for regression problems data generation. A description of the benchmark datasets used in this research and the data preprocessing applied for all datasets close the chapter.

4.1. A REVIEW ON SYNTHETIC DATASETS

The use of benchmark datasets is widely regarded as a way to evaluate and compare ML algorithms. Several data repositories are available on the Internet, and one of the most widely used is the UCI Machine Learning Repository (Dua and Graff, 2019). The use of benchmark datasets allows researchers to evaluate different versions of their algorithms and compare the results against previous research. However, it is not possible to make any assumptions about the properties of the real data, such as trends, noise or stationarity, since they are generally unknown.

The use of synthetic datasets enables controlled experiments (Shaker and Hullermeier, 2015). It is possible to build bench test data that perform specific behaviours, such as controlled levels of noise, the inclusion of irrelevant features, changes in the distribution of variables, and types of dependence among targets

(Read et al., 2012). These allow researchers to assess model assumptions and mechanisms developed for specific tasks, such as feature selection or concept drift detection. Some advantages of the use of synthetic datasets include: easy to reproduce, low cost of storage and transmission, as well as knowledge of the ground truth about the data (Bifet et al., 2009; Sun et al., 2016).

In the context of concept drift, the use of synthetic datasets is particularly convenient, since it is not possible to assure the presence and/or type of concept drift in real datasets. When evaluating drift detection mechanisms, relevant change detection measurements such as the *probability of true change detection*, *probability of false alarms* and *delay of detection* (Gama et al., 2014), can only be assessed on synthetic datasets where the points of change are known. Many researchers rely on synthetic datasets and several strategies have been developed for regression and classification problems.

One of the first synthetic datasets reported in the literature are LED and Waveform (Breiman et al., 1984), developed to evaluate classification algorithms. LED consists of a 7-dimensional binary vector, where “1” means that the light in respective position is on and “0” means that the light off. The combination of a given vector represents a number on a digital clock. The task is to classify the true number based on the vector, considering a faulty device that inverts the value of each vector position with a 10% chance. The authors also suggest a variation of the problem adding noise variables. It has been popular within ML community to evaluate classification algorithms (Bifet et al., 2009; Brzezinski and Stefanowski, 2014; Brzezinski and Stefanowski, 2014b; Jiang et al., 2015; Pietruczuk et al., 2017). Furthermore, Sun et al. (2016) simulate concept drift by interchanging the relevant variables.

The Waveform dataset is a more complex example and consists of a classification problem with three classes, where each class is based a convex combination of two out of three different waveforms, resulting in a 21-d vector. A version with a 40-d vector is also studied, where 19 irrelevant attributes are added. Some research that relies on this strategy to generate data include Breiman, 1996; Bifet et al., 2009; Brzezinski and Stefanowski, 2014; Sun et al., 2016; and Pietruczuk et al., 2017.

Schlimmer and Granger Jr. (1986) suggested the use of a dataset based on three features (size, colour and shape), with three levels each. Three different definitions of the concept were defined: (1) *size* = *small* and *colour* = *red*, (2) *colour* = *green* or *shape* = *circular*, and (3) *size* = *(medium or large)*. They used the dataset to evaluate how their algorithm reacts when a change in the definition happens, e.g. definition (1) is switched to (2). Some researches that make use of this dataset are Bifet et al. (2009) and Ghazikhani et al. (2013). Sun et al. (2018) applied a slightly different concept drift, where the rules are modified instead of replaced.

In the field of regression, Friedman (1991) used nonlinear functions (Eqs. 4.1 and 4.2) to generate a synthetic dataset to evaluate a method for regression modelling of high dimensional data.

$$y = 0.1e^{4x_1} + \frac{4}{1 + e^{-20(x_2 - \frac{1}{2})}} + 3x_3 + 2x_4 + x_5 + 0 * \sum_{i=6}^{10} x_i + \varepsilon \tag{4.1}$$

$$y = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 + 0 * \sum_{i=6}^{10} x_i + \varepsilon \tag{4.2}$$

Each variable of the feature vector is independently generated in the unit hypercube, i.e. in the interval [0, 1], following a uniform distribution. As can be observed, the last 5 attributes are irrelevant for the output. The noise ε follows a Gaussian distribution with 0 mean and variance equal to 1. Some researches rely on the function described in Eq. 4.2 and its variations for data stream regression, such as Breiman (1996), Ikonomovska et al. (2011a) and Ikonomovska et al. (2011b). The latter simulates drift by changing the domain of input variables, changing function parameters, and misplacing variables. Nadungodage and Xia (2014) applied several variations of a function similar to Eq. 4.2 to simulate abrupt and gradual drift by replacement of the concepts (functions). Furthermore, ensemble learning using Friedman functions were studied in Hansen (2000) and Chen and Yao (2009).

Karalic (1992) suggested three functions to generate data for regression problems that accommodate the use of categorical variables. The categorical

variable conditions the function that generates the target variable, changing the function coefficients and therefore creating different hyperplanes according to the value of the categorical attribute. The datasets, LINE, LEXP and LOSC, eq. 4.3, 4.4 and 4.5, respectively were used to evaluate regression trees and were also used by Ikonomovska et al. (2011b).

$$f(x) = \begin{cases} 1 + 2x_2 + x_3, & if\ x_1 = v_1 \\ -4 - 2x_2 - x_3, & if\ x_1 = v_2 \end{cases} \tag{4.3}$$

$$f(x) = \begin{cases} 1 + 2x_2 + 3x_3 - e^{-2(x_4 + x_5)}, & if\ x_1 = v_1 \\ 1 - 1.2x_2 - 3.1x_3 + e^{-3(x_4 + x_5)}, & if\ x_1 = v_2 \end{cases} \tag{4.4}$$

$$f(x) = \begin{cases} 1 + 1.5x_2 + x_3 + \sin(2(x_4 + x_5))e^{-2(x_2 + x_4)}, & if\ x_1 = v_1 \\ -1 - 2x_2 - x_3 + \sin(3(x_4 + x_5))e^{-3(x_3 - x_4)}, & if\ x_1 = v_2 \end{cases} \tag{4.5}$$

Agrawal et al. (1992) proposed a classification dataset that consists of 9 attributes, from which 3 are categorical and 1 is derived from 2 other attributes. The training instances are randomly created and the class label of each instance is given by 5 different functions of increasing complexity (Bifet et al., 2009; Pietruczuk et al., 2017).

Synthetic data for classification problems, built based on DTs, are proposed by Domingos and Hulten (2000). The instance space is composed of 100 binary attributes. For each tree level, a number of nodes are replaced by leaves and the rest are split using an attribute chosen randomly. At a given depth, the splitting process stops and all the remaining growing nodes become leaves. To each leave, it is assigned a class in a random manner. A similar idea is applied by Ikonomovska et al. (2011a), using 10 attributes to build synthetic datasets for regression problems. Classification algorithms that used this technique as bench test include Read et al. (2012), Jiang et al. (2015), Shaker and Hullermeier (2015), and Pietruczuk et al. (2017). Brzezinski and Stefanowski (2014a) and Brzezinski and Stefanowski (2014b) simulate concept drift by alternating different trees.

Besides Friedman functions, Hansen (2000) also used the functions SinC (eq. 4.6), Gabor (eq. 4.7) and Multi (eq. 4.8) to evaluate ensembles:

$$y = \frac{\sin(x)}{x} \tag{4.6}$$

$$y = \frac{2}{\pi}\exp[-2(x^2 + y^2)]\cos[2\pi(x + y)] \tag{4.7}$$

$$y = 0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5 \tag{4.8}$$

These functions were also applied by Chen and Yao (2009) to evaluate negative correlation learning for NN ensembles.

A synthetic dataset based on *d*-dimensional hyperplane is suggested by Hulten et al. (2001), following the form showed in eq. 4.9.

$$\sum_{i=1}^{d} w_i x_i = w_0 \tag{4.9}$$

where the vector w represents the hyperplane coefficients and the vector x represents the variables, uniformly distributed in the interval [0, 1]. The instances are labelled positive when $\sum_{i=1}^{d} w_i x_i \geq w_0$ and negative otherwise. This setting not only allows the control of the importance of each variable (e.g. a weight = 0 means the corresponding variable does not contribute to the output) but also allows simulating concept drift by rotating the hyperplane through its weights, which results in conditional change (Wang et al., 2003; Fan, 2004; Gao et al., 2008; Bifet et al., 2009; Ghazikhani et al., 2013; Brzezinski and Stefanowski, 2014a; Brzezinski and Stefanowski, 2014b; Jiang et al., 2015; Sun et al., 2016).

Another approach to simulate the conditional change is applied by Shaker and Hullermeier (2015). Instead of rotating the hyperplane, two datasets are created, each based on a different set of weights. The drift occurs by replacing the stream of instances from the first dataset by the instances of the second, with a probability given by a sigmoidal function. A similar approach is applied by other authors to apply gradual drift in other types of datasets, such as Read et al. (2012), Nadungodage

and Xia (2014), Shaker and Hullermeier (2015) and Ikonomovska et al. (2011b). Nadungodage and Xia (2014) use a linear function to increase the occurrence probability of instances of the replacing concept and also simulate abrupt drift by suddenly replacing the current concept by a new one.

The hyperplanes allow the simulation of feature change by changing the distribution of the instances in the attribute space. Furthermore, by controlling the value of w_0 it is possible to adjust the balance between classes (Gao et al., 2008) and evaluate algorithms designed to deal with imbalanced datasets. Shaker and Hullermeier (2015) used the hyperplane approach not only to generate classification datasets but also to simulate regression problems. Instead of defining in which side of the hyperplane an instance *x* lies on to assign a class, the target is given by absolute distance ($d_1 = |w^t.x|$ and $d_2 = |w^t.x|^3$ - *w* is the normal vector of the hyperplane) from the point *x*, which represents the instance, to the hyperplane.

A simple dataset, (SEA – Streaming Ensemble Algorithm) was proposed by Street and Kim (2001) to evaluate concept drift. Instances with three attributes, from which only 2 are relevant, are randomly generated in the interval [0, 10] and classes are assigned according to the sum of the first two attributes (a_1 and a_2), i.e. class 1 if $a_1 + a_2 \leq \theta$ and class 2 otherwise. They simulate concept drift by changing the threshold θ (Ghazikhani et al., 2013; Brzezinski and Stefanowski, 2014b; Sun et al., 2016; Sun et al., 2018). Noise is introduced by swapping the class of 10% of the instances. Despite its simplicity, many research relies on this dataset to evaluate ML algorithms (Bifet et al., 2009; Brzezinski and Stefanowski, 2014a; Jiang et al., 2015; Pietruczuk et al., 2017).

Polikar et al. (2001) suggested an artificial dataset based on concentrical circles, where each formed ring is assigned a class. The aim is to assess incremental learning, where new rings (classes) are presented to the classification algorithm as time evolves. Oza and Russel (2001) develop a model for classification dataset generation based on binary attributes. The attributes are set based not only on the probabilities of an attribute given a class but also in the probabilities of the next attribute. This way, it is possible to evaluate the effectiveness of the boosting algorithm compared to bagging.

Liu and Zio (2016) generated synthetic datasets for analysis of regression data streams. Concepts are created by summing up different variables and the drift is applied by suddenly swapping the stream of instances from one concept by another. Besides SEA and STAGGER, Sun et al. (2018) also used classification datasets based on rotating, circle and sine concepts, as shown in Eqs. 4.10, 4.11 and 4.12, respectively. The drift is applied by changing the value of θ.

$$\begin{cases} x_1 \leftarrow (x_1 - a) * cos\theta - (x_2 - b)sin\theta + a \\ x_2 \leftarrow (x_1 - a) * cos\theta + (x_2 - b)sin\theta + b \end{cases} \tag{4.10}$$

$$(x_1 - a)^2 + (x_2 - b)^2 \leq / > \theta \tag{4.11}$$

$$\mathrm{asin}(bx_1 + \theta) + c \leq / > x_2 \tag{4.12}$$

Table 4.1 summarizes the main works, organized chronologically, that rely on synthetic datasets to evaluate classification and regression algorithms, along with the strategy applied for data generation and the type of drift when applicable.

Table 4.1: Summary of related works using synthetic datasets (C: Classification, R: Regression).

Authors	Year	Approach	Task	Drift	Types of drift
Domingos and Hulten	2000	Random trees.	C	No	-
Hulten et al.	2001	Hyperplane	C	No	Gradual.
Polikar et al.	2001	Concentric circles.	C	No	-
Oza and Russell	2001	Binary attributes conditionally dependent upon the class label and next attribute value.	C	No	-
Wang et al.	2003	Hyperplane	C	Yes	Gradual.
Fan	2004	Hyperplane	C	Yes	Gradual.
Gao et al.	2008	Hyperplane	C	Yes	Sudden.
Chen and Yao	2009	Mexican Hat, Friedman, Gabor, Multi, Plane, Polynomial, Sinc, Synth, Overlap, Bumpy and Relevance.	C, R	No	-
Bifet et al.	2009	SEA, STAGGER, Hyperplane, RBF, LED, Waveform and Agrawal Generator.	C	Yes	Gradual and Sudden.
Ikonomovska et al.	2011a	Random trees and modified Friedman.	R	No	-
Ikonomovska et al.	2011b	Friedman, Losc and Lexp.	R	Yes	Gradual, Sudden and Recurrent.
Read et al.	2012	Random trees and RBF	C	Yes	Gradual.
Ghazikhani et al.	2013	SEA, STAGGER and Hyperplane.	C	Yes	Gradual and Sudden.
Nadungodage et al.	2014	Friedman based functions.	R	Yes	Gradual and Sudden.
Brzezinski and Stefanowski	2014a	Hyperplane, SEA, Random trees, RBF, LED and Waveform.	C	Yes	Gradual, Sudden, Recurrent, Incremental and Mixed.
Brzezinski and Stefanowski	2014b	Hyperplane, RBF, SEA, Radom trees and LED.	C	Yes	Gradual, Sudden, Recurrent and Mixed.
Shaker and Hullermeier	2015	Random trees and Hyperplane.	C, R	Yes	Gradual.
Jiang et al.	2015	SEA, Random trees, Hyperplane, RBF and LED.	C	Yes	Gradual, Sudden and Recurrent.
Sun et al.	2016	Hyperplane, SEA, LED and Waveform.	C	Yes	Gradual and Sudden.
Liu and Zio	2016	Nonlinear functions.	R	Yes	Sudden and Recurrent.
Pietruczuk et al.	2017	Agrawal Generator, Hyperplane, LED, Random trees, RBF, SEA and Waveform.	C	No	-
Sun et al.	2018	Hyperplane, Rotating, Circle, Sine and Boolean Concepts.	C	Yes	Sudden.

Based on the available techniques for data generation, a need for a regression data generator that is readable at low dimensions and can be easily expanded to an arbitrary number of dimensions was identified. It also needs to accommodate different types of data drift simulations and allow for the use of categorical attributes.

4.2. AN ALTERNATIVE APPROACH FOR DATA GENERATION

The main approaches for regression data generation rely on nonlinear functions. In general, these functions are limited in terms of dimensionality, i.e. number of relevant features, and difficult to simulate the various types of drift. To overcome these issues, in this research it is proposed the use of the functions presented on the CEC (Congress on Evolutionary Computation) 2005 Special Session on Real-Parameter Optimisation functions (Suganthan et al., 2005). The set of functions used in CEC consists of continuous nonlinear hyperplanes with various shapes and the optimisation algorithms are challenged to find the global minimum, i.e. the minimum *f*(*x*). These functions offer a range of features that make them suitable to simulate regression problems, where the task is to predict the value of *f*(*x*). They allow visual inspection for problems with one or two attributes, are easily scalable to any number of attributes and also offer several possibilities to simulate various types of concept drift. Additionally, it is possible to create theoretically infinite length data streams.

From the 25 functions presented in CEC, five were selected, based on empirical analysis, to generate streams of data and assess the algorithms discussed in this thesis: 1) Shifted Sphere (f_1); 2) Shifted Schwefel (f_2); 3) Shifted Rotated High Conditioned Elliptic (f_3); 4) Shifted Rotated Griewank (f_7); 5) Shifted Rotated Weierstrass (f_{11}). Each function is computed according to Eqs. 4.13 – 4.17, respectively.

$$f_1(x) = \sum_{i=1}^{D} (x_i - o_i)^2 + f\_bias \tag{4.13}$$

$$f_2(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j - o_j \right)^2 + f\_bias \tag{4.14}$$

$$f_3(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f\_bias$$

(4.15)

$$f_7(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f\_bias$$

(4.16)

$$f_{11}(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k\,max} [a^k \cos\left(2\pi b^k (z_i + 0.5)\right)] \right) - D \sum_{k=0}^{k\,max} [a^k \cos(2\pi b^k . 0.5)] + f\_bias$$

(4.17)

where o is the coordinate of the global minimum, $z = (x - o)M$, M is an orthogonal matrix in case of F3 and a linear transformation matrix in case of F7 and F11, a=0.5, b=3 and k=20. The shapes of each function for two attributes are shown in Figure. 4.1.

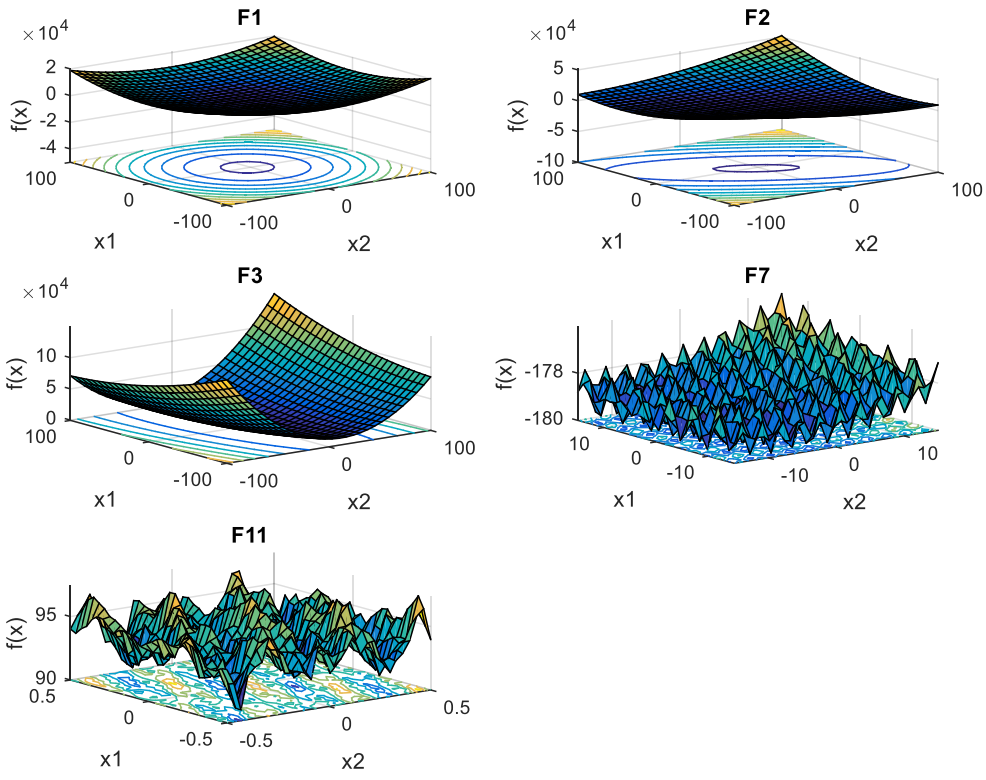

Figure 4.1: 3-Dimensional plots of f_1, f_2, f_3, f_7 and f_{11} CEC functions.

In this thesis, three main types of drift are simulated: gradual drift, abrupt drift (Tsymbal, 2004; Krawczyk et al., 2017) and data expansion (Ikonomovska et al., 2011b), as detailed in Section 2.1. Another common type of drift is the recurrent drift (Krawczyk et al., 2017), i.e. a concept is replaced and after a certain period time, it appears back, such as seasons in weather prediction. This type of drift is simulated in the same manner as the abrupt drift and its explicit evaluation is important when the algorithm possesses long term memory mechanisms. Therefore, in this thesis, recurrent drift is not explicitly evaluated.

***Gradual drift*:** Two strategies are used for gradual drift, hyperplane rotation and function replacement. The former represents a change in the concept itself, analogous to the effect of wear tool in the prediction equipment's performance or the effect of global warming on weather prediction. Gradual drift based on hyperplane rotation is applied by many authors (Wang et al., 2003; Fan, 2004; Gao et al., 2008; Ghazikhani et al., 2013; Brzezinski and Stefanowski, 2014b; Sun et al., 2016). For the selected functions, this can be easily achieved by changing the position of function's global minimum, which causes a rotation in the hyperplane. An illustrative example is shown in Figure. 4.2, where the global minima (o) of function f_1 is moved 2 times, by 50% in each axis in each iteration.

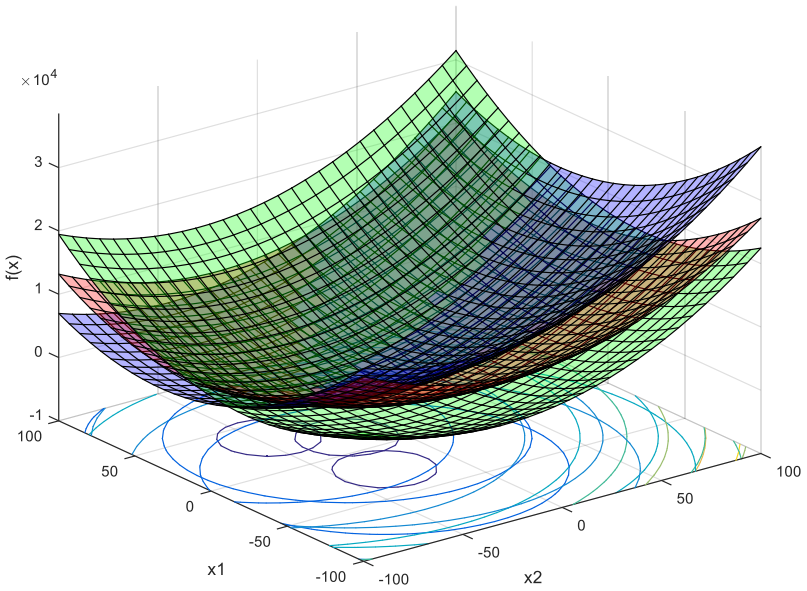

Figure 4.2: Rotation of f_1 function.

The function replacement is another common way to simulate gradual drift (Read et al., 2012; Nadungodage et al., 2014; Shaker and Hullermeier, 2015). Given a stream of data generated according to function *A*, instances generated by function *B* start to appear with a small probability. This probability is gradually increased, up to a point where all the new instances are generated by function *B*. This process is analogous to the market of a given company when the interest of a group of consumers is gradually replaced by the demand from another group.

***Abrupt drift*:** The abrupt drift works based on the same principle of gradual function replacement; however, after the drifting point, all the new instances are generated by a new function (Ghazikhani et al., 2013; Nadungodage et al., 2014; Liu and Zio, 2016), instead of gradually replaced. A practical example of abrupt drift is the imposition of trading tariffs that can suddenly affect companies and/or economies.

***Data expansion*:** data expansion is simulated by changing the bounds of the input space. A related approach is studied in (Ikonomovska et al., 2011b). At some extent, the data expansion can be used to simulate cases where the training data does not represent the process generating data as a whole. For instance, an attribute can be defined in the range [0 7] in the training process, and then expanded to the range [0 10] during the test, in order to evaluate how the algorithm adapts. As an example, data expansion can happen when a system designed to predict houses prices, trained with houses of a given size range, are exposed to houses with sizes beyond the previous range. In this case, several features can be affected (number of rooms, number of bathrooms, neighbourhood average income, etc).

4.3. BENCHMARK DATASETS

In this section, the real-world datasets chosen to evaluate the algorithms presented in this research are introduced. The datasets were chosen not only based on the number of samples, which should be big enough to simulate a stream of data, but also considering the number of features and diversity of application domains. Four benchmark datasets were used in this thesis to evaluate the models, three from a well-known public domain source (UCI data repository -

https://archive.ics.uci.edu/ml/datasets.php): appliances energy prediction, condition-based maintenance and wine quality. The fourth dataset (California housing) is from StatLib repository (lib.stat.cmu.edu). A summary of the main features of each dataset, i.e. the number and type of predictive attributes (A) and the number of data samples (N), is presented in Table 4.2:

Table 4.2: Benchmark dataset features (N - # data samples, A - # features).

Name	N	A
California Housing (Housing)	20640	8
Wine quality (Quality)	4898	11
Condition based maintenance (Maintenance)	11934	14
Appliances energy prediction (Energy)	20640	26

In this chapter, besides the benchmark datasets and data preprocessing, an alternative approach for generating regression datasets is presented. This approach offers a range of possibilities that include: an arbitrary number of predictive attributes, the capability to intuitively simulate various types of drift, and create theoretically infinite data streams. The aforementioned features are important to validate the algorithms presented in this research and the data assumptions they are designed to address.

5. NEURAL NETWORKS WITH RANDOM WEIGHTS

The amount of research on randomised networks has grown immensely in recent years and numerous publications can be found in the literature, especially under the Huang's (2004) terminology, the Extreme Learning Machine (ELM). NNRWs are able to tackle not only classification and regression problems but also feature learning and clustering problems (Huang, 2014; Alaba et al., 2019) in a wide range of applications. Deep Learning techniques, such as auto-encoders and convolutional neural networks, based on NNRWs have also been investigated recently (Cao et al., 2018).

The main appeal of the NNRWs is their simplicity of implementation and high learning speed compared to the traditional NNs. Huang et al. (2004) reported learning speed thousands of times faster than SVMs and NNs trained with Back-Propagation (BP) algorithms; moreover, the good generalisation capability makes the NNRWs a promising technique for many applications. In the remainder of this chapter, an overview of the development of NNRW is presented, followed by an in-depth analysis of its structure. The focus of this analysis is on the randomised version of the Single-hidden Layer Feedforward Neural Network (SLFNN) for regression problems.

This chapter discusses the key elements of the NNRW model from a theoretical perspective, and justify its selection for the ensemble learning approach. Hyperparameter optimisation will be presented in Chapter 6 and the ensemble learning strategies will be developed in Chapter 7.

5.1. DEVELOPMENT OF NEURAL NETWORKS WITH RANDOM WEIGHTS

One of the main representatives of the NNRWs, the RVFL, was proposed by Pao et al. (1992). The main idea was to transform the architecture of an SLFNN into a flat net, where the weights between the input layer and the hidden layer (W_H), as well as the thresholds (B), are generated randomly. Given an input X and a continuous function $g(\cdot)$, the transformation $g(X \cdot W_H + B)$, along with the original

input X (the so-called direct-link) become the inputs of the flat net and only the weights between the new inputs and the output layer are optimised.

In the same year, Schmidt et al. (1992) studied the effects of the hidden layer random parameters in an SLFNN. They found out that output layer weights are significantly more important than the hidden layer weights; however, the experiments were carried out on small datasets. Similar to Schmidt's idea, the ELM algorithm was proposed by Huang et al. (2004), where the main difference was in the optimisation procedure to find the output weights. Whilst ELM applied a Moore-Penrose generalised inverse, Schmidt et al. (1992) used a numerical method.

The theoretical learning capability of NNRW has been demonstrated in several studies. Igelnik and Pao (1995) presented theoretical justification for RVFL and showed that RVFL is a universal approximator of continuous functions. Huang et al. (2006) showed through an incremental constructive method that the ELMs are universal approximators for any continuous target function, given a constant piecewise activation function is provided.

The randomness of the NNRWs is responsible for creating learning instability i.e. two NNRWs with equal structures have different generalisation performances. This was investigated by Fu et al. (2015) through a series of experiments with ELMs. They experimentally confirmed the instability caused by the random initialisation and also found out that instability decreases when the ELMs are combined in an ensemble.

Ding et al. (2014) explored the main ELM variants, which include incremental ELM, pruning ELM, error-minimised ELM, two-stage ELM, online sequential ELM, evolutionary ELM, voting-based ELM, ordinal ELM, fully complex ELM and symmetric ELM. Deng et al. (2015) further explore semi-supervised and unsupervised ELM variants, as well as ELM autoencoders and multilayer ELMs. The use of NNRWs spans over a wide range of practical problems, such as classification, regression, pattern recognition, forecasting and diagnosis, and image processing (Ding et al., 2014). Some interesting applications where NNRWs have been applied successfully include ship detection, image quality assessment and online visual tracking (Deng et al., 2015). NNRW popularity also motivated research investigating efficient implementations methods, such as the one carried out by Martínez-Villena et al.

(2014), who studied the hardware implementation of RVFLs and proposed three computation architectures. This allows the use of RVFLs in embedded real-time systems where the use of personal computers is not possible.

An extensive evaluation of the RVFL hyperparameters is performed by Zhang and Suganthan (2016) for classification problems. Some of their findings are discussed in section 5.3. However, as pointed out by Alaba et al. (2019), finding an effective hidden layer structure still an open problem in the literature. In the next section, the structural elements of the NNRWs are presented, highlighting the fundamental difference between the main representatives, i.e. the RVFL and ELM.

5.2. NNRW ARCHITECTURE

In this section, the main structure of NNRWs is analysed, along with their tuning hyperparameters, aiming at the comprehension of the differences between the main NNRW representatives, i.e. RVFL and ELM, and the effects of the hyperparameters. This analysis is important to create the building blocks for constructing an effective and optimised NNRW. Finding a good NNRW structure is of fundamental importance for an effective NNRW implementation (Scardapane and Wang, 2017). The standard SLFNN structure for regression and its elements are illustrated in Figure 5.1. The fundamental difference to an SLFNN for classification lies in the output layer, where an additional activation function is applied and more than one node can be present.

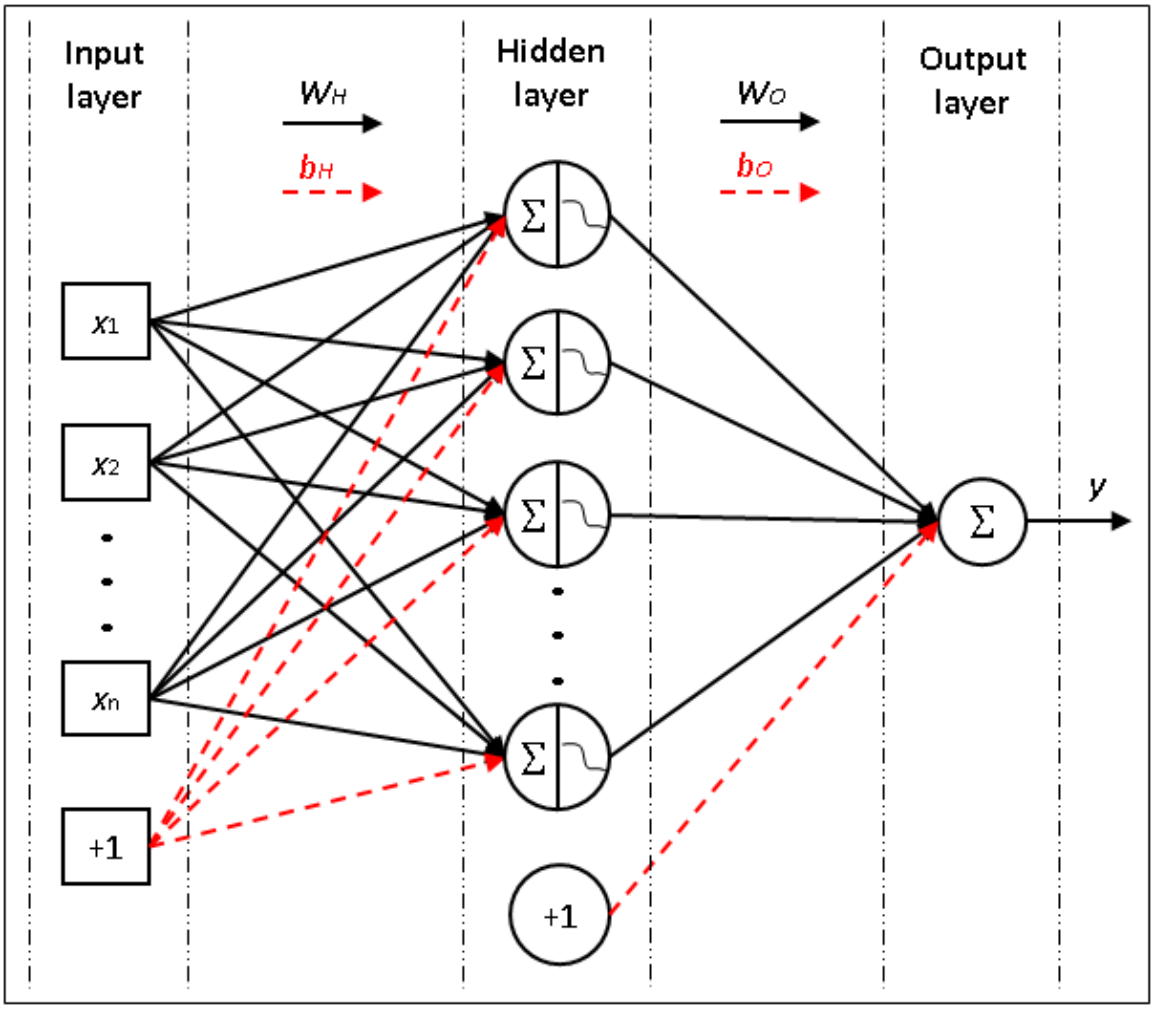

Figure 5.1: Single-hidden-layer feedforward neural network architecture.

In the SLFNN, a prediction *y* is obtained by feeding forward a given pattern x through the network and computing the corresponding operations. Each node i in the hidden layer receives the dot product of the input pattern x and the weights w_{Hi}, connecting the inputs to the respective node. A threshold value b_{Hi} is added to the resulting value and then, an activation function $g(*)$ is applied. The described computation is illustrated in Eq. 5.1:

$$h_i = g(\boldsymbol{x} \cdot \boldsymbol{w}_{Hi} + b_{Hi}) \tag{5.1}$$

The output $\hat{y}$ is obtained by computing the sum of the products of the N hidden layer outputs h_i and the weights connecting the hidden nodes to the output node (w_O). The threshold value b_O is then added to the resulting value, as shown in Eq. 5.2.

$$\hat{y} = \sum_{i=1}^{N} h_i \cdot w_{Oi} + b_O$$

(5.2)

The complete feedforward process can be described according to the Eq. 5.3.

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{N} g(\boldsymbol{x} \cdot \boldsymbol{w}_{Hi} + b_{Hi}) \cdot w_{Oi} + b_O = g(\boldsymbol{x} \cdot \boldsymbol{W}_H + \boldsymbol{b}_H) \cdot \boldsymbol{w}_O + b_O$$

(5.3)

The SLFNN training process consists of adjusting the free parameters $\boldsymbol{W}_H$, $\boldsymbol{b}_H$, $\boldsymbol{w}_O$ and b_O according to an optimisation objective. In general, the objective function is, given a training set $(\boldsymbol{X}, \boldsymbol{y})$, to minimise the error between the predicted vector $\hat{\boldsymbol{y}}(\boldsymbol{X})$ and the true vector $\boldsymbol{y}$**.** The adjustment of the free parameters is characterised by non-convex optimisation, usually performed by BP algorithms, which are iterative and recursive methods based on the chain rule for computing the derivatives. Although many successful applications of SLFNNs with BP algorithm are reported in the literature, some drawbacks may include, slow convergence and local minima. Additionally, adjustment of the learning parameters (learning rate, number of epochs, etc) is not a trivial task and could lead to poor generalisation or overfitting.

Schmidt's work (Schmidt et al., 1992) explored the idea of fixing randomly the weights from the input layer to the hidden layer ($\boldsymbol{W}_H$) and the biases ($\boldsymbol{b}_H$), arguing that these elements are of less importance for the overall performance and optimising only $\boldsymbol{w}_O$ and b_O is sufficient for good generalisation performance. The RVFL (Pao et al., 1992) shares a similar idea; however, in this case, the authors include a *direct link* connecting the inputs to the output node. The resulting architecture of RVFL is shown in Figure 5.2.

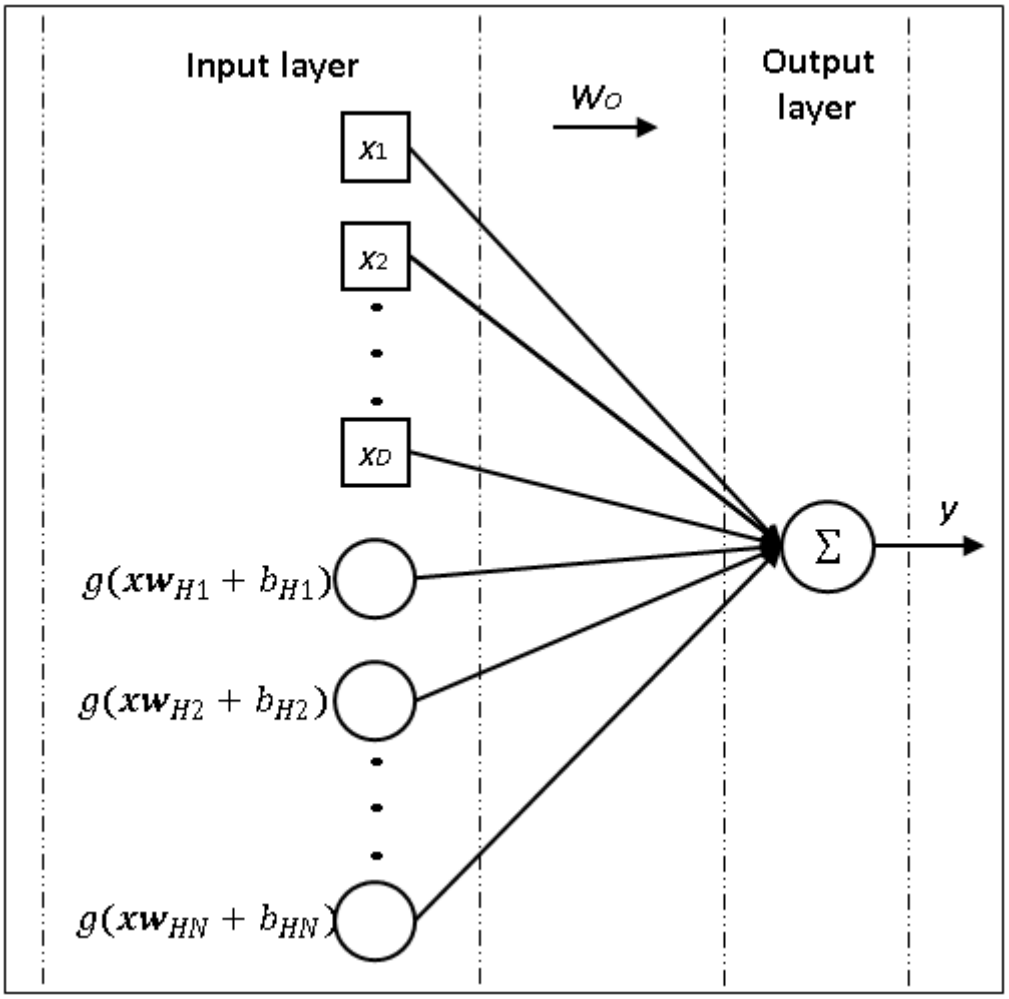

Figure 5.2: RVFL architecture.

The ELM follows the same structure of Schmidt et al. (1992), except for the output bias, which is not present, as in RVFL. The main advantage of the randomisation lies in the learning process, which becomes of convex nature and can be solved by analytical methods. Additionally, since the learning process does not rely on derivatives, as is the case of BP learning algorithms, almost any nonzero activation functions can be successfully applied (Huang et al., 2004).

Given a training set $(\boldsymbol{X}, \boldsymbol{y})$, the output from the hidden layer can be described as Eq. 5.4.

$$\boldsymbol{H} = g(\boldsymbol{X} \cdot \boldsymbol{W}_H + \boldsymbol{b}_H) \tag{5.4}$$

where $\boldsymbol{W}_H$ and $\boldsymbol{b}_H$ are randomly chosen and kept fixed. The function that describes the predicted vector $\hat{\boldsymbol{y}}$ is written as a linear system, according to Eq. 5.5.

$$\hat{\boldsymbol{y}} = \boldsymbol{H} \cdot \boldsymbol{w}_O \tag{5.5}$$

The optimised set of weights $\boldsymbol{w}_O$ is the one minimises the difference that $\boldsymbol{y} - \hat{\boldsymbol{y}}$, and the optimisation function can be described as 5.6.

$$min \, \|\boldsymbol{y} - \hat{\boldsymbol{y}}\| = min \, \|\boldsymbol{y} - \boldsymbol{H} \cdot \boldsymbol{w}_O\| \tag{5.6}$$

The optimisation algorithm applied by Schmidt et al. (1992), referred to as Fisher solution, can be written as Equation 5.7:

$$\boldsymbol{w}_O^* = (\boldsymbol{H}^T \cdot \boldsymbol{H})^{-1} \cdot \boldsymbol{H}^T \cdot \boldsymbol{y} \tag{5.7}$$

which is equivalent to the Least Squares (LS) estimator. The computation of $(\boldsymbol{H}^T \cdot \boldsymbol{H})^{-1}$ may lead to instability if $\boldsymbol{H}^T \cdot \boldsymbol{H}$ is singular or nearly singular. This issue can be addressed using the ridge regression, introduced by Hoerl and Kennard (1970), which consists of small positive quantities added to the diagonal of $\boldsymbol{H}^T \cdot \boldsymbol{H}$ (Equation 5.8).

$$\boldsymbol{w}_O^* = (\boldsymbol{H}^T \cdot \boldsymbol{H} + \lambda \cdot \boldsymbol{I})^{-1} \cdot \boldsymbol{H}^T \cdot \boldsymbol{y} \tag{5.8}$$

where λ is a small constant value and $\boldsymbol{I}$ is the identity matrix. The λ is also known as a regularisation factor since it penalises large weights in the optimisation process. Alternatively, one can rely on the Moore-Penrose pseudo-inverse (Huang et al., 2004), as used in ELM, as described in 5.9.

$$\boldsymbol{w}_O^* = \boldsymbol{H}^\dagger \cdot \boldsymbol{y} \tag{5.9}$$

where $\boldsymbol{H}^\dagger$ refers to the Moore-Penrose pseudo inverse (Huang et al., 2004). In this research, the ridge regression method is applied since preliminary results, not reported in this thesis, showed a better generalisation capability compared to the Moore-Penrose approach. The advantage of ridge regression was also observed by Zhang and Suganthan (2016).

In the next section, the design and tuning decisions involved in the construction of the NNRWs are detailed.

5.3. NNRW HYPERPARAMETERS

The hyperparameters that affect the NNRWs can be mainly divided into two categories: the design hyperparameters and the tuning hyperparameters. In this research, the NNRW design hyperparameters are described as the ones that affect the architecture of the NNRW, which include the number of nodes, use of the output bias and use of the direct link. The tuning hyperparameters include the random weights and bias scaling factor, the regularisation factor and the activation function.

Number of nodes (*N*): Number of hidden nodes in the hidden layer. Zhang and Suganthan (2016) evaluated values from 3 to 203 nodes, with a step size of 20, however, the number of nodes were optimised separately from the remaining hyperparameters. Two variants of ELM try to automatically establish the number of nodes: the pruning ELM (Rong et al., 2008, Miche et al., 2010) and the incremental ELM (Huang et al., 2006), however, these methods require additional steps to determine the most adequate number of hidden nodes, which are dependent on the random weights initialisation.

Regularisation factor (*R*): The regularisation factor is responsible for penalizing large weights in the ridge regression optimisation process. The set of values analysed by Zhang and Suganthan (2016) ranged from 6E-5 to 32.

Scaling factor (*S*): This hyperparameter determines the interval in which the weights between the input layer and hidden layer are initialized. The weights are randomly generated from a uniform distribution and kept fixed afterwards. A commonly used approach is to generate *W* from a uniform distribution within the interval [-1 1] (Schmidt et al., 1992; Pao et al., 1992; Huang et al., 2004; Ding et al., 2017). The scaling factor is multiplied by the random weights set and changes its distribution interval. The effect of initial weights in RVFL was investigated by Zhang and Suganthan (2016), where the authors showed that the adjustment of the scaling factor produces statistically differences in the algorithms' performance.

Activation function (*A*): A nonlinear function applied to the hidden nodes. A very popular activation function is the sigmoid function (Schmidt et al., 1992; Pao et al., 1992; Huang et al., 2004; Ding et al., 2017). Huang (2014) also mention other nonlinear piecewise continuous functions, such as Fourier (sine), Hardlimit and

Gaussian. Zhang and Suganthan (2016) evaluated the effects of the Sigmoid (Eq. 5.10), Sine (Eq. 5.11), Hardlimit (Eq. 5.12), Tribas (Eq. 5.13), Radbas (Eq. 5.14) and Sign (Eq. 5.15) functions and found out that the Radbas function achieved better performance. In this research, besides the functions evaluated in Zhang and Suganthan (2016), two other functions are assessed: the hyperbolic tangent sigmoid (tansig - Eq. 5.16) and the rectifier linear unit (relu - Eq. 5.17). The former is a popular activation function used in NNs while the latter has become one of the most used activation functions in deep learning applications.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{5.10}$$

$$g(x) = \sin(x) \tag{5.11}$$

$$g(x) = \begin{cases} 1, & x \geq 0 \\ 0, & otherwise \end{cases} \tag{5.12}$$

$$g(x) = \max(1 - |x|, 0) \tag{5.13}$$

$$g(x) = \exp(-x^2) \tag{5.14}$$

$$g(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & otherwise \end{cases} \tag{5.15}$$

$$g(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{5.16}$$

$$g(x) = \begin{cases} x, & x > 0 \\ 0, & otherwise \end{cases} \tag{5.17}$$

Direct link (*D*): It refers to the links connecting the input nodes to the output node and is applied in RVLF applications. It is not considered in the popular ELM algorithm, however, the study of Zhang and Suganthan (2016) suggests that the use of direct link enhances the accuracy of randomized SLFNs.

Output bias (*Ob*): It is a threshold value applied to the output node. One of the learning principles of ELM, elaborated by Huang (2014), states that the output nodes should have no bias, while Zhang and Suganthan (2016) did not find significant differences in accuracy when the output bias is used. Preliminary experiments performed in this research showed that the output bias is an important factor in NNRW performance for some datasets.

In this chapter, after a brief review of randomised NN algorithms, the structure of SLFNN with random weights was demonstrated and the design decisions involved in building an effective NNRW were discussed. The main representatives of NNRWs are the RVFL and the ELM and the fundamental difference between them is in the use of the direct link in RVFL. The main advantage of NNRWs is their lower training complexity compared to BP algorithms, which allows finding the optimal set of parameters in a fraction of the time and avoids getting stuck in local minima. Additionally, NNRWs show good accuracy and are easy to implement for both regression and classification problems. The tuning of NNRW's hyperparameters is an important factor in its performance. A previous study carried out by Zhang and Suganthan (2016) gave an overview of the hyperparameter's effect on NNRW's performance. In their study, they found that the RVFL's direct link, not present in ELMs, is responsible for enhancing the NNRW accuracy.

In the next chapter, the hyperparameter tuning of the NNRW is performed on the datasets used in this research. To this end, a new hyperparameter optimisation algorithm is presented and benchmarked against a popular optimisation algorithm, the GA. The optimised NNRW will be used as based models for building the ensembles demonstrated in Chapter 7.

6. HYPERPARAMETER OPTIMISATION

This chapter aims at finding optimised NNRW settings for the use in the ensemble, in Chapter 7. As any other ML algorithm, the NNRW rely on the adjustment of several hyperparameters, which must be set by the user and plays an important role in the algorithm's performance. Although some default settings for ML algorithms can be found in literature or implementation packages, the diversity of problems and applications make it difficult to think of a *one fits all* solution. Finding a good set of hyperparameters is not a trivial task and may require not only expert's experience but also an extensive process of trial and error, or, as some authors refer to, a *black art* (Snoek et al., 2012; Smith, 2018).

Before moving to the NNRW ensemble analysis in the data stream environment, it is important to analyse the NNRW's hyperparameters and find good settings that help improve the overall performance of the ensemble. To this end, a new hyperparameter optimisation algorithm is proposed in this research. The new algorithm is based on the analysis of properties of Design of Experiments (DOE), a widely used tool for process optimisation, which allows a systematic evaluation of not only the effect and importance of each hyperparameter but also the effect and importance of the interactions among them.

In the remainder of this chapter, a literature review explores the developments in the field of hyperparameter optimisation in section 6.1, followed by the description of the methodology in section 6.2. The experimental protocol is presented in section 6.3 and the results and discussion, in section 6.4 close this chapter.

6.1. A BRIEF REVIEW ON HYPERPARAMETER OPTIMISATION

Usually, ML algorithms have several hyperparameters and their adjustment are an important aspect to be taken into consideration. A proper adjustment of hyperparameters is key to achieve superior performance of ML algorithms and is related to the characteristics of the dataset (Di Martino et al., 2011). A popular approach for hyperparameter tuning is Grid-search, where sets of values for each

hyperparameter are defined by the user and all the combinations are evaluated. This approach can be time-consuming and leads to searching over not promising regions of the search space. Additionally, in case of continuous variables, the search is limited to the pre-defined values, which may be time-consuming for high granularity or ineffective for low granularity of the variable's values. Some reasons for the popularity of Grid-search are highlighted by Bergstra and Bengio (2012), which include simplicity to implement, trivial parallelisation, and usually better results than purely manual optimisation.

A more effective technique, the Random Search (RS), was presented in Bergstra and Bengio (2012). Different from Grid-search, instead of evaluating all hyperparameter combinations, in RS the combinations are selected randomly and the values of continuous variables are defined based on user-defined distributions, avoiding granularity issues. The authors demonstrated that the RS avoids exploring non-promising search space and achieve competitive results compared to Grid-search.

Some approaches apply sequential model-based optimisation (SMBO) techniques. Bergstra et al. (2011) proposed two greedy SMBO methods for tuning NNs and Deep Belief Networks (DBNs). The proposed method outperformed RS in tuning DBNs but showed similar results for NNs. Thornton et al. (2013) apply SMBO for hyperparameter tuning as part of a broader system that aims not only hyperparameter tuning but also model selection. The approach simultaneously evaluates different classification models and hyperparameter settings along with feature selection methods.

SMBO methods require the setting of a surrogate function. This function will indicate the regions to be explored by the algorithm, as well as the optimisation criterion. These choices may highly influence not only the number of iterations for convergence but also the quality of the results. Another drawback of SMBO methods, highlighted by Maclaurin et al. (2015) is the inability to deal with problems with many hyperparameters. They develop a gradient-based approach to overcome this issue; however, it works with the assumption of a continuous search space. It relies on gradient descend algorithms that may need hundreds of iterations in the search process and may not work well in the presence of non-smooth functions.

Alternative approaches for hyperparameter tuning are the evolutionary algorithms (EA), more specifically the GA. GA has been successfully applied for hyperparameter optimisation. It does not require assumptions about the function that describes the hyperparameter space and is able to perform a directed search from an initial population of random samples. Lessmann et al. (2005) proposed a combination of GA and SVM, where the GA is used to find the best SVM structure by changing the kernel type, the kernel parameters and the regularisation parameter. SVM tuning is also addressed by Chatelain et al. (2007) and Guo et al. (2008). The former applies a multi-objective approach using the NSGA-II algorithm for hyperparameter tuning, which considers the trade-off between false rejection and false acceptance rates. The latter relied on Particle Swarm Optimisation (PSO) to analyse the effects of different kernel functions for LS-SVM.

Di Martino et al. (2011) successfully applied GA to optimise the two hyperparameters of a classification SVM with Radial Basis Function (RBF) kernel. They not only evaluated the effects of different fitness functions but also benchmarked the proposed technique with Grid-search and other ML techniques. Barros et al. (2014) use GA to optimise the design components and the respective hyperparameters of DTs, achieving superior results compared to traditional DT algorithms, i.e. CART (Classification and Regression Trees), C4.5 and REP. Young et al. (2015) applied GA to optimise a Convolutional Neural Network (CNN) algorithm for an image classification benchmark dataset.

Despite the advantage of EA algorithms for hyperparameter tuning compared to SMBO based algorithms, these methods may suffer from slow convergence, especially when a high number of hyperparameters are involved. Furthermore, these approaches do not make sense of the underlying function that describes the effect of each hyperparameter in the optimisation process or the importance of each hyperparameter. Bergstra and Bengio (2012) showed through a Gaussian process analysis that, for the same algorithms, in most datasets, a few sets of hyperparameters are more important for algorithm’s performance and they differ according to the dataset. Additionally, they mentioned the fact that hyperparameter search space is more sensitive in some dimensions than others.

Based on the drawbacks of the existing approaches and taking advantage of the observations pointed out by Bergstra and Bengio (2012), a new approach is proposed in this research. The approach uses the features of the ANOVA (Analysis of Variance) to determine the hyperparameter with the more sensitive search space at each step, i.e. the hyperparameter with higher effect on the algorithm's variability. Once identified, this hyperparameter is adjusted, reducing the overall search space for the next iteration. Additionally, the proposed method takes into consideration, in the optimisation process, the interaction among hyperparameters, which may improve the effectiveness of the hyperparameter tuning.

6.2. METHODOLOGY

The general factorial experiment (Montgomery, 2012) has been widely used in process and product optimisation. Previous experiments (Almeida and Steiner, 2013; Almeida et al., 2019) showed that the use of full factorial DOE is useful to identify the hyperparameters that have the highest effect on algorithm's performance in both optimisation problems and supervised learning tasks. The proposed approach takes advantage of the ANOVA to explicitly determine the most sensitive hyperparameter and its statistical significance at each iteration. It relies on the use of the factorial experiment to analyse the effects and tune the hyperparameters.

Considering a two-factor experiment, where each factor represents an algorithm hyperparameter, the response, i.e. the algorithm's measure of performance, when the hyperparameter *H1* is set at the *i*th level (*i* = 1, 2, …, *a*) and the hyperparameter *H2* is set at the *j*th level (*j* = 1, 2, …, *b*) for the *k*th replicate (*k* = 1, 2, …, *n*), is denoted as y_{ijk}. Each observation can then be represented by the *effects model* (Montgomery, 2012), as shown in Equation (6.1).

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \epsilon_{ijk} \tag{6.1}$$

where μ is the overall mean, τ_i is the effect of hyperparameter *H1* at level *i*, β_j is the effect of the hyperparameter *H2* at level *j*, $(\tau\beta)_{ij}$ is the effect of the interaction

between the hyperparameters, and ϵ_{ijk} is a random error. The treatment and interaction effects are defined as deviations from the overall mean, consequently, $\sum_{i=1}^{a}\tau_i = 0,\ \sum_{j=1}^{b}\beta_j = 0,$ and $\sum_{i=1}^{a}(\tau\beta)_{ij} = \sum_{j=1}^{b}(\tau\beta)_{ij} = 0.$ Through the analysis of variance, the hypothesis of equality of different levels of each hyperparameter (Equations 6.2 and 6.3), as well as the interaction (Equation 6.4) between them, are evaluated.

$$H_0: \tau_1 = \tau_2 = \ ... = \tau_a = 0$$

$$H_1: at\ least\ one\ \tau_i \neq 0 \tag{6.2}$$

$$H_0: \beta_1 = \beta_2 = \ ... = \beta_a = 0$$

$$H_1: at\ least\ one\ \beta_i \neq 0 \tag{6.3}$$

$$H_0: (\tau\beta)_{ij} = 0\ for\ all\ i, j$$

$$H_1: at\ least\ one\ (\tau\beta)_{ij} \neq 0 \tag{6.4}$$

In order for the null hypothesis to be true, the mean squares must estimate the variance (σ^2). The expected mean squares, for the hyperparameter *H1* and *H2*, the interaction between them and the mean squared error are given by the Equations 6.5, 6.6, 6.7 and 6.8, respectively.

$$E(MS_{H1}) = E\left(\frac{SS_{H1}}{a-1}\right) = \sigma^2 + \frac{bn\sum_{i=1}^{a}\tau_i^2}{a-1} \tag{6.5}$$

$$E(MS_{H2}) = E\left(\frac{SS_{H2}}{b-1}\right) = \sigma^2 + \frac{an\sum_{j=1}^{b}\beta_j^2}{b-1} \tag{6.6}$$

$$E(MS_{H1H2}) = E\left(\frac{SS_{H1H2}}{(a-1)(b-1)}\right) = \sigma^2 + \frac{n\sum_{i=1}^{a}\sum_{j=1}^{b}(\tau\beta)_{ij}^2}{(a-1)(b-1)} \tag{6.7}$$

$$E(MS_E) = E\left(\frac{SS_E}{ab(n-1)}\right) = \sigma^2$$

(6.8)

If there are differences between different levels of each hyperparameter or in the interaction, the corresponding mean square will be larger than MS_E. The error term ϵ_{ijk} is assumed to be normally and independently distributed with constant variance σ^2, the mean square ratios, in this example MS_{H1}/MS_E, MS_{H2}/MS_E, and MS_{H1H2}/MS_E, follows the *F* distribution with *a* - 1, *b* - 1 and (*a* - 1)(*b* - 1) degrees of freedom in the numerator, respectively, and *ab*(*n* – 1) degrees of freedom in the denominator. Larger mean squares ratios suggest that the null hypothesis does not hold, which can be confirmed by the analysis of the critical region of the *F* distribution. The ANOVA table summarises the results in terms of mean squares and statistical significance.

Based on the capabilities of the full factorial experiment, an automatic hyperparameter optimisation algorithm was developed in this research. The approach takes advantage of the information from the SS computation to prioritise the adjustment of each hyperparameter according to its importance. In this research, the importance of a hyperparameter is related to its effect on the algorithm's accuracy due to its adjustment, i.e., the variability of the algorithm's accuracy when the hyperparameter is adjusted, which can be captured by computing the SS of the error. Therefore, the new algorithm is referred to as SSHT (Sum of Squares Hyperparameter Tuning). To deal with continuous hyperparameters, such as the number of nodes or regularisation factor, SSHT not only accepts predefined values (in case they are treated as categorical) but is also able to perform interpolation based on lower and upper limits defined by the user. One limitation of the current approach is tackling dependent hyperparameters, that becomes inactive according to a certain set of another hyperparameter, e.g. the number of nodes in the second hidden layer of a NN when a single hidden layer is evaluated (Bergstra et al., 2011; Thornton et al., 2013).

The algorithm starts computing the full factorial experiment, i.e. evaluating all hyperparameter combinations. For each combination, at least two evaluations, i.e. two runs of the algorithm using the same levels for each hyperparameter, are

required in order to compute the error SS of each treatment, the higher the number of evaluations, more robust results are achieved. This is especially useful for weak algorithms, such as NNRW, where variations in the initial random weights and training data produce a high variability in the output.

The results of factorial experiments are used to compute the SS for both main effects and two-factor effects, with their respective F-value and significance level. The hyperparameter with higher effect on the algorithm's variability is selected to be adjusted. Before computing the averages of each hyperparameter level to define the best one, a filter is applied to consider the interaction of the chosen hyperparameter with the others. The interactions are analysed and, for each statistically significant interaction, the worst level of the hyperparameters with significant interaction with the chosen one is temporarily disabled. This aims to avoid the effect of interactions when defining the level of the chosen hyperparameter where the higher accuracy was achieved.

As an example, considering a ML algorithm with hyperparameters A, B and C, with two levels (Low and High) each, all possible combinations are computed, i.e. [A_L, B_L, C_L], [A_L, B_L, C_H], [A_L, B_H, C_L], [A_L, B_H, C_H], [A_H, B_L, C_L], [A_H, B_L, C_H], [A_H, B_H, C_L], [A_H, B_H, C_H]. Each combination must be computed at least two times and the results are used to compute the ANOVA table, which shows the F-score and significance of the main effects (A, B and C) and the interactions (AxB, AxC and BxC). The algorithm chooses the main effect with higher F-score and, before computing the average of low and high levels, checks for significant interactions, e.g. if A is the main effect with higher F-score, the interactions AxB and AxC are analysed. In the case of the interaction AxB, for example, being statistically significant, the average error of B_L and B_H are computed and the experiments with the worst level of B are temporarily disabled. This process is repeated for all A interactions and then the average error of A levels are computed, using the remaining experiments.

Once the averages of A levels are computed, the setting of A has two possibilities, according to the type of hyperparameter, i.e. continuous and categorical. The latter case is the simpler one, the best level of A is set, the experiments with the worst level are excluded, and a new ANOVA is computed for the analysis of the next hyperparameter. In case of a continuous hyperparameter, an intermediate level is

defined and new experiments with the new level, combined with all levels of the remaining hyperparameter are performed. The new experiments are combined with the existing ones, i.e. the combination of all hyperparameters with the best level of A and a new ANOVA is computed for the use in the next iteration. The process is illustrated in Figure 6.1.

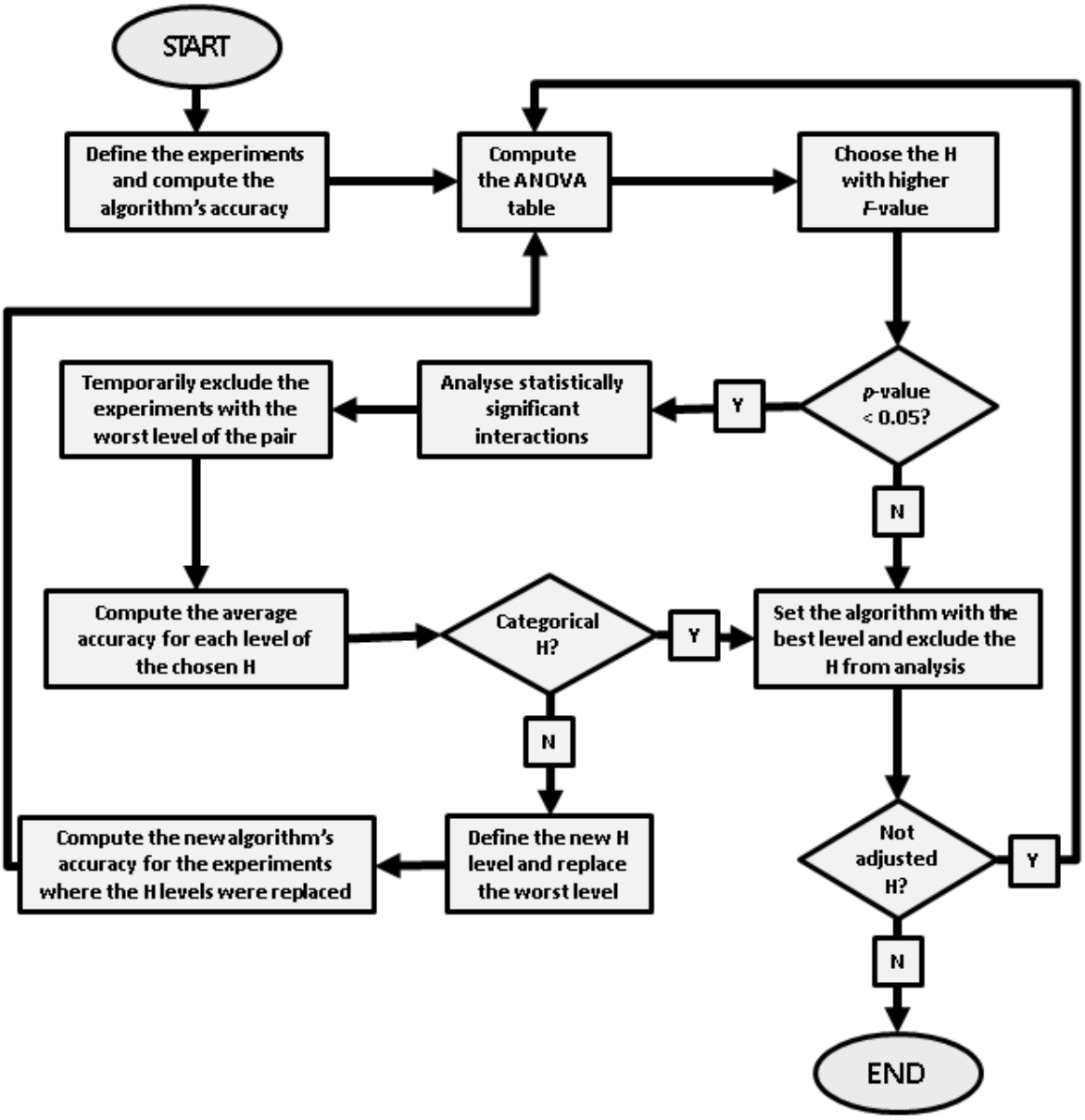

Figure 6.1: SSHT procedure.

In order to evaluate the effectiveness of the proposed hyperparameter tuning approach, it is compared to a metaheuristic approach, as described in the next section.

6.3. EXPERIMENTAL PROTOCOL

In this section, the evaluation protocol used to assess the proposed approach is described. The experiments are carried out using the four benchmark datasets considered in this research (House, Maintenance, Energy and Quality). Additionally, five synthetic datasets are generated using the functions described in Chapter 4 (F1, F2, F3, F7 and F11). The synthetic datasets are created with 15 predictive variables, randomly generated within the interval [0, 10] and containing 10,000 observations. Both benchmark and synthetic datasets attributes are standardised (mean 0 and variance 1).

The experiments aim to understand how each hyperparameter affects the accuracy of the NNRW and all adjustable factors of NNRW, as described in Chapter 5, are analysed. The hyperparameters are divided into two main types, continuous and categorical/binary hyperparameters. The continuous ones are the number of nodes (N), regularisation factor (R) and initial random weights (W) and the categorial/binary hyperparameters are the activation function (A), use of direct link (D) and use of output bias (Ob). Table 5.1 shows the range of values to be searched for NNRW hyperparameter optimisation. It is important to note that the continuous hyperparameters are described in terms of limits, within which the search will be performed, and that the number of nodes lies in the integer domain.

Table 6.1: Hyperparameters levels for the first set of experiments.

Hyperparameter	Range of values	Type
N	[20, 150]	Integer (step = 1)
R	[0.01, 1.50]	Continuous
W	[0.1, 1.5]	Continuous
A	[relu, logsig, tansig, sin, hardlim, tribas, radbas, sign]	Categorical/Binary
D	[False, True]	Categorical/Binary
Ob	[False, True]	Categorical/Binary

For the means of comparison, a widely used technique, the GA (Pinto et al., 2013), is considered. The GA has been successfully applied for many optimisation

problems. It is relatively easy to implement and avoids the drawbacks of SMBO based approaches. In this thesis, a combined strategy is applied to encode the GA chromosomes, where one cluster deals with continuous variables and the other deals with categorical and binary ones. This requires special attention to the crossover procedure, which relies on two different strategies applied according to the cluster. For the continuous cluster, a convex combination of the selected parents is computed, according to Eq. 6.9.

$$p_{new} = \alpha p_1 + (1 - \alpha) p_2 \tag{6.9}$$

where p_1 and p_1 are the selected parents and α is a random value uniformly distributed within the interval $[-\gamma,\ 1 + \gamma]$.

For the categorical cluster, the popular uniform crossover (Pinto et al., 2013) is applied. The crossover strategy is illustrated in Figure. 6.2.

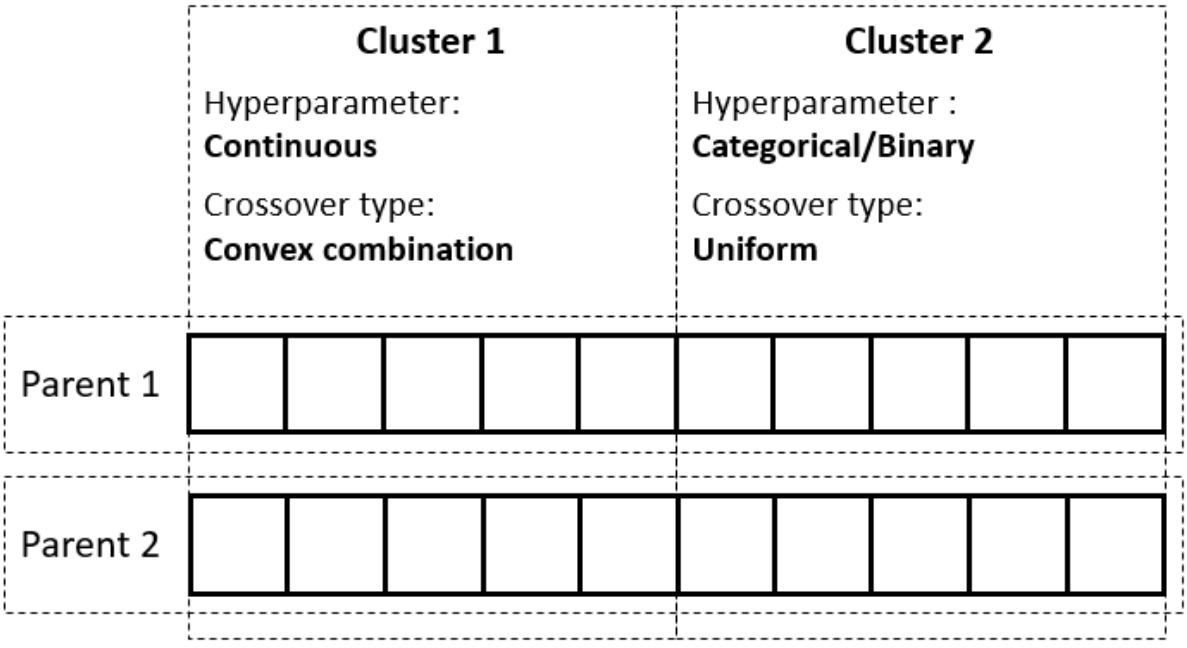

Figure 6.2: Crossover strategy for hyperparameter optimisation using GA.

When a pair of individuals are selected for the crossover, both clusters are subjected, at the same time, to their respective crossover strategies. The remaining GA parameters were adjusted manually, based on preliminary experiments, and are defined as follows:

- Number of iterations: 30;
- Population size: 200;
- Crossover percentage: 60%;
- Mutation percentage: 10%;
- Mutation rate: 2%;
- γ = 0.2;
- Stopping criteria: 5 iterations.

For each experiment, a sample of 4,000 observations is selected from the dataset, from which 2,000 are used for hyperparameter optimisation and 2,000 are used for evaluation. The data for hyperparameter optimisation is fed to the algorithms, which use 70% for training and 30% for validation, i.e. 1,400 observations are used to train the NNRW with a given set of hyperparameters and 600 are used to compute the MSE for validation. Once the hyperparameters are optimised, 70% of the evaluation data is used to train a model and the remaining 30% is used to test the accuracy of the optimised model. This process is repeated 10 times and each experiment is repeated 30 times for each dataset.

The results are discussed in the next section.

6.4. RESULTS AND DISCUSSION

The proposed hyperparameter tuning algorithm proved an effective method to optimise the NNRW and a promising tool to advance the field of hyperparameter

tuning. Compared to the GA algorithm the main achievements are: very competitive accuracy and fewer number of evaluations required. The results are summarised in Table 6.2, which presents the average MSE for each hyperparameter set found by SSHT and GA, and Table 6.3, which brings the number of evaluations required by each algorithm.

Table 6.2: Average and standard deviation MSE resulted from NNRW optimised by SSHT and GA.

		F1	F2	F3	F7	F11	Energy	House	Quality	Maint.
SSHT	**Avg**	**211.8**	**37981.1**	**2542.4**	82.1	47.2	9472.9	4.12E+9	0.556	1.74E-7
	Std	31.1	19202.2	1024.6	3.4	1.3	887.3	2.63E+8	0.029	3.02E-8
GA	**Avg**	237.1	116317.8	6178.6	83.2	47.4	9696.3	4.20E+9	**0.538**	1.77E-7
	Std	18.7	104891.8	1047.7	3.4	1.7	1000.7	2.78E+8	0.030	3.90E-8
	Diff	-10.7%	-67.3%	-58.9%	-1.4%	-0.4%	-2.3%	-2.0%	3.4%	-2.0%
	***p*-value**	<< 0.01	<< 0.01	<< 0.01	0.188	0.630	0.372	0.253	0.023	0.707

The results of Table 6.2 show that the hyperparameter sets found by SSHT achieved similar average MSE compared to GA. Considering a significance level of 5%, the *p*-value of the *t*-test indicates that the average MSE for F7, F11, Energy, House and Maintenance are statistically equal. GA resulted in slightly better accuracy in the Quality dataset, while SSHT performed better in F1, F2 and F3.

Table 6.3: Descriptive statistics of the number of evaluations needed for SSHT and GA to NNRW optimisation.

		F1	F2	F3	F7	F11	Energy	House	Quality	Maint.
SSHT	**Avg**	700.1	1469.6	756.8	1091.1	1090.9	636.0	585.9	663.7	1314.7
	Std	58.3	69.3	38.2	193.2	229.7	130.1	69.6	111.1	79.1
	Min	632	1232	656	832	768	512	512	512	1056
	Max	816	1584	840	1600	2048	1024	768	992	1456
GA	**Avg**	2416.7	2888.0	2211.3	1684.0	1828.7	1436.7	1660.7	1712.0	3168.0
	Std	827.5	1162.6	973.0	687.4	777.4	460.2	553.8	485.5	951.2
	Min	1040	1320	1040	1040	1040	1040	1040	1040	1180
	Max	3840	4400	4400	4400	4400	2580	3000	2720	4400

In terms of the number of evaluations (Table 6.3), SSHT not only needed fewer evaluations compared to GA (up to 1/3 of the evaluations needed by GA) but also better consistency. The much lower standard deviation and amplitude (the difference between the minimum and the maximum number of evaluations) indicate that SSHT achieves similar results in every run. This fact summed to the equal or better accuracy (The accuracy on Quality dataset could be considered statistically equal if 1% significance is required) puts SSHT as a very competitive strategy for hyperparameter tuning.

It is important to investigate the values of the hyperparameters returned by the tuning algorithms and analyse their effects on the set up of NNRW. Firstly, the activation function recommended by each algorithm for each problem is summarised in Table 6.4.

Table 6.4: Number of times each activation function was recommended by the optimisation algorithms.

		hardlim	logsig	radbas	relu	sign	sin	tansig	tribas
F1	**SSHT**	0	0	13	16	0	0	0	1
	GA	0	0	3	27	0	0	0	0
F2	**SSHT**	0	0	30	0	0	0	0	0
	GA	0	0	25	5	0	0	0	0
F3	**SSHT**	0	0	28	2	0	0	0	0
	GA	0	0	5	25	0	0	0	0
F7	**SSHT**	5	3	2	6	4	3	5	2
	GA	2	11	3	1	3	1	7	2
F11	**SSHT**	5	7	2	3	2	6	2	3
	GA	1	13	1	1	6	1	3	4
Energy	**SSHT**	3	8	5	3	1	2	3	5
	GA	3	17	0	5	0	1	3	1
House	**SSHT**	0	1	2	20	0	0	2	5
	GA	0	1	9	18	0	0	1	1
Quality	**SSHT**	2	1	6	15	1	2	0	3
	GA	0	3	3	22	0	1	1	0
Maint.	**SSHT**	0	0	0	0	0	29	1	0
	GA	0	0	2	0	0	28	0	0
Total		**21**	**65**	**139**	**169**	**17**	**74**	**28**	**27**

The first result to be highlighted is that the *relu* was the most recommended activation function, followed by *radbas* and sin (which was responsible for the better accuracy in Maintenance dataset). In general, both algorithms agreed on the activation function recommendations, with the main exceptions observed in the F1 and F3 datasets. In F1, while GA concentrated its recommendations in *relu* function, SSHT divided its recommendations mainly between *relu* and *radbas*.

The fact that SSHT hyperparameter sets showed better accuracy compared to GA in F1 dataset (Table 6.1) raised the question if the use of *radbas* was responsible for that difference. The average MSE of the 13 *radbas* the 16 *relu* recommendations resulted in 180.6 and 234.9, respectively, a relatively large difference that confirms the advantage of *radbas* for this dataset. In the case of F3, where SSHT showed an advantage of 58.9% in terms of accuracy, SSHT recommendations concentrated on *radbas*, while GA focused on *relu*. For F2 (where SSHT showed bigger advantage compared to GA), House and Maintenance datasets, both algorithms behaved the same in terms of the activation function. For F7, F11 and Energy datasets, there was no clear advantaged of one activation function over another.

Table 6.5 shows the results related to the use of direct link and output bias, i.e. the number of times they were set as true or false.

Table 6.5: Number of times SSHT and GA recommend the activation of direct link and output bias.

		Direct link		Output bias	
		True	False	True	False
F1	SSHT	30	0	30	0
	GA	30	0	30	0
F2	SSHT	30	0	25	5
	GA	30	0	25	5
F3	SSHT	30	0	30	0
	GA	30	0	30	0
F7	SSHT	8	22	30	0
	GA	8	22	27	3
F11	SSHT	8	22	30	0
	GA	10	20	26	4
Energy	SSHT	30	0	30	0
	GA	17	13	18	12
House	SSHT	30	0	30	0
	GA	23	7	26	4
Quality	SSHT	29	1	30	0
	GA	26	4	28	2
Maint.	SSHT	30	0	30	0
	GA	23	7	30	0

Both algorithms showed similar results, except on Energy dataset, where GA divided its recommendations between *true* and *false* for both direct link and output bias, while SSHT recommended *true* for both hyperparameters in all runs. The use of output bias is an important feature to increase the accuracy of the NNRW. Both algorithms recommended the use of output bias most of the time (98.1% in case of SSHT and 88.9% in case of GA) for all datasets, this hyperparameter was also the main source of NNRW variance. The direct link was also recommended most of the time; however, in this case, for two datasets (F7 and F11) the use of direct link was avoided by the algorithms. An in-depth look at the averages of the two direct link settings showed no important difference between the treatments. In order to check if there is a statistical difference between NNRW with or without the direct link in F7 and F77 datasets, additional experiments were performed. By using an arbitrary set of hyperparameters, 100 runs were executed and the average MSE were computed. The results are shown in Table 6.6.

Table 6.6: Average and standard deviation MSE resulted by NNRW with and without the direct link in datasets F7 and F11.

		Direct link		
		True	**False**	***p*-value**
F7	**MSE**	82.6	81.6	0.015
	std	2.7	2.6	
F11	**MSE**	47.0	46.3	<< 0.010
	std	1.2	1.3	

In fact, considering a confidence level of 5% for the paired *t*-test, an advantage is observed when the direct link is deactivated in F7 and F11 datasets.

In Table 6.7, the mean and standard deviation of the number of nodes, weights and regularisation factor recommendations are presented.

Table 6.7: Average optimised N, W and R hyperparameters.

		N		**W**		**R**	
		Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
F1	**SSHT**	148.3	4.7	0.2	0.1	0.15	0.26
	GA	141.9	6.8	0.8	0.4	0.48	0.25
F2	**SSHT**	148.3	3.0	0.1	0.0	0.04	0.13
	GA	139.7	10.2	0.2	0.3	0.31	0.38
F3	**SSHT**	147.2	4.4	0.1	0.1	0.06	0.19
	GA	145.1	6.3	0.6	0.3	0.63	0.34
F7	**SSHT**	24.5	7.1	0.8	0.7	0.51	0.61
	GA	59.8	29.8	0.5	0.4	0.68	0.34
F11	**SSHT**	24.9	8.2	0.6	0.6	0.41	0.60
	GA	53.3	31.9	0.7	0.4	0.78	0.34
Energy	**SSHT**	44.4	45.7	0.6	0.6	0.99	0.70
	GA	73.2	30.1	0.6	0.4	0.74	0.44
House	**SSHT**	122.9	42.9	0.8	0.6	0.94	0.67
	GA	95.2	27.4	0.6	0.3	0.87	0.30
Quality	**SSHT**	82.3	54.4	0.4	0.4	1.10	0.57
	GA	87.0	25.5	0.6	0.3	0.73	0.33
Maint	**SSHT**	132.0	26.0	0.9	0.2	0.01	0.00
	GA	116.5	17.4	0.8	0.1	0.01	0.00

In general, both algorithms showed similar values for N, W and R. The base number of nodes varies according to the problems, although it is generally accepted that higher accuracy is obtained by increasing the number of nodes. In all cases, the initial weights range lies in the interval [-0.9 0.9]. In the case of F1, F3 and F11, where the more important differences in accuracy were observed, additional experiments showed that the different W recommendations from SSHT and GA algorithms resulted in significant differences in terms of accuracy. Similar behaviour was observed in the analysis of R.

The proposed SSHT algorithm for hyperparameter tuning proved a competitive approach for NNRW optimisation. The accuracies of the hyperparameter settings found by SSHT were equal to or better than the accuracies of GA settings, in most of the datasets. Furthermore, while GA required an average of 2111.8 evaluations to find the optimised hyperparameter settings for all problems, SSHT needed on average 923.2 evaluations to converge, a reduction of 56.3%. This is an important advantage, especially when tuning computationally expensive algorithms such as DNNs or high dimensional SVMs.

The proposed approach uses a simple interpolation technique to achieve convergence for continuous hyperparameters. More advanced and effective searching techniques, such as Bayesian optimisation or gradient-based techniques, could potentially improve the search process and reduce the required number of evaluations. During the optimisation process, when SSHT finds two levels that are statistically equal, it arbitrarily chooses the one that resulted in better accuracy. It is possible to easily take advantage of this mechanism by including a criterion that considers other optimisation measures, such as computing time, for example.

In Table 6.8 the optimised set of hyperparameters for each problem is summarised. The average values of SSHT were selected for F1, F2 and F3 datasets and the values found by GA were selected for the Quality dataset. For the remaining datasets, where the results were statistically equal, an average of both techniques is used.

Table 6.8: Final optimised NNRW hyperparameters for each dataset.

	A	D	O	N	W	R
F1	radbas	True	True	148	0.2	0.15
F2	radbas	True	True	148	0.1	0.04
F3	radbas	True	True	147	0.1	0.06
F7	relu	False	True	42	0.7	0.60
F11	relu	False	True	39	0.6	0.59
Energy	logsig	True	True	59	0.6	0.87
House	relu	True	True	109	0.7	0.91
Quality	relu	True	True	87	0.6	0.73
Maint.	sin	True	True	124	0.9	0.01

In this chapter, the SSHT, a new technique for hyperparameter optimisation, has been presented. The results demonstrated that SSHT is an effective tool for hyperparameter optimisation. SSHT showed similar convergence compared to the GA, however, the SSHT showed better consistency, i.e. achieved similar results in every run, and also required a fewer number of evaluations to find the optimised hyperparameter set, which results in an important computational advantage compared to the GA. Despite the popularity of the ELM, the results indicate an advantage of RVLF in terms of accuracy. In most of the cases, the best accuracy was achieved using the direct link, a mechanism that is present only in the RVFL architecture. Additionally, the results showed significant improvement in accuracy when the output bias is present, while Huang (2014) argues that the output bias should not be active and Zhang and Suganthan (2016) did not find a statistical difference when it is not activated. The optimised NNRW hyperparameter setting found in this chapter will support the development of the ensemble, in the next chapter. For each dataset, the ensemble will be built using the respective optimised NNRW as base models for the evaluation on the simulated data streams.

7. A NEW ENSEMBLE APPROACH FOR DATA STREAM REGRESSION

In this chapter, a new bagging ensemble method based on NNRW is developed. The proposed approach, bagging NNRW (B-NNRW) aims to deal with the online regression problems in the presence of concept drift with competitive accuracy and better computational efficiency compared to the existing methods. The proposed algorithm takes advantage of the efficiency of NNRWs to build a homogeneous ensemble and enables effective updating of the model to accommodate possible concept drifts.

The proposed approach relies on an initial buffer of training data to build the initial ensemble. The ensemble is built using the bagging meta-algorithm, which creates bootstrap samples of data that are used to train the base models and helps to increase the ensemble's diversity. Although some of the online ensemble approaches do not rely on data buffering, these methods require that a considerable amount of training samples are presented to the model before it reaches an acceptable level of accuracy (Oza and Russell, 2001; Ikonomovska et al., 2015). The update of the developed ensemble is executed by tracking the base model's performance and scoring them accordingly. When a model achieves a pre-determined level of the negative score, it is replaced by a new model. To evaluate the proposed algorithm, synthetic datasets simulating various types of drift are used, along with benchmark datasets from public data repositories.

In the next section, the B-NNRW methodology is presented. Before evaluating the proposed approach, more details on data generation are outlined in Section 7.2, followed by the approach for data scaling applied in this research, on Section 7.3 and a brief discussion on the ensemble size, carried out in Section 7.4.

7.1. METHODOLOGY

In this section, the proposed ensemble algorithm for data stream regression is demonstrated. The algorithm applies the bagging meta-algorithm to create a

diversified ensemble of NNRWs. To cope with concept drift, the weight of each model is dynamically updated (i.e. at every new instance), based on the exponentially smoothed error. Additionally, a replacement mechanism base on individual's performance helps to improve the ensemble accuracy and also keeps the accuracy under control on the occurrence of various types of concept drift.

The algorithm starts by buffering the first samples of the data stream to form the training set and then the bagging meta-learning is applied. The bagging meta-learning consists of creating *M* bootstrapped samples from the training data, where each sample has the same number of instances of the training data and is used to build a model. From each sample, 70% of the data are used for training and the remaining 30% are used for validation. The validation set is used to compute the *MSE* for each model *m*. The *MSE* is then used to compute the weight of each ensemble member, using Eq. 7.1.

$$w_m = \frac{1}{MSE_m} \tag{7.1}$$

The process is illustrated in Figure 7.1.

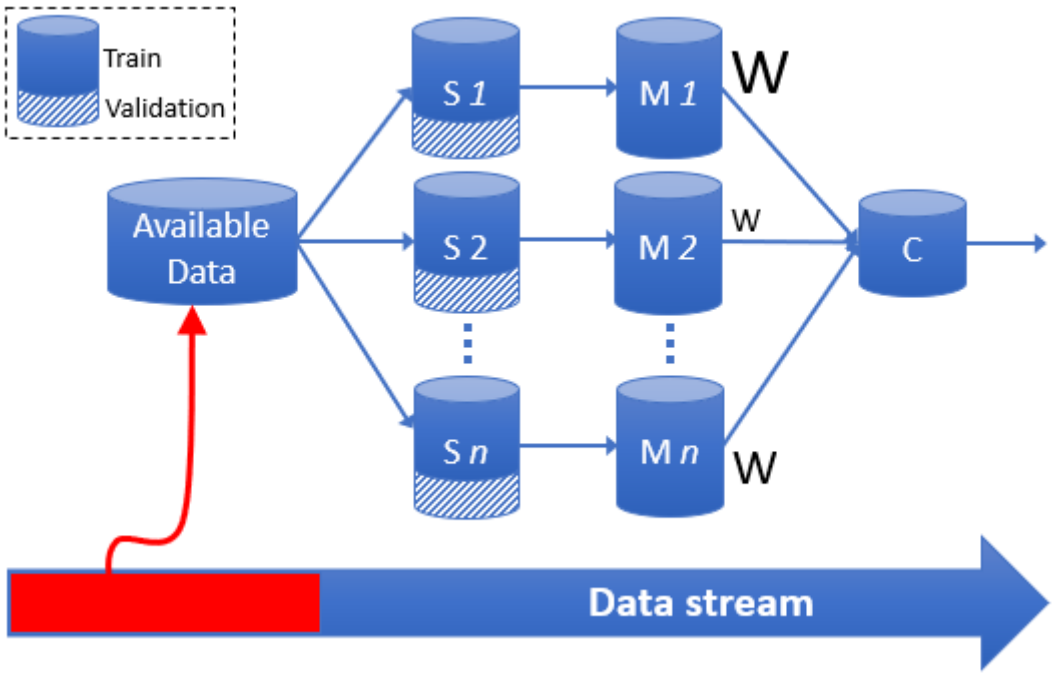

Figure 7.1: Initial B-NNRW ensemble. The different W sizes represent the weights attributed to each model according to their accuracy.

Once the ensemble is built, the output of a new instance x is predicted according to Eq. 7.2.

$$y_E(x) = \frac{\sum_{m=1}^{M} w_m * \hat{y}_m(x)}{\sum_{m=1}^{M} w_m} \tag{7.2}$$

where $\hat{y}_m$ is the output of model *m* and w_m is the model's weight. As the ensemble performs the predictions on the data stream, two updating mechanisms become active, the weight updating and the model contribution.

The weight updating aims at dimming the importance of the less accurate models in the final decision. Relying on the last squared error may be ineffective since the model's accuracy for a single observation may be far from its overall accuracy. On the other hand, global MSE may not represent the current accuracy of the model. In the case of concept drift, the lower accuracy on new instances may not be immediately reflected on the global MSE, therefore, it may be slow to identify when the model's accuracy is decreasing. To overcome this, the *MSE* is updated using an exponential moving average filter, henceforth referred to as Exponentially Smoothed *MSE* (*ESMSE*), computed as Eq. 7.3.

$$ESMSE_n = \begin{cases} \alpha.SE_n + (1-\alpha).MSE, & if\ n = 1 \\ \alpha.SE_n + (1-\alpha).ESMSE_{n-1}, & otherwise \end{cases} \tag{7.3}$$

where α is a tuning parameter where the user can regulate the sensitivity of the model to current error. A large α will give more importance to the last error and the algorithm will respond faster to a decrease in the model accuracy, reducing its importance when computing the model's weight. On the other hand, a small α will make the *MSE* less sensitive to short term errors.

The replacement mechanism works by evaluating the models' performance and eventually replacing the low performing members. The evaluation can be carried out using any ML performance metric, such as MSE, MAPE, accuracy, precision or F1 score. The replacement mechanism proposed in this research is easy to implement and, different from traditional concept drift detection mechanisms, activates the update of the ensemble regardless of the occurrence of concept drift. This helps to improve the accuracy of the ensemble while no concept drift is

detected. The replacement of low performing members for more accurate ones increases the overall accuracy and also encourages the continuous improvement of the ensemble accuracy. There will be always comparatively low performing members subject to be replaced. Additionally, there is no need to tune the drift detection mechanism or tune the number of models to be replaced.

The replacement is activated when the model's score decreases relative to the model's lifetime for a given period of time. A model receives a score when it meets its performance targets. In this research, the target was established in terms of accuracy related to the accuracy of the ensemble. More specifically, when the model's error is lower than the average of all ensemble members, 1 point is added to its score (S_m), as shown in Eq. 7.4.

$$S_m = \begin{cases} S_m + 1, & if\ e_m < \frac{1}{M}\sum_{m=1}^{M} e_m \\ S_m, & otherwise \end{cases} \tag{7.4}$$

The lifetime of a model (L_m) is computed as the number of instances that have been presented to the model for prediction. Using the model's lifetime it is possible to compute the relative score (*RS*), as shown in Eq. 7.5.

$$RS_m = \frac{S_m}{L_m + 1} \tag{7.5}$$

When a model loses its accuracy its score will remain constant and therefore, its relative score will decrease. Preliminary experiments using fixed replacement intervals showed that, in most of the cases, the replaced models had shown low performance for long periods before the updating point. This would result in a constant decrease of *RS* and therefore could be used to trigger the replacement of the model. The trigger *T* is computed according to Eq. 7.6, as follows:

$$T_m = \begin{cases} T_m + 1, & if\ RS_n^m < RS_{n-1}^m \\ T_m - 1, & otherwise \end{cases} \tag{7.6}$$

The model is replaced when *T* reaches a user-defined threshold. A lower threshold will result in higher sensitivity to concept drift and therefore a higher number of replacements. The opposite behaviour is expected when the threshold value is increased. This approach allows any kind of performance metric to be used to evaluate the model's performance. It also allows that a model recovers its accuracy if it performs poorly for a short period, avoiding unnecessary computations. The replacement strategy for an ensemble with 6 base models is illustrated in Figure 7.2 and the B-NNRW procedure is summarised in the diagram shown in Figure 7.3.

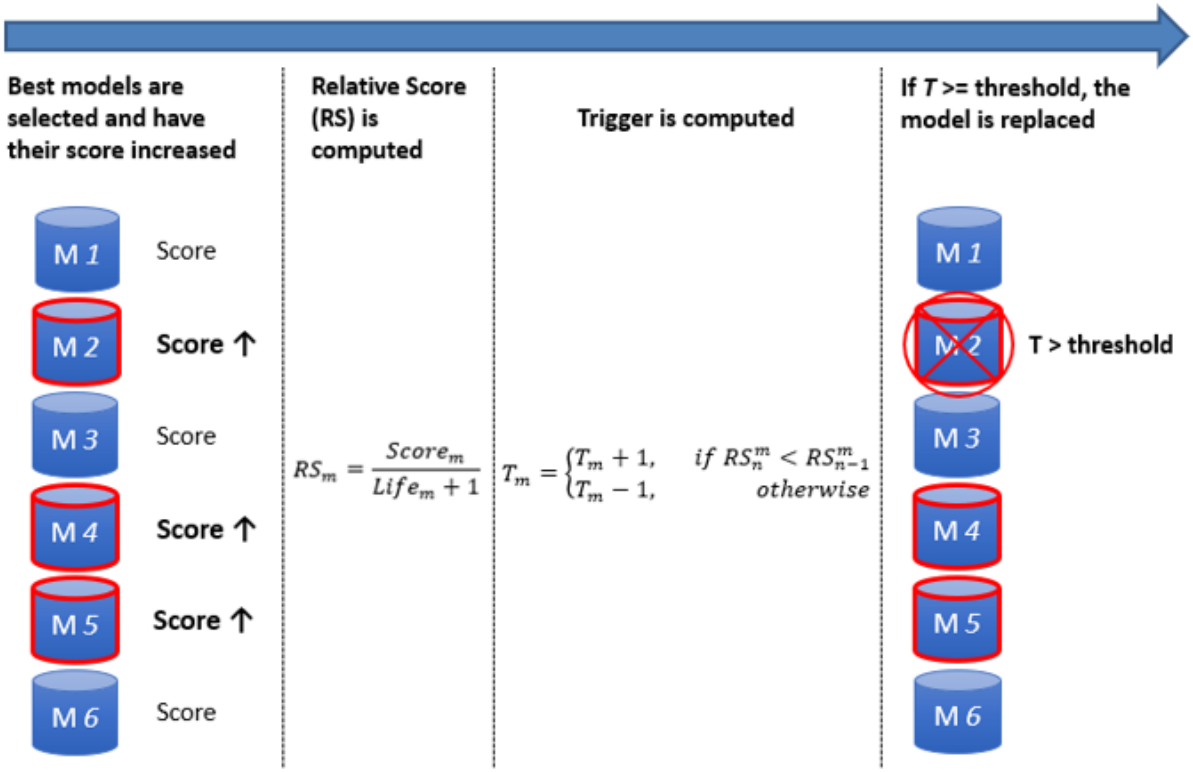

Figure 7.2: B-NNRW replacement strategy.

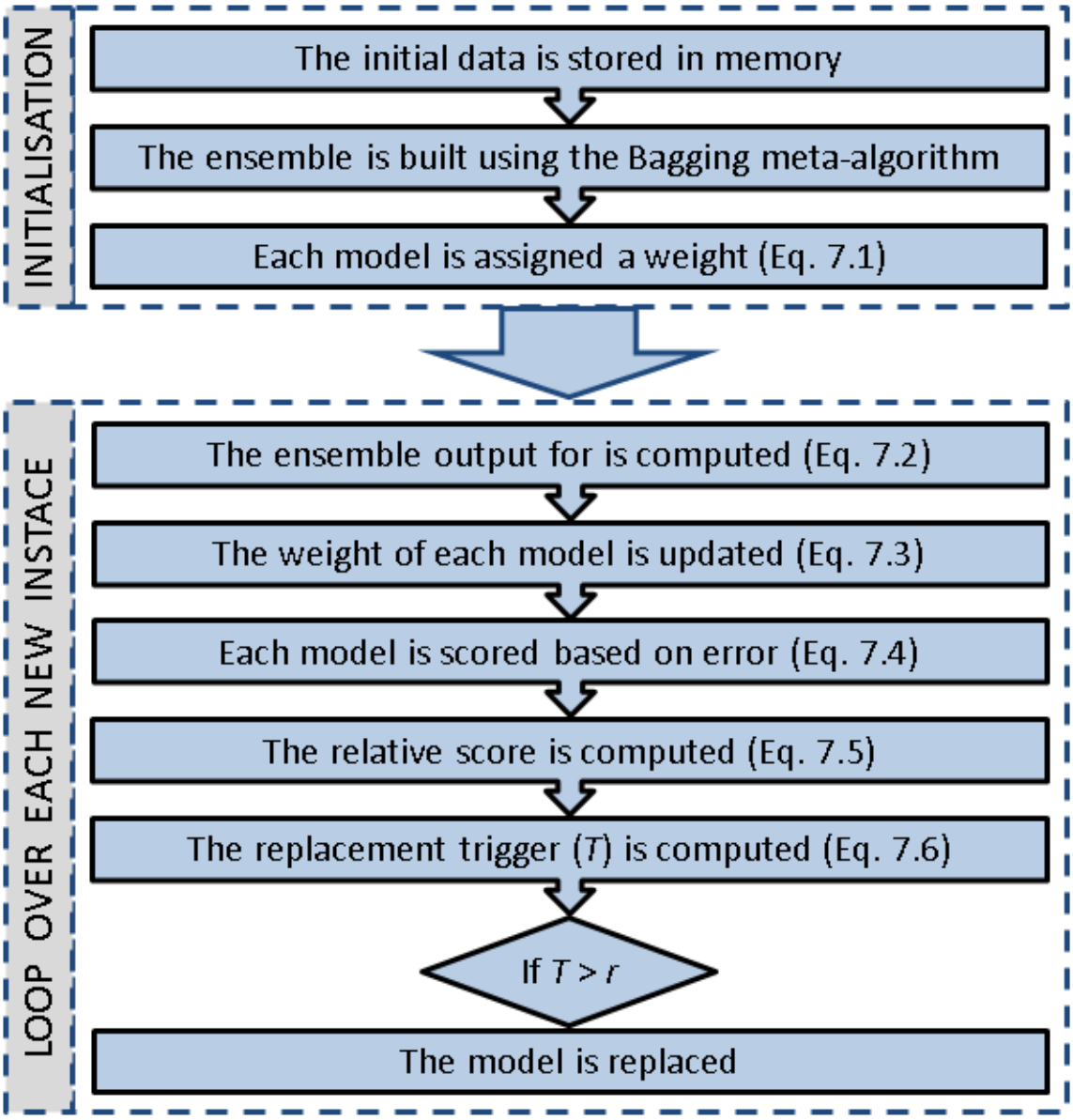

Figure 7.3: B-NNRW procedure.

In the next section, the details regarding the synthetic data generation are presented.

7.2. DATA STREAM GENERATION

For the assessment of the effectiveness of the strategies proposed in this research, different synthetic datasets were generated. For each function described in Section 4.2, five scenarios were simulated which include no drift, two types of gradual

drift (gradual rotation and gradual replacement), abrupt drift and data expansion. The five scenarios are created as follows:

No drift: The data were generated without drift according to Eqs. 4.13 - 4.16;

Gradual rotation: The gradual rotation applies function rotation by changing the position of the function global minimum, as described in section 4.2. For each new instance, the position of the global minimum is moved by a percentage of the input range. The directions for each dimension were randomly selected and approximately half of the dimensions moves in the positive direction while the remaining move in the negative direction. The dimensions in which the global minimum is moved were randomly chosen at each new instance. The settings for the gradual rotation are summarised in Table 7.1.

Table 7.1: Gradual rotation settings.

	F1	F2	F3	F7	F11
Lower bound	6.0	5.0	5.0	0.0	0.0
Upper bound	10.0	6.0	7.5	100.0	40.0
Range	4.0	1.0	2.5	100	40
Percentage change	0.010%	0.015%	0.010%	0.015%	N.A
Absolute change	0.00040	0.00015	0.00025	0.01500	N.A

An exception is the function F11. For this function, changing the global minimum does not rotate the hyperplane; therefore it has a limited effect on drift simulation. To effectively simulate gradual drift for function F11, a change in the function bias value is applied. The initial F11 bias, described in section 4.2, is incremented by 0.005 at each new instance (25/number of instances).

Gradual replacement: This drift was simulated by generating different hyperplanes and gradually replacing one by another. In this research, a linear replacement was applied where the replacement of the hyperplane H1 by hyperplane H2 starts with zero probability before the drifting point, reaches 50% probability in the drift point and ends with 100% probability, where H1 is completely replaced. This process is carried out during 50% of the distance between the drift points. Considering a distance

between drift points of 1,000 instances the replacement starts 250 instances before the drifting point and ends 250 instances after the drifting point. The process is illustrated in Figure. 7.4.

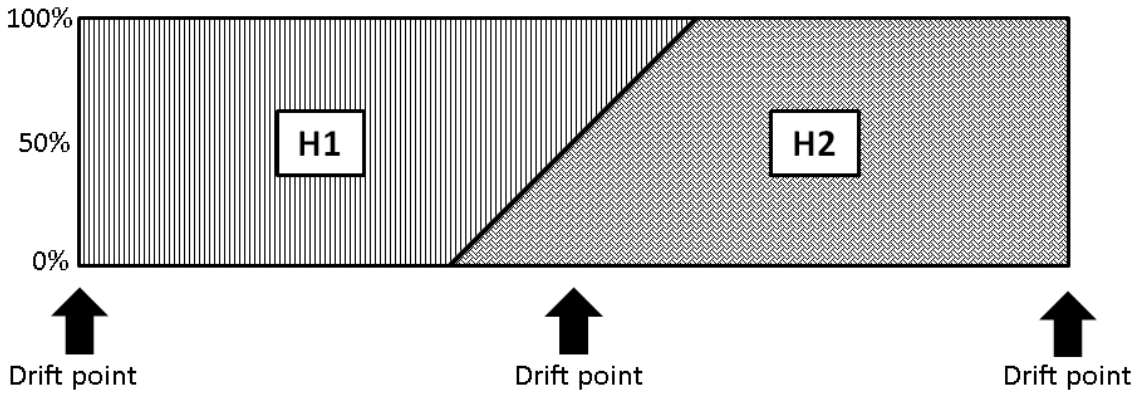

Figure 7.4: Probabilities of data being generated according to the hyperplanes H1 and H2 related to the drift points.

The hyperplanes are distinguished by a shift in the value of function bias. For all functions, given a hyperplane H_n, the hyperplane H_{n+1} obtained by incrementing the bias of function that generated H_n by 5.

Abrupt drift: The abrupt drift is similar to the gradual replacement; however, instead of generating instances according to a transition probability, all the instances were generated from the new hyperplane after the point of drift. The process is illustrated in Figure. 7.5.

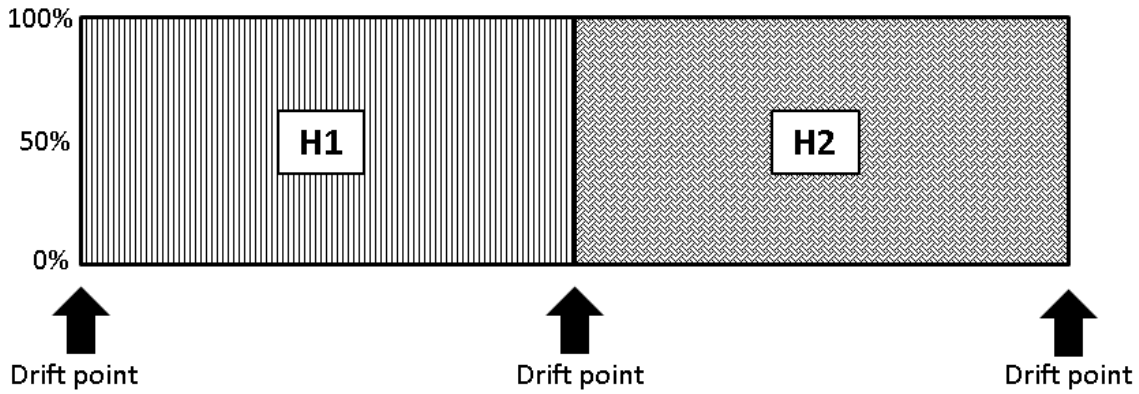

Figure 7.5: Probabilities of data being generated according to the hyperplanes H1 and H2 related to the drift points.

Data expansion: The data expansion simulates a change in the distribution of inputs. The initial domain represents half of the final range, centred at the middle of the final domain (Section 4.2). The input domain is expanded at each drift point and both dimension and direction are chosen randomly. The expansion step is equal to half the original range divided by the number of drift points and split in both directions. An illustrative example of the expansion on a 2-dimensional domain is shown in Figure. 7.6.

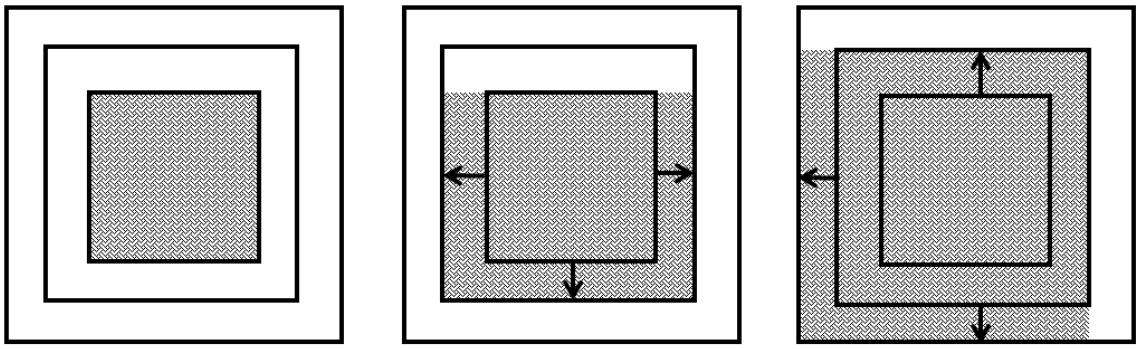

Figure 7.6: Arbitrary data expansion example for a 2-D domain.

7.3. DATA SCALING

The feature scaling techniques are widely used as a preprocessing step in ML applications. These techniques aim at normalising the range of variables to help to improve the accuracy of ML algorithms, especially those based on Euclidean distance avoiding that the variables with wide ranges or big values have a disproportional contribution to the learning process.

A common scaling approach is the min-max normalisation, where the values of the independent variables are scaled, in general within the interval [0, 1] and [-1, 1]. In the case of data streams, the use of normalisation can generate inadequate scaling when a change in the attribute distribution is observed. This requires that the minimum and maximum values are properly adjusted, which is not a trivial task. Additionally, the normalisation is more sensitive to outliers and may require some form of previous data cleaning.

A more adequate approach for data streams is the standardisation, where each feature adjusted to have zero-mean and unit variance. The standardisation is computed as shown in Eq. 7.7, where μ and σ are the mean and the standard deviation of the sample x, respectively.

$$x' = \frac{x + \mu}{\sigma} \tag{7.7}$$

The standardisation is less sensitive to outliers. Furthermore, the mean and standard deviation can be computed incrementally, which makes it suitable for dealing with changes in data distribution without making assumptions about the adequate minimum and maximum attribute values. The incremental average is computed as Eq. 7.8.

$$\mu_n = \mu_{n-1} + \frac{x_n - \mu_{n-1}}{n} \tag{7.8}$$

The incremental standard deviation can be computed according to Eq. (7.9). It requires actual incremental average (μ_n), computed in Eq. (7.8), and the previous incremental average (μ_{n-1}).

$$\sigma_n = \sqrt{\frac{(n-2)\sigma_{n-1}^2 + (n-1)(\mu_{n-1} - \mu_n)^2 + (x_n - \mu_n)^2}{n-1}} \tag{7.9}$$

Using the incremental average and standard deviation allows an effective scaling without storing past data (only the previous standard deviation and average need to be stored).

7.4. ENSEMBLE SIZE

One important decision in the design of an ensemble is its size. The most adequate ensemble size may be influenced by a number of factors, which include base learner algorithm, training data and computational constraints. As the ensemble

size grows, a computational burden is added not only in terms of memory but also in terms of processing time. Additionally, adding a new model only improves the ensemble accuracy if the new model produces different predictions from the other members, which can be achieved by manipulating the training data used to induce the new model. For a limited training dataset, it is expected that, as the number of base models grows, the ensemble accuracy gain decreases as the probability of having similar models increases.

The accuracy of different ensemble sizes was analysed for both synthetic and benchmark datasets. In the case of synthetic datasets, 10,000 observations with 15 attributes are generated for each function (F1, F2, F3, F7 and F11) with no drift. Each evaluation consists of randomly selecting training and testing sets, training the ensemble using the respective optimised set of hyperparameters for each dataset (obtained in Chapter 6), and computing the MSE on the testing data. A similar approach is applied to the benchmark data, using all the observations available (description on Section 4.3). In both synthetic and benchmark datasets, the training and testing sets are composed of 1,000 and 500 observations, respectively. Ensemble sizes ranging from 5 to 100, with increments of 5, were evaluated. Each evaluation (ensemble size/dataset) was run 30 times, and the average test MSE was computed. The results are illustrated in Figure 7.7, where the y-axis shows the average MSE and the *x*-axis shows the number of base models.

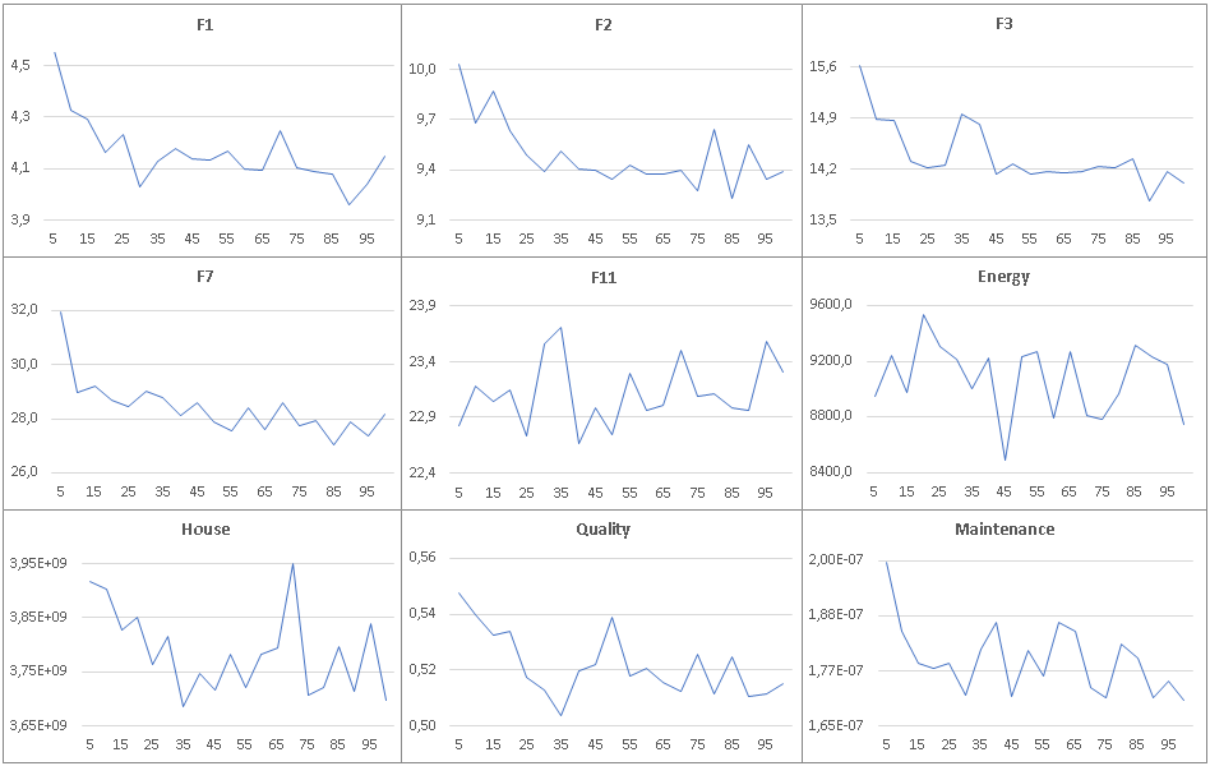

Figure 7.7: Average MSE according to the ensemble size for each dataset.

The results show the effect of ensemble size on MSE. It is possible to observe that the accuracy converges for ensembles with approximately 30 models, except for F11 and Energy datasets, where the accuracies do not seem to improve as the ensemble size grows. Since adding new models to the ensemble only improves the accuracy if they are different from the existing ones, the linear correlations among the members of the ensemble were computed. The average correlations of the 30 runs of each ensemble size for each dataset are shown in Figure 7.8.

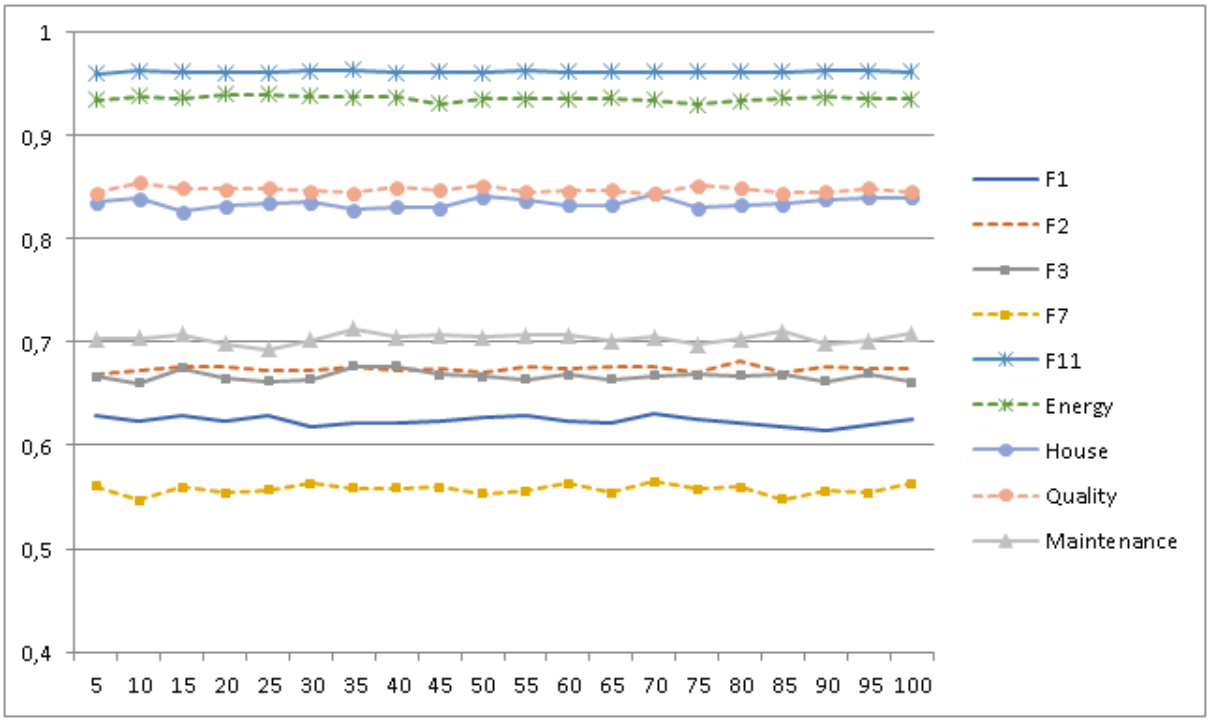

Figure 7.8: Average correlation according to the ensemble size for each dataset.

From Figure 7.8 it is possible to highlight that the average correlations keep constant as the ensemble grows. Additionally, since the same approach was applied for all datasets, the results suggest that the characteristic of the learning problem plays an important role in the ensemble diversity. The average correlation of each dataset is summarised in Table 7.2.

Table 7.2: Average pairwise linear correlation among the ensemble members.

	F1	F2	F3	F7	F11	Energy	House	Quality	Maintenance
Corr	0.624	0.674	0.667	0.557	0.962	0.936	0.835	0.848	0.704

The F11 and Energy datasets showed the highest levels of correlation, which explains why adding new models seems not to improve the ensemble's accuracy (Figure 7.7).

In the next section, the effects of the ensemble's hyperparameters, the smoothing factor (α) and the replacement threshold (r) are analysed.

7.5. B-NNRW ADJUSTING

The adjustment of the error smoothing factor (α) and the replacement threshold (r) is an important factor for the effectiveness of the proposed ensemble. In this section, an analysis of the effects of different α and r values is carried out not only assess how they affect the ensemble's accuracy but also the ensemble's computational efficiency.

For this purpose, synthetic data streams using the functions described in Section 4.2 were generated simulating the various types of drift, as described in Section 7.2. Each data stream has 5,000 observations and 15 features. The first 1,000 observations were used for training and the remaining 4,000 observations were used for testing in a prequential mode (Bifet et al., 2010), i.e. each observation was used for test and then train the model. Additionally, the benchmark datasets were also used for assessing the α and r. In this case, the first 1,000 data points were used for training and the remaining data points for testing in a prequential mode.

The α affects the sensitivity of each model to changes in the environment that may cause a loss of accuracy. In Tables 7.3 and 7.4, the effects of α on the accuracy and number of replacements r, respectively, are presented. The accuracy was measured by prequential MSE and the number of replacements refers to how many models were replaced through the stream.

Table 7.3: Average and standard deviation MSE according to the value of α.

α	0.005		0.010		0.050		0.100		0.200	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
No drift										
F1	**6.51**	0.52	**6.55**	0.53	6.86	0.60	7.13	0.63	7.46	0.57
F2	**11.07**	1.93	**10.92**	1.57	11.53	2.26	11.65	1.85	12.08	1.92
F3	**16.57**	1.77	**16.24**	1.63	17.35	1.82	17.69	1.97	18.48	1.96
F7	**15.34**	2.99	**14.91**	2.95	15.86	2.86	16.43	3.22	17.21	3.56
F11	**23.75**	0.54	**23.76**	0.53	23.89	0.52	23.94	0.58	24.06	0.53
Gradual rotation										
F1	**21.49**	11.00	**21.85**	10.94	**24.14**	10.44	29.80	13.34	46.26	32.95
F2	23.59	19.18	**19.88**	19.67	25.08	20.26	25.71	28.45	31.99	25.74
F3	59.42	35.86	**50.80**	25.50	63.73	30.48	71.55	38.97	90.00	66.96
F7	52.60	30.55	**48.08**	24.80	57.58	30.96	58.12	31.74	83.69	61.94
F11	**42.68**	3.82	**42.02**	4.22	44.58	4.13	47.84	5.14	54.77	7.59
Gradual replacement										
F1	21.56	2.82	**20.68**	2.43	23.68	2.59	28.14	6.44	47.39	21.22
F2	28.64	3.98	**26.85**	3.63	30.10	3.44	32.44	3.48	39.75	8.23
F3	35.31	3.76	**33.87**	3.74	37.44	3.02	40.58	3.52	51.74	12.78
F7	**34.40**	5.50	**33.99**	5.10	37.12	5.05	39.56	6.06	53.47	14.06
F11	40.41	2.79	**39.00**	2.42	40.23	2.25	42.30	2.64	47.41	5.82
Abrupt										
F1	21.88	3.27	**20.74**	2.44	23.70	2.76	28.07	6.52	44.24	18.61
F2	28.93	3.67	**27.03**	3.43	29.86	3.08	32.13	3.77	40.59	12.24
F3	35.82	4.00	**33.59**	3.49	37.63	2.78	40.88	3.27	50.91	12.18
F7	34.87	4.96	**33.14**	4.58	36.36	4.50	40.26	6.20	52.87	16.85
F11	40.73	3.12	**39.15**	2.49	40.78	2.32	42.67	2.82	47.21	5.47
Data expansion										
F1	6.22	1.27	**5.96**	0.96	6.76	1.03	7.47	1.50	9.96	2.86
F2	**11.26**	2.53	**11.14**	2.24	12.22	2.89	13.07	3.11	14.90	4.50
F3	14.62	3.41	**13.66**	2.88	15.40	3.15	16.74	3.53	18.11	4.01
F7	**9.15**	1.87	**9.04**	1.73	9.92	2.14	10.09	2.25	10.55	2.14
F11	**24.04**	0.51	**24.01**	0.62	24.35	0.62	24.53	0.61	25.00	0.65

In general, the lower values of α resulted in better ensemble accuracy. The bold shaded values in Table 7.3 indicate the lower MSE for each dataset and more than one value highlighted in the same line means that the values are statistically similar, according to the paired *t*-test with a significance level of 95%. The results indicate the best ensemble accuracies were achieved when setting α at 0.010 and it also contributes to more stable ensembles since at this setting lower standard deviations were also achieved. The adjustment of α also showed a different impact on accuracy, according to the type of drift. The ensemble accuracy on datasets with no drift and simulating data expansion are less affected by different values of α, while in datasets simulating gradual and abrupt drift, the ensemble is highly affected by the increase of α. The different effects of the smoothing factor according to the type of drift is illustrated in Figure 7.9.

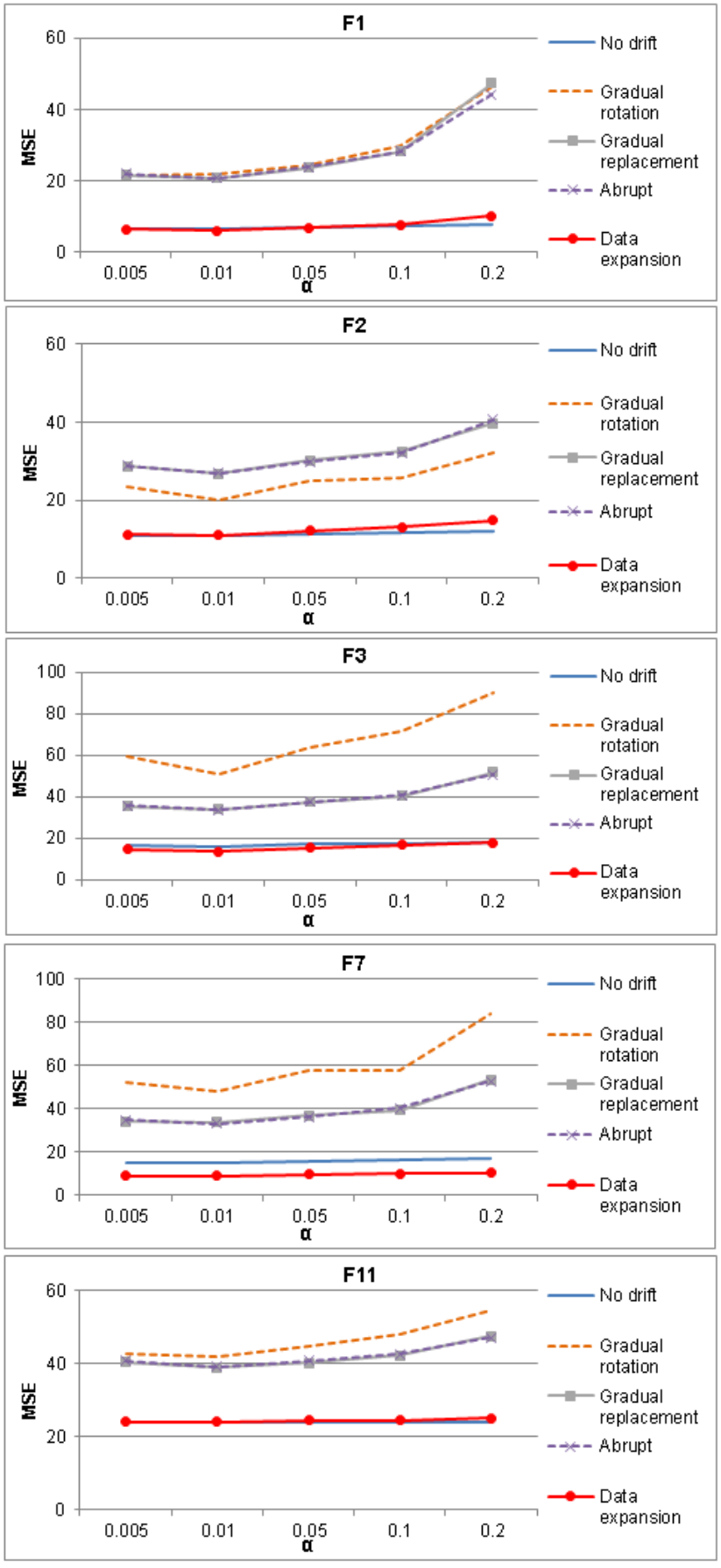

Figure 7.9: Effect of α on MSE for different types of drift.

The smoothing factor has an important impact on the number of replacements. The replacement is not only related to the accuracy of the ensemble but also affects the ensemble's computational performance. The higher the number of replacements, the higher is the computational cost due to the training of new models. The average number of replacements for each value of r is presented in Table 7.4.

Table 7.4: Average and standard deviation of the number of replacements according to the α value.

α	0.005		0.010		0.050		0.100		0.200	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
No drift										
F1	73.1	30.3	57.1	25.0	23.7	8.5	12.4	7.1	3.2	2.7
F2	58.8	19.8	55.8	20.3	29.9	10.8	19.6	7.4	10.3	3.9
F3	64.4	22.8	60.9	24.1	30.7	11.0	18.9	7.1	10.2	5.0
F7	52.5	19.9	53.3	19.3	29.7	12.8	17.1	7.4	7.7	4.1
F11	56.8	20.8	46.0	20.4	19.0	8.9	11.1	5.8	4.8	3.3
Gradual rotation										
F1	125.8	60.7	127.0	63.5	70.9	29.7	52.6	22.2	31.2	18.1
F2	99.2	51.6	91.7	49.9	54.9	31.4	38.0	18.3	27.0	16.6
F3	125.6	57.3	127.8	63.8	81.1	35.7	59.7	29.0	39.3	17.3
F7	101.9	38.8	107.2	51.4	61.1	27.0	43.6	18.3	30.9	14.2
F11	127.8	55.4	135.2	68.5	95.3	36.9	74.0	23.9	53.0	14.8
Gradual replacement										
F1	114.6	47.0	118.3	58.8	62.6	23.3	44.7	16.2	23.8	14.4
F2	95.4	35.7	104.9	49.2	57.2	18.9	42.9	12.6	29.8	8.6
F3	94.4	28.1	100.0	46.9	53.2	16.3	39.2	9.8	26.5	9.7
F7	81.1	30.1	80.0	29.1	48.8	14.3	37.1	10.0	23.0	11.0
F11	93.7	29.0	101.1	44.6	68.7	25.2	52.6	16.3	37.9	10.4
Abrupt										
F1	111.3	47.3	109.3	53.8	60.0	21.8	43.1	14.5	25.0	13.8
F2	91.2	32.0	96.1	44.1	55.5	17.4	42.4	11.5	29.8	10.2
F3	90.5	28.7	95.5	41.7	51.7	15.4	38.9	9.7	26.7	9.6
F7	76.3	25.3	80.0	30.1	48.3	14.2	36.9	10.1	23.1	11.4
F11	91.5	29.9	94.7	41.8	63.1	22.5	49.2	14.5	37.6	10.1
Data expansion										
F1	103.2	44.4	100.4	51.2	50.9	18.9	36.0	10.0	21.7	10.4
F2	90.6	37.6	77.0	39.0	36.4	10.3	27.5	8.3	14.2	9.1
F3	98.9	37.8	100.0	48.3	52.5	18.6	38.9	10.9	27.9	8.1
F7	50.1	30.0	36.6	21.4	29.9	17.7	24.9	14.6	21.1	13.4
F11	59.4	19.8	48.8	17.8	25.9	7.6	18.4	6.5	10.0	5.2

From Table 7.4 it is possible to observe that lower values of α are not only responsible for higher accuracy but also for a higher number of replacements. This trade-off must be taken into consideration when adjusting the smoothing factor. An important advantaged of the proposed approach is that the number of replacements adjusts to the type of drift. The algorithm replaced fewer models when there is no drift and, on the other hand, more models were replaced in gradual rotation drift, where the drift is constant through the data stream. The average number of replacements for all datasets, according to the different types of drift, are shown in Table 7.5.

Table 7.5: Average number of replacements for each type of data drift.

α	**0.005**	**0.010**	**0.050**	**0.100**	**0.200**	**Avg**
No drift	61.1	54.6	26.6	15.8	7.2	**33.1**
Gradual rotation	116.1	117.8	72.6	53.6	36.3	**79.3**
Gradual replacement	95.8	100.8	58.1	43.3	28.2	**65.3**
Abrupt	92.1	95.1	55.7	42.1	28.4	**62.7**
Data expansion	80.4	72.6	39.1	29.1	19.0	**48.1**

The replacement threshold r also influences the algorithm's rate of replacement. Table 7.6 shows the average MSE according to the r value.

Table 7.6: Average and standard deviation MSE according to threshold value r.

r	200		300		400		500		600	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
No drift										
F1	**6.68**	0.63	6.86	0.65	6.89	0.70	7.02	0.66	7.07	0.68
F2	**11.27**	1.90	**11.26**	1.61	**11.49**	1.80	**11.63**	2.30	**11.58**	2.10
F3	**16.80**	1.97	**17.07**	1.85	17.30	1.98	17.53	1.90	17.65	2.19
F7	**15.67**	3.31	**16.14**	3.01	**15.95**	3.31	**16.19**	3.48	**15.79**	2.98
F11	**23.80**	0.57	**23.90**	0.55	**23.89**	0.53	**23.87**	0.61	23.95	0.50
Gradual rotation										
F1	**20.73**	8.24	25.01	11.64	26.44	12.98	33.21	19.18	38.16	33.35
F2	**21.75**	15.53	**22.29**	14.16	**27.19**	31.86	**23.90**	18.69	31.12	29.11
F3	**55.32**	24.76	**58.47**	28.55	67.45	36.88	74.03	56.00	80.24	58.24
F7	**54.85**	31.94	**56.28**	36.63	**57.21**	31.24	**58.40**	35.46	73.34	57.43
F11	**40.66**	3.51	42.83	3.99	45.77	4.80	49.35	5.97	53.28	7.27
Gradual replacement										
F1	**22.22**	4.44	24.09	6.55	27.12	11.16	32.63	18.08	35.38	19.10
F2	**28.71**	4.74	**29.77**	5.29	31.03	5.15	32.98	6.58	35.28	8.61
F3	**36.40**	5.04	**36.91**	5.06	39.52	7.64	42.00	10.70	44.13	12.28
F7	**35.99**	5.94	**36.72**	7.07	39.01	9.03	41.26	10.82	45.54	15.20
F11	**39.35**	3.17	**39.91**	2.90	41.50	3.05	43.10	3.64	45.48	6.15
Abrupt										
F1	**22.32**	4.62	23.64	5.94	26.83	8.95	30.79	14.25	35.06	18.36
F2	**28.53**	4.52	30.08	4.51	31.13	6.05	33.16	8.90	35.63	11.16
F3	**36.42**	5.03	**37.47**	5.67	39.73	7.33	41.30	8.76	43.92	12.47
F7	**35.25**	5.98	**36.56**	7.95	39.48	7.90	41.09	12.88	45.11	15.89
F11	**38.94**	2.76	40.37	2.95	42.04	3.10	43.70	4.06	45.50	5.34
Data expansion										
F1	**6.47**	1.41	**6.62**	1.46	7.08	1.64	7.67	2.35	8.53	3.04
F2	**11.98**	2.86	**12.03**	2.98	**12.57**	3.59	12.90	4.12	13.11	3.38
F3	**14.56**	3.46	**14.97**	3.39	15.95	3.94	16.42	4.07	16.63	3.46
F7	**9.65**	2.13	**9.79**	2.03	**9.87**	2.04	**9.66**	2.30	**9.78**	2.06
F11	**24.25**	0.65	**24.24**	0.68	24.42	0.65	24.51	0.73	24.51	0.76

In general, lower threshold values resulted in better ensemble accuracy. The best MSE values are highlighted (bold), and when more than one value is highlighted, it means that the values are statistically similar, according to a paired t-test with 95% confidence. The adjustment of r has a lower impact on the ensemble's accuracy compared to α, especially when there is no drift and for gradual rotation and data expansion. The effects of r on MSE are illustrated in Figure 7.10.

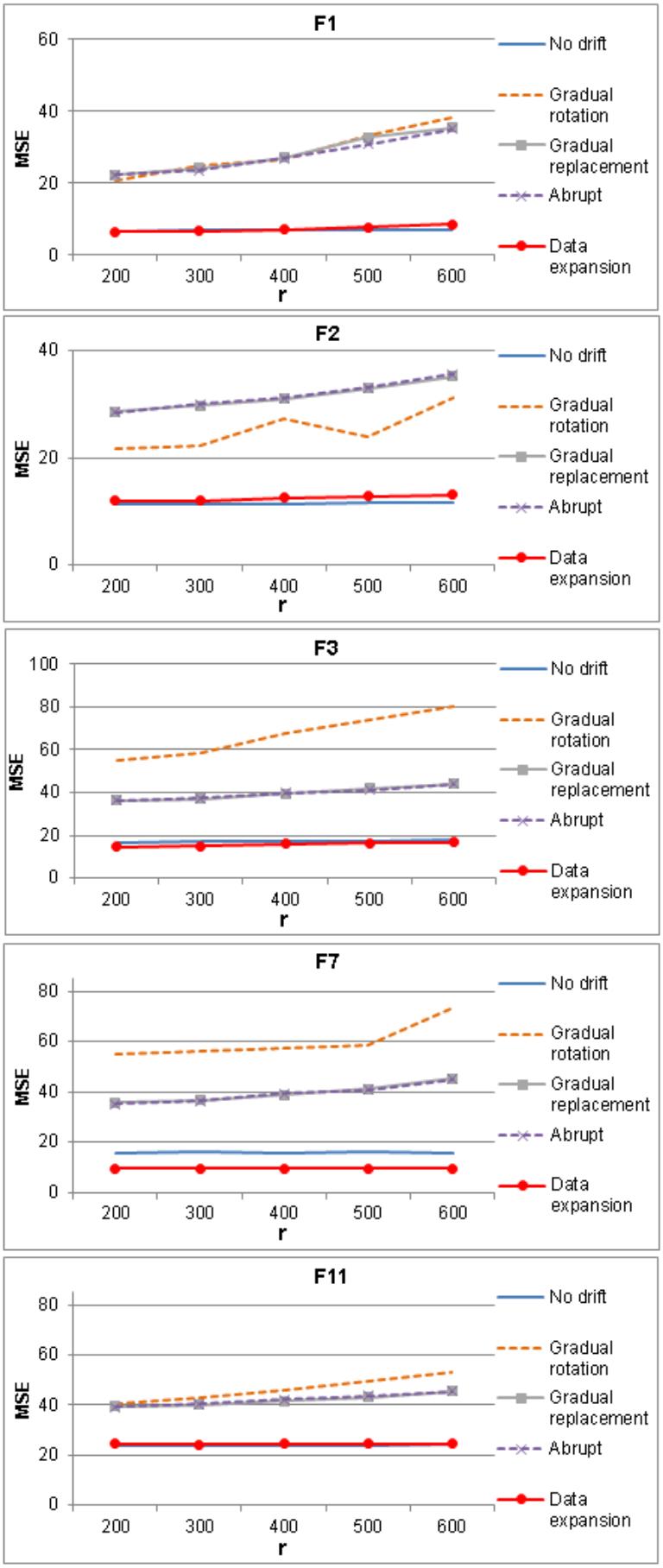

Figure 7.10: Effect of r on MSE for different types of drift.

The effect of r on the number of replacements is shown in Table 7.7.

Table 7.7: Average and standard deviation of the number of replacements according to the r value.

r	200		300		400		500		600	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
No drift										
F1	56.8	45.8	40.6	32.6	29.1	23.8	23.7	19.3	19.2	16.3
F2	51.9	30.6	40.2	23.5	32.3	19.6	26.8	16.5	23.3	14.1
F3	57.7	35.7	43.0	27.4	32.6	20.0	27.7	17.5	24.2	15.8
F7	48.1	30.0	36.8	22.7	28.7	18.5	25.5	16.9	21.1	14.4
F11	45.9	32.6	32.7	24.9	24.0	18.9	19.4	15.4	15.6	13.1
Gradual rotation										
F1	136.1	82.3	99.5	49.6	72.2	32.9	55.9	26.5	43.8	21.2
F2	96.3	69.0	71.9	44.2	59.8	34.3	44.2	22.7	38.6	18.2
F3	145.2	76.3	102.1	48.2	77.5	32.3	60.0	23.4	48.7	19.1
F7	106.2	61.7	80.6	41.5	63.5	32.5	51.9	24.2	42.5	20.4
F11	162.2	67.9	115.7	39.6	86.7	25.3	66.9	17.4	53.8	13.6
Gradual replacement										
F1	122.3	71.4	87.2	45.9	64.7	32.5	49.6	26.8	40.2	22.5
F2	102.4	58.2	76.1	37.3	60.3	27.4	49.4	20.9	42.0	17.8
F3	93.4	53.7	73.9	38.5	57.8	27.8	47.7	22.5	40.5	19.3
F7	77.6	39.3	63.7	30.6	49.9	22.1	43.6	21.0	35.2	18.4
F11	107.3	44.9	84.3	33.4	64.6	22.8	53.0	16.9	44.7	15.0
Abrupt										
F1	115.2	66.9	83.7	43.9	63.3	30.7	48.3	24.1	38.2	21.0
F2	96.4	51.5	70.9	33.2	58.4	25.0	48.8	20.2	40.6	17.5
F3	89.1	49.2	71.7	35.7	56.6	27.4	46.6	20.7	39.2	17.9
F7	77.1	37.7	60.4	27.5	49.9	22.0	42.1	19.3	35.1	18.1
F11	103.0	43.1	79.0	30.3	61.1	21.2	50.5	16.5	42.6	13.8
Data expansion										
F1	102.8	66.7	73.7	41.1	55.5	29.4	44.1	23.6	36.1	19.4
F2	79.1	54.7	58.3	38.3	44.0	27.7	35.0	20.4	29.3	17.6
F3	98.8	59.8	74.5	39.3	58.5	27.9	47.3	20.6	39.1	17.5
F7	51.2	30.4	39.7	24.4	31.3	20.2	26.6	17.3	22.5	14.9
F11	60.8	18.9	54.4	19.8	28.7	9.7	18.6	8.4	8.9	4.7

Based on the effects of r on the MSE and on the number of replacements, a good commitment between accuracy and computational efficiency was achieved when setting the threshold at 300. At this level, statistically similar accuracy is achieved in most of the cases, compared to setting the threshold at 200. The average increase in MSE at this level is of 3.6%, while the number of replacements is reduced, on average, by 32.6%.

In the case of benchmark datasets, the smoothing factor affects the accuracy in different ways. While in Quality and Maintenance datasets the best accuracy is achieved setting α 0.010, as in the synthetic datasets, in Energy and House datasets the best accuracy is achieved with higher α values. The results are summarised in Table 7.8 and the best results are highlighted (bold shaded values). The results were subjected to paired t-test with 95% of confidence and more than one value is highlighted for the same dataset indicate that the results are statistically equal.

Table 7.8: Average and standard deviation of MSE according to the α value.

α	**0.005**		**0.010**		**0.050**		**0.100**		**0.200**	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
Energy	1.02E+4	5.53E+2	9.90E+3	4.51E+2	9.62E+3	4.87E+2	9.38E+3	5.86E+2	**9.06E+3**	6.24E+2
House	4.65E+9	2.74E+8	4.36E+9	2.52E+8	3.90E+9	3.00E+8	**3.78E+9**	3.76E+8	**3.79E+9**	4.07E+8
Quality	0.504	0.004	**0.501**	0.005	0.508	0.011	0.516	0.012	0.526	0.009
Maint.	4.96E-7	2.69E-7	**3.76E-7**	1.10E-7	4.23E-7	6.88E-8	4.76E-7	7.09E-8	5.80E-7	9.46E-8

Similarly to synthetic datasets, lower values of α are related to a higher number of replacements; however, a higher number of replacements is not necessarily related to better accuracy, as can be observed by the results in Energy and House datasets, where the better accuracy was achieved with a lower number of replacements. The average number of replacements for the benchmark datasets is presented in Table 7.9.

Table 7.9: Average and standard deviation of the number of replacements according to the α value.

α	0.005		0.010		0.050		0.100		0.200	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
Energy	88.8	47.7	58.6	20.8	24.0	6.2	17.8	4.8	14.0	3.6
House	158.8	91.9	135.7	71.9	87.7	34.0	76.4	29.0	64.4	26.2
Quality	73.6	38.5	49.4	24.6	19.0	8.0	10.0	6.2	3.4	3.0
Maint.	152.1	67.5	182.7	91.6	88.9	29.3	66.3	17.3	50.8	10.6

The threshold values had a similar effect on benchmark datasets, compared to the effect on synthetic datasets. Lower r value resulted in better accuracy and also is responsible for a higher number of replacements. The effects of r on MSE and on the number of replacements are shown in Tables 7.10 and 7.11, respectively.

Table 7.10: Average and standard deviation of MSE according to the r value.

r	200		300		400		500		600	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
Energy	**9.16E+3**	5.15E+2	9.40E+3	6.12E+2	9.71E+3	5.99E+2	9.84E+3	5.73E+2	1.00E+4	6.45E+2
House	**3.83E+9**	5.18E+8	**3.92E+9**	4.00E+8	4.07E+9	3.67E+8	4.25E+9	4.26E+8	4.41E+9	4.11E+8
Quality	**0.502**	0.008	0.507	0.012	0.511	0.011	0.515	0.012	0.518	0.013
Maint.	**4.60E-7**	2.29E-7	**4.30E-7**	1.36E-7	**4.53E-7**	1.16E-7	4.88E-7	1.10E-7	5.20E-7	1.61E-7

Table 7.11: Average and standard deviation of the number of replacements according to the r value.

r	200		300		400		500		600	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
Energy	66.2	60.5	44.0	32.3	34.6	24.0	31.2	20.7	27.3	16.4
House	187.2	90.2	119.4	45.6	88.0	31.9	70.8	22.1	57.5	21.3
Quality	56.7	49.9	37.7	33.7	25.7	22.1	19.8	17.3	15.5	14.0
Maint.	153.4	105.6	131.7	79.4	101.0	51.4	83.9	39.5	70.9	30.0

Using the threshold r set at 300 results in an average reduction of 43.5% on the number of replacements, compared to r set at 200. This has a limited effect on accuracy, in case of House and Maintenance datasets, the average accuracies are statistically similar and in case of Energy and Maintenance, the increase on average MSE is of 2.6% and 1.1%, respectively.

Base on the analysis of the different settings of α and r, it was possible to establish their values for the next set of experiments, where the proposed ensemble is compared to an existing approach for data stream regression from literature. For all datasets, r is set at 300, where good commitment between accuracy and the number of replacements is achieved. The α is set at 0.01 for all datasets except the Energy and House, which showed better accuracy when α is set at higher values, for these datasets α is set at 0.2.

7.6. B-NNRW VALIDATION

In this section, the proposed algorithm, using the α and r defined in the previous section, is compared to two approaches, the O-DNNE (Ding et al., 2017) and a single SLFNN. The SLFNN used in this research is trained using the Levenberg-Marquardt backpropagation algorithm. The network is built using the built-in Matlab® implementation with its default hyperparameters, except the number of neurons, which is optimised using the SSHT algorithm. The initial range for the number of nodes is [20, 250] and the optimised values found by SSHT are summarised in Table 7.12.

Table 7.12: Number of SLFNN hidden nodes for each dataset.

	F1	F2	F3	F7	F11	Energy	House	Quality	Maint.
Number of nodes	20	20	20	20	20	20	20	20	250

In this research, an online approach for the SLFNN is applied. For each new instance, after the prediction is performed, the network error on that instance is backpropagated through the network and its weights are updated. The approach is therefore called O-SLFNN.

The second algorithm, the O-DNNE, is a recent ensemble algorithm from literature proposed to deal with regression data streams. This approach achieved good results compared to benchmark algorithms; however, since all the base models must be optimised at the same time, the training and updating process becomes

computationally expensive, especially when the number of nodes or the number of base models is increased. A brief description of O-DNNE and how its optimisation is affected by an increase in the number of base models and/or hidden nodes is presented in the next section.

7.6.1. The online DNNE

The Online DNNE (Ding et al., 2017) is an approach derived from the decorrelated neural network ensemble (DNNE) to deal with online regression problems. The DNNE algorithm builds an ensemble of single-hidden layer NNRWs and incorporates the concept of negative correlation learning (NCL) in the training process to create a well-diversified set of models. The main idea behind NCL is to train the models simultaneously in a way that their individual errors are decorrelated since no major gains can be obtained from a combination of outputs if they are positively correlated (Rosen, 1996). Given a data set of size N consisting of pairs (x_n, y_n) and $f_i(x_n)$ the output of sample x_n of the ith model in the ensemble of size (M), the error function for the ith model can be written as Equation 7.10.

$$E_i = \sum_{n=1}^{N} \frac{1}{2} (f_i(x_n) - y_n)^2 \tag{7.10}$$

Rosen (1996) propose a modification in the error function (Equation 7.10) to include a decorrelation penalty term p_i, resulting in the following learning error (Equation 7.11):

$$e_i = \sum_{n=1}^{N} \left[\frac{1}{2} (f_i(x_n) - y_n)^2 - \lambda p_i(x_n) \right] \tag{7.11}$$

where $\lambda \in [0, 1]$ is a regularization factor. Alhamdoosh and Wang (2014) adopted the penalty term formulated in Equation 7.12:

$$p_i(x_n) = (f_i(x_n) - \bar{f}(x_n)) \sum_{j \neq i} (f_j(x_n) - \bar{f}(x_n)) \tag{7.12}$$

where $\bar{f}(x_n)$, which is used instead of the target value y_n to reduce the correlation among ensemble individuals mutually. The final DNNE consists of a set of weights $B_{ens} = [\beta_{11}, \dots, \beta_{1L}, \dots, \beta_{M1}, \dots, \beta_{ML}]^T_{ML\times 1}$, where β_{ij} is the output weight of the *j*th hidden node of the *i*th model, and can be obtained by solving the following linear system (Equation 7.13):

$$\hat{B}_{ens} = H^{+}_{corr} T_h \tag{7.13}$$

The H^{+}_{corr} is generalized pseudo-inverse (Rao and Mitra, 1971) of matrix H_{corr}. The hidden-target matrix $T_h = [\varphi(1,1), \dots, \varphi(1,L), \dots, \varphi(M,1), \dots, \varphi(M,L)]^T_{ML\times 1}$, where $\varphi(i,j)$ models the correlation between the *j*th hidden neuron of the *i*th base model and is computed as Equation 7.14:

$$\varphi(i,j) = \sum_{n=1}^{N} g_{ij}(x_n) y_n \tag{7.14}$$

where g_{ij} is the output of *j*th hidden neuron from the *i*th model given a data sample x_n.

Finally, H_{corr} is an *ML*x*ML*, where each element is given the following condition:

$$H_{corr}(p,q) = \begin{cases} C_1 \varphi(m,n,k,l) & if\ m = k; \\ C_2 \varphi(m,n,k,l) & otherwise. \end{cases}$$

where *p*, *q* = 1,…, *M*x*L*; $m = \left\lceil \frac{p}{L} \right\rceil$; $n = \big((p-1) mod\ L\big) + 1$; $k = \left\lceil \frac{q}{L} \right\rceil$; $l = \big((q-1) mod\ L\big) + 1$. The constants C_1 and C_2, and the correlation between the *j*th hidden neuron of the *i*th base model and *l*th hidden neuron of the *k*th base model $\varphi(m,n,k,l)$, are formulated as shown in Equations 7.15, 7.16 and 7.17, respectively.

$$\varphi(i,j,k,l) = \sum_{n=1}^{N} g_{ij}(x_n) g_{kl}(x_n) \tag{7.15}$$

$$C_1 = 1 - 2\lambda \frac{(M-1)^2}{M^2} \tag{7.16}$$

$$C_2 = 2\lambda \frac{M-1}{M^2} \tag{7.17}$$

The online version of DNNE (Ding et al., 2017), both H_{corr} and T_h are updated according to new data simply by adding the H_{corr} and T_h computed using the new data and then adding up to the existing H_{corr} and T_h, as shown in Equations 7.18 and 7.19, respectively:

$$H_{corr} = H_{corr}^{old} + H_{corr}^{old} \tag{7.18}$$

$$T_h = T_h^{old} + T_h^{new} \tag{7.19}$$

For further details, the reader can refer to Ding et al. (2017). Once H_{corr} and T_h are updated, the $\hat{B}_{ens}$ is recomputed according to Equation 7.13.

The online-DNNE can process effectively a single new data sample due to the fact that the processing cost of computing Equation 7.13 is not affected by the number of samples to be evaluated. However, the computation of H_{corr}^{+} is very sensitive to the number of NNRW hidden nodes as well as the number of models. Considering an ensemble with *n* nodes and *m* models, an increment of one node results in an increment in the size of H_{corr} matrix in the order of $m^2(n^2 + 2n + 1)$; likewise, an increment of one model in the ensemble increases the size of H_{corr} in the order of $n^2(m^2 + 2m + 1)$. Attempts to overcome the issue of computational constraints due to Moore-Penrose computation have been carried out (Cao et al., 2016), but the authors found that the results do not apply to regression problems since they do not satisfy stability conditions.

The O-DNNE hyperparameters were optimised using the SSHT algorithm and the best hyperparameter setting for each dataset is shown in Table 7.13.

Table 7.13: Optimised O-DNNE hyperparameters for each dataset.

	F1	F2	F3	F7	F11	Energy	House	Quality	Maint.
Number of Models	15	15	15	5	5	15	5	15	10
Number of Nodes	150	150	150	150	20	150	150	20	150
Initial weights range	0.1	0.11	0.1	0.1	0.1	0.1	0.8	0.1	0.8
Regularisation factor	0.01	0.01	0.38	0.01	0.01	0.01	0.01	0.01	0.38

The update of the O-DNNE is executed at pre-defined intervals, i.e. the new instances are buffered and when it reaches 500 instances, the O-DNNE is updated.

7.6.2. B-NNRW performance on synthetic datasets

In this section, the performance of B-NNRW is compared to O-SLFNN and O-DNNE on synthetic data streams in the presence of concept drift. The algorithms are compared in terms of accuracy and computational time. Each algorithm runs 10 times on each dataset and the following metrics are computed, the prequential MSE for each run, the training time and the testing time, i.e. the time the algorithm takes to process the entire stream, including the prediction and updating time. The results are organized by accuracy and computational time for each time of drift and are submitted to a paired t-test with 95% confidence to check for statistically equal means. The accuracy of each algorithm, measure in MSE, are presented in Tables 7.14 - 7.18 for datasets with no drift, gradual rotation, gradual replacement, abrupt drift and data expansion, respectively.

Table 7.14: Average MSE and standard deviation for each algorithm on datasets with NO DRIFT.

	B-NNRW		O-DNNE		O-SLFNN	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
F1	6.33	0.43	**4.72**	0.40	11.68	1.50
F2	**10.60**	1.00	14.68	1.53	19.65	1.85
F3	16.29	1.70	**10.16**	1.07	34.18	8.32
F7	14.40	3.24	**8.71**	0.97	20.98	3.07
F11	23.60	0.58	**22.89**	0.58	47.05	1.69

The results in Table 7.14 show how the algorithms performed on synthetic datasets without the presence of concept drift. The average and standard deviation of MSE of each algorithm are presented and the best results are highlighted. The O-DNNE showed better accuracy in all datasets except F2, where the B-NNRW performed better. The results also showed the advantage of ensemble approaches compared to single models, the B-NNRW and O-DNNE reduced the error, on average by 45.1% and 53.0%, respectively.

The results are also illustrated in Figure 7.11, where a 30-period SE moving average is shown for dataset F1.

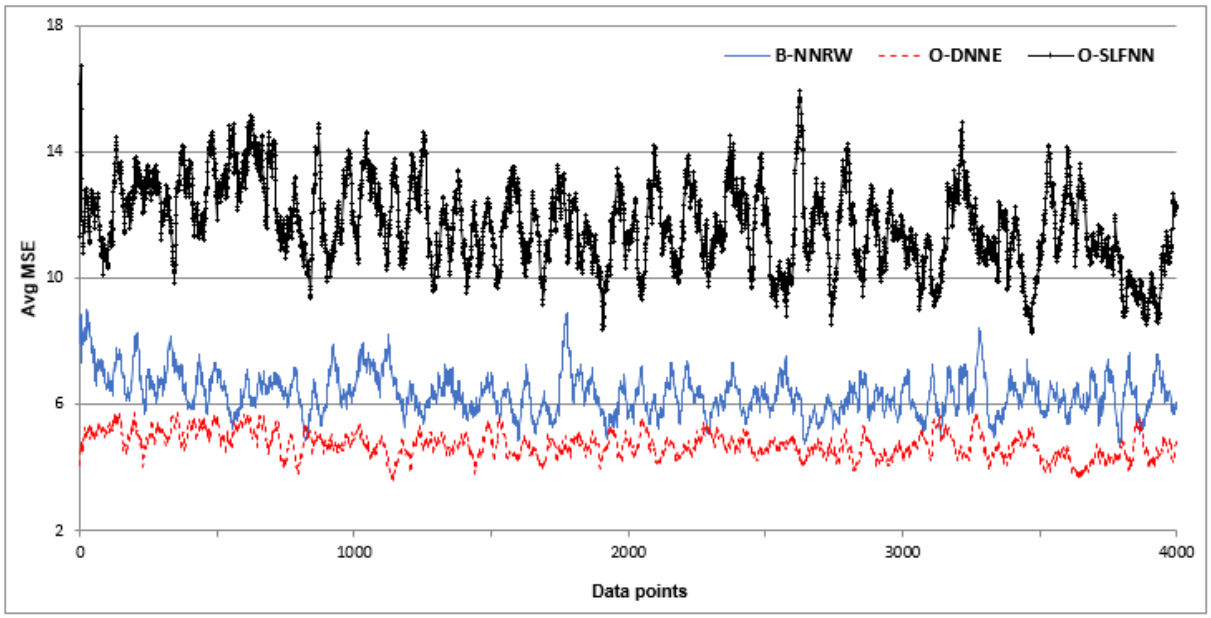

Figure 7.11: Moving average SE for each algorithm on F1 dataset with no drift.

The updating mechanisms have a limited effect on the accuracy through data stream and do not improve the accuracy significantly. This behaviour is also observed in the remaining synthetic datasets with no drift.

The accuracy of the algorithms on datasets with drift simulated by hyperplane gradual rotation is shown in Table 7.15.

Table 7.15: Average MSE and standard deviation for each algorithm on datasets with GRADUAL ROTATION.

	B-NNRW		O-DNNE		O-SLFNN	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
F1	22.10	6.96	69.93	46.34	**17.89**	3.61
F2	**16.54**	7.51	71.60	82.39	24.59	7.43
F3	**51.94**	28.85	310.95	163.14	**44.46**	12.16
F7	**31.46**	18.82	189.66	126.18	**28.57**	6.29
F11	**39.01**	0.55	100.30	1.79	51.90	1.53

The results for gradual rotation showed that the B-NNRW resulted in better accuracy in two datasets (F2 and F11). Moreover, statistically similar accuracy, compared to the O-SLFNN, was observed in F3 and F7, while the O-SLFNN performed better in dataset F1. The O-DNNE performed poorly in this type of drift; however, its accuracy could be improved by increasing the updating frequency at the expense of an increase in computational time. Figure 7.12 shows the 30-period moving average SE for each algorithm through the F2 simulated data stream with gradual rotation.

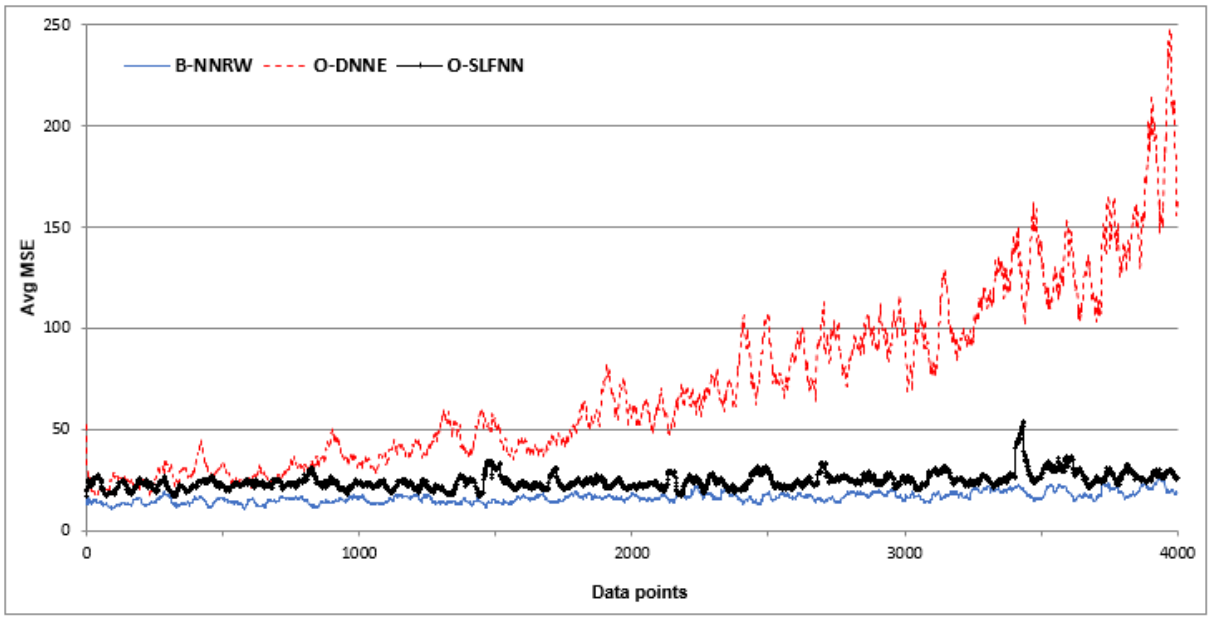

Figure 7.12: Moving average SE for each algorithm on F2 dataset with gradual rotation.

Table 7.16 shows the results of each algorithm on datasets with gradual replacement.

Table 7.16: Average MSE and standard deviation for each algorithm on datasets with GRADUAL REPLACEMENT.

	B-NNRW		O-DNNE		O-SLFNN	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
F1	**19.12**	0.57	52.68	2.29	**18.49**	3.81
F2	**26.40**	3.83	65.46	2.81	**26.28**	5.31
F3	**31.13**	1.25	58.59	4.24	40.25	9.18
F7	32.67	4.58	58.62	2.67	**25.73**	3.76
F11	**37.99**	1.65	72.12	1.46	52.62	1.56

The B-NNRW produced a lower error in F3 and F11 datasets when the gradual replacement is present. In datasets F1 and F2, it showed statistically similar results compared to O-SLFNN and O-SLFNN performed better on F7. Similarly to the gradual rotation, the O-DNNE was not effective to tackle gradual replacement, which could be solved by improving the updating frequency. The moving average SE for F7 dataset is illustrated in Figure 7.13.

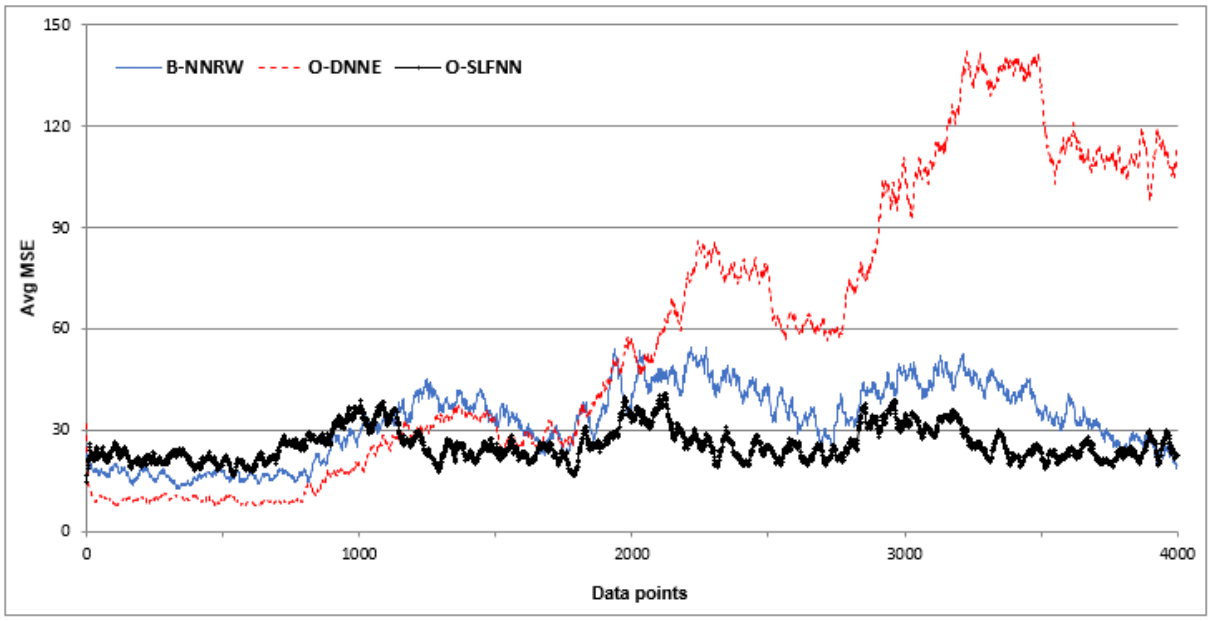

Figure 7.13: Moving average SE for each algorithm on F7 dataset with gradual replacement.

It is possible to observe the accuracy of O-DNNE increases when it updates between the drift points (500, 1500, 2500 and 3500), while the updating at the drifting points decrease the accuracy since the algorithm updates with old concepts. In general, the O-SLFNN adapts to drift much faster compared to the ensemble algorithms, this is mainly due to two factors: there is only one model to update and it the model is updated at every new instance. The latter implies unnecessary computation in the absence of drift, as observed in Figure 7.10.

The accuracy in datasets with abrupt drift is shown in Table 7.17. For the same reasons observed on datasets simulating gradual replacement, the O-DNNE as not able to deal with the abrupt drift adequately. The B-NNRW showed better accuracy in datasets F2, F3 and F11, while O-SLFNN performed better in F7. In dataset F1, both B-NNRW and O-SLFNN showed statistically similar results.

Table 7.17: Average MSE and standard deviation for each algorithm on datasets with ABRUPT drift.

	B-NNRW		**O-DNNE**		**O-SLFNN**	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
F1	**19.13**	0.62	54.09	1.57	**18.12**	3.49
F2	**25.15**	2.31	63.74	2.03	30.34	5.17
F3	**31.91**	1.40	60.13	3.94	39.47	7.62
F7	31.77	4.01	58.37	1.47	**25.62**	2.05
F11	**37.34**	1.18	73.06	1.57	55.97	1.42

The behaviour of the algorithms through the simulated F1 data stream with abrupt drift is illustrated in Figure 7.14.

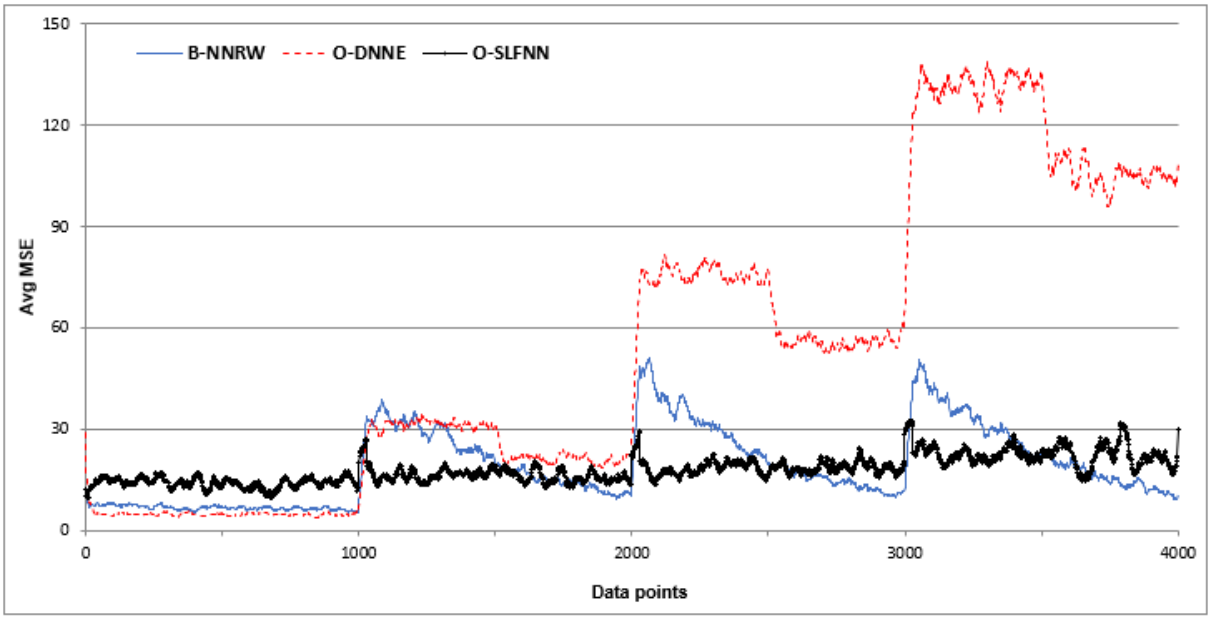

Figure 7.14: Moving average SE for each algorithm on F1 dataset with abrupt drift.

Similarly to the gradual replacement, the O-SLFNN adapts fast to drift, while B-NNRW needs to replace all the models with old accuracy before the levels of accuracy are re-established.

The last type of drift analysed in this research is data expansion, which results are shown in Table 7.18. For this type of drift, B-NNRW and O-DNNE showed statistically equal results in all datasets. This type of drift results in more complex changes in the relationship between input and output, compared to the previously simulated drifts. In this case, the O-SLFNN did not update effectively and performed poorly compared to B-NNRW and O-DNNE.

Table 7.18: Average MSE and standard deviation for each algorithm on datasets with DATA EXPANSION.

	B-NNRW		**O-DNNE**		**O-SLFNN**	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
F1	**5.71**	0.64	**6.18**	1.21	11.73	1.83
F2	**11.17**	1.78	**13.80**	3.74	26.21	7.71
F3	**12.23**	1.97	**14.65**	3.89	26.91	5.20
F7	**8.75**	1.41	**8.24**	1.06	18.44	3.43
F11	**23.59**	0.35	**23.40**	0.51	46.17	1.09

The effect of data expansion on the accuracy of the algorithms is illustrated in Figure 7.15, for the F3 data stream.

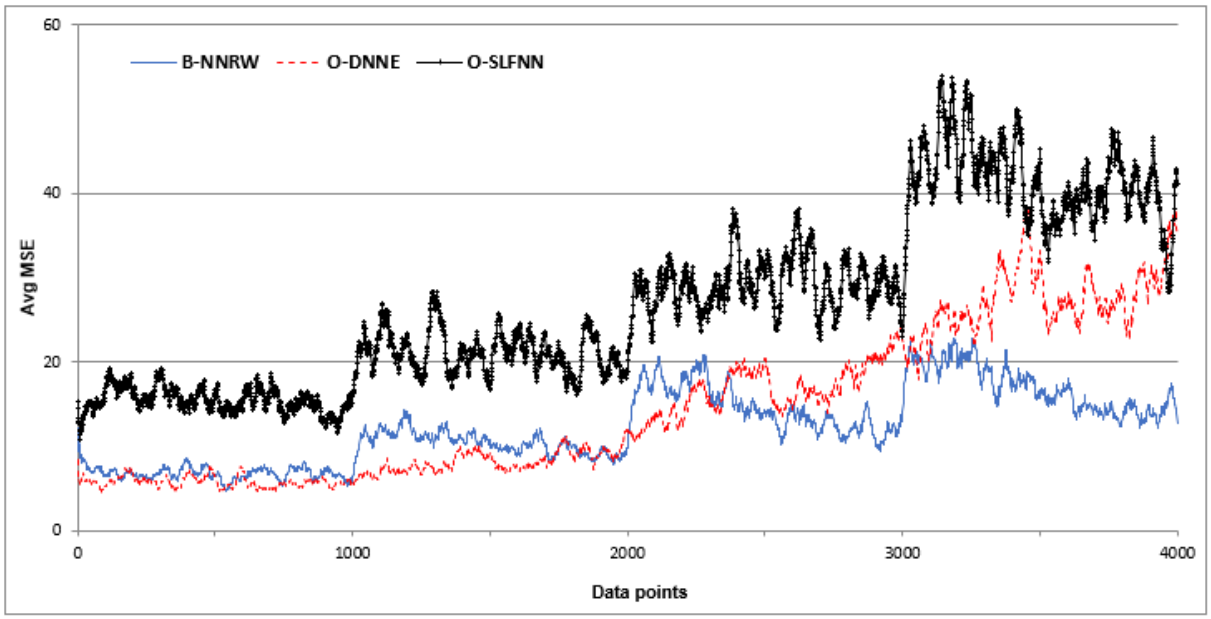

Figure 7.15: Moving average SE for each algorithm on F3 dataset with data expansion.

The O-DNNE showed a good learning capability for datasets with data expansion. It is possible to observe that the average accuracy does not change abruptly in the drift points, as it happens to B-NNRW and O-SLFNN. The B-NNRW recover its accuracy level as the low performing members are replaced, but slowly compared to other types of drift. This is explained by the fact that the input distribution before the drift is contained within the new distribution, i.e. instances of the new distribution were already learnt by the models and may slow the replacing process.

Another important aspect of algorithms for data stream is the computational time. The training and testing time that each algorithm took to process the entire data stream were collected. The training time refers to the time spent to train the models using the initial data, while the testing time includes the predictions and updating of the models. The average training and testing time of each dataset for all types of drift were averaged and the results are shown in Table 7.19.

Table 7.19: Average computational time and standard deviation for each algorithm and dataset.

	B-NNRW		O-DNNE		O-SLFNN	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
Training time (seconds)						
F1	**0.236**	0.071	37.053	0.877	0.638	0.231
F2	**0.221**	0.007	37.054	0.803	0.541	0.152
F3	**0.223**	0.010	37.066	0.740	0.627	0.177
F7	**0.048**	0.004	3.183	0.069	0.487	0.078
F11	**0.044**	0.002	0.054	0.001	0.693	0.512
Testing time (seconds)						
F1	**5.231**	0.262	314.788	6.053	108.368	0.856
F2	**4.916**	0.218	315.073	6.027	107.797	2.255
F3	**5.088**	0.243	314.989	6.089	108.135	0.591
F7	**3.878**	0.043	44.829	0.849	107.733	0.521
F11	3.910	0.118	**3.514**	0.063	115.745	13.635

In terms of computational time, the proposed algorithm showed a significant advantage compared to O-DNNE and O-SLFNN. The only exception was in dataset F11, in this case, the best accuracy of O-DNNE were achieved with a reduced number of base models (5) and nodes (20), while in datasets F1, F2 and F3, the number of base models and nodes for the best accuracy were 15 and 150, respectively. In dataset F7, where O-DNNE showed moderate computational time, the optimised number of base models and nodes were 5 and 150, respectively. These results demonstrate the drawbacks of O-DNNE when the number of models and/or nodes need to be increased for better accuracy. The O-SLFNN showed good training times and lower testing times compared to O-DNNE, however, the testing time is approximately 20 times higher compared to B-NNRW.

7.6.3. B-NNRW performance on benchmark datasets

In this section, the proposed algorithm is further validated by evaluating its performance on benchmark datasets from public data repositories, described in Section 4.3. For each dataset, the first 1.000 data points were used for test and the remaining data points were used to simulate the data streams. Each experiment was

run 10 times and the average and standard deviation of MSE were computed. The results are shown in Table 7.20 and the lower MSE for each problem is highlighted.

Table 7.20: Average MSE and standard deviation for each algorithm on benchmark datasets.

	B-NNRW		O-DNNE		O-SLFNN	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
Energy	8.76E+03	0.34E+03	13.67E+03	0.65E+03	**6.43E+03**	1.30E+03
House	**3.57E+09**	0.32E+09	13.87E+09	3.20E+09	**3.42E+09**	0.16E+09
Quality	**0.500**	0.002	0.532	0.003	0.909	0.045
Maintenance	5.31E-07	3.74E-07	**1.90E-07**	0.91E-07	27.10E-07	8.86E-07

The algorithms showed different performances according to the dataset. The B-NNRW achieved the best accuracy in Quality, where the MSE was 6.0% and 45.0% lower compared to O-DNNE and O-SLFNN, respectively. In House dataset, B-NNRW and O-DNNE achieved statistically equal results while O-DNNE error was approximately 4 times higher. The O-SLFNN resulted in lower MSE in Energy dataset and the best algorithm in Maintenance was the O-DNNE. The smoothed MSE for through the stream is illustrated in Figures 7.16, 7.17, 7.18 and 7.19 for Energy, House, Quality and Maintenance problems, respectively.

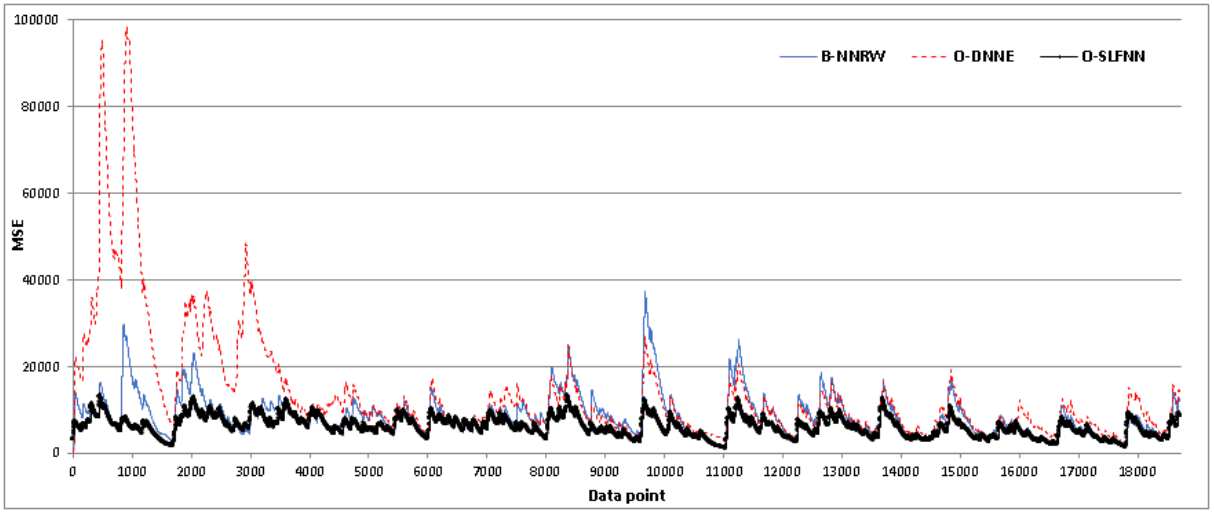

Figure 7.16: Smoothed MSE for each algorithm on Energy dataset.

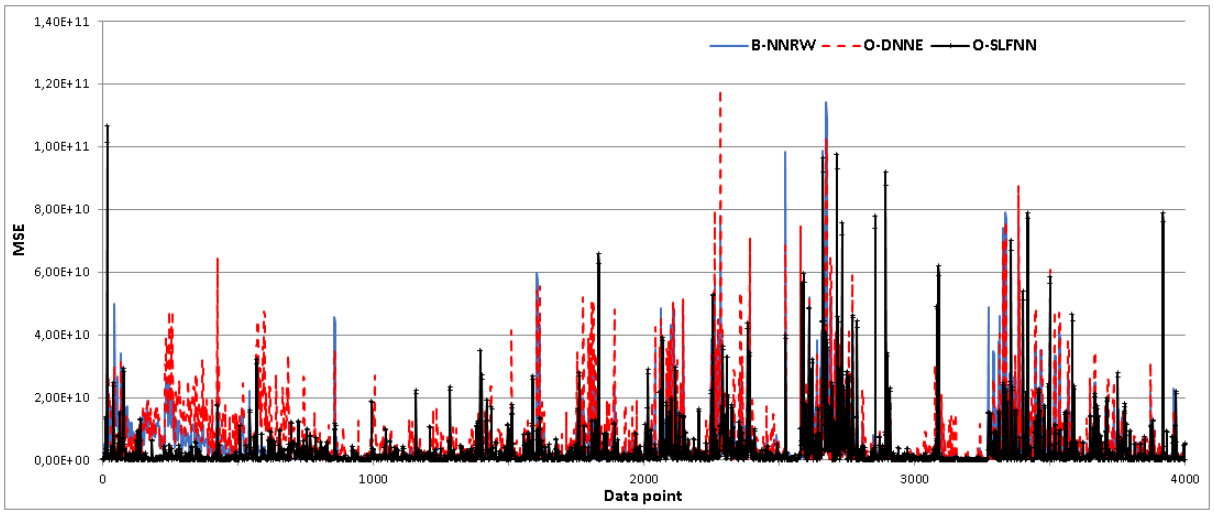

Figure 7.17: Smoothed MSE sample for each algorithm on House dataset.

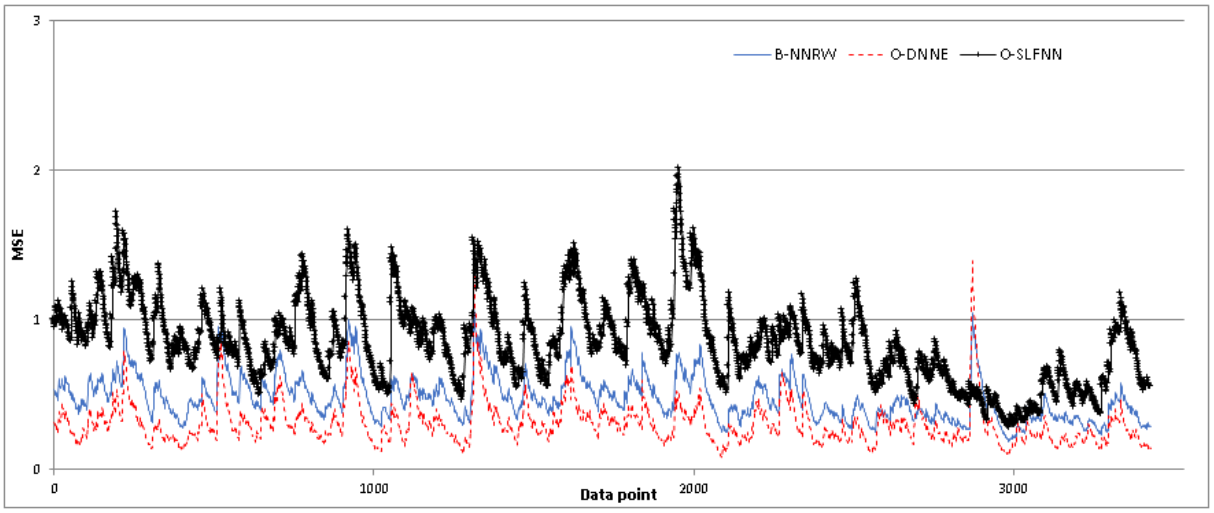

Figure 7.18: Smoothed MSE sample for each algorithm on House dataset.

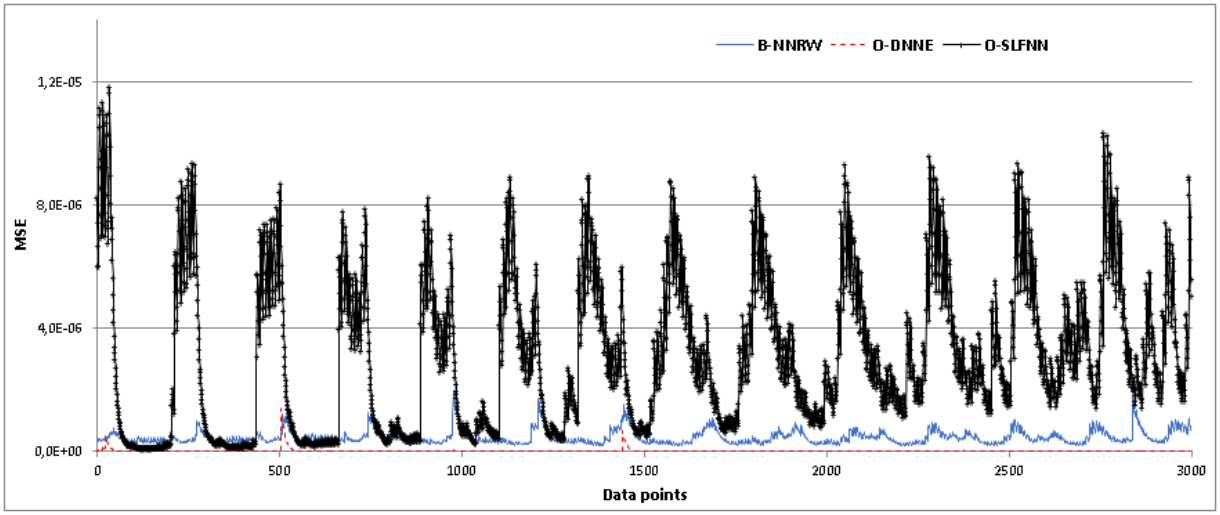

Figure 7.19: Smoothed MSE sample for each algorithm on House dataset.

Besides the MSE, the Mean Absolute Percentage Error (MAPE) was also computed. The MAPE is computed according to the Eq. 7.20

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i} * 100\%$$

Eq. 7.20

where y_i and $\hat{y}_i$ are the true values and the predicted value, respectively. The results are summarised in Table 7.21

Table 7.21: Average MAPE and standard deviation for each algorithm on benchmark datasets.

	B-NNRW		**O-DNNE**		**O-SLFNN**	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
Energy	**52.23%**	2.82%	92.71%	2.23%	**49.04%**	11.52%
House	24.34%	2.16%	39.09%	1.54%	**21.61%**	1.04%
Quality	**9.74%**	0.05%	10.00%	0.02%	12.41%	0.34%
Maintenance	0.05%	0.02%	**0.01%**	0.00%	0.10%	0.01%

By the MAPE values, it is possible to analyse how much each algorithm deviates from the true value on average and its impact on the final results. For example, choosing the wrong algorithm for Maintenance problem would have a lower impact on the results compared to the impact of the wrong choice on House dataset. It is important to note that the datasets are used for simulating data streams and for practical applications, a more in-depth analysis of the data would be required.

The results in terms of computational time (elapsed time) for the benchmark datasets are shown in Table 7.22.

Table 7.22: Average computational time and standard deviation for each algorithm and dataset.

	B-NNRW		O-DNNE		O-SLFNN	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
Training time (seconds)						
Energy	**0.15**	0.11	37.42	0.18	1.45	0.44
House	**0.16**	0.01	3.20	0.05	0.38	0.07
Quality	**0.13**	0.00	0.50	0.00	0.35	0.04
Maintenance	**0.20**	0.01	14.96	0.14	94.22	15.46
Testing time (seconds)						
Energy	**19.90**	0.05	1477.71	2.46	776.85	189.83
House	**18.76**	0.06	220.34	0.64	447.16	1.04
Quality	**3.98**	0.03	12.02	0.20	92.86	0.26
Maintenance	**12.16**	0.63	374.17	0.70	4741.52	16.37

Although the B-NNRW did not achieve the best accuracy on all datasets, it was able to process the entire data streams much faster compared to O-DNNE and O-SLFNN, reaching up to 390 times faster compared to O-SLFNN and 74 times faster compared to O-DNNE.

In this chapter, a new ensemble for data stream regression with concept drift was developed. The results showed that the proposed algorithm, the B-NNRW, is a competitive alternative for solving data stream regression problems, especially when time constraints are involved. It was demonstrated the trade-off between accuracy and the number of base models and, for simplicity, the evaluation of the ensemble on the data streams was carried out using 30 base models for all datasets. From the results showed in Section 7.4, it is possible to observe that the accuracy of B-NNRW could be improved by increasing the ensemble size in most of the datasets (F1, F2, F3, F7, House and Maintenance).

The algorithm updates constantly without the need to determine the updating frequency, adapting to all types of drift without the need of any assumption about the type of concept drift. The updating mechanism increases or decreases the number of replacements according to characteristics of the dataset to adapt to the various types of drift. The main drawback identified during the experiment is the slow recovery of accuracy levels when an abrupt drift occurs. This is due to the fact that, when an abrupt data drift occurs, all the models are affected and the replacement tends to be

constant until new models, trained on the new concept, force the old models to be replaced. Effective mechanisms to deal with abrupt drift could significantly increase the algorithm’s performance in terms of accuracy.

8. CONCLUSIONS

This research has explored the data stream regression problem in the presence of concept drift. The big data paradigm has assigned new challenges for the ML algorithms that include processing data on high speed in a continuous manner and adapting to changes in the environment. Despite the increasing amount of research regarding data streams and concept drift, only a few published methods for data stream regression can be found in the literature. The new approaches presented in this thesis are proven to have considerably enhanced the state-of-the-art in the area of data stream regression and hyperparameter optimisation.

This chapter is structured in four sections. Firstly, the main findings and conclusions are presented. Then, it is shown to what level research objectives established at the beginning of this study are fulfilled. The research contributions are outlined in Section 8.3, and Section 8.4 closes this study by outlining some limitations and suggestions for future work.

8.1. RESEARCH FINDINGS AND CONCLUSIONS

Several characteristics of data streams must be observed when developing ML models. The nature of data streams requires computationally efficient algorithms, both in terms of memory use and processing time. For this reason, non-iterative algorithms are preferred for this type of problem.

The issue of concept drift, where the underlying concept that represents the process being modelled changes over time, has been addressed in this thesis. Some authors have worked on identifying the main types of drift; however, it is still not possible to precisely determine when it happens or which type of drift taking place in real-world applications. The existing approaches for concept drift either rely on drift detector mechanisms or update the model at a fixed rate. In the former case, the model can generate false alarms and be ineffective when more than one type of drift

is present in data. In the latter case, an inadequate adjustment of the updating rate can lead to unnecessary computations or slow reaction to concept drift.

The ensembles have been successfully applied to solve data stream regression and classification problems. For this type of task, some advantages of ensembles compared to single models can be highlighted: flexibility, high accuracy and computational efficiency. The flexibility allows the ensemble to incorporate various mechanisms to adapt to concept drift. The ensembles are easier to parallelise, and it is computationally more efficient to train several small models than a single model, especially when the computational complexity of the model increases exponentially according to the model's size.

The assessment of data stream regression algorithms requires synthetic datasets where various types of concept drift can be simulated. In this research, a need for data generation approaches where the algorithms can be effectively evaluated was identified. The existing approaches for regression data generation are limited in terms of dimensions, which make it difficult to access the performance of the algorithm on high dimensional datasets. It is also difficult to check the function shape and therefore difficult to simulate the various types of concept drift.

In general, the ensemble approaches from the literature do not address base model optimisation. The optimisation of base models can increase the overall accuracy of the ensemble and also allows the use of fewer base models, improving the ensemble's computational efficiency. Different variations of NNRWs can be found in literature, the main representatives are the RVFL and the ELM. Their optimisation involves not only tuning hyperparameters but also design hyperparameters; additionally, only a few research has studied the main difference between them.

8.2. RESEARCH ACHIEVEMENTS

This research was aimed at developing an ensemble algorithm to solve data stream regression problems with concept drift. To accomplish the research aim,

several research objectives were set out at the beginning of this study. These objectives have been achieved as explained below:

Objective 1: Develop a robust methodology for generating synthetic data streams and simulating concept drift.

This objective was fulfilled in Chapter 4, where functions designed originally for evaluating optimisation algorithms were adapted to simulate regression data streams. The approach allows simulation of various types of concept drift and can generate datasets with any number of predictive variables. Moreover, it is possible to visualise the shape of the functions at low dimensions (1 or 2 attributes), which allows assessing the effect of the simulated drift on the data.

Objective 2: Analysis of the randomised NN approaches and their main differences.

This objective was addressed in Chapter 5, where the main differences between the existing NNRWs approaches were identified and analysed, and the design decisions involving the construction of an effective NNRW were discussed. It was found that, despite the ELM popularity, design elements of RVFL increase the accuracy of randomised NNs.

Objective 3: Development of new hyperparameter tuning algorithms.

This objective was fulfilled in Chapter 6, where a new algorithm for hyperparameter optimisation, the SSHT was proposed. The algorithm works by analysing the effect of each hyperparameter on the model's variability and therefore prioritising the search for the optimal value.

Objective 4: Development of effective updating mechanisms to cope with concept drift.

This objective was achieved in Chapter 7, where a new ensemble algorithm based on NNRWs was developed. The new algorithm incorporates an updating mechanism that replaces base models based on their contribution to the ensemble. The updating mechanism

constantly evaluates each model's performance and does not need any previous assumption about the type of drift or when the drift is expected to occur.

Objective 5: Test and validate the proposed approaches using synthetic and benchmark datasets and comparing with existing methods from the literature.

This objective was achieved in Chapters 6 and 7. In Chapter 6, the new hyperparameter optimisation algorithm was validated on the optimisation of NNRWs and the results were compared to the GA algorithm. The SSHT was able to achieve better convergence to optimised hyperparameter sets with fewer evaluations compared to GA. In Chapter 7, the new ensemble algorithm for data stream regression was compared to the existing approaches from the literature. The algorithm was able to adapt to all types of drift and showed competitive accuracy with much less computational effort.

8.3. RESEARCH CONTRIBUTIONS

The main contribution of this research is the development of a new ensemble algorithm, the B-NNRW, that has improved the state-of-the-art in the data stream regression with concept drift problems. The experiments demonstrated that the proposed approach can achieve competitive results compared to existing approaches in all datasets and can adapt to all types of concept drift.

The assessment on synthetic datasets with concept drift showed that B-NNRW achieved better accuracy on 7 problems, statistically similar accuracy on 10 problems and worse accuracy on 3 problems, compared to the best technique on each problem. In the cases where B-NNRW showed better accuracy, it reduced the error by 25.7%, on average, while in the 3 cases where other techniques showed better accuracy, the error was reduced 19.9% on average. The proposed algorithm also reduced significantly the computational time by 75.1% and 95.8%, on average, compared to O-DNNE and O-SLFNN, respectively.

This research study has also contributed to the knowledge by:

- Proposing a new approach for simulating data streams for regression problems with various types of concept drift and with any number of predictive variables.
- Advancing the understanding of the NNRWs from an optimisation perspective and incorporating it into a bagging ensemble updating algorithm.
- Proposing a promising hyperparameter optimisation algorithm that is able to find optimised hyperparameter sets with less computational effort. The SSTH showed better convergence with 55.7% fewer evaluations, on average, compared to GA.
- Disseminating an article on data stream regression accepted to be published by the Soft Computing journal, 2019.

8.4. LIMITATIONS AND FUTURE RESEARCH OPPORTUNITIES

The proposed data generation approach is able to successfully simulate regression data streams and various types of drift and allows simulation of datasets with a high number of features. Following the common practice in concept drift studies, the proposed approach for data generation simulates only one type of concept drift at a time. This might not be realistic in practical applications and combining more than one drift could potentially enhance the assessment of the data stream algorithms. Moreover, the effectiveness of the proposed approach to simulate classification problems could be investigated.

The novel hyperparameter optimisation algorithm was effective in optimising the NNRW hyperparameters. The SSHT algorithm uses a simple interpolation method for searching the optimised values of continuous hyperparameters. The SSHT can be substantially improved by replacing the interpolation method for more advanced techniques, such as gradient-based models. A more effective search

mechanism could reduce even further the number of evaluations. Additionally, comparing the SSHT to other techniques, such as Random Search or SMO based techniques could improve the validation of the algorithm and establish its advantages and disadvantages compared to the existing methods.

The main strength of the proposed ensemble for regression problems was its computational efficiency. The algorithm achieved competitive accuracy, however, investigating the causes of lower accuracy in some datasets could improve the overall effectiveness of B-NNRW. It is important to note that the ensemble size was fixed at 30 base models. Further research and in-depth analysis of the effects of ensemble size on the accuracy is required to achieve better performance.

One drawback of the B-NNRW identified in this research is the slow reaction to sudden drift compared to an online approach. This is due to the fact that, when a sudden data drift happens, all the existing base models are affected. It is necessary that models trained on the new concept are included in the ensemble in order to force the replacement of models trained in old concepts. Some potential solutions to this issue include:

- Using a few base models that are kept on the ensemble and are updated on an online basis. These models would adapt faster to the new concept and force the replacement of models trained with data from old concepts.
- Including an abrupt drift detection mechanism in order to trigger additional model replacement.

REFERENCES

Agrawal, R.; Ghosh, S.; Imielinski, T.; Iyer, B.; Swami, A. **An Interval classifier for database mining applications.** In: Proceedings of the 18th VLDB Conference, Vancouver, British Columbia, Canada. 560-573, 1992.

Aiolfi, M.; Timmermann, A. **Persistence in forecasting performance and conditional combination strategies.** Journal of Economics. 135 (1): 32-53, 2006.

Aksu, C.; Gunter, S.I. **An empirical analysis of the accuracy of SA, OLS, ERLS and NRLS combination forecasts.** International Journal of Forecasting. 8: 27-43, 1992.

Alaba, P.A.; Popoola, S.I.; Olatomiwa, L.; Akanle, M.B.; Ohunakin, O.S.; Adetiba, E.; Alex, O.D.; Atayero, A.A.A; Daud, W.M.A.W. **Towards a more efficient and cost-sensitive extreme learning machine: A state-of-the-art of recent trend.** Neurocomputing. 350: 70-90, 2019.

Alhamdoosh, M.; Wang, D. **Fast decorrelated neural network ensembles with random weights.** Information Sciences. 264: 104-117, 2014.

Almeida, R.; Goh, Y.M.; Monfared, R.; Steiner, M.T.A.; West, A. **An ensemble based on neural networks with random weights for online data stream regression**. Soft Computing. 1-21, 2019.

Almeida, R.; Steiner, M.T.A. **Aplicação de uma rede neural aumentada ajustada por delineamento de experimentos para resolução de problemas de corte e empacotamento**. In: Proceedings of the XVI Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM 2013), Rio de Janeiro, Brazil. 2013.

Armstrong, J.S. **Principles of forecasting: a handbook for researches and practitioners.** Kluwer Academic Publishing, Boston. 30, 2001.

Barros, R.C.; Basgalupp, M.P.; Freitas, A.A; Carvalho, A.C.P.L.F **Evolutionary Design of Decision-Tree Algorithms Tailored to Microarray Gene Expression Data Sets**. IEEE Transactions on Evolutionary Computation. 18(6): 873-892, 2014.

Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. **Algorithms for hyper-parameter optimization.** NIPS'11 Proceedings for the 24th International Conference on Neural Information Processing Systems (NIPS). 2546-2554, 2011.

Bergstra, J.; Bengio, Y. **Random search for hyper-parameter optimization.** Journal of Machine Learning Research. 13: 281-305, 2012.

Bhardwaj, M.; Bhatnagar, V.; Sharma, K. **Cost-effectiveness of classification ensembles.** Pattern Recognition. 54: 84-96, 2016.

Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. **MOA: Massive online analysis.** Journal of Machine Learning Research. 11, 1601-1604, 2010.

Bifet, A.; Holmes, G.; Pfahringer, B.; Kirkby, R.; Gavaldà, R. **New ensemble methods for evolving data streams.** Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 139-148, 2009.

Bishop, C.M. **Pattern recognition and machine learning**. Springer-Verlag Berlin, Heidelberg. 2006.

Breiman, L. **Bagging predictors.** Machine Learning. 24: 123-140, 1996.

Breiman, L. **Random forests**, Machine Learning. 45 (1): 5-32, 2001.

Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. **Classification and Regression Trees**. Wadsworth, Belmont, CA, 1984.

Brown, G.; Wyatt, J.; Harris, H.; Yao, X. **Diversity creation methods: a survey and categorization.** Information Fusion. 6: 5-20, 2005.

Bruce, R. **Ensemble Learning using decorrelated neural networks.** Connected Sciences. 8: 373-384, 1996.

Brzezinski, D.; Stefanowski, J. **Combining block-based and online methods in learning ensembles from concept drifting data streams**. Information Sciences, 265: 50-67, 2014a.

Brzezinski, D.; Stefanowski, J. **Reacting to different types of concept drift: The accuracy updated ensemble algorithm**. IEEE Transactions on Neural Networks and Learning Systems, 25(1): 81-94, 2014b.

Bunn, D.W. **A Bayesian approach to the linear combination forecasts.** Operations Research. 325-329, 1975.

Cao, F.; Wang, D.; Zhu, H.; Wang, Y. **An iterative learning algorithm for feedforward neural networks with random weights.** Information Sciences. 328, 546-557, 2016.

Cao, W.; Wang, X.; Ming, Z.; Gao, J. **A review on neural networks with random weights.** Neurocomputing. 275, 278-287, 2018.

Chatelain, C.; Adam, S.; Lecourtier, Y.; Heutte, L.; Paquet, T. **Multi-objective optimization for SVM model selection.** In: Proceeding of the International Conference on Document Analysis and Recognition, ICDAR 2007. 1: 427-431, 2007.

Chen, H.; Yao, X. **Regularized negative correlation learning for neural network ensembles**. IEEE Transactions on Neural Networks. 20 (12): 1962-1979, 2009.

Deng, C.W.; Huang, G-B.; Xu, J.; Tang, J-X. **Extreme learning machines: new trends and applications**. Science China Information Sciences. 58: 1-16, 2015.

Di Martino, S.; Ferrucci, F.; Gravino, C.; Sarro, F. **A genetic algorithm to configure support vector machines for predicting fault-prone components.** in: International Conference on Product Focused Software Process Improvement. PROFES 2011. Lecture Notes in Computer Science. 6759: 247-261, 2011.

Ding, J.; Wang, H.; Li, C.; Chai, T.; Wang, J. **An online learning neural network ensemble with random weights for regression of sequential data stream**. Soft Computing, 21 (20): 5919-5937, 2017.

Ding, S.; Xu, X.; Nie, R. **Extreme learning machine and its applications**. Neural Computing and Applications. 25 (3-4): 549-556, 2014.

Domingos, P.; Hulten, G. **Mining high-speed data streams**. Knowledge Discovery and Data Mining. 71-80, 2000.

Dua, D. and Graff, C. **UCI Machine Learning Repository** [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019.

Elwell, R.; Polikar, R. **Incremental learning of concept drift in nonstationary environments.** IEEE Transactions on Neural Networks. 22 (10): 1517-1531, 2011.

Fan, W. **Systematic data selection to mine concept-drifting data streams**, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), Seattle, Washington, USA, 128-137, 2004.

Farid, D. M.; Zhang, L.; Hossain, A.; Rahman, C.M.; Strachan, R.; Sexton, G.; Dahal, K. **An adaptive ensemble classifier for mining concept drifting data streams.** Expert Systems with Applications. 40: 5895-5906, 2013.

Friedman, J.H. **Multivariate adaptive regression splines.** The Annals of Statistics. 19(1): 1-141, 1991.

Fu, A.; Dong, C.; Wang, L. **An experimental study on stability and generalization of extreme learning machines.** International Journal of Machine Learning and Cybernetics. 6: 129-135, 2015.

Gállego, P.P.; Quevedo, J.R.; Coz, J.J. **Using ensembles for problems with characterizable changes in data distribution: A case study on quantification.** Information Fusion. 34: 87-100, 2017.

Gama, J.; Medas, P.; Castillo, G.; Rodrigues, P. **Learning with drift detection**. In: SBIA Brazilian Symposium on Artificial Intelligence, Springer Verlag. 286-295, 2004.

Gama, J.; Zliobaite, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. **A survey on concept drift adaptation**. ACM Computing Surveys. 46(4): 1-37, 2014.

Gao, J.; Ding, B.; Han, J.; Fan, W.; Yu, P.S. **Classifying data streams with skewed class distributions and concept drifts.** IEEE Internet Computing, November/December. 37-49, 2008.

Ghazikhani, A.; Monsefi, R.; Yazdi, H.S. **Ensemble of online neural networks for non-stationary and imbalanced data streams.** Neurocomputing, 122: 535-544, 2013.

Gomes, H.M.; Barddal, J.P.; Enembreck, F.; Bifet, A. **A survey on ensemble learning for data stream classification**. ACM Computing Surveys. 50-2(23):1-36, 2017.

González-Castro, V.; Alaiz-Rodríguez, R.; Alegre, E. **Class distribution estimation based on the Hellinger distance**. Information Sciences. 218: 146-164, 2013.

Granger, C.W.J.; Ramanathan, R. **Improved methods of combining forecasts.** Journal of Forecasting. 3 (2): 197-204, 1984.

Guo, X.C.; Yang, J.H.; Wu, C.G.; Wang, C.Y.; Liang, Y.C. **A novel LS-SVMs hyper-parameter selection based on particle swarm optimization.** Neurocomputing. 71:3211-3215, 2008.

Guo, Y.; Jiao, L.; Wang, S.; Wang, S.; Liu, F.; Rong K.; Xiong, T. **A novel dynamic rough subspace based selective ensemble.** Pattern Recognition. 45: 1638-1652, 2015.

Hansen, J. **Combining predictors: Meta machine learning methods and bias/variance & ambiguity decompositions.** PhD dissertation. Department of Computer Science, University of Aarhus, Aarhus, Denmark, 2000.

Hoerl, A.E; Kennard, R.W. **Ridge regression: Biased estimation for nonorthogonal problems.** Technometrics. 12 (1): 55-67, 1970.

Hofer, V.; Krempl, G. **Drift mining in data: A framework for addressing drift in classification.** Computational Statistics and Data Analysis. 57: 377-391, 2013.

Huang, G-B.; Zhu, Q-Y.; Siew, C-K. **Extreme learning machine: A new learning scheme of feedforward neural networks**. In: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks. 2, 985-990, 2004.

Huang, G-B.; Chen, L.; Siew, C-K. **Universal approximation using incremental constructive feedforward networks with random hidden nodes**. IEEE Transactions on Neural Networks. 17(4): 879-892, 2006.

Huang, G-B. **An insight into extreme learning machines: Random neurons, random features and kernels**. Cognitive Computing. 6(3), 376-390, 2014.

Hulten, G.; Spencer, L.; Domingos, P. **Mining time-changing data streams.** In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, San Francisco, California. 97-106, 2001.

Igelnik, B.; Pao, Y-H. **Stochastic choice of basis functions in adaptive function approximation and the functional-link net.** IEEE Transactions on Neural Networks. 6(6): 1320-1329, 1995.

Ikonomovska, E.; Gama, J.; Dzeroski, S. **Incremental multi-target model trees for data streams.** in: Proceedings of the 2011 ACM Symposium on Applied Computing. Taiwan, 988-993, 2011a.

Ikonomovska, E.; Gama, J.; Dzeroski, S. **Learning model trees from evolving data streams.** Data Mining and Knowledge Discovery. 23: 128-168, 2011b.

Ikonomovska, E.; Gama, J.; Dzeroski, S. **Online tree-based ensembles and option trees for regression on evolving data streams.** Neurocomputing. 150: 458-470, 2015.

Iwashita, A.S.; Albuquerque, V.H.C.; Papa, J. P. **Learning concept drift with ensembles of optimum-path forest-based classifiers**. Future Generation Computer Systems. 95: 198-211, 2019.

Jiang, Y.; Zhao, Q.; Lu, Y. **Adaptive ensemble with human memorizing characteristics for data stream mining**. Mathematical Problems in Engineering. 2015.

Jose, V.R.R.; Winkler, R.L. **Simple robust averages of forecasts: Some empirical results.** International Journal of Forecast. 24 (1): 163-169, 2008.

Karalic, A. **Linear regression in regression tree leaves**. Technical Report, 1992.

Kadlec, P.; Gabrys, B. **Local learning-based adaptive soft sensor for catalyst activation prediction**. AIChE Journal. 57(5): 1288-1301, 2011.

Kittler, J.; Hatef, M.; Duin, R.P.; Matas, J. **On combining classifiers.** IEEE Transactions on Pattern Analysis and Machine Intelligence. 20: 226-239, 1998.

Kolter, J.Z.; Maloof, M.A. **Using additive expert ensembles to cope with concept drift**. In: Proceedings of the 22nd ACM International Conference on Machine Learning (ICML'05), Bonn, Germany, 2005, 449-456.

Krawczyk, B.; Minku, L.L.; Gama, J.; Stefanoswski, J.; Wozniak, M. **Ensemble learning for data stream analysis: A survey**. Information Fusion. 37, 132-156, 2017.

Kuncheva, L.I.; Whitaker, C.J. **Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy**. Machine Learning. 5 (2): 181-207, 2003.

Lemke, C.; Gabrys, B. **Meta-learning for time series forecasting in the nn gc1 competition.** In: Proceedings 16th IEEE International Conference on Fuzzy Systems, Barcelona. 2258-2262, 2010.

Lessmann, S.; Stahlbock, R.; Crone, S. F. **Optimizing hyperparameters of support vector machines by genetic algorithms.** CSREA Press, vol. 1. 74-82, 2005.

Liang, N-Y; Huang, G-B; Saratchandran P.; Sundararajan, N. **A fast and accurate online sequential learning algorithm for feedforward networks**. IEEE Transactions on Neural Networks. 17(6), 1411-1423, 2006.

Liu, Y.; Yao, X. **Ensemble learning via negative correlation.** Neural Networks. 12: 1399-1404, 1999.

Liu, J.; Zio, E. **A SVR-based ensemble approach for drifting data streams with recurring patterns**. Applied Soft Computing, 47: 553-564, 2016.

López, E.M.; Verdejo, V.G.; Vidal, A.R.F. **A new boosting design of support vector machine classifiers.** Information Fusion. 25: 63-71, 2015.

Maclaurin, D.; Duvenaud, D.; Adams, R.P. **Gradient-based hyperparameter optimization through reversible learning.** arXiv: 1502.03492, 2015.

Margineantu, D.D.; Dietterich, T.G. **Pruning adaptive boosting.** In: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann. 211-218, 1997.

Martínez-Villena, J.M.; Rosado-Muñoz, A.; Soria-Olivas, E. **Hardware implementation methods in random vector functional-link networks**. Applied Intelligence. 41: 184-195, 2014.

Masud, M.M.; Gao, J.; Khan, L.; Han, J. **A practical approach to classify evolving data streams: Training with limited amount of labeled data.** IEEE International Conference on Data Mining. 929-934, 2008.

Miche, Y.; Sorjamaa, A.; Bas, P.; Simula, O.; Jutten, C.; Lendasse, A. **OP-ELM: Optimally pruned extreme learning machine**. IEEE Transactions on Neural Networks. 21(1): 158-162, 2010.

Montgomery, D.C. **Design and analysis of experiments.** John Wiley & Sons. 8ed. 2012.

Mousavi, R.; Eftekhari, M. **A new ensemble learning methodology based on hybridization of classifier ensemble selection approaches.** Applied Soft Computing. 37: 652-666, 2015.

Nadungodage, C.H.; Xia, Y. **Online multi-dimensional regression analysis on concept-drifting data streams.** International Journal of Data Mining, Modelling and Management, 6(3): 217-238, 2014.

Omari, A.; Vidal, A.R.F. **Post-aggregation of classifier ensembles.** Information Fusion. 26: 96-102, 2015.

Oza, N.; Russell, S. **Experimental comparisons of online and batch versions of bagging and boosting.** KDD ’01. 359-364, 2001.

Pao, Y-H.; Takefuji, Y. **Functional-link net computing: Theory, system architecture, and functionalities.** IEEE Computer. 25(5) 76-79, 1992.

Pietruczuk, L.; Rutkowski, L.; Jaworski, M.; Duda, P. **How to adjust an ensemble size in stream data mining?** Information Sciences, 381: 46-54, 2017.

Pinto, A.F.; Montevechi, J.A.B.; Marins, F.A.S.; Miranda, R.C. **Algoritmos genéticos: Fundamentos e aplicações**. In: Lopes, H.S.; Rodrigues, L.C.A.; Steiner, M.T.A. Meta-heurísticas em pesquisa operacional. Curitiba: Omnipax, 1ed. 2013.

Polikar, R.; DePasquale, J.; Mohammed, H.S.; Brown, G.; Kuncheva, L.I. **Learn++.MF: A random subspace approach for the missing feature problem.** Pattern Recognition. 43 (11): 3817-3832, 2010.

Polikar, R.; Udpa, L.; Udpa, S.S.; Honavar, V. **Learn++: An incremental learning algorithm for supervised neural networks.** IEEE Transactions on Systems Man and Cybernetics. Part C: Application Review. 31 (4): 497-508, 2001.

Rao, C.R.; Mitra, S.K. **Generalized inverse of matrices and its applications**. Wiley, New York, 1971.

Read, J.; Bifet, A.; Holmes, G.; Pfahringer, B. **Scalable and efficient multi-label classification for evolving data streams.** Machine Learning. 88: 243-272, 2012.

Ren, S.; Liao, B.; Zhu, W.; Li, K. **Knowledge-maximized ensemble algorithm for different types of concept drift.** Information Sciences. 430-431: 261-281, 2018.

Rong, H.J.; Ong, Y.S.; Than, A.H.; Zhu, Z. **A fast pruned-extreme learning machine for classification problem**. Neurocomputing. 72(1-3):359-366, 2008.

Rokach, L. **Ensemble-based classifiers.** Artificial Intelligence Review. 33: 1-39, 2010.

Rosen, B.E. **Ensemble learning using decorrelated neural networks.** Connection Science. 8 (3-4), 373-393, 1996.

Scardapane, S.; Wang, D. **Randomness in neural networks: an overview.** Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 7(2): 1-18, 2017.

Schapire, R.E. **The strength of weak learnability**. Machine Learning, 5 (2): 197-227, 1990.

Schlimmer, J.C.; Granger Jr., R.H. **Incremental learning from noisy data.** Machine Learning, 1: 317-354, 1986.

Schmidt, W.F.; Kraaijveld, M.A.; Duin, R.P.W. **Feed forward neural networks with random weights**. In: Proceedings of the 11th IAPR International Conference on Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, IEEE, 1-4, 1992.

Sharkey, A.; Sharkey, N. **Combining diverse neural networks.** Knowledge Engineering Review. 12 (3): 231-247, 1997.

Shaker, A.; Hullermeier, E. **Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study.** Neurocomputing, 150: 250-264, 2015.

Smith, L.N. **A disciplined approach to neural network hyper-parameters: Part 1 – Learning rate, batch size, momentum, and weight decay.** US Naval Research Laboratory Technical Report 5510-026, 2018.

Snoek, J.; Larochelle, H.; Adams, R.P. **Practical Bayesian optimization of machine learning algorithms.** Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12 - Volume 2: 2951-2959, 2012.

Soares, S.G.; Araújo. R. **An on-line weighted ensemble of regressor models to handle concept drifts**. Engineering Applications of Artificial Intelligence, 37(0), 392-406, 2015a.

Soares, S.G.; Araújo, R. **A dynamic and on-line ensemble regression for changing environments**. Expert Systems with Applications, 42, 2935-2948, 2015b.

Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y-P.; Auger, A.; Tiwari, S. **Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session**

on Real-Parameter Optimization. Technical Report, Nanyang Technological University, Singapore, 2005.

Sun, Z.; Song, Q.; Zhu, X.; Sun, H.; Xu, B.; Zhou, Y. **A novel ensemble method for classifying imbalanced data.** Pattern Recognition. 48: 1623-1637, 2015.

Sun, Y.; Wang, Z.; Liu, H.; Du, C.; Yuan, J. **Online ensemble using adaptive windowing for data streams with concept drift**. International Journal of Distributed Sensor Networks, 2016.

Sun, Y.; Tang, K.; Zhu, Z.; Yao, X. **Concept drift adaptation by exploiting historical knowledge.** IEEE Transactions on Neural Networks and Learning Systems. 29(10): 4822-4832, 2018.

Street, W.N.; Kim, Y. **A streaming ensemble algorithm (SEA) for large-scale classification.** In: Proceeding of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA. 377-382, 2001.

Thornton, C.; Hutter, F.; Hoss, H.H; Leyton-Brown, K. **Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms**. In: Proceedings of the Knowledge Discovery and Data Mining, KDD '13. 847-855, 2013.

Tsai, C-J.; Lee, C-I; Yang, W-P. **Mining decision rules on data streams in the presence of concept drifts**. Expert Systems with Applications. 36, 1164-1178, 2009.

Tsymbal, A. **The problem of concept drift: definitions and related work.** Technical report. Department of Computer Science, Trinity College, Dublin, 2004.

Tumer, K.; Ghosh, J. **Error correlation and error reduction in ensemble classifier.** Connection Science. 8 (3-4): 385-404, 1996.

Wang, H.; Fan, W.; Yu, P.S.; Han, J. **Mining concept-drifting data streams using ensemble classifiers**, in: Proceedings of International Conference on Knowledge Discovery and Data Mining, SIGKDD, Washington DC, USA. 226-235, 2003.

Wolpert, D.H. **Stacked generalization**. Neural Networks. 5, 241-259, 1992.

Wozniak, M.; Graña, M.; Corchado, E. **A survey of multiple classifier systems as hybrid systems.** Information Fusion. 16: 3-17, 2014.

Yaqoob, I.; Hashem, I.A.T.; Gani, A.; Mokhtar, S.; Ahmed, E.; Anuar, N.B.; Vasilakos, A.V. **Big data: From beginning to future**. International Journal of Information Management, 36(6), 1231-1247, 2016.

Yin, X.C.; Huang, K.; Hao, H.W. **DE²: Dynamic ensemble of ensembles for learning nonstationary data.** Neurocomputing. 165: 14-22, 2015.

Young, S.R.; Rose, D.C.; Karnowski, T.P.; Lim, S-H.; Patton, R.M. **Optimizing Deep Learning Hyper-Parameters Through an Evolutionary Algorithm**. MLHPC2015, 15-20, 2015.

Zhang, P.; Gao, B.J.; Liu, P.; Shi, Y.; Guo, Li. **A framework for application-driven classification of data streams**. Neurocomputing. 92, 170-182, 2012.

Zhang, L.; Suganthan, P.N. **A survey of randomized algorithms for training neural networks.** Information Sciences. 364-365, 146-155, 2016.

Zhu, X.; Zhang, P.; Lin, X.; Shi, Y. **Active learning from stream data using optimal weight classifier ensemble.** IEEE Transactions on Systems, Man, and Cybernetics. 40 (6): 1607-1620, 2010.

Zliobaite, I.; Pechenizkiy, M.; Gama, J. **An overview of concept drift applications**, in: Japkowicz, N; Stefanowski, J. Big Data Analysis: New Algorithms for a New Society, Springer, 91-114, 2016.