An ensemble based on neural networks with random weights for online data stream regression

Soft Computing (2020) 24:9835–9855
https://doi.org/10.1007/s00500-019-04499-x

METHODOLOGIES AND APPLICATION

An ensemble based on neural networks with random weights for online data stream regression

Ricardo de Almeida[1,2] · Yee Mey Goh[2] · Radmehr Monfared[2] · Maria Teresinha Arns Steiner[1] · Andrew West[2]

Published online: 9 November 2019

Abstract
Most information sources in the current technological world are generating data sequentially and rapidly, in the form of data streams. The evolving nature of processes may often cause changes in data distribution, also known as concept drift, which is difficult to detect and causes loss of accuracy in supervised learning algorithms. As a consequence, online machine learning algorithms that are able to update actively according to possible changes in the data distribution are required. Although many strategies have been developed to tackle this problem, most of them are designed for classification problems. Therefore, in the domain of regression problems, there is a need for the development of accurate algorithms with dynamic updating mechanisms that can operate in a computational time compatible with today's demanding market. In this article, the authors propose a new bagging ensemble approach based on neural network with random weights for online data stream regression. The proposed method improves the data prediction accuracy as well as minimises the required computational time compared to a recent algorithm for online data stream regression from literature. The experiments are carried out using four synthetic datasets to evaluate the algorithm's response to concept drift, along with four benchmark datasets from different industries. The results indicate improvement in data prediction accuracy, effectiveness in handling concept drift, and much faster updating times compared to the existing available approach. Additionally, the use of design of experiments as an effective tool for hyperparameter tuning is demonstrated.

Keywords Ensembles · Data stream regression · Neural networks with random weights · Hyperparameter adjustment

1 Introduction

The field of machine learning has been developing rapidly and proved useful in modelling complex real-life applications. In many application domains such as social networks, financial industries, and engineering monitoring systems, data are generated in continuous flows in the form of data streams. Such data format requires the models 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).

Usually, traditional supervised learning approaches assume that the data probability distribution does not change between training data and the application data. Such assumption 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

Communicated by V. Loia.

✉ Radmehr Monfared
R.P.Monfared@lboro.ac.uk

[1] Industrial and Systems Engineering Graduate Program, Pontifícia Universidade Católica do Paraná, R. Imaculada Conceição 1155, Curitiba CEP: 80215-901, Brazil

[2] Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE11 3TU, UK

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. Besides the evolving nature of data, other properties that make the prediction task in data streams challenging include infinite length, high dimensionality, orderliness, non-repetitive, high speed, and time varying (Masud et al. 2008; Farid et al. 2013). These characteristics demand for algorithms that can process the data under time constraints with a right level of accuracy and can adapt rapidly to change in the data distribution.

A significant research effort has been made in recent years towards data stream mining, although most of the attention has been directed to supervised classification problems. Krawczyk et al. (2017) recognised an evident lack of research dedicated to the regression online learning algorithms, as also stated in Ikonomovska et al. (2015). A promising research direction in the context of modelling data streams are the ensemble methods (Krawczyk et al. 2017). 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. 2011). 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 about different concepts and new models can be easily trained to cope with new concepts that may appear. Hence, they can efficiently deal with evolving data streams and achieve superior accuracy compared to single models.

A crucial aspect of ensemble design is the choice of base models. Artificial neural networks (ANNs) is a successful model widely used in the field of machine learning for tasks such as classification and regression. Despite its success, some problems such as slow convergence and local minima have led to the emergence of research towards neural networks with random weights (NNRW) (Cao et al. 2018). NNRW was introduced by Pao and Takefuji (1992), which proposed the random vector functional link (RVFL). The main idea of such models is to randomly initialise the weights between input and hidden layers, which are kept fixed during the optimisation process, and optimise the weights between the hidden and output layers. This process results in a much lower training complexity compared to traditional ANN training algorithms.

In this article, the authors propose a new bagging ensemble method based on NNRW. The proposed approach, bagging NNRW (B-NNRW), is developed to deal with online regression problems and takes advantage of the efficiency of NNRWs and bootstrapping mechanisms to build the ensemble. The approach enables the use of different updating strategies to accommodate possible concept drifts. Three main updating mechanisms are evaluated. These include *weighting* (B-NNRW), *pruning* (BP-NNRW), and *replacement* (BR-NNRW). The updating process is performed at fixed intervals in a batch mode, i.e. data samples are stored before the updating is applied. This approach avoids the assumption of the presence and type of concept drift and also improves the accuracy in case of insufficient training data. The proposed approach relies on a primary buffer of training data to build the initial ensemble. 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 authors have evaluated the proposed approach by comparing its performance with a recent ensemble algorithm, O-DNNE (online-decorrelated neural network ensemble), developed by Ding et al. (2017), and have demonstrated an apparent enhancement to the existing approach.

As a further contribution reported by this article, the use of factorial experiments is examined and proven to be effective to adjust the algorithm's hyperparameters. The current hyperparameter optimisation approaches do not consider the importance of each hyperparameter or the interaction between them. The use of factorial experiments offers a systematic way to identify the hyperparameters that have higher effect in the algorithm's variability. Furthermore, it is possible to identify significant interactions between hyperparameters, which helps to understand how the adjustment of one hyperparameter affects another and achieve a better hyperparameter tuning.

The remainder of this article is structured as follows: in Sect. 2, the authors report state of the art in the domain of data stream prediction and Sect. 3 describes the methodology for hyperparameter tuning and the new ensemble approach to data stream regression. The experiments and resultsare outlined and discussed in Sects. 4 and 5 concludes the article and suggests further research directions.

2 Literature review

The evolving nature of data has presented as an important challenge in the field of machine learning due to several factors, such as a change in consumer preferences, change in economic dynamics, or change in environmental conditions. Besides concept drift (data mining and predictive analytics), this phenomenon is also found in the literature as covariate shift or dataset shift (pattern recognition) and non-stationary (signal processing) (Zliobaite et al. 2016).

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:

(1) Online versus batch algorithms: online algorithms learn one instance at a time while batch algorithms learn chunks of instances;
(2) Single model versus ensemble-based approaches; and
(3) Active versus passive approaches: active approaches rely on drift detector mechanisms while passive approaches assume constant drift and update the model continuously.

Ensemble approaches have been successfully applied in both classification and regression problems. The classical ensemble approaches include boosting (Schapire 1990), staking (Wolpert 1992), bagging (Breiman 1996), and random forests (Breiman 2001), and many variants 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, and the underperforming parts can be pruned to adapt to changes and usually generate more accurate concept description compared to single models (Bifet et al. 2009). 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.

2.1 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 highlights 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. Another method that trains a new model for every new chunk of data to cope with data evolution is presented by Masud et al. (2011). The classification is performed using k-NN as base models and is designed to be effective in problems with a limited amount of labelled data. Furthermore, this approach also incorporates a novels 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 non-stationary 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 detecting new classes is presented by Farid et al. (2013). The authors train three Decision Trees in a boosting manner, i.e. creating subsets of the training data based on instance weighting. A new Decision Tree 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 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 the 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, oldest first (the oldest model is excluded) and weakest first (the weakest model is excluded). The latter proved 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 underperforming models when the accuracy of the ensemble decreases.

Two algorithms based on online Hoeffding-based regression trees (Ikonomovska et al. 2010), 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 with low-accuracy 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. Their algorithm is an online version of DNNE (Alhamdoosh and Wang 2014) that is based on a decorrelated 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 Sect. 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.

2.2 Base models

The challenges posed by data streams require base models that are not only accurate but also computationally efficient. ANNs have been successfully applied for classification and regression tasks in many fields. However, some issues may make the ANN model difficult for implementation. These include high computational cost, a high number of hyperparameters, and convergence issues. ANN training process is usually based on the optimisation of a nonlinear least squares problem, where the derivatives of the loss function are back-propagated to each layer to control the weights' adjustment. This may cause slow

Table 1 Ensemble approaches developed to deal with data streams in the presence of concept drift

References	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. (2011)	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-based classifiers that consider approaches to train the new models (full memory, no memory, and window of fixed size)

convergence and/or convergence to local minima (Zhang and Suganthan 2016). Cao et al. (2018) yet point out the model selection uncertainty as an additional drawback of ANNs.

Back in the early 1990s, Schmidt et al. (1992) evaluated the use of random weights in the hidden layer of a single-hidden-layer ANN to analyse the behaviour of ANNs in terms of learning. At the same year, Pao and Takefuji (1992) proposed a similar approach called random vector functional link (RVFL). The mentioned approaches share the same principle, i.e. to randomly generate the weights between input and hidden layers, which are kept fixed during the training process. Only the weights between the hidden layer and output layer are optimised, which transform the optimisation function in a linear least squares minimisation that can be solved in a single step using pseudo-inverse algorithms or ridge regression. In this sense, the attention of the research community towards the NNRW has been increasing due to its efficiency compared to traditional ANNs.

The NNRW capabilities make it a good choice to deal with high-dimensional datasets and online applications where computational efficiency is an essential requirement. The usually lower accuracy compared to the traditional ANNs can be compensated with the use of ensembles. Several approaches can be used to introduce diversity in the ensemble and increase its accuracy, such as varying initial random weights, varying the topology of ANNs, varying the training algorithm, and varying the training datasets (Masoudnia and Ebrahimpour 2014).

Following a comprehensive review of the existing literature and methods, a number of problems with the current approaches are identified. These include:

(a) Need for development of faster algorithms that can be effectively updated under restricted time constraints;
(b) Lack of a systematic approach for hyperparameter adjustment;
(c) Need for accuracy improvement, which is always a desirable property, especially in response to concept drift.

The drawbacks of current approaches in dealing with online data stream regression with concept drift motivate the development of an ensemble algorithm using NNRW as

base models for data stream regression to improve computational efficiency and accuracy.

3 Methodology

In this section, the methodology adopted to develop the proposed approach is described. The novelty of this approach is to combine a bootstrap sampling with random feature selection to train a highly diversified pool NNRWs. In the proposed approach, the least accurate base models are replaced and deactivated at updating points, while the highly accurate models have their decision power increased through a weighting mechanism. Firstly, the use of design of experiments as an alternative for hyperparameter tuning is outlined. Then, a description of the NNRW algorithm and the B-NNRW approach is presented. Finally, some limitations of the existing method, i.e. O-DNNE, are discussed. The performance of the proposed B-NNRW algorithm will be evaluated in Sect. 4 using eight datasets (four synthetic and four benchmark datasets).

3.1 Hyperparameter optimisation

Usually, machine learning algorithms have several hyperparameters, and their adjustment is an important aspect to be taken into consideration. Besides the manual hyperparameter adjustment, another popular approach is grid search, which is used to perform an exhaustive search through all combinations of predefined levels of hyperparameters to find the best combinations. Other methods for hyperparameter optimisation include random search, Bayesian optimisation, and evolutionary algorithms (Hutter et al. 2015; Francescomarino et al. 2018). It was observed in all previous approaches that the importance of each hyperparameter and the information regarding the interactions among them are neglected.

In this research, the use of design of experiments (DOE) (Montgomery 2012) for hyperparameter adjustment, specifically the full factorial design, is proposed by the authors. The factorial design relies on the computation of all combinations of predefined levels of hyperparameters. However, similar to grid search, it offers a systematic way of analysing not only the sensitiveness of each hyperparameter but also the interactions among them.

The sensitiveness refers to the amount of change in the algorithm's response due to a change in the hyperparameter level. This can help to identify the most critical hyperparameters and direct the effort to their optimisation, especially for algorithms with a high number of hyperparameters. When a high number of hyperparameters are involved, the tuning can be done in two steps. Firstly, experiments with fewer levels are carried out to identify the importance of each hyperparameter. Secondly, a new experiment is executed for fine-tuning, keeping hyperparameters with low importance at fixed levels and therefore reducing the search space.

The interaction among hyperparameters may also play a critical role in hyperparameter optimisation. Using DOE, it is possible to identify significant hyperparameter interactions, i.e. different response patterns of one hyperparameter for different levels of a second hyperparameter. As a hypothetical example, one could observe that for the level 1 of hyperparameter A, the accuracy of the algorithm increases when the level of hyperparameter B changes from 1 to 2, while the accuracy decreases, for the same change in B, when A is at level 2.

This article will adopt the full factorial DOE to adjust the hyperparameters of both B-NNRW and O-DNNE. The authors highlight that this approach is only used to tune the hyperparameters of the initial model fitting, which are kept fixed through the evaluation of the simulated stream.

3.2 Neural networks with random weights

The use of NN with randomised weights appeared simultaneously in the works of Schmidt et al. (1992) and Pao and Takefuji (1992). While the former authors were interested in evaluating the effect of the parameters in the hidden layer, the latter proposed the RVFL network. Both approaches shared a similar architecture, which is a fully connected feed-forward neural network, as shown in Fig. 1, except for the fact that Schmidt's approach does not consider the use of direct links between the input layer and the output layer.

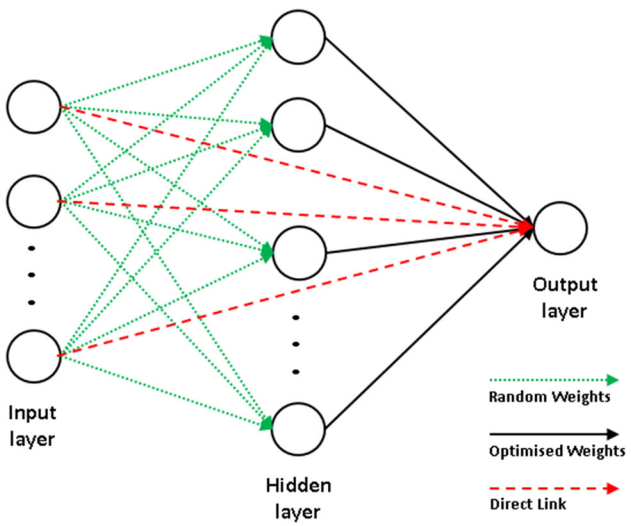

Fig. 1 Single-hidden-layer feed-forward neural network

In both cases, the weights between the input layer and the hidden layer are chosen randomly and are kept fixed during the training process. Only the weights between hidden and output layers are optimised. The main advantage of this procedure is that it converts the non-convex optimisation of parameters into a convex optimisation problem, where the global minimum can be fast approximated using a pseudo-inverse technique, such as Moore–Penrose or ridge regression.

In this work, the authors considered an NNRW with no direct link using ridge regression as the learning algorithm. The learning function can be mathematically expressed by Eq. (1):

$$\boldsymbol{T} = g(\boldsymbol{X} \cdot \boldsymbol{W}_H + \boldsymbol{B}) \cdot \boldsymbol{W}_O \tag{1}$$

where $\boldsymbol{T}$ is the target vector, $\boldsymbol{X}$ is the set of input training data, $\boldsymbol{W}_H$ is the set of weights from the input layer to the hidden layer, $\boldsymbol{B}$ is the bias vector, and $\boldsymbol{W}_O$ is the vector of weights from the hidden layer to the output layer. The function $g(\bullet)$ is the activation function, in this article the sigmoid function, given by Eq. (2):

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

Since the NNRW learning process does not rely on derivatives, as is the case of back-propagation learning algorithms, almost any nonzero activation function can be successfully applied (Huang et al. 2004).

Due to the fact that $\boldsymbol{W}_H$ and $\boldsymbol{B}$ are randomly chosen and kept fixed during the training process, the training function becomes a linear system and can be summarised as Eq. (3):

$$\boldsymbol{T} = \boldsymbol{H} \cdot \boldsymbol{W}_O \tag{3}$$

where $\boldsymbol{H}$ is the output from the hidden layer and is computed as Eq. (4).

$$\boldsymbol{H} = g(\boldsymbol{X} \cdot \boldsymbol{W}_H + \boldsymbol{B}) \tag{4}$$

The optimised set of weights $\boldsymbol{W}_o$ is the one that satisfies Eq. (5):

$$\min \|\boldsymbol{H} \cdot \boldsymbol{W}_O - \boldsymbol{T}\| \tag{5}$$

The optimisation algorithm applied by Schmidt et al. (1992), referred to as Fisher solution, can be written as Eq. (6):

$$\boldsymbol{W}_O^* = \left(\boldsymbol{H}^T \cdot \boldsymbol{H}\right)^{-1} \cdot \boldsymbol{H}^T \cdot \boldsymbol{T} \tag{6}$$

which is equivalent to the least squares (LS) estimator. The computation of $\left(\boldsymbol{H}^T \cdot \boldsymbol{H}\right)^{-1}$ may lead to instability if the matrix resulted from $\boldsymbol{H}^T \cdot \boldsymbol{H}$ 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 $\boldsymbol{H}^T \cdot \boldsymbol{H}$, given by Eq. (7).

$$\boldsymbol{W}_O^* = \left(\boldsymbol{H}^T \cdot \boldsymbol{H} + \lambda \cdot \boldsymbol{I}\right)^{-1} \cdot \boldsymbol{H}^T \cdot \boldsymbol{T} \tag{7}$$

where λ is a small constant value and $\boldsymbol{I}$ is the identity matrix, also known as penalty term, since it penalises large weights in the optimisation process.

3.3 Proposed bagging of NNRW approach

In this section, a new bagging ensemble of NNRW (B-NNRW) to tackle online regression problems is presented. The main advantage compared with O-DNNE is the potential ability to cope with problems with high rates of data arrival and also with high-dimensional data that usually require more complex models. Two interrelated factors contribute to this advantage, and they are mainly related to the matrix inversion needed for the model optimisation process. Primarily, the fact that the computational complexity is $O(M(N^3))$, in case of B-NNRW, while O-DNNE computational complexity is $O((MN)^3)$. In addition, since each model is optimised separately in B-NNRW, it is prone to be parallelised.

Both algorithms update without making any assumptions on the type or rate of drift. The diversity of B-NNRW is mainly introduced through bootstrapping the samples, not only to generate new training sets as the original bagging algorithm but also bootstrapping the features from the training set. The number of features that are used to build each model is given according to the percentage of total features, and several values are evaluated in this article. Three updating approaches are adopted and evaluated: weighting, pruning, and replacement.

(a) *Weighting* Each model in the ensemble is assigned a weight according to its accuracy in the most recent data chunk. Given the most recent chunk of data C, and an ensemble of M elements (m = 1, 2, …, M), the weight of each model is computed according to Eq. (8):

$$w_m = \frac{1}{mse_m} \tag{8}$$

where mse_m is the mean square error of the mth model, computed on C. Therefore, the output y_E of the ensemble for a data sample $\boldsymbol{x}_n$ is calculated according to Eq. (9):

$$y_E(\boldsymbol{x}_n) = \frac{\sum_{m=1}^{M} w_m * \hat{y}_m(\boldsymbol{x}_n)}{\sum_{m=1}^{M} w_m} \tag{9}$$

where $\hat{y}_m(\boldsymbol{x}_n)$ is the individual prediction of the mth model on the data sample $\boldsymbol{x}_n$.

(b) *Pruning* Pruning consists of temporarily deactivating the less accurate models of the ensemble. For each data chunk C, only Q models with the lowest error ($Q < M$) are eligible to participate in the final

ensemble decision. Given a pruning rate $p \in \mathbb{R}^{(0,1]}$, the number of Q models that participate in the prediction is given by Eq. (10):

$$Q = p * M \tag{10}$$

The idea behind keeping the deactivated models is that they may carry useful information learnt from past examples and can be helpful in future predictions, such as in the case of recurrent data concepts. The algorithm keeps track of the deactivated models' accuracy, and once one of them is included in the Q models with the lowest error, it is reactivated and used in the prediction of the subsequent chunk of data. The updating B-NNRW with pruning mechanism is referred to as BP-NNRW.

(c) *Replacement* Replacement consists of training new models, one at a time, using labelled data from the last chunk of data. The data chunk is randomly divided into training (85%) and validation (15%) sets. If the accuracy of the newly trained model is better than the worst existing model, then the worst model is replaced by the new model. This process is repeated until the desired number of new models is trained. Given a replacement rate $r \in \mathbb{R}^{[0,1]}$, the number of new models M_{new} is computed as Eq. (11):

$$M_{\text{new}} = \text{round}(r * M) \tag{11}$$

The replacement not only keeps the ensemble up to date with the most recent data concepts but also works as a natural selection mechanism since low-performing models are constantly excluded. The updating B-NNRW with replacement mechanism is referred to as BR-NNRW.

Figure 2 illustrates the B-NNRW procedure. It is assumed that an initial amount of data is available to build the initial model. The updating approaches above are applied at each predetermined updating point, which is given by the number of data samples.

3.4 Online DNNE

The O-DNNE (Ding et al. 2017) is an approach derived from the decorrelated neural network ensemble (DNNE) to deal with online regression problems. DNNE builds an ensemble of single-hidden-layer NNRWs, as described in Sect. 3.2, 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 (Rosen 1996) since no gains can be obtained from a combination of outputs if they are positively correlated. Given a dataset of size N consisting of pairs $(\boldsymbol{x}_n, y_n)$ and $f_i(\boldsymbol{x}_n)$ the output of sample $\boldsymbol{x}_n$ of the ith model in the ensemble of size (M), the error function for the ith model can be written as Eq. (12).

$$E_i = \sum_{n=1}^{N} \frac{1}{2} (f_i(\boldsymbol{x}_n) - y_n)^2 \tag{12}$$

Rosen (1996) proposed a modification in the error function (Eq. 12) to include a decorrelation penalty term p_i, resulting in the error model given by Eq. (13):

$$e_i = \sum_{n=1}^{N} \left[\frac{1}{2} (f_i(\boldsymbol{x}_n) - y_n)^2 - \lambda p_i(\boldsymbol{x}_n) \right] \tag{13}$$

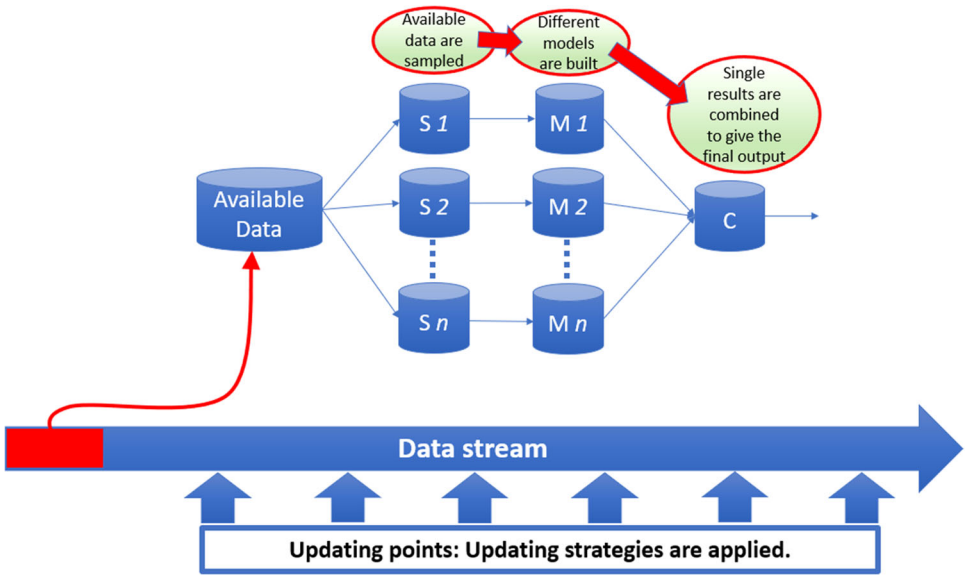

Fig. 2 B-NNRW ensemble procedure

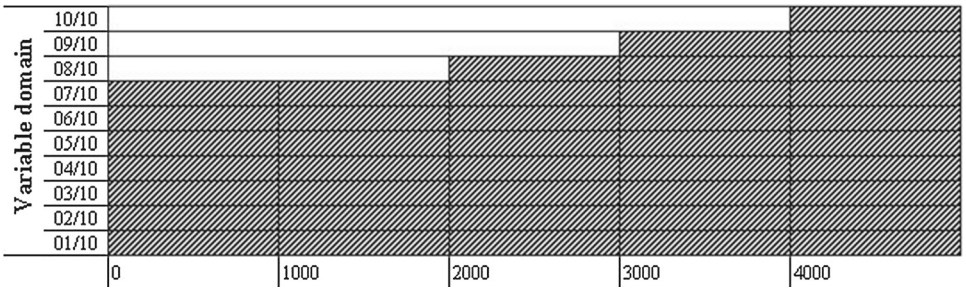

Fig. 3 Synthetic dataset generation schematics showing the drifting points at 2000, 3000, and 4000 samples

where $\lambda \in [0, 1]$ is a regularisation factor. Alhamdoosh and Wang (2004) adopted the penalty term formulated in Eq. (14):

$$p_i(\boldsymbol{x}_n) = (f_i(\boldsymbol{x}_n) - \bar{f}(\boldsymbol{x}_n)) \sum_{j \neq i} \left(f_j(\boldsymbol{x}_n) - \bar{f}(\boldsymbol{x}_n)\right) \tag{14}$$

where $\bar{f}(\boldsymbol{x}_n)$ is the average output, 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 $\boldsymbol{B}_{\text{ens}} = [\beta_{11}, \ldots, \beta_{1L}, \ldots, \beta_{M1}, \ldots, \beta_{ML}]^T_{ML \times 1}$, where β_{ij} is the output weight of the jth hidden node of the ith model and can be obtained by solving the following linear system given by Eq. (15):

$$\hat{\boldsymbol{B}}_{\text{ens}} = \boldsymbol{H}^{+}_{\text{corr}} \boldsymbol{T}_h \tag{15}$$

The $\boldsymbol{H}^{+}_{\text{corr}}$ is generalised pseudo-inverse of matrix $\boldsymbol{H}_{\text{corr}}$. The hidden–target matrix $\boldsymbol{T}_h = [\varphi(1,1), \ldots, \varphi(1,L), \ldots, \varphi(M,1), \ldots, \varphi(M,L)]^T_{ML \times 1}$, where $\varphi(i,j)$ models the correlation between the jth hidden neuron of the ith base model and the target function $G(\cdot)$ (Eq. 4) and is computed as Eq. (16):

$$\varphi(i,j) = \sum_{n=1}^{N} g_{ij}(\boldsymbol{x}_n) y_n \tag{16}$$

Finally, $\boldsymbol{H}_{\text{corr}}$ is an $ML \times ML$, where each element is given the following condition:

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

where p, $q = 1, \ldots, ML$; $m = \frac{p}{L}$; $n = ((p-1) \bmod L) + 1$; $k = \frac{q}{L}$; $l = ((q-1) \bmod L) + 1$. The constants C_1 and C_2 and the correlation between the jth hidden neuron of the ith base model and lth hidden neuron of the kth base model $\varphi(m,n,k,l)$ are formulated as Eqs. (17), (18), and (19), respectively.

$$\varphi(i,j,k,l) = \sum_{n=1}^{N} g_{ij}(\boldsymbol{x}_n) g_{kl}(\boldsymbol{x}_n) \tag{17}$$

$$C_1 = 1 - 2\lambda \frac{(M-1)^2}{M^2} \tag{18}$$

$$C_2 = 2\lambda \frac{M-1}{M^2} \tag{19}$$

In the online version of DNNE (Ding et al. 2017), both $\boldsymbol{H}_{\text{corr}}$ and $\boldsymbol{T}_h$ are updated according to new data simply by adding the $\boldsymbol{H}_{\text{corr}}$ and $\boldsymbol{T}_h$ computed using the new data ($\boldsymbol{H}^{\text{new}}_{\text{corr}}$ and $\boldsymbol{T}^{\text{new}}_h$, respectively) and then adding up to the existing $\boldsymbol{H}_{\text{corr}}$ and $\boldsymbol{T}_h$, ($\boldsymbol{H}^{\text{old}}_{\text{corr}}$ and $\boldsymbol{T}^{\text{old}}_h$, respectively) as shown in Eqs. (20) and (21), respectively:

$$\boldsymbol{H}_{\text{corr}} = \boldsymbol{H}^{\text{old}}_{\text{corr}} + \boldsymbol{H}^{\text{new}}_{\text{corr}} \tag{20}$$

$$\boldsymbol{T}_h = \boldsymbol{T}^{\text{old}}_h + \boldsymbol{T}^{\text{new}}_h \tag{21}$$

For further details, the reader could refer to Ding et al. (2017). Once $\boldsymbol{H}_{\text{corr}}$ and $\boldsymbol{T}_h$ are updated, the $\boldsymbol{B}_{\text{ens}}$ is recomputed according to Eq. (15).

The O-DNNE can effectively process a single new data sample due to the fact that the processing cost of computing Eq. (15) is not affected by the number of samples to be evaluated. However, the computation of $\boldsymbol{H}^{+}_{\text{corr}}$ 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 $\boldsymbol{H}_{\text{corr}}$ matrix in the order of $m^2(n^2 + 2n + 1)$; likewise, an increment in one model in the ensemble increases the size of $\boldsymbol{H}_{\text{corr}}$ in the order of $n^2(m^2 + 2m + 1)$.

4 Experiments

In this section, the description of the benchmark datasets used in this article is presented, followed by the hyperparameter analysis and tuning using DOE. Finally, the B-NNRW algorithm and its variants (BP-NNRW and BR-NNRW) are analysed and compared to the O-DNNE.

4.1 Datasets

The experiments are carried out using eight datasets, four synthetic datasets (Ding et al. 2017) and four benchmark datasets. The synthetic datasets are used to evaluate how the algorithms respond to drift, which is simulated by expanding the variable domain. For each synthetic dataset, 5000 samples are generated. The variable domain of each attribute is divided into ten parts, with seven parts used to create the first 2000 samples. A new part is added at every 1000 samples, expanding the domain and including new data never presented to the algorithms, as illustrated in Fig. 3.

The benchmark datasets were chosen from public repositories based on the number of features, the number of samples to simulate the stream, and the diversity of application domains. A summary of the main features of each dataset, i.e. the number of predictive attributes and the number of data samples, is presented in Table 2.

At data pre-processing stage, the *standardisation* feature scaling (zero mean and unit variance) was applied to the attributes of all datasets to avoid the effects of large differences in scale. It also prevents the impact of outliers when compared to the *normalisation* feature scale. The data transformation is given by Eq. (22):

$$\boldsymbol{X}_{\text{new}} = \frac{\boldsymbol{X} - \mu(\boldsymbol{X})}{\sigma(\boldsymbol{X})} \tag{22}$$

Additionally, a random Gaussian noise $e \in N(0, 0.02)$ is added to the input variables of the synthetic datasets. In Sects. (4.2 and 4.3), the authors analyse the hyperparameter optimisation using DOE.

4.2 Hyperparameter adjustment

In this section, the authors detail the experimental protocol to tune the models' hyperparameters using full factorial DOE. The full factorial relies on the two-way ANOVA (Montgomery 2012) and works under the following assumptions: populations are normally distributed, populations have equal variances, and samples are randomly and independently drawn.

For the purpose of hyperparameter tuning, a real application is simulated where only a portion of the data is

Table 2 Dataset features

Name	Dataset	# samples	# attributes
3-D Mex. Hat (Mex)	Synthetic	5000	2
Friedman #1 (Fried1)	Synthetic	5000	5
Friedman # 3 (Fried3)	Synthetic	5000	4
Multi (Multi)	Synthetic	5000	5
California housing[a] (Housing)	Benchmark	20,640	8
Wine quality[b] (Quality)	Benchmark	4898	11
Condition-based maintenance[b] (Maintenance)	Benchmark	11,934	14
Appliances energy prediction[b] (Energy)	Benchmark	20,640	26

[a]StatLib repository: https://lib.stat.cmu.edu

[b]UCI repository: https://archive.ics.uci.edu/ml/datasets.html

Table 3 Set of evaluated hyperparameters for DNNE and B-NNRW

Factors	Algorithm	Levels				
M	B-NNRW	40	60	80	100	120
	DNNE	3	6	9	12	15
N	B-NNRW	8x	10x	12x	14x	16x
	DNNE	60	80	100	120	140
R	B-NNRW	0.0001	0.0010	0.0050	0.0100	0.0500
	DNNE	0.1	0.2	0.3	0.4	0.5
W	B-NNRW	[− 0.50, 0.50]	[− 0.75, 0.75]	[− 1.00, 1.00]	[− 1.25, 1.25]	[− 1.50, 1.50]
	DNNE	[− 0.005, 0.005]	[− 0.020, 0.020]	[− 0.035, 0.035]	[− 0.050, 0.050]	[− 0.065, 0.065]
A	B-NNRW	0.6	0.7	0.8	0.9	1.0
	DNNE	–	–	–	–	–

available. The first 1000 data samples of each dataset were used, randomly divided into 70% for training and 30% reserved for testing. The hyperparameters to be optimised for both B-NNRW and DNNE algorithms are described as follows:

- *Number of models* (M) Number of base models that compose the ensemble;
- *Number of nodes* (N) Number of hidden nodes for each base model;
- *Regularisation factor* (R) In the case of B-NNRW, the regularisation factor is responsible for penalising large weights in the optimisation process. For DNNE, it acts to control the decorrelation term in the optimisation function;
- *Random weights range* (W) This hyperparameter determines the interval in which the initial random weights are uniformly distributed. Although the authors (Alhamdoosh and Wang 2014; Ding et al. 2017) suggest the weights of DNNE to be set in the interval $[-1, 1]$, this hyperparameter plays a vital role in the accuracy of the models. The effect of initial weights in RVFL was investigated by Zhang and Suganthan (2016);
- *Number of attributes* (A) This hyperparameter is exclusive of B-NNRW. It determines the fraction of total inputs that are randomly selected to train each NNRW base model.

Five levels of each hyperparameter are investigated, and each treatment (a combination of hyperparameters) was run ten times. The hyperparameter levels are summarised in Table 3.

It is important to note that for B-NNRW, N is a function of the number of inputs. For a dataset with ten inputs and $A = 0.8$, N is equal to the resulting number of inputs ($10 \times 0.8 = 8$) times N. The analysis of hyperparameter tuning and the resulting hyperparameter set for each algorithm is presented in the next section.

4.3 Hyperparameter analysis

The results of experiments show that not only each hyperparameter has a different level of importance in the hyperparameter optimisation but also the levels of importance change for different problems. Tables 4 and 5 show the first three most important sources of variability, given by the F_0 statistic. The small p values ($\ll 0.01$) indicate that the results are statistically significant. The analysis of the importance of each factor can help to prioritise the optimisation of the hyperparameters that have more influence in the model's performance.

The adjustment of W is the most important hyperparameter to the tuning of DNNE, except in the Mex dataset that showed no statistically significant differences among different treatments. In the case of B-NNRW algorithm, the critical factor depends on the problem, although the number of features was found to be the most important hyperparameter for synthetic datasets. The results also showed that some interactions between factors also resulted in relevant sources of variability. An illustrative example is an interaction between the N and W of B-NNRW in the quality dataset, which is statistically significant and is responsible for 10.8% of the total variability.

Table 4 DNNE significance (p value), F0 statistic, the percentage of explained variance, and cumulative percentage of explained variance

	p value	F_0	%	Cum%		p value	F_0	%	Cum%
A) Mex					B) Fried1				
$\boldsymbol{P} \times \boldsymbol{M}$	0.06	1.6	19.0	19.0	$\boldsymbol{W}$	$\ll 0.01$	28,287.6	93.7	93.7
$\boldsymbol{M} \times \boldsymbol{W}$	0.17	1.3	15.6	34.6	$\boldsymbol{M}$	$\ll 0.01$	957.8	3.2	96.9
$\boldsymbol{N}$	0.36	1.1	12.8	47.4	$\boldsymbol{M} \times \boldsymbol{W}$	$\ll 0.01$	547.8	1.8	98.7
C) Fried3					D) Multi				
$\boldsymbol{W}$	$\ll 0.01$	4719.8	88.1	88.1	$\boldsymbol{W}$	$\ll 0.01$	5361.3	93.1	93.1
$\boldsymbol{M}$	$\ll 0.01$	335.7	6.3	94.4	$\boldsymbol{M}$	$\ll 0.01$	160.9	2.8	95.9
$\boldsymbol{M} \times \boldsymbol{W}$	$\ll 0.01$	149.3	2.8	97.2	$\boldsymbol{M} \times \boldsymbol{W}$	$\ll 0.01$	96.6	1.7	97.6
E) Housing					F) Quality				
$\boldsymbol{W}$	$\ll 0.01$	122.5	69.7	69.7	$\boldsymbol{W}$	$\ll 0.01$	124.3	40.2	40.2
$\boldsymbol{M}$	$\ll 0.01$	19.5	11.1	80.8	$\boldsymbol{N}$	$\ll 0.01$	104.9	33.9	74.0
$\boldsymbol{N}$	$\ll 0.01$	7.0	4.0	84.7	$\boldsymbol{R}$	$\ll 0.01$	35.4	11.4	85.5
G) Maintenance					H) Energy				
$\boldsymbol{W}$	$\ll 0.01$	39,694.5	98.7	98.7	$\boldsymbol{W}$	$\ll 0.01$	286.8	46.5	46.5
$\boldsymbol{M}$	$\ll 0.01$	210.8	0.5	99.2	$\boldsymbol{N}$	$\ll 0.01$	233.9	37.9	84.3
$\boldsymbol{M} \times \boldsymbol{W}$	$\ll 0.01$	126.5	0.3	99.5	$\boldsymbol{M}$	$\ll 0.01$	36.2	5.9	90.2

M number of models, N number of nodes, W random weights range, R regularisation factor

Table 5 B-NNRW significance (p value), F_0 statistic, the percentage of explained variance, and cumulative percentage of explained variance

	p value	F_0	%	Cum%		p value	F_0	%	Cum%
A) Mex					B) Fried1				
A	$\ll 0.01$	53.4	55.2	55.2	***A***	$\ll 0.01$	183,045.5	91.9	91.9
W	$\ll 0.01$	18.5	19.1	74.3	***P***	$\ll 0.01$	7160.7	3.6	95.5
P	$\ll 0.01$	6.9	7.2	81.5	***N***	$\ll 0.01$	5398.2	2.7	98.3
C) Fried3					D) Multi				
A	$\ll 0.01$	33,207.2	93.2	93.2	***A***	$\ll 0.01$	146,166.6	99.8	99.8
P	$\ll 0.01$	1485.6	4.2	97.3	***W***	$\ll 0.01$	146.2	0.1	99.9
N	$\ll 0.01$	477.6	1.3	98.7	***N***	$\ll 0.01$	41.4	0.0	99.9
E) Housing					F) Quality				
W	$\ll 0.01$	334.9	45.6	45.6	***N***	$\ll 0.01$	39.2	28.7	28.7
R	$\ll 0.01$	121.5	16.6	62.2	***A***	$\ll 0.01$	24.3	17.8	46.6
A	$\ll 0.01$	121.4	16.5	78.7	***W***	$\ll 0.01$	15.4	11.3	57.9
G) Maintenance					G) Energy				
R	$\ll 0.01$	235,822.8	64.2	64.2	***N***	$\ll 0.01$	173.9	41.0	41.0
W	$\ll 0.01$	58,369.6	15.9	80.1	***M***	$\ll 0.01$	45.2	10.7	51.7
A	$\ll 0.01$	40,241.0	11.0	91.0	***A***	$\ll 0.01$	41.5	9.8	61.5

M number of models, *N* number of nodes, *R* regularisation factor, *W* random weights range, *A* number of attributes

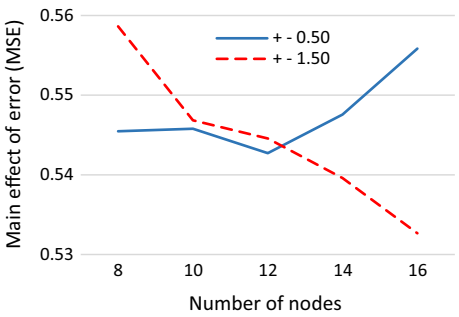

Fig. 4 Effects of the number of nodes (N) given different initialisation weights for the quality dataset (B-NNRW)

Figure 4 shows the different effect of the number of nodes according to the levels of weight initialisation. This type of analysis is not possible when the traditional hyperparameter tuning approaches are applied.

Following the analysis of DOE results, supported by the results of paired t test and Wilcoxon tests, the hyperparameters for each problem were optimised. The best set of hyperparameters for each problem is summarised in Table 6.

The optimised models are evaluated in a simulated data stream, as described in Sects. 4.4 and 4.5.

4.4 Data stream evaluation set-up

The second set of experiments was aimed at evaluating the performance of the algorithms in the data stream environment. The optimised models, obtained from the model fitting process, are run in a simulated data stream that consists of the remaining data from the hyperparameter tuning process. The effectiveness of BP-NNRW (B-NNRW with pruning) and BR-NNRW (B-NNRW with replacement) is evaluated for different ranges of pruning (p) and replacement (r), respectively. The range of p evaluated in this experiment is $p = [0.1, 0.3, 0.5, 0.7, 0.9]$. The r range investigated is defined as $r = [0.1, 0.2, 0.3, 0.4, 0.5]$.

The algorithms are updated at fixed intervals. Once the algorithms start the predictions, the data samples are stored until the updating point is reached. It is assumed that the true label of the stored data is available at this point; therefore, they can be used to compute the evaluation metrics and update the models. In the case of synthetic datasets, where a drift occurs at every 1000 samples, the updating interval is set at every 250 samples. For the benchmark datasets, since no drift is reported, several updating points are evaluated (at every 250, 500, 1000, and 1500 samples). The updating process of DNNE follows the O-DNNE procedure described in Ding et al. (2017). The algorithms are compared in terms of MSE over the simulated data stream. Each experiment is run for 20 times using a random seed, and the results are compared in terms of MSE. Furthermore, all the results are submitted to the t test with 95% confidence to check the statistical significance. The algorithms were coded by the authors using the Matlab® software version 2017b.

Table 6 Best set of hyperparameters for each problem: B-NNRW and DNNE algorithms

	Mex	Fried1	Fried3	Multi	Housing	Quality	Maintenance	Energy
B-NNRW								
P	0.05	0.0001	0.0001	0.0001	0.0010	0.0050	0.0001	0.0050
W	± 1.5	± 0.5	± 0.75	± 0.5	± 1.50	± 1.00	± 1.50	± 1.00
N	10 ×	16 ×	14 ×	16 ×	14 ×	14 ×	16 ×	14 ×
M	40	100	100	80	80	60	40	100
A	0.9	0.9	0.9	0.9	0.7	0.7	1.0	0.8
DNNE								
P	0.5	0.4	0.2	0.1	0.1	0.1	0.3	0.5
W	± 0.05	± 0.065	± 0.065	± 0.05	± 0.005	± 0.005	± 0.065	± 0.035
N	120	100	60	140	60	60	60	120
M	9	12	9	9	12	6	3	12

4.5 Results and discussion

The results of O-DNNE and B-NNRW strategies are compared in this section using the optimised hyperparameter sets for synthetic and benchmark datasets.

(a) Synthetic datasets

First, the response of the algorithms in the synthetic datasets is discussed. The main results are summarised in Table 7.

In general, the replacement strategy (BR-NNRW) resulted in better accuracy than the pruning (BP-NNRW) and weight updating (B-NNRW) strategies. An analysis based on statistical tests, specifically the t test with 95% of confidence, showed that different accuracies are achieved according to the level of replacement. For the Mex dataset, 0.2 and 0.3 were statistically equal and resulted in better accuracy. For Fried1 and Multi, all levels are statistically different, and the best result is achieved with a 0.5 replacement rate. The increase in accuracy for the Fried3 dataset is observed until the rate of 0.3, from where onwards the results are statistically equal.

When compared to O-DNNE, the BR-NNRW achieved statistically better accuracy in Mex, Fried1 and Multi datasets, while O-DNNE performed better in Fried3 dataset. However, before the first point of drift (the first 1000 test samples), the O-DNNE showed better accuracy, suggesting a better learning capability. After the first point of drift, at the test sample 1001, BR-NNRW resulted in better accuracy, showing a better ability to cope with concept drift. The results are summarised in Table 8, where the MSE of O-DNNE and BR-NNRW (with the best replacement rate, according to Table 7) on test data is presented. The test data shown are separated by the samples on each

Table 7 General results of B-NNRW and its variants and O-DNNE for the synthetic datasets

	Mex		Fried1		Fried3		Multi	
	MSE	SD	MSE	SD	MSE	SD	MSE	SD
O-DNNE	2.718E−02	0.000	5.514	0.002	0.872E−02	0.000	0.268	0.000
B-NNRW	2.697E−02	0.000	10.943	0.033	1.803E−02	0.000	0.516	0.001
BP-NNRW								
0.1	2.696E−02	0.000	10.884	0.045	1.773E−02	0.000	0.515	0.001
0.3	2.695E−02	0.000	10.793	0.037	1.661E−02	0.000	0.513	0.001
0.5	2.695E−02	0.000	10.731	0.044	1.599E−02	0.001	0.512	0.001
0.7	2.694E−02	0.000	10.613	0.031	1.498E−02	0.000	0.511	0.001
0.9	2.695E−02	0.000	10.457	0.051	1.402E−02	0.001	0.509	0.001
Average	2.695E−02		10.696		1.587E−02		0.512	
BR-NNRW								
0.1	2.683E−02	0.000	3.374	0.054	1.601E−02	0.000	0.065	0.002
0.2	2.680E−02	0.000	2.547	0.046	1.548E−02	0.000	0.046	0.001
0.3	2.680E−02	0.000	2.294	0.032	1.499E−02	0.000	0.039	0.001
0.4	2.683E−02	0.000	2.176	0.018	1.492E−02	0.000	0.037	0.000
0.5	2.686E−02	0.000	2.106	0.028	1.497E−02	0.000	0.035	0.001
Average	2.682E−02		2.499		1.527E−02		0.044	

Table 8 Accuracy (MSE) of BR-NNRW and O-DNNE by the samples on each domain in the test data

	0–1000	1001–2000	2001–3000	3001–4000	Updating time (s)
Mex					
O-DNNE	3.614E−02	3.026E−02	2.448E−02	1.786E−02	2.393
BR-NNRW	3.577E−02	3.004E−02	2.405E−02	1.735E−02	0.003
Fried1					
O-DNNE	0.107	2.625	5.737	13.586	3.152
BR-NNRW	0.410	1.616	2.245	4.153	0.015
Fried3					
O-DNNE	0.680E−02	0.770E−02	1.073E−02	0.964E−02	0.539
BR-NNRW	1.495E−02	1.552E−02	1.552E−02	1.397E−02	0.009
Multi					
O-DNNE	0.112E−02	8.803E−02	34.332E−02	63.884E−02	3.462
BR-NNRW	0.335E−02	3.437E−02	4.811E−02	5.454E−02	0.012

domain. Table 8 also shows the average updating time, in seconds, where the computational advantage of BR-NNRW to keep updated to the data can be observed.

The smoothed error curves shown in Fig. 5 illustrate how the BR-NNRW and O-DNNE algorithms perform on each dataset.

For synthetic datasets with known drifts, BR-NNRW has been shown to improve the accuracy compared to B-NNRW and BP-NNRW. The best results were achieved with different replacement rates, highlighting the importance of keeping old knowledge (low replacement rate) or discarding old knowledge (high replacement rate) according to the problem at hand. The O-DNNE showed considerable learning capability, achieving better accuracy in the first part of the test data, which is within the same domain of the training data. BR-NNRW, on the other hand, showed an advantage to cope with concept drift not only in terms of accuracy but especially in terms of computational efficiency.

In the next section, the tests of the algorithms on benchmark datasets of four varied application domains with larger sample sizes and attributes are presented.

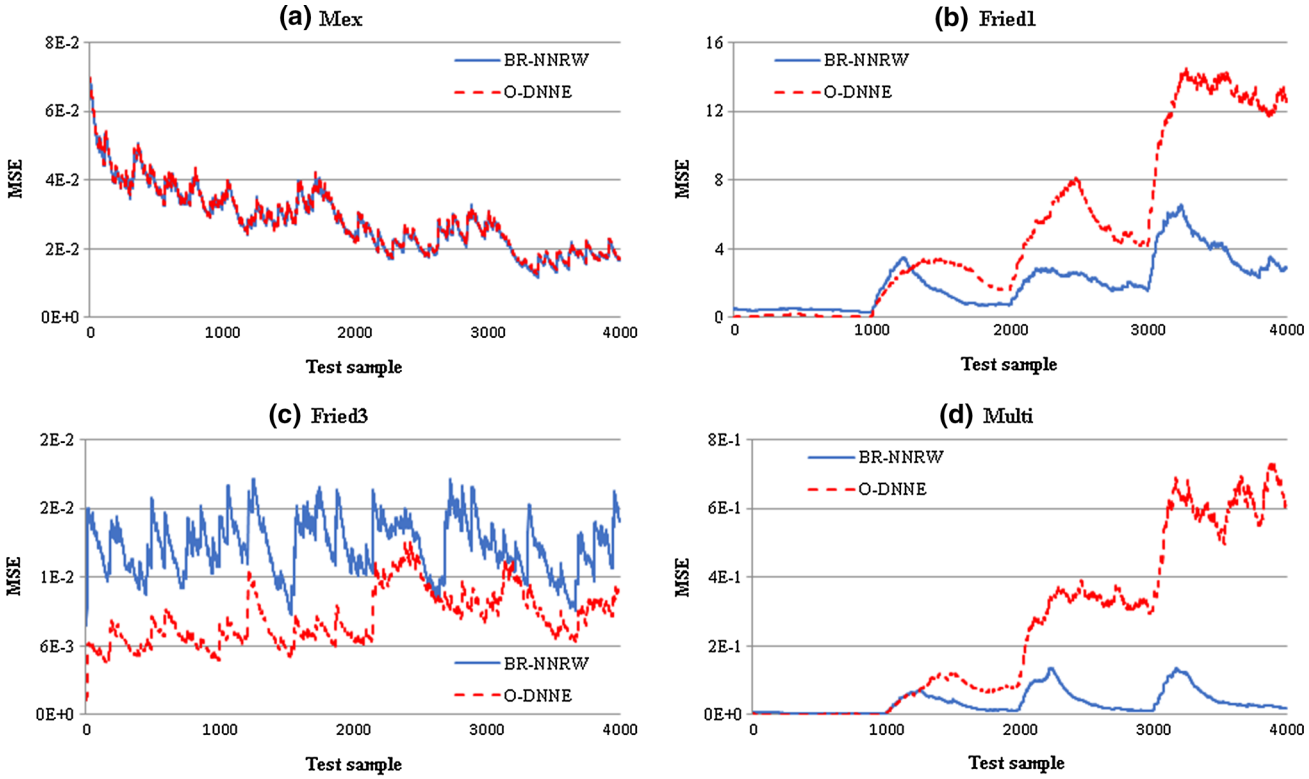

Fig. 5 Effects of the number of nodes (N) given different initialisation weights for the quality dataset (B-NNRW)

(b) Benchmark datasets

Next, the performance of O-DNNE and B-NNRW, along with its variations (BP-NNRW and BR-NNRW), is analysed at different updating intervals on the benchmark datasets. The main results are summarised in Tables 9, 10, 11, and 12.

The first result that should be highlighted is the effect of updating interval. In general, for each problem, all algorithms showed a similar pattern in terms of the best updating interval. This trend is illustrated in Fig. 6, which shows the average accuracy of all algorithms for each dataset. In this figure, the MSE was normalised between 0 and 1 for each algorithm and then averaged.

The results show that the choice of updating frequency is related to the problem that is being studied, which in turn may be related not only to the level of drift present in data but also to data sufficiency. The best updating interval for Maintenance and Quality datasets is likely to be near 1000 observations. Better accuracy is achieved in Energy dataset when it is updated more frequently, i.e. smaller data chunks, while an opposite behaviour is observed in Housing dataset, which can indicate that no dataset shift is occurring in this dataset.

From Tables 9, 10, 11, and 12, it is possible to observe the advantage of the replacement strategy compared to the pruning strategy. In Fig. 7, the results of BP-NNRW and BR-NNRW, averaged over the chunk size, are summarised to illustrate the effects of the different levels of pruning and replacement.

The pruning (BP-NNRW) approach can prevent the vote of low-performing members of the ensemble in the final decision. If properly adjusted, the pruning can increase the accuracy of the model significantly compared to weight update only, as given in Tables 9, 10, 11, and 12. The replacement (BR-NNRW) not only avoids the use of low-performing members due to the exclusion of them but also includes new members trained with the most recent data. Although the flat lines shown in Fig. 7b, c, d for the levels of replacement suggest no difference in response for different levels, especially for Energy and Maintenance datasets, the results of t test with 95% of significance showed that there are statistically significant differences between the replacement levels. The difference between various levels of replacement is shown in Fig. 8, considering the average results of the best updating frequency of each dataset.

For the Energy and Housing datasets, the best accuracy was achieved with a lower rate of replacement (0.1). It should be pointed out that in both cases, the difference between 0.1 and 0.2 rates of replacement was statistically significant, according to the t test. In Quality and Maintenance datasets, B-NNRW required higher rates of replacement to achieve their best performance. The best replacement rate for Quality was 0.5, while for Maintenance, the best accuracy was achieved by replacing 40% of

Table 9 Results of O-DNNE and B-NNRW variations for Housing problem at various model updating intervals

Housing								
Updating interval	250		500		1000		1500	
	MSE	SD	MSE	SD	MSE	SD	MSE	SD
O-DNNE	1.08E+10	1.53E+07	9.85E+09	1.51E+07	9.13E+09	2.06E+07	8.70E+09	2.98E+07
B-NNRW	1.14E+10	1.93E+08	1.04E+10	1.48E+08	9.83E+09	1.96E+08	9.18E+09	1.26E+08
BP-NNRW								
0.1	1.12E+10	1.50E+08	1.02E+10	1.82E+08	9.47E+09	2.05E+08	9.09E+09	1.88E+08
0.3	1.09E+10	1.13E+08	9.92E+09	1.97E+08	9.31E+09	1.56E+08	8.84E+09	2.13E+08
0.5	1.08E+10	1.11E+08	9.85E+09	1.59E+08	9.19E+09	1.46E+08	8.67E+09	1.11E+08
0.7	1.07E+10	1.09E+08	9.84E+09	1.78E+08	9.17E+09	1.48E+08	8.59E+09	1.57E+08
0.9	1.09E+10	8.89E+07	1.01E+10	2.34E+08	9.35E+09	2.33E+08	8.88E+09	1.80E+08
Average	1.09E+10		9.99E+09		9.30E+09		8.81E+09	
BR-NNRW								
0.1	9.83E+09	9.96E+07	1.02E+10	6.66E+08	9.47E+09	1.70E+08	8.41E+09	1.63E+08
0.2	1.07E+10	1.02E+08	1.21E+10	1.21E+09	9.99E+09	1.42E+08	8.70E+09	1.46E+08
0.3	1.11E+10	1.36E+08	1.30E+10	1.12E+09	1.05E+10	1.34E+08	9.20E+09	1.06E+08
0.4	1.14E+10	1.17E+08	1.30E+10	1.02E+09	1.12E+10	1.37E+08	9.70E+09	1.33E+08
0.5	1.17E+10	1.04E+08	1.34E+10	8.99E+08	1.18E+10	1.63E+08	1.02E+10	2.26E+08
Average	1.10E+10		1.09E+10		9.82E+09		9.00E+09	

Table 10 Results of O-DNNE and B-NNRW variations for the Quality problem at various model updating intervals

Quality								
Updating interval	250		500		1000		1500	
	MSE	SD	MSE	SD	MSE	SD	MSE	SD
O-DNNE	0.549	0.006	0.557	0.008	0.546	0.005	0.575	0.009
B-NNRW	0.584	0.004	0.580	0.003	0.565	0.005	0.571	0.003
BP-NNRW								
0.1	0.577	0.005	0.574	0.006	0.559	0.005	0.566	0.004
0.3	0.575	0.006	0.571	0.005	0.557	0.007	0.564	0.005
0.5	0.575	0.007	0.567	0.005	0.553	0.007	0.567	0.006
0.7	0.575	0.006	0.569	0.006	0.557	0.006	0.570	0.007
0.9	0.605	0.008	0.599	0.010	0.585	0.013	0.601	0.009
Average	0.581		0.576		0.562		0.574	
BR-NNRW								
0.1	0.562	0.003	0.546	0.005	0.532	0.004	0.550	0.004
0.2	0.550	0.005	0.536	0.003	0.519	0.003	0.538	0.003
0.3	0.546	0.004	0.533	0.004	0.513	0.003	0.532	0.004
0.4	0.539	0.004	0.535	0.004	0.510	0.003	0.528	0.003
0.5	0.537	0.004	0.535	0.004	0.509	0.004	0.526	0.003
Average	0.547		0.537		0.516		0.535	

Table 11 Results of O-DNNE and B-NNRW variations for the Maintenance problem at various model updating intervals

Maintenance								
Updating interval	250		500		1000		1500	
	MSE	SD	MSE	SD	MSE	SD	MSE	SD
O-DNNE	5.95E−07	3.56E−08	5.27E−07	4.49E−08	5.13E−07	6.62E−08	8.73E−07	1.77E−07
B-NNRW	3.84E−06	3.06E−07	3.69E−06	3.93E−07	3.52E−06	4.25E−07	3.86E−06	4.05E−07
BP-NNRW								
0.1	3.72E−06	3.53E−07	3.39E−06	2.483E−07	3.33E−06	4.49E−07	3.45E−06	4.06E−07
0.3	3.56E−06	3.03E−07	3.06E−06	4.642E−07	3.03E−06	2.83E−07	3.16E−06	2.71E−07
0.5	3.07E−06	3.08E−07	2.92E−06	3.773E−07	2.78E−06	3.80E−07	2.90E−06	2.90E−07
0.7	2.79E−06	2.48E−07	2.70E−06	2.982E−07	2.45E−06	3.03E−07	2.70E−06	3.79E−07
0.9	2.51E−06	2.60E−07	2.36E−06	3.903E−07	2.11E−06	3.35E−07	2.37E−06	4.47E−07
Average	3.13E−06		2.89E−06		2.74E−06		2.92E−06	
BR-NNRW								
0.1	1.04E−06	2.12E−08	9.22E−07	2.40E−08	5.61E−07	5.51E−08	9.11E−07	1.37E−07
0.2	1.08E−06	1.53E−08	9.47E−07	1.99E−08	5.15E−07	3.79E−08	8.35E−07	8.70E−08
0.3	1.10E−06	1.58E−08	9.56E−07	1.88E−08	5.03E−07	4.89E−08	8.45E−07	8.01E−08
0.4	1.12E−06	1.81E−08	9.64E−07	1.53E−08	4.98E−07	3.38E−08	8.19E−07	7.95E−08
0.5	1.13E−06	9.95E−09	9.73E−07	1.42E−08	5.04E−07	2.44E−08	8.58E−07	8.14E−08
Average	1.09E−06		9.53E−07		5.16E−07		8.53E−07	

the models. The adjustment of the rate of replacement is an additional challenge for the use of BR-NNRW; however, the gain in accuracy greatly outweighs the effort.

A comparison between the algorithm with a more accurate updating strategy (BR-NNRW) and a recent online data stream regression algorithm from literature (O-DNNE) is summarised in Table 13. The BR-NNRW is set with the best replacement rate for each dataset. Table 13 shows the average MSE and average updating time (in seconds), i.e. the computational processing time spent to

Table 12 Results of O-DNNE and B-NNRW variations for Energy problem at various model updating intervals

Energy								
Updating interval	250		500		1000		1500	
	MSE	SD	MSE	SD	MSE	SD	MSE	SD
O-DNNE	1.72E+04	8.45E+02	2.48E+04	2.67E+03	2.60E+04	2.34E+03	2.54E+04	2.16E+03
B-NNRW	3.50E+04	2.79E+03	3.31E+04	2.24E+03	3.48E+04	2.04E+03	3.52E+04	3.32E+03
BP-NNRW								
0.1	3.54E+04	2.30E+03	3.34E+04	2.10E+03	3.48E+04	1.81E+03	3.38E+04	2.94E+03
0.3	3.45E+04	2.05E+03	3.30E+04	2.02E+03	3.26E+04	2.46E+03	3.19E+04	2.09E+03
0.5	3.46E+04	1.32E+03	3.30E+04	2.28E+03	3.36E+04	2.07E+03	3.13E+04	2.78E+03
0.7	3.81E+04	2.15E+03	3.45E+04	2.01E+03	3.56E+04	2.34E+03	3.35E+04	2.59E+03
0.9	5.32E+04	3.55E+03	4.93E+04	2.78E+03	4.71E+04	2.84E+03	4.38E+04	4.10E+03
Average	3.92E+04		3.66E+04		3.67E+04		3.49E+04	
BR-NNRW								
0.1	1.39E+04	3.51E+02	1.63E+04	4.27E+02	1.74E+04	4.44E+02	1.74E+04	7.55E+02
0.2	1.45E+04	3.62E+02	1.63E+04	5.93E+02	1.63E+04	6.21E+02	1.65E+04	5.55E+02
0.3	1.48E+04	3.85E+02	1.71E+04	7.09E+02	1.62E+04	4.76E+02	1.61E+04	3.72E+02
0.4	1.51E+04	3.95E+02	1.82E+04	5.77E+02	1.63E+04	3.86E+02	1.62E+04	5.44E+02
0.5	1.54E+04	3.71E+02	1.87E+04	6.04E+02	1.62E+04	4.11E+02	1.66E+04	4.91E+02
Average	1.47E+04		1.73E+04		1.65E+04		1.66E+04	

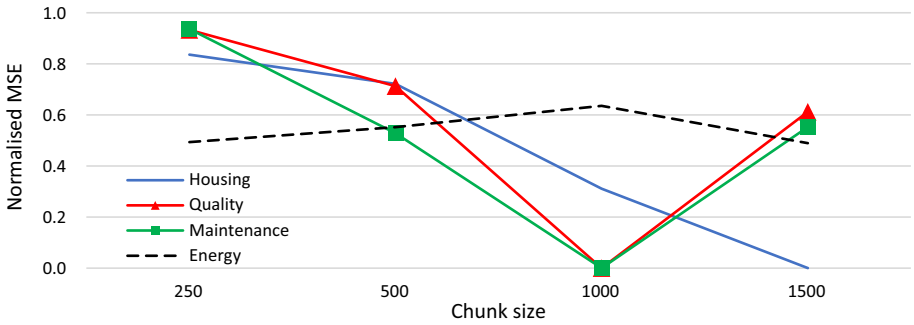

Fig. 6 Datasets average normalised MSE for each chunk size

execute the updating process of each algorithm, considering the best updating interval for each dataset (Fig. 6).

Both BR-NNRW and O-DNNE algorithms showed statistically similar results, according to the t tests with 95% significance, in Maintenance dataset. Furthermore, in this dataset, the O-DNNE showed an advantage in processing time due to the reduced number of models resulted from the hyperparameter tuning; however, as the number of models and nodes increases, the exponential increase in the $\boldsymbol{H}_{\text{corr}}$ matrix (Sect. 3.4) makes the model updating computationally expensive. In the other benchmark datasets, the results showed the advantage of BR-NNRW compared to O-DNNE, in terms of both accuracy and updating time. The replacement mechanism was able to effectively update the ensemble and keep/improve the accuracy through the simulated stream of data. Although no concept drift was reported in the benchmark datasets studied in this paper, the advantage of the use of updating methods compared to static algorithms is clear. This is shown in Fig. 9, where a comparison between the smoothed error of B-NNRW with no updating (static) and BR-NNRW (dynamic) is presented.

From Fig. 9, it is possible to observe that, except in Housing dataset, the use of updating strategies was able to improve the results compared with their static versions, i.e. no update applied. In the case of Housing dataset, it can be concluded that the data used for training may not have been sufficient to train the models, or no concept drift was observed during the evaluation period. The behaviour of the error in the Quality dataset may indicate a recurring drift, due to the decrease of accuracy achieved by the static model through the stream of data. The Maintenance dataset

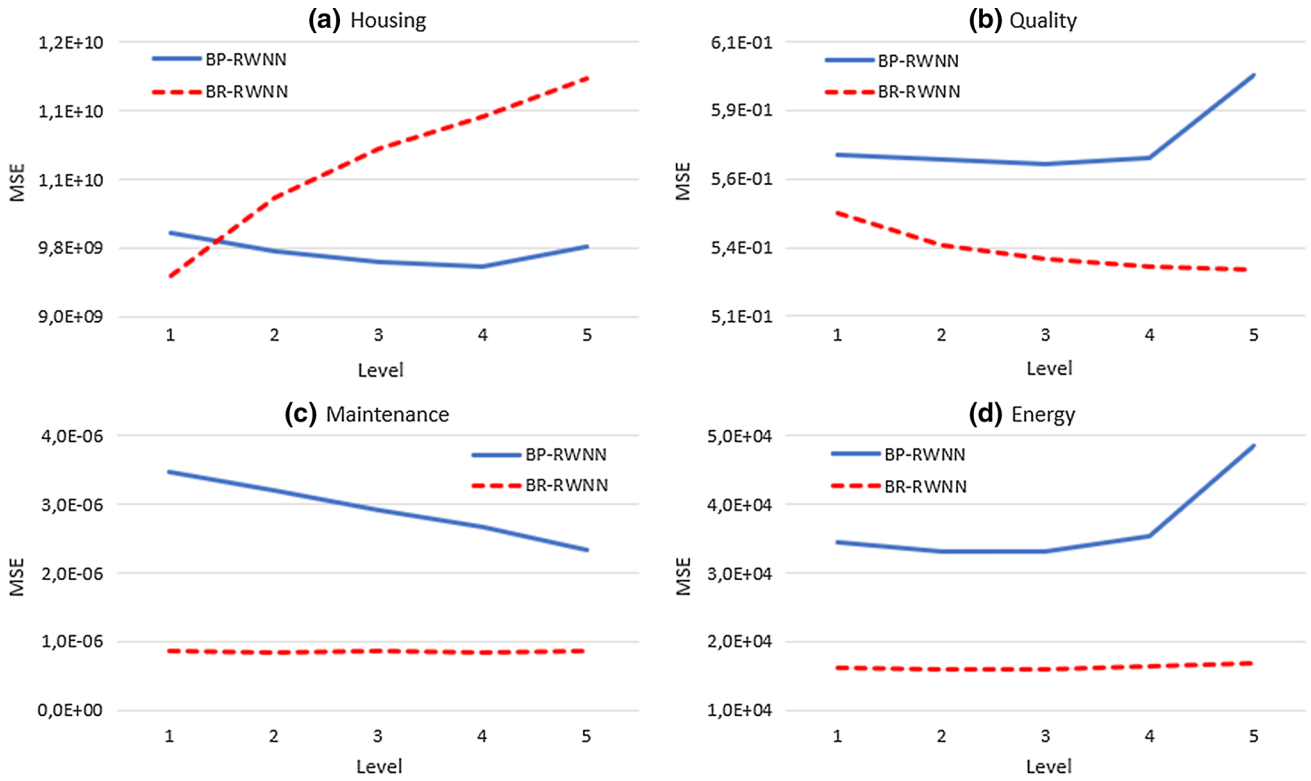

Fig. 7 Average results of pruning (BP-NNRW) and replacement (BR-NNRW) strategies

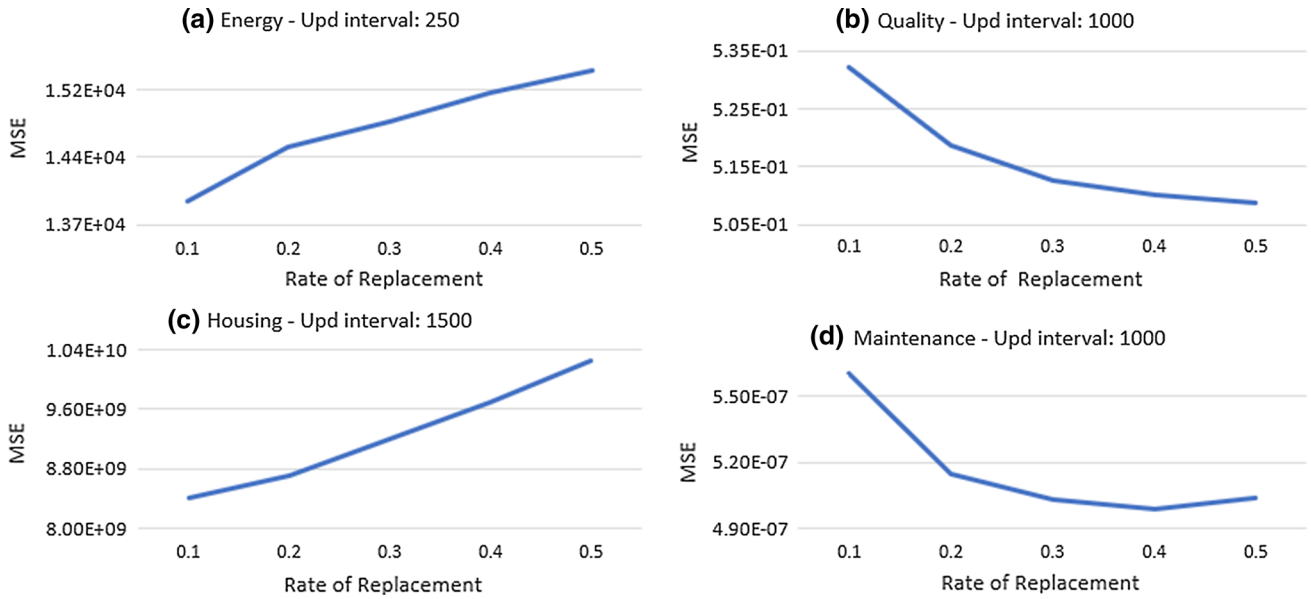

Fig. 8 Average MSE for B-NNRW for different rates of replacement levels

showed stronger evidence of concept drift not only due to the difference in accuracy between static and dynamic models but also due to the high level of replacement required. Both static and dynamic models start at a similar level of accuracy. However, as the time evolves, the static models lose accuracy, while the updating process keeps the error of dynamic models at low levels. The Energy dataset benefited from the updating strategies, especially in the beginning of the evaluation period, where the MSE shows a decrease before it stabilises, which could indicate that the

Table 13 O-DNNE versus B-NNRW

Dataset	Housing		Energy		Maintenance		Quality	
Interval/rep. rate	1500/0.1		250/0.1		1000/0.4		1000/0.5	
	Avg	SD	Avg	SD	Avg	SD	Avg	SD
MSE								
BR-NNRW	**8.41**E+**09**	1.63E+08	**1.39**E+**04**	3.51E+02	4.98E−07	3.38E−08	**0.509**	0.004
O-DNNE	8.70E+09	2.98E+07	1.72E+04	8.45E+02	5.13E−07	6.62E−08	0.546	0.005
% decrease in MSE	3.4%		18.9%		2.9%		6.8%	
Updating time (s)								
BR-NNRW	**0.023**	0.002	**0.044**	0.004	0.135	0.006	**0.088**	0.005
O-DNNE	1.599	0.029	5.037	0.083	**0.080**	0.004	0.322	0.012
Times faster	69.7		113.6		0.6		3.7	

The bold figures indicate the best value for each dataset. In case of Maintenance dataset, the MSE for both algorithms are statistically similar (confidence interval of 95%)

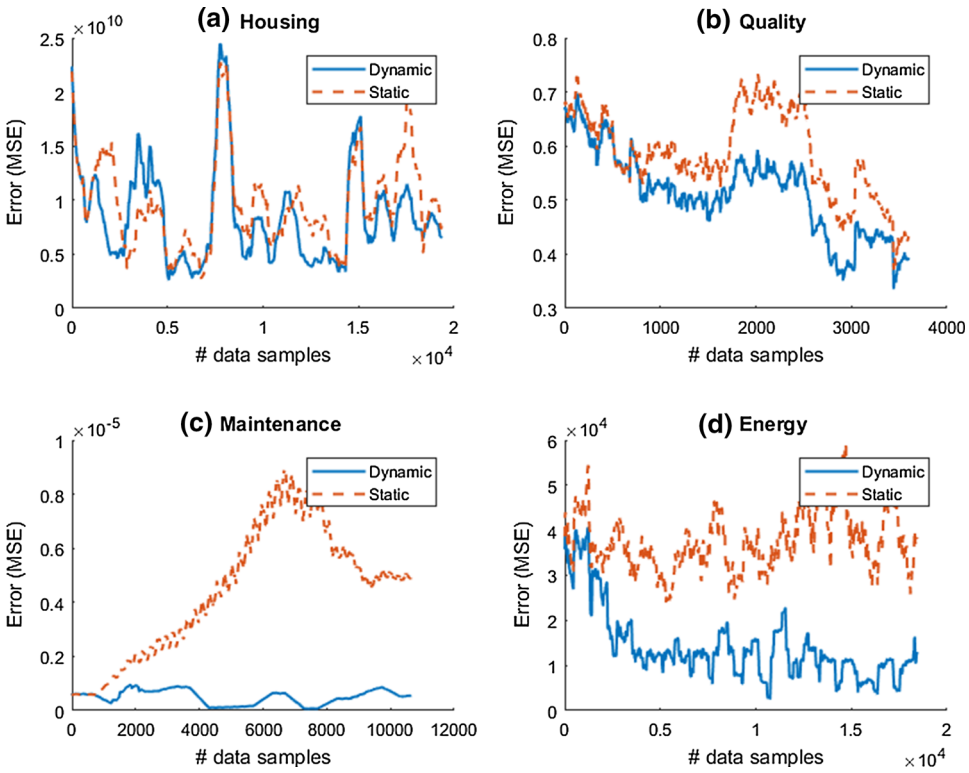

Fig. 9 Comparison between static B-NNRW and dynamic BP-NNRW

data used for training may not be sufficient to represent all the concepts present in data; this idea is reinforced by the low rate of replacement required for this dataset.

5 Conclusions

In this article, the authors have reviewed the available solutions to the online data stream regression problems and identified a need for a faster and more accurate approach. The development of a new B-NNRW method is presented, which is designed to adapt to the evolving nature of the data streams by updating the ensemble in specified intervals to maintain the accuracy of the predictions. This was made possible through combining a bootstrap sampling with random feature selection to train a highly diversified pool of NNRWs. Synthetic datasets for simulating concept drift were used to validate the ability of the proposed algorithm to deal with changing data concepts.

Additionally, datasets from various industries were selected to evaluate the potential enhancement to the prediction model in different dataset types. A series of

experiments were carried out on datasets from Housing, Maintenance, Energy, and Quality Control applications, and the results were compared with the existing O-DNNE method. The results with the proposed updating mechanism showed an average of 8% improvement in the accuracy of predictions across all four types of datasets, with up to 47 times shorter computational time. Furthermore, the use of DOE proved a promising technique to optimise the hyperparameters systematically and can be applied to any ML algorithm.

As part of future research, the next step will be the development of strategies to automatically define the rate of replacement in the proposed approach. The results of the experiments indicated that such automated updating mechanism should also be linked to the types of the datasets. Such a development could potentially improve the suitability of the proposed method for industrial applications.

Acknowledgements This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001 and Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, UK.

Funding Ricardo de Almeida Ph.D. study has been financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Finance Code 001, and the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, UK.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

Alhamdoosh M, Wang D (2014) Fast decorrelated neural network ensembles with random weights. Inf Sci 264:104–117

Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R (2009) New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 139–148

Breiman L (1996) Bagging predictors. Mach Learn 24:123–140

Breiman L (2001) Random forests. Mach Learn 45:5–32

Brown G, Wyatt J, Harris H, Yao X (2005) Diversity creation methods: a survey and categorization. Inf Fusion 6:5–20

Bruce R (1996) Ensemble learning using decorrelated neural networks. Connect Sci 8:373–384

Cao W, Wang X, Ming Z, Gao J (2018) A review on neural networks with random weights. Neurocomputing 275:278–287

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

Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw 22(10):1517–1531

Fan W (2004) Systematic data selection to mine concept-drifting data streams. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, pp 128–137

Farid DM, Zhang L, Hossain A, Rahman CM, Strachan R, Sexton G, Dahal K (2013) An adaptive ensemble classifier for mining concept drifting data streams. Expert Syst Appl 40:5895–5906

Francescomarino CD, Dumas M, Federici M, Ghidini C, Maggi FM, Rizzi W, Simonetto L (2018) Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Inf Syst 74(1):67–83

Gállego PP, Quevedo JR, Coz JJ (2017) Using ensembles for problems with characterizable changes in data distribution: a case study on quantification. Inf Fusion 34:87–100

Gao J, Ding B, Han J, Fan W, Yu PS (2008) Classifying data streams with skewed class distributions and concept drifts. IEEE Internet Comput 12(6):37–49

Gomes HM, Bardal JP, Enembreck F, Bifet A (2017) Survey on ensemble learning for data stream classification. ACM Comput Surv 50(2):23:1–23:36

Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67

Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the 2004 IEEE international joint conference on neural networks, vol 2, pp 985–990

Hutter F, Lücke J, Schmidt-Thieme L (2015) Beyond manual tuning of hyperparameters. KI – Künstliche Intelligenz 29(4):329–337

Ikonomovska E, Gama J, Dzeroski S (2010) Learning model trees from evolving data streams. Data Min Knowl Discov 23:1–41

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

Iwashita AS, Albuquerque VHC, Papa JP (2019) Learning concept drift with ensembles of optimum-path forest-based classifiers. Future Gener Comput Syst 95:198–211

Kadlec P, Gabrys B (2011) Local learning-based adaptive soft sensor for catalyst activation prediction. Am Inst Chem Eng 57(5):1288–1301

Kolter JZ, Maloof MA (2005) Using additive expert ensembles to cope with concept drift. In: Proceedings of the 22th ACM international conference on machine learning, pp 449–456

Krawczyk B, Minku LL, Gama J, Stefanoswski J, Wozniak M (2017) Ensemble learning for data stream analysis: a survey. Inf Fusion 37:132–156

Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17(6):1411–1423

Liu Y, Yao X (1999) Ensemble learning via negative correlation. Neural Netw 12:1399–1404

Masoudnia S, Ebrahimpour R (2014) Mixture of experts: a literature survey. Artif Intell Rev 42:275–293

Masud MM, Gao J, Khan L, Han J (2008) A practical approach to classify evolving data streams: training with limited amount of labeled data. In: IEEE international conference on data mining, pp 929–934

Masud MM, Gao J, Khan L, Han J, Thuraisingham B (2011) Classification and novel class detection in concept-drifting data streams under time constrains. IEEE Trans Knowl Data Eng 23(6):859–874

Montgomery DC (2012) Design and analysis of experiments, 8th edn. Wiley, Hoboken

Oza NC, Russell S (2001) Online bagging and boosting. In: Proceedings of the eighth international workshop on artificial intelligence and statistics, pp 105–112

Pao Y-H, Takefuji Y (1992) Functional-link net computing: theory, system architecture, and functionalities. IEEE Comput 25(5):76–79

Ren S, Liao B, Zhu W, Li K (2018) Knowledge-maximized ensemble algorithm for different types of concept drift. Inf Sci 430–431:261–281

Rokach L (2010) Ensemble-based classifiers. Artif Intell Rev 33:1–39

Rosen BE (1996) Ensemble learning using decorrelated neural networks. Connect Sci 8(3–4):373–393

Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227

Schmidt WF, Kraaijveld MA, Duin RPW (1992) 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, pp 1–4

Soares SG, Araújo R (2015a) An on-line weighted ensemble of regressor models to handle concept drifts. Eng Appl Artif Intell 37:392–406

Soares SG, Araújo R (2015b) A dynamic and on-line ensemble regression for changing environments. Expert Syst Appl 42:2935–2948

Tsai C-J, Lee C-I, Yang W-P (2009) Mining decision rules on data streams in the presence of concept drifts. Expert Syst Appl 36:1164–1178

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

Tumer K, Ghosh J (1996) Error correlation and error reduction in ensemble classifiers. Connect Sci 8(3–4):385–404

Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 226–235

Wolpert DH (1992) Stacked generalization. Neural Netw 5:241–259

Yin XC, Huang K, Hao HW (2015) DE^2: dynamic ensemble of ensembles for learning nonstationary data. Neurocomputing 165:14–22

Zhang L, Suganthan PN (2016) A survey of randomized algorithms for training neural networks. Inf Sci 364–365:146–155

Zliobaite I, Pechenizkiy M, Gama J (2016) An overview of concept drift applications. In: Japkowicz N, Stefanowski J (eds) Big data analysis: new algorithms for a new society. Springer, Berlin, pp 91–114