MORStreaming: A Multioutput Regression System for Streaming Data

Hang Yu, *Member, IEEE*, Jie Lu, *Fellow, IEEE*, and Guangquan Zhang

***Abstract*—With the continuous generation of huge volumes of streaming data, streaming data regression has become more complicated. A regressor that predicts two or more outputs, i.e., multioutput regression, is commonly used in many applications. However, current multioutput regressors use a batch method to handle data, which presents compatibility issues for streaming data as they need to be analyzed online. To address this issue, we present a multioutput regression system, called MORStreaming, for streaming data. MORStreaming uses an instance-based model to make predictions because this model can quickly adapt to change by only storing new instances or by throwing away old instances. However, learning instances in our regression system are constrained by online demand and need to consider the relationship between outputs. Therefore, MORStreaming consists of two algorithms: 1) an online algorithm based on topology networks which is designed to learn the instances and 2) an online algorithm based on adaptive rules which is designed to learn the correlation between outputs automatically. Experiments involving both artificial and real-world datasets indicate MORStreaming can achieve superior performance compared with other multioutput methods.**

***Index Terms*—Instance-based model, multioutput regression, online learning, streaming data.**

NOMENCLATURE

To make the notations easier to follow, we summarize the notations as follows.

λ	Number of samples during one learning period.
$F_1'(t)$	Winner. The instance F_i is nearest to the input vector.
$F_2'(t)$	Second winner. The instance F_i is second nearest to the input vector.
$age_{\max}$	Lifetime of each edge. Once the F_1' is connected to the F_2' with an edge, the "age" of the edge is "0" at first; subsequently, the age of all edges linked to the F_1' is increased by "1." Once the age of an edge exceeds $age_{\max}$, the edge is deleted.
ω_i	Winning times of F_i. The initial value is 0, then increase 1 once the F_i is selected as the winner.
S	Set of rules R_i.
R_i	ith rule represents the relationship between outputs and is an implication in the form $A \Longrightarrow C$.
A	Conjunction of logical operators based on input attributes.
C	Kind of correlation of output attributes.
L	Logical operators which have distinct forms depending on the input attribute. For example, if inputs are real numbers, literals can have the forms $L = (x_1 \leq v)$ or $L = (x_2 > v)$, meaning the value of the first input attribute x_1 of one instance F_i must be less than or equal to v, and the second input attribute $x_2(t)$ must be greater than v, respectively.
v	Split point.
I_L	Set of instances $\{F_i \in I : x_i \leq v\}$.
I_R	Set of instances $\{F_i \in I : x_i > v\}$.
K	Kernel function.
h	Bandwidth.
D	Underlying data distribution.

I. INTRODUCTION

WITH the arrival of the big data era, many applications are generating huge amounts of streaming data [1]–[3]. However, the evolving characteristics of streaming data raise new challenges for batch-based machine learning methods [4] and result in the proposal of stream mining systems. A stream mining system should meet four requirements: 1) *fast:* data are processed in limited time; 2) *memory-efficient:* previously processed data are discarded; 3) *noniterative:* data are learned one at a time; and 4) *adaptive:* learning model can be adjusted to adapt to concept drift [5], [6].

Although many stream mining systems have been proposed, compared with classification [7] and clustering [8], regression [9]–[12] has attracted less attention as it is a more complex problem. Furthermore, in an increasing number of practical applications, multiple outputs instead of one output are predicted, i.e., multioutput regression [13]–[15]. Multioutput regression is a more complicated regression problem because these outputs may have a structure to represent the relationship between outputs. Commonly, multioutput regression can be divided into global and local methods based on the structure of outputs [16]. Global methods use

Manuscript received February 9, 2021; revised June 3, 2021; accepted August 1, 2021. This work was supported by the Australian Research Council (ARC) under Discovery Grant DP190101733. This article was recommended by Associate Editor H. Zhu. *(Corresponding author: Jie Lu.)*

Hang Yu is with the Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia, and also with the Faculty of Computer Engineering and Science, Shanghai University, Shanghai 200444, China (e-mail: hang.yu@student.uts.edu.au).

Jie Lu and Guangquan Zhang are with the Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: jie.lu@uts.edu.au; guangquan.zhang@uts.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2021.3102978.

Digital Object Identifier 10.1109/TSMC.2021.3102978

the idea of classical learning algorithms for single output, i.e., to predict the multiple outputs as a whole. In contrast, the idea of local methods is simpler, i.e., to decompose the multioutputs into multiple single outputs, then use classical learning algorithms for a single output to predict each output. However, most existing methods are usually computationally intensive and are unsuitable for dealing with streaming data.

This article proposes a system called MORStreaming to address the multioutput regression problems in streaming data. The system uses an instance-based model [17] for regression, which means MORStreaming can easily adapt to a new concept by simply storing new instances or throwing old instances away [18]. However, since the training data are streaming data, MORStreaming needs to consider three challenges: 1) instances can only be obtained in an online manner; 2) the structure of outputs is unknown and can change due to concept drift; and 3) how to use the instances and the structure of outputs to make a prediction. In order to solve these challenges, in the learning stage, MORStreaming learns rules to reflect the different structures of outputs and then classifies all the obtained instances into different rules according to the input of data. In the prediction stage, rules are found which cover the predicted data according to the input of the predicted data, then only the instances which fall under these rules can be used to make a prediction in relation to this predicted data.

The main contributions of this article are as follows.

1) Transferring the structures of outputs into rules. The rules, based on Hoeffding's bound [19], are learned not only online but are also easily adjusted to fit the drift of the outputs' structures. Based on these types of rules, our proposed algorithm does not fall into the local or global categories. Instead, each rule can specialize in the related subsets of the outputs to represent various structures.
2) Developing an online algorithm that inherits the topology network to learn instances. This algorithm shows high robustness to noise as it inherits the advantages of topology learning [20]. In addition, when the algorithm is used to obtain the instances, it does not need to set the number of instances, and obtaining and discarding instances is also in a self-starting scheme.
3) An instance-based regression model is established. In this model, only instances that have the same structure as the outputs can be used to make a prediction. Furthermore, the correlation between the outputs is utilized well by a multioutput nonparametric function. Hence, the accuracy of the prediction improves without significantly increasing learning time.

The remainder of this article is structured as follows. Related works are introduced in Section II. Our proposed multioutput regression system, MORStreaming, is introduced in Section III. The experiments used to analyze and verify our proposed algorithm are shown in Section IV. Section V concludes this article and proposes future works.

II. Related Works

In this section, we introduce some works which are related to our research and classify them into two categories: 1) streaming data regression and 2) multioutput regression.

A. Data Stream Regression

In the past ten years, compared with the classification and clustering of data streams, regression on data streams has attracted less attention, with a few notable exceptions. Of these, the Hoeffding tree-based methods [21] have attracted a lot of attention from researchers, and many modifications of the original method have been proposed. For example, a Hoeffding tree-based method called Fast Incremental Model Trees with Drift Detection [22] is proposed, and it uses Hoeffding's inequality to choose the best splitting attribute. Gomes *et al.* [23] proposed an adaptive random forest by assembling the Hoeffding tree. Ikonomovska *et al.* [24] also modified the Hoeffding tree method to present option trees and ensembles of option trees for regression. Adaptive Model Rules [25] is another relevant representative of streaming data regression and makes predictions by learning rules. Similar to Adaptive Model Rules, some evolving fuzzy systems [26] were also proposed. Evolving fuzzy systems also need to learn rules, but the rules are specified by a conjunction of fuzzy logical operations on the input attributes, and the Takagi–Sugeno–Kang (TSK) rule in consequence. Support vector regression (SVR) is also extended to the online version for streaming data regression. To extend v-SVR, Gu *et al.* [27] proposed an incremental v-SVR algorithm, and Yu *et al.* [28], [29] proposed some online SVR algorithms for nonstationary data streams. In contrast to model-based algorithms, IBLStreams [30] is an instance-based learning system, where the prediction result can be estimated by the weighted mean of the k-nearest neighbor instances. However, these methods only consider whether there is a single output that needs to be predicted.

B. Multioutput Regression

Multioutput regression is an important branch of structured-output learning. Assume $\{(X(1), Y(1)), \ldots, (X(N), Y(N))\}$ is the training data, where $X(i)$ is a d-dimensional vector $[x_1(i) \ldots x_d(i)]^T$ describing the inputs of the ith data, and Y_i is an m-dimensional vector $\left[y_1(i) \ \ldots \ y_m(i)\right]^T$ of the outputs. The task of multioutput regression is to learn a model $f(X) \rightarrow Y$ that maps the inputs $X(i)$ to the outputs $Y(i)$.

Next, we briefly introduce several classical methods for multioutput regression, categorized as local methods and global methods.

1) Local methods are based on the idea of converting the multioutput regression problem into m single-output problems, then learning a model for each output, and finally concatenating all m predictions. For example, Hoerl and Kennard [31] separated the multivariate output into multiple univariate outputs and then utilized the ridge regression to make a prediction. Next, inspired by stacked generalization [32], the multioutput regressor stacking method is proposed [33]. The regressor

chain method [13] is another important local method based on the idea of chaining single-output models. Zhang *et al.* [34] presented a multioutput SVR. This type of regression develops a vector virtualization method to build a multioutput model that takes into account the relations between all the outputs.

2) Global methods are mainly based on simultaneously predicting all the outputs using a single model that can capture all the internal relations between them. Izenman [35] proposed a reduced-rank regression, which adds a rank of constraint on the estimated outputs. Tuia *et al.* [36] integrated a constraint-based system into the process of learning multiobjective regression trees. Struyf and Dzeroski [37] also developed a multioutput SVR method by extending the single-output SVR to multioutputs while maintaining the advantages of a compact and sparse solution using a cost function. Aho *et al.* [38] presented a novel method for learning ensemble rules for multioutput regression and treating multiple numeric outputs as a whole.

In summary, these multioutput regression methods are usually computationally intensive, i.e., instances need to be processed multiple times, and hence are not suited to dealing with streaming data. In addition, these methods do not have a mechanism to handle the uncertain and nonstationary characteristics of streaming data.

III. Multiple-Output Regression System

In this section, we introduce the details of our proposed multiple-output regression system, MORStreaming.

A. Learning Instances

The aim of the learning instances algorithm is to obtain a set of instances which enable us to learn a predictor with the same (or higher) accuracy than the original set. However, in the previous algorithms [30], [39], the instances were chosen depending on the initial state of the clustering, i.e., the number and position of the initial clusters. However, it is quite challenging to suitably set the initial state for streaming data because the prior knowledge of streaming data is unknown. In recent years, there have been advances in the area of clustering algorithms [40], a follow-on from online topology learning algorithms. However, this type of algorithm, such as a self-organized incremental neural network (SOINN) [41] is proposed for unsupervised problems, as the learning process is very complex and at least four parameters require manual settings. However, MORStreaming is proposed for the supervised problem, so, in our MORStreaming algorithm, only two parameters λ and $age_{\max}$ need to be set manually. However, the accuracy of the learning result is the same as that obtained by SOINN.

Next, we introduce how MORStreaming learns instances. Assume T number of input data $Z(t) = \{X(t), Y(t)\}$ arrives at timestamp t, where $X(t) \in \mathbb{R}^d$, and $Y(t) \in \mathbb{R}^m$. Like SOINN, the learning objective of our method is also formally defined as a minimization of the reconstruction error

$$\sum_{t\in T}\sum_{i\in N}\mu_i(t)d_i^2(t) \tag{1}$$

where $d_i(t)$ defines the Euclidian distance (L_2-norm) between the input data $Z(t)$ and the instances $F_i(t) \in \mathbb{R}^{d+m}$ as follows:

$$d_i^2(t) = ||Z(t) - F_i(t)||^2, 1 \le t \le T, 1 \le i \le N \tag{2}$$

and N is the set of instances, but

$$0 \le \mu_i(t) \le 1, \quad \sum_{i\in N}\mu_i(t) = 1). \tag{3}$$

The difference in the $\mu_i(t)$ value is the most significant variation between our method and SOINN. In SOINN, when an instance F_i is the nearest instance [with minimum $d_i^2(t)$] to input data $Z(t)$, $\mu_i(t) = 1$, and the remaining $\mu_i(t) = 0$. However, in our method, the $\mu_i(t)$ of the instance F_i is calculated as

$$\mu_i(t) = d_i^2(t)\left(\sum_{j=1}^{N} d_j^2(t)\right)^{-1} \tag{4}$$

and can represent the degree to which input data $Z(t)$ is close to an instance $F_i(t)$. Hence, the instance $F_i(t)$ with a maximum $\mu_i(t)$ is the nearest instance to the input data $Z(t)$.

Like SOINN, the first step in learning instances is to identify whether to insert a new input data $Z(t)$ as a new instance $F_i(t)$, but our method has its own standards of judgment. Assume the topology network already has N instances $\{F_1(t), \ldots, F_N(t)\}$ at timestamp t. When a new input data $Z(t)$ arrives, it starts finding the winner $F_1'(t)$ and the second winner (s-winner) $F_2'(t)$ based on μ_i. If $\mu_1(t)$ of the winner $F_1'(t)$ or $\mu_2(t)$ of the s-winner $F_2'(t)$ is more than the threshold T_i, furthermore no edge connects $F_1'(t)$ and $F_2'(t)$, they will be connected with an edge, and setting the age of the edge as 0. Subsequently, the age of all edges linked to $F_1'(t)$ is also increased by 1. However, an edge will be deleted if its age exceeds $\text{age}_{\max}$. Otherwise, the new input data $Z(t)$ is inserted into the topology network as a new instance $F_{N+1}(t)$.

As for the value of the threshold T_i, if an instance $F_i(t)$ is connected to neurons, threshold T_i is calculated using the maximum value of $\mu_i(t)$ of connected instances

$$T_i = \max_{j\in \text{con}_i} u_j(t) \tag{5}$$

where con_i is a set of all instances that connect to this instance $F_i(t)$. However, if the instance $F_i(t)$ does not have instances connected to it, threshold T_i is defined as the minimum value of $\mu_i(t)$ of other instances

$$T_i = \min_{j\in AN\setminus\{i\}} \mu_j(t) \tag{6}$$

where AN is a set of all existing instances.

The next step is to update $F_1'(t)$ whereby all instances of $F_i(t)$ that connect to $F_1'(t)$ are updated. When we obtain an instance $F_i(t)$, we assume the relationship between an old instance $F_i(t)$ and new instance $F_i(t+1)$ is as follows:

$$F_i(t+1) = F_i(t) + \Delta\ F_i(t+1) \tag{7}$$

where the $\Delta F_i(t+1)$ is

$$\Delta\ F_i(t+1) = \frac{\mu_i^2(t+1)(Z(t+1) - F_i(t))}{U_i(t+1)} \tag{8}$$

and the denominator $U_i(t+1) \in \mathbb{R}$ is defined as

$$U_i(t+1) = U_i(t) + \mu_i^2(t+1) \tag{9}$$

where $U_i(t)$ is defined as follows:

$$U_i(t) = \sum_{t=1}^{T} \mu_i^2(t). \tag{10}$$

Hence, the winner $F_1'(t)$ is updated as follows:

$$F_1'(t+1) = F_1'(t) + \gamma_i(t+1)\Delta\ F_1'(t+1) \tag{11}$$

where the parameter $\gamma_i(t)$, $(0 \le \gamma_i \le 1)$ is the reciprocal of the winning times ω_i of winner $F_1'(t+1)$. As for other instances $F_i(t+1)$ that connect to the winner $F_1'(t+1)$, these are updated as follows:

$$F_i(t+1) = F_i(t) + \eta \cdot \gamma_i(t+1) \cdot \Delta\ F_i(t+1) \tag{12}$$

where η is a learning rate and is recommended as 0.01 [41]. Based on (11) and (12), all instances F_i can be updated in an online manner. In addition, the learning procedure is robust to noisy data since instances need to sustain a topology network [41].

B. Learning Structured Outputs

The most significant difference between single-output and multioutput regression is that multioutput regression needs to consider the structure or correlation of the output attributes. Hence, in this section, we introduce how to learn the structure between output attributes. Typically, multioutput predictors can be divided into local and global strategy approaches. However, there are three challenges when using these approaches to handle streaming data: 1) unlike batch-based methods, once one sample of streaming data is handled, it will be discarded; 2) the structure of output attributes is neither local nor global. For example, assuming there are three output attributes $\{y_1,\ y_2,\ y_3\}$ of the training data, where only attribute y_1 has a relationship with attribute y_2, the structure of these three output attributes should be presented as $\{(y_1, y_2), (y_3)\}$; and 3) output attributes have a structure under certain conditions. For example, only the third input attribute x_3 is less than 10, the first output attribute y_1 has a relationship with the third output attribute y_3.

To solve the aforementioned challenges, the modularity property of a rule set is used to distinguish the local and global methods [19]. In MORStreaming, rule R_i is an implication in the form $A \Longrightarrow C$ where antecedent A is a conjunction of literals L. When rule R_i covers one instance F_i, consequent C returns a kind of correlation of output attributes, such as $\{(y_1, y_2), (y_3)\}$. Rule R_i is said to cover one instance F_i if, and only if, input $X(t)$ of F_i satisfies all the literals in A. In addition, a default rule D exists in MORStreaming with a set of n learned rules $R = \{R_1, \ldots, R_n\}$ to build a rule set. Default rules assume there are no relationships between any two output attributes.

Furthermore, to learn rule set R from streaming data, instances only need to be learned once, and the rule set can be continuously grown. In MORStreaming, growing the rule set means learning a new rule $R_l\ A \Longrightarrow C$ or a rule R_l is expanded by adding a new literal to antecedent A_l. Of course, the literal can only be added if there is strong evidence that the new literal is the best one among the set of candidates. Hence, in order to evaluate the merit of splitting an input attribute x_j given the split-point v, the mean-variance ratio (MVR) function is used and defined as

$$MVR\big(x_j, v\big) = \frac{1}{m}\sum_{o=1}^{m} VR_o\big(x_j, v\big) \tag{13}$$

where $VR_o(X_j, v)$ assesses the merit of splitting the input attribute x_j given split-point v concerning output attribute y_o, and m is the number of output attributes. The variance ratio (VR) is defined as

$$VR_o\big(x_j, v\big) = 1 - \frac{1}{2}\frac{\mathrm{var}_o(I_L)}{\mathrm{var}_o(I)} - \frac{1}{2}\frac{\mathrm{var}_o(I_R)}{\mathrm{var}_o(I)} \tag{14}$$

$$\mathrm{var}_o(I) = \frac{1}{m-1}\sum_{i=1}^{m-1}\big(\rho_{i,o}(I) - \bar{\rho}_o(I)\big)^2 \tag{15}$$

$$\rho_{i,o}(I) = \left|\frac{\mathrm{cov}(y_i(I), y_o(I))}{\sigma_i\sigma_o}\right| \tag{16}$$

where $y_i(I)$ is the set of ith output attribute of instances that are covered by rule R_i since its last expansion. $\overline{\rho}_o$ is the mean of value ρ_o of output attribute $\rho_{i,o}$, and σ_i is the standard deviation of $y_i(I)$.

However, considering the literal is chosen from streaming data, we also utilize the Hoeffding bound to guarantee the new literal is the best one in the candidate set. According to [21], the Hoeffding bound is defined as

$$\epsilon = \sqrt{\frac{P^2 ln(1/\delta)}{2n}} \tag{17}$$

where P represents a range and is set to 1, δ is the probability of the true mean of a random variable r will not differ from the sample mean less than ϵ, and n is the number of instances. This equality can be utilized to determine the minimum number n of instances required to expand a rule R_i, i.e., if $r > \epsilon$, where $r = \mathrm{MVR}_{\mathrm{Best}} - \mathrm{MVR}_{\mathrm{2ndBest}}$ means the score difference between the two best potential literals, we can state that the current literal is the best literal with probability $1 - \delta$, and there is no need to collect more instances. Hence, the rule can then be expanded by this literal.

Although the Hoeffding bound guarantees MVR will increase in the process of the learning rule set, c does not guarantee that VR is increased for all output attributes, and may only be a subset of output attributes. For this reason, after selecting the best literal, only the output attributes whose VR is increased are considered to be correlated. Let C_l be the current correlation of the multioutputs according to rule R_l, and E_{best} is the set of instances in the corresponding best branch. The new correlation C_l' is defined as

$$C_l' = \left\{Y_o : Y_o \in C_l \wedge \frac{\mathrm{var}_o(E_{\mathrm{best}})}{\mathrm{var}_o(E)} > 1\right\} \tag{18}$$

which means the set of output attributes effectively increases in var after the expansion of R_l. Furthermore, to continue to save information for the other output attributes, we add a complementary rule R_o, containing the set of output attributes that were pruned after the split into a rule set. The antecedent of R_o is equal to the antecedent of R_l before the split and R_o will only be learned for the output attributes $Y_o \in C'_o$, where C'_o is defined as

$$C'_o = C_l / C'_l. \tag{19}$$

Lines 6–13 of Algorithm 1 show how to learn rules in MORStreaming. From Algorithm 1, we can see rule set R is empty in the initial step and initializing statistics L_D for default rule R_D. Then, when a new instance F_i is available, the statistics necessary to expand rule R_l are updated. If $r > \epsilon$, this rule is expanded. If the expansion of this rule causes a specialization of rule R_l on a subset of the current output attributes, a complementary rule R_o is also created and added to the rule set. If no rule covers $X(t)$, the statistics of the default rule L_D are updated and a new default rule R'_d is created if the default rule R_d is expanded.

C. Instanced-Based Prediction Model

When data $\{X \in \mathbb{R}^d\}$ need to be predicted, the rules which cover them need to be found first, then it is necessary to obtain instances that follow these rules to make a prediction. Assume rule R_l is used to obtain instances $F_l = \{X \in \mathbb{R}^d, Y \in \mathbb{R}^m\}$. Based on these instances, we propose a nonparametric regression function to make a prediction for these data. When we predict one output $\hat{y}$, if no output attribute has a relationship with it, $\hat{y}$ will be predicted using the following function:

$$\widehat{y}\left(\hat{X}\right) = \frac{\sum_{i=1}^{n} K_h\left(X_i, \hat{X}\right) y_i}{\sum_{i=1}^{n} K_h\left(X_i, \hat{X}\right)} \tag{20}$$

where K is kernel function and h is the bandwidth. However, for the sake of notational simplicity, if one output attribute y of these data has a relationship with y_j according to rule R_l, our proposed multioutput nonparametric regression function uses the co-outputs in the expression for the estimator described as follows:

$$\widehat{y}\left(\hat{X}\right) = \frac{\sum_{i=1}^{n}\left[K_{H_1}^{y_i}\left(X_i, \hat{X}\right) y_i + K_{H_2}^{y_i y_i}\left(X_i, \hat{X}\right) y_i\right]}{\sum_{i=1}^{n}\left[K_{H_1}^{y_i}\left(X_i, \hat{X}\right) + K_{H_2}^{y_i y_i}\left(X_i, \hat{X}\right)\right]} \tag{21}$$

where K^{y_i} and $K^{y_i y_j}$ are the kernels that reflect the influence of the predicted output of y_i and y_j of instance F_i, respectively.

When one output attribute $\hat{y}_i$ is covered by multiple rules, the final values of $\hat{y}_i$ are calculated by

$$\widehat{y}^*\left(\hat{X}\right) = \frac{\sum_{N}^{k=1} \theta_{R_k} \widehat{y}_i\left(\widehat{X}\right)}{\sum_{k=1}^{N} \theta_{R_k}} \tag{22}$$

where N is the number of rules which cover input data $\hat{X}$, and θ_{R_k} is the number of instances under rule R_k.

Algorithm 1 MORStreaming

Input: Sequence $\{Z\}$, age_{max}, λ, η
Output: the topology network of instances F_i
1: Initialize set AN with the first 2 input data drawn from $\{Z\}$
2: while $\{Z\}$ is not empty **do**
3: Find winner F'_1 and second winner F'_2 from set AN, as
$\boldsymbol{F'_1 = F_i argmax_{i \in AN} \mu_i}$
$\boldsymbol{F'_1 = F_i argmax_{i \in AN / \{F'_1\}} \mu_i}$
4: **if** $\mu_1 < T_1$ **and** $\mu_2 < T_2$ **then**
5: Add an instance F_i with weight $Z(t)$ to AN.
6: for each $R_l \in S(X(t))$ **do**
7: $R_c \leftarrow R_l$ and update $\mathbf{L_l}$
8: expanded $\leftarrow$ expend(R_l)
9: **if** expanded = TRUE **then**
10: Compute C'_o as in eq. (19).
11: $C_o \leftarrow C'_o$ **and** $R \leftarrow R \cup \{R_c\}$
12: **end if**
13: **end for**
14: **if** $S(X(t)) = \emptyset$ **then // S(X(t)) is defined as the set of rules that cover x.**
15: update $\mathbf{L_D}$ **and** expanded $\leftarrow$ expend(D)
16: **if** expanded = TRUE **then**
17: $R \leftarrow R \cup D$
18: $D \leftarrow 0$
19: **end if**
20: end if
21: else
22: Increase the winning times ω_i of winners by 1.
23: Update instances as
$\mathbf{F'_1(t+1) = F'_1(t) + \gamma_1(t+1) \Delta F'_1(t+1)}$
$\mathbf{F'_i(t+1) = F'_i(t) + \eta \cdot \gamma_i(t+1) \Delta F'_i(t+1)}$ for all instances F'_i that connect to F'_1
24: Increase the age of all edges linked with F'_1 by 1.
25: **for all** edge $age_{i,j} > age_{max}$ do
26: Remove the edge connecting F_i and F_j **.**
27: **end for**
28: **end if**
29: **if** number of input instances divides λ **then**
30: Remove instances F_i that only one instance connects to it.
31: Delete redundant rules
32: **end if**
33: end whilecc

D. Handling Concept Drift and Noise

A mining algorithm that is proposed for streaming data needs the ability to handle concept drift [42]. In our research problem, concept drift can be seen from a local and global viewpoint. From a local viewpoint (i.e., at each rule), one rule can change to a redundant rule since the structure of the outputs under these rules has changed, so this rule is deleted to save memory and improve prediction accuracy. Hence, in Algorithm 1, after one iteration has finished, the operation of deleting redundant rules will be implemented (see the 15th step of Algorithm 1). Each rule is associated with a drift detection test. Since each rule covers a region (a hyperrectangle) of the instance space, when concept drift occurs in some part of the instance space, the error over that part of the instance space will increase. Hence, we monitor the Page–Hinckley (PH) test [43] of error at each rule. If it is over the threshold (set to 50), this rule is identified as a redundant rule and is deleted. From a global viewpoint, the underlying distribution changes, that is, $D(t+1) \neq D(t)$ after timestamp t.

This drift means the current topology network cannot represent the real underlying distribution, and new structures of the outputs also occur. However, since MORStreaming continuously updates instances according to the underlying distribution D of data, the new instances which fit the new distribution will be continuously inserted into the topology network and old instances will also be updated to fit the new distribution according to the 8th step of Algorithm 1, with an increasing amount of newly arriving data. When new instances are learned, new rules that represent the new structures of the output attributes are also learned. In addition, handling noise [44] is also very important because the noise will impact the prediction accuracy. In MORStreaming, handling noise is from a global viewpoint. After each iteration, the advantages of topology learning will result in fewer instances connected to this noise. Hence, a sample rule can effectively delete noise (refer to details from line 14 of Algorithm 1).

E. Complexity Analysis

As for time complexity, according to Algorithm 1, the important component in MORStreaming is to learn instances. The most time-consuming operation of learning instances is to find thresholds T_1 and T_2, which is $O(N^2)$, and N means the number of instances. Once each iteration of learning instances is finished, for example, 200 data are processed, new rules based on the obtained instances are devised to explore the structure between the output attributes or grow already learned rules such as R_i using learned instances F_i whose inputs are covered by rule R_i. For example, assume rule R_1 is learned in the first iteration to represent a relationship between each output attribute. In the second iteration, instances F_i whose input $X(t)$ is covered by rule R_1 will be used to grow rule R_1. Otherwise, the remaining instances F_i will be used to learn new rules; hence, the complexity of learning rules depends on the number of instances, so it is $O(N)$. Next, the redundant rules need to be deleted, and the complexity of this operation also depends on the number of instances. Hence, in summary, the complexity of MORStreaming is $O(N^2)$. In the prediction stage, once a sample is inputted, the structure between its output attributes will be obtained according to learned rules R_i, then this structure will be utilized to improve the accuracy of prediction. As for space complexity, the most time-consuming operation is saving instances, which is $O(N)$. In addition, rules R_i also need to be saved and its complexity is $O(1)$. Hence, in summary, the space complexity of our MORStreaming is $O(N)$.

IV. Experiments

In this section, the advantages of the MORStreaming system will be illustrated by comparing it with other multioutput methods. Different scenarios using both artificial and real-world datasets are involved in the experiments. In addition, all experiments are implemented in Python 3.7 version on Windows 10, running on a PC with system configuration Intel Core i7 processor (2.40 GHz) with 16-GB RAM.

A. Evaluation Settings

To evaluate the performance of each multioutput method, the prequential strategy introduced in [45] is employed. In this strategy, a sample is used to update a learning model after it is evaluated by this learning model. Furthermore, to compare overall performance, we calculated their mean value for all data streams. In addition, according to the suggestion of [46], we employ a windowed measurement method and a tumbling window of size 200. More importantly, to reduce the effects of randomness in our evaluation, all compared methods were performed at least 30 times for each dataset. Hence, the average of the repetitions was taken as the performance.

In this sense, we validate the performance of the algorithms in terms of three aspects, i.e., predictive performance, model size, and running time. To validate the predictive performance, the average root-mean-square error (ARMSE) was calculated, considering both an overall measurement using all the arrived samples and errors in the sliding window. The ARMSE is defined as

$$\text{ARMSE} = \frac{1}{d}\sum_{d}^{t=1}\sqrt{\frac{\sum_{i=1}^{N}\left(y_i^t - \widehat{y}_i^t\right)}{N}}. \tag{23}$$

In addition, the running time (s) of each system and the total model size (MB) consumed by the learning models were also recorded. In all cases, metrics were recorded in intervals of 200 samples.

Next, we discuss the hyperparameters of MORStreaming. In MORStreaming, there are two main groups of parameters in the learning stage: 1) expansion rules and 2) learning instances. In order to expand the rule set, the parameters include probability δ, the minimum number n of samples which are required to expand rule R_i, and the Hoeffding bound ϵ. However, even though the Hoeffding bound ε decreases considerably when more instances are obtained, it is not efficient to select literals in practice if they are only dependent on the Hoeffding bound. Hence, a threshold τ is defined in our proposed MORStreaming system, and if $\epsilon < \tau$, the literal with the higher MVR is chosen and the rule is expanded considering the literal. In order to learn instances, the parameters include parameter λ, which is used to define the frequency of neuron removal, and parameter $age_{\max}$ is defined as the lifetime of each edge. However, in the MORStreaming system, parameter λ equals parameter n in rule expansion. In the prediction stage, kernel function K and bandwidth h also need to be set.

B. Artificial Datasets

MORStreaming has two defining characteristics. First, it uses a different method to the traditional local or global method. Second, it uses a topology learning method to learn instances. Hence, to illustrate the advantages of using a different method, two extensions of MORStreaming were presented following the idea of classical local strategy (MORStreaming-L) and global strategy (MORStreaming-G). In our local strategy, a rule set is learned independently for each output, and the final prediction results are calculated by concatenating the individual prediction results of each rule. In contrast,

TABLE I
ARTIFICIAL DATASETS

Dataset	SAMPLE SIZE	Inputs	Outputs
2Dplanes	256000	20	8
FriedD	256000	10	4
FriedAsyncD	256000	10	4
MV	256000	20	9

TABLE II
ARSME FOR ARTIFICIAL DATASETS

	2Dplanes	FiredD	FriedAsyncD	MV
**-L*	**2.736**	**8.643**	**6.976**	**24.112**
*	2.741	8.709	7.102	24.539
**-G*	3.477	10.977	9.234	35.343
**-K50*	2.845	9.236	8.472	33.684
**-K200*	2.739	8.716	7.122	24.342

TABLE III
MODEL SIZE FOR ARTIFICIAL DATASETS

	2Dplanes	FriedD	FriedAsyncD	MV
**-L*	**89.676**	**401.166**	**403.392**	**689.482**
*	97.345	412.008	417.396	709.459
**-G*	154.023	460.349	506.423	820.943
**-K50*	92.609	406.755	411.387	701.307
**-K200*	129.347	434.831	465.391	780.365

TABLE IV
RUNNING TIME FOR ARTIFICIAL DATASETS

	2Dplanes	FriedD	FriedAsyncD	MV
**-L*	28.290	1203.685	1139.272	1648.291
*	15.532	670.717	606.633	805.970
**-G*	**10.652**	**500.134**	**489.306**	**576.004**
**-K50*	12.598	579.060	**553.601**	734.702
**-K200*	17.378	770.631	705.491	979.109

a literal is chosen in the global strategy based on a compromise on reducing the MVR with respect to all output attributes. Furthermore, to illustrate the advantages of the topology learning method, an extension of MORStreaming is presented using a KNN method to learn the instances (MORStreaming-K).

Next, to demonstrate the advantages of MORStreaming, we first conducted experiments with four artificial datasets: 1) 2Dplanes; 2) FriedD; 3) FriedAsyncD; and 4) MV. The datasets were designed by Duarte and Gama [19] and are used in many research studies on multioutput regression for data streams [47]. As for concept drift, the FriedD, FriedAsyncD, and MV datasets contain concept drift. To be more specific, concept drift in the FriedD dataset occurs simultaneously for all the output variables in the middle location of the data stream, while concept drift in the FriedAsyncD dataset occurs asynchronously. Finally, the design of the MV dataset was inspired by the homonym dataset [19]. Details of the four artificial datasets are described in Table I.

As for the parameters, first, the parameters which refer to the rule expansion were set to $n = 200$ (i.e., equal to the size of the window), $\tau = 0.05$, and $\delta = 0.0000001$, according to the typical configurations in [19]. Then, as parameter λ also needs to equal the size of the window, it is also set to 200. However, to set the parameter $age_{\max}$, we select 6000 samples to evaluate the prediction performance and utilize a grid search method from the set [50, 100, 200]. Based on the prediction result, parameters $age_{\max}$ was set to 50. In the prediction stage, we utilize the Gaussian kernel and set the bandwidth using a maximum-likelihood estimation [48] method. In addition, due to the KNN algorithm in MORStreaming-K, an important parameter k representing the number of nearest neighbors needs to be set.

The comparative results are shown in Tables II–IV, where "*" represents MORStreaming for convenience. In Table II, MORStreaming-L achieved slightly lower ARMSE in the artificial datasets than the other methods. Table III shows that MORStreaming-G needed fewer model sizes than the other methods. In addition, MORStreaming-K50 required less running time than the other artificial dataset methods (Table IV). However, from these three perspectives, MORStreaming outperformed the other methods. The reasons for the higher performance are as follows: 1) MORStreaming needs fewer model sizes than MORStreaming-L, although MORStreaming produces a slightly higher error rate and higher running time than MROStreaming-L; 2) MORStreaming has higher predictive accuracy and needs less running time than MORStreaming-G; and 3) the predictive performance of MORStreaming is more stable than MORStreaming-K because MORStreaming-K is dependent on the K value. If the K value increases, ARMSE decreases, but it needs more model sizes, and thereby using more running time than MORstreaming. In summary, MORStreaming outperforms MORStreaming-L, and since it can be used to learn a rule set, our proposed method has an advantage over MORStreaming-G. When compared with MORStreaming-K, the better performance of MORStreaming illustrates the advantage of using a topology learning method to learn instances.

In addition, we not only compare the ARMSE for the whole dataset but also the evolution of the ARMSE through the time spent. The line plots are presented for the evaluated datasets, 2Dplanes, FriedD, FriedAsyncD, and MV in Fig. 1. From these results, different patterns emerge depending on the considered dataset. For example, since 2Dplanes is a stationary dataset, the error rate does not change after convergence. However, for the FriedD dataset, due to the concept drift that occurs after about 140 000 samples, the error rate will increase incrementally, but as our MORStreaming has strategies to handle concept drift, the error rate will finally converge. The FriedAsyncD dataset has extra concept drift, so the error rate will increase by two, but the error rate can also converge after about 240 000 samples. The MV dataset

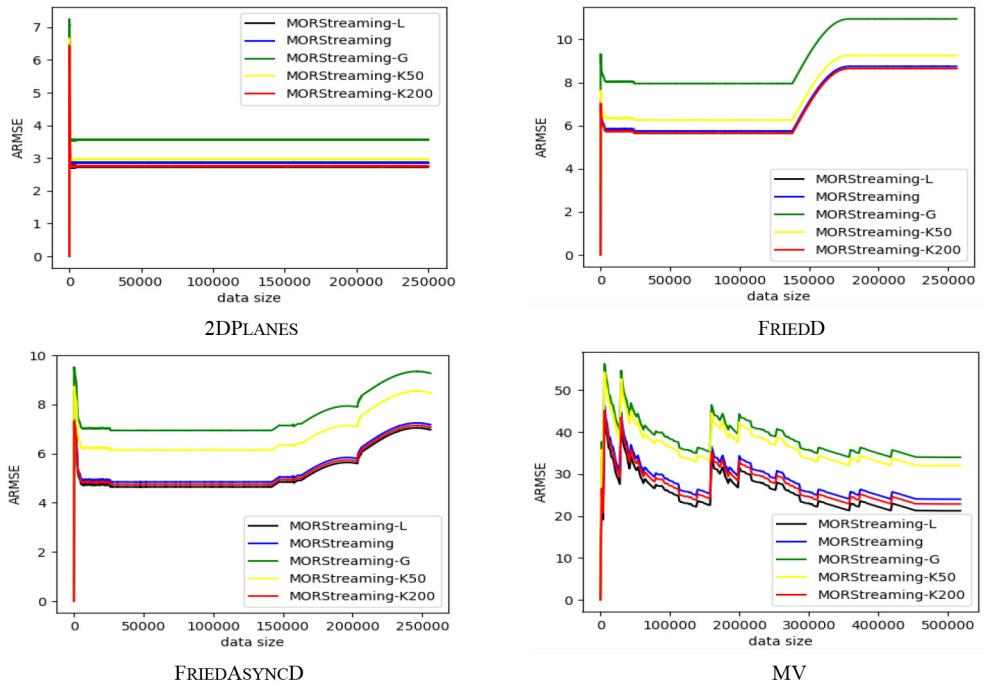

Fig. 1. ARMSE based on different datasets.

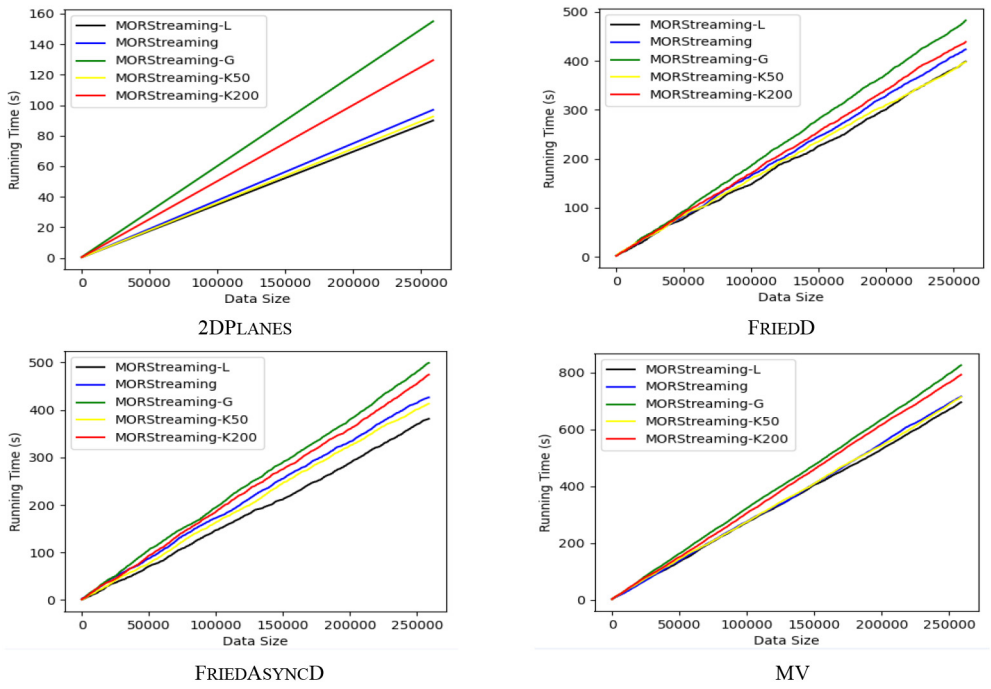

Fig. 2. Running time based on different datasets.

not only contains concept drift but also includes noise, so the change in accuracy is not as smooth as the change in accuracy based on the FriedD and FriedAsyncD datasets, and fluctuates. However, the convergence of ARMSE also occurs after 420 000 samples. Hence, we can see that all the methods display the same behavior when implemented in the same dataset, based on the AMRSE; that is, although the convergence of ARMSE displays different patterns because the pattern of concept drift is different in different datasets, the convergence of AMRSE can be guaranteed when more data were processed.

Next, the evolution of the running time was also considered in each dataset (Fig. 2). From Fig. 2, we can see similar behaviors, i.e., an approximately linear relationship between the time each multioutput regression method spent merged for all datasets. Hence, these experimental results show that the time complexity of each system is not too large. The experimental results also show that the number of saved instances

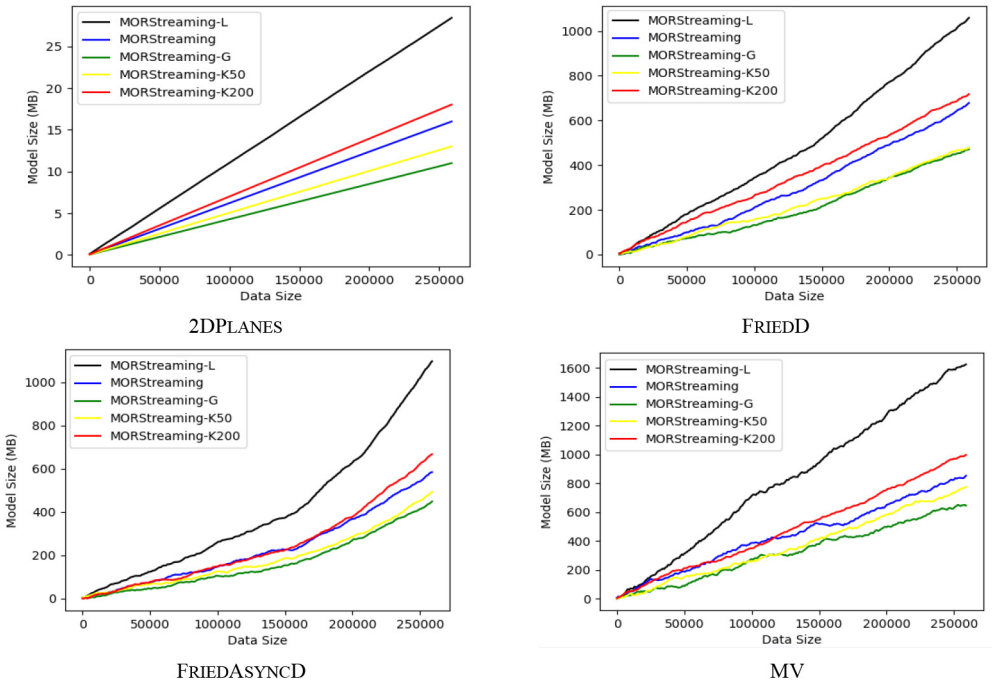

Fig. 3. Model size based on different datasets.

TABLE V
DESCRIPTION OF THE COMPARED METHODS

Acronym	Description
MTR-HT$_{Mean}$	The variant of MTR-HT which calculates the mean of the outputs at the leaf nodes
MTR-HT$_{Perception}$	The variant of MTR-HT which learns a perceptron model of outputs at the leaf nodes
iSOUP-Tree	The variant of Hoeffding-Tree method which dynamically selects rules between the MTR-HT$_{Mean}$ and MTR-HT$_{Perception}$
iSOUP-HT	The variant of Hoeffding-Tree method which use the stacked regressors for making predictions
MTR-IBL	The variant of IBLStreams which learns a single instance-based model for each output
MTR-SVR	The variant of online SVR which learns a single S for each output

will impact the time which is spent to learn the rules and make predictions.

Finally, we compare the evolution of the model size over time. Fig. 3 shows the comparative results. From Fig. 3, we can see the MORStreaming-G achieves the best performance. Similar to the experimental results for running time, the relationship between the amount of memory spent by different systems over time was linear for nearly all the datasets. Furthermore, when more data were learned, the clustering results, that is, the topology network, become reliable, and fewer new instances and new rules need to be learned, so the running time is shorter if the model size is larger with the same data size.

TABLE VI
REAL-WORLD DATASETS

Dataset	SAMPLE SIZE	Inputs	Outputs
Bicycles	17379	22	3
Eunite03	8064	29	5
RF1	9005	64	8
RF2	7679	576	8
SCM1d	9803	280	16
SCM20d	8966	61	16
Traffic Flow	12590	12	4
Sydney Trains	20000	20	8

C. Real-World Datasets

The experimental results for some real-world datasets are presented in this section. The comparison experiments were carried out with rule-based systems (MTR-HT$_{Mean}$ [19] and MTR-HT$_{Perceptron}$ [19]), decision tree-based systems (standard iSOUP-Tree [47] and stacked iSOUP-Tree [47]), and MORStreaming. In addition, two single-output regressors, IBLStreams (instance-based algorithm) [30] and online SVR [49], were also extended for comparison purposes. Table V summarizes the main characteristics of each method, including their acronyms, which will be used from here onward.

TABLE VII
COMPARISON RESULTS OF ARMSE ON REAL-WORLD DATASETS

	MTR-HT$_{\text{Mean}}$	MTR-HT$_{\text{Perception}}$	iSOUP-Tree	iSOUP-HT	MORStreaming	MTR-SVR	MTR-IBL
Bicycles	**86.73 (1)**	141.68 (7)	101.39 (5)	132.97 (6)	87.97 (2)	94.51 (4)	90.63 (3)
Eunite03	25.86 (3)	26.89 (6)	25.96 (4)	**22.32 (1)**	23.86 (2)	28.63 (7)	26.19 (5)
RF1	23.15 (5)	28.03 (6)	13.02 (2)	20.04 (3)	**11.03 (1)**	19.19 (4)	24.07 (7)
RF2	26.87 (2)	60.97 (5)	**23.51 (1)**	57.67 (4)	35.95 (3)	67.55 (7)	65.65 (6)
SCM1d	245.91 (4)	354.72 (7)	**215.42 (1)**	317.70 (6)	279.62 (5)	235.66 (3)	356.19 (7)
SCM20d	246.80 (5)	198.31 (4)	147.39 (2)	170.41 (3)	**139.58 (1)**	247.31 (6)	250.10 (7)
Traffic flow	6.91 (3)	5.38 (2)	7.28 (4)	11.09 (5)	**4.94 (1)**	19.05 (7)	18.97 (6)
Trains	415.45 (4)	429.94 (5)	328.83 (3)	317.67 (2)	**301.08 (1)**	520.04 (7)	487.38 (6)
AVE Rank	3.375	5.25	2.75	3.75	**2**	5.63	5.88

TABLE VIII
COMPARISON RESULTS OF ARMAE ON REAL-WORLD DATASETS

	MTR-HT$_{\text{Mean}}$	MTR-HT$_{\text{Perception}}$	iSOUP-Tree	iSOUP-HT	MORStreaming	MTR-SVR	MTR-IBL
Bicycles	0.82 (2)	0.98 (3)	1.16 (4)	1.27 (5)	**0.79 (1)**	1.92 (6)	1.97 (7)
Eunite03	0.52 (2)	0.83 (5)	0.92 (6)	1.05 (7)	**0.46 (1)**	0.59 (3)	0.62 (4)
RF1	0.56 (1)	2.13 (5)	1.79 (4)	1.67 (3)	0.59 (2)	3.02 (6)	3.42 (7)
RF2	1.77 (2)	1.91 (4)	1.87 (3)	1.94 (5)	**1.59 (1)**	2.51 (6)	2.67 (7)
SCM1d	0.95 (2)	2.02 (5)	1.76 (3)	1.75 (4)	**0.86 (1)**	2.42 (7)	2.30 (6)
SCM20d	**0.44 (1)**	0.96 (3)	1.23 (4)	1.49 (5)	0.48 (2)	1.91 (6)	2.10 (7)
Traffic flow	1.02 (2)	1.47 (3)	1.66 (5)	1.59 (4)	**0.79 (1)**	2.01 (6)	2.04 (7)
Trains	1.29 (4)	1.65 (5)	1.03 (2)	1.21 (3)	**0.91 (1)**	1.69 (7)	1.67 (6)
AVE Rank	2.00	4.13	3.88	4.5	**1.25**	5.88	6.38

Next, some of the parameters are fixed for each algorithm and these are the same for the artificial experiment. The attempts of splitting were performed once at intervals of 200 samples. The probability δ for the calculation of the Hoeffding bound was set to 0.0000001 and the tie-break parameter τ was set to 0.05. In all the datasets, we want to provide a "warm" start for the evaluations, so we employ 2000 samples to initiate the tree predictors. The perceptron weights in AMRules, iSOUP-Tree, and linear regressor were set to uniform random values in the $[-1, 1]$ interval, and new leaf nodes in iSOUTP-Tree can inherit their ancestors' weights. The evaluation strategy is the same as the strategy introduced in Section IV-A. The experimental results still include three aspects: 1) predictive performance; 2) running time; and 3) model size.

As for real-world datasets, a summary of each dataset used in our evaluations is given in Table VI. Their detailed descriptions are as follows.

1) *Bicycles:* This dataset has been used in two research studies for the multioutput regression problem of data streams [19], [47]. This dataset describes the count of rental bikes in an hour during the period between 2011 and 2012 in the "Capital" bikeshare system [19]. The data also contain seasonal and weather information for each rent case. So this data can be used to predict the count of each type of users.
2) *Eunite03:* This dataset was first reported in the competition for the 3rd European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (2003), and then employed in research on multioutput regression for data streams [19]. The dataset concerns the continuous production process of manufactured glass. The input features of each observation include the parameters that were employed in producing the glasses, while the outputs refer to glass quality.
3) *RF1 and RF2*: These two datasets were obtained from the U.S. National Weather and were first reported by [13]. Then, they were employed in research on multioutput regression for data streams [19], [47]. The RF1 and RF2 datasets describe the river network flows in the future 48 h at specific locations. However, in the first dataset, RF1, the sensor data is only used. In contrast, in the second dataset, i.e., RF2, the information of precipitation forecast (expected rainfall) for each of the measurement sites is added.
4) *SCM1d and SCM20d:* These two datasets were first proposed by [13] and were extracted from the Trading Agent Competition in the Supply Chain Management tournament in 2010. Then, these two datasets were applied to research on multioutput regression for data streams [19], [47]. Each observation corresponds to a day in the tournament (220 days for each tournament and 18 games during the whole tournament). The input features correspond to the prices considering a specific day in the tournament. Each dataset has

TABLE IX
COMPARISON RESULTS OF RUNNING TIME ON REAL-WORLD DATASETS

	MTR-HT$_{\text{Mean}}$	MTR-HT$_{\text{Perception}}$	iSOUP-Tree	iSOUP-HT	MORStreaming	MTR-SVR	MTR-IBL
Bicycles	7.9 (2)	14.6 (7)	14.1 (5)	14.6 (6)	10.6 (4)	8.3 (3)	**6.9 (1)**
Eunite03	11.6 (3)	15.3 (5)	16.6 (6)	17.6 (7)	12.6 (4)	10.9 (2)	**10.3 (1)**
RF1	175.8 (4)	220.3 (7)	191.3 (5)	203.2 (6)	173.5 (3)	163.2 (2)	**120.4 (1)**
RF2	**150.4 (1)**	171.9 (4)	187.8 (7)	186.9 (6)	159.1 (3)	173.5 (5)	157.6 (2)
SCM1d	**809.1 (1)**	816.5 (5)	814.7 (3)	850.7 (7)	811.1 (2)	815.4 (4)	817.7 (6)
SCM20d	100.9 (3)	122.3 (7)	115.2 (6)	112.7 (5)	109.1 (4)	97.3 (2)	**80.4 (1)**
Traffic flow	6.7 (4)	7.4 (5)	8.2 (6)	10.6 (7)	3.6 (2)	5.2 (3)	**2.9 (1)**
Trains	25.6 (3)	31.8 (7)	29.0 (5)	30.2 (6)	22.4 (2)	27.9 (4)	**20.6 (1)**
AVE Rank	2.63	5.88	5.38	6.25	3.00	3.00	**2.25**

TABLE X
COMPARISON RESULTS OF MODEL SIZE ON REAL-WORLD DATASETS

	MTR-HT$_{\text{Mean}}$	MTR-HT$_{\text{Perception}}$	iSOUP-Tree	iSOUP-HT	MORStreaming	MTR-SVR	MTR-IBL
Bicycles	4.4 (2)	4.5 (3)	4.6 (4)	5.0 (5)	**2.5 (1)**	10.3 (6)	12.7 (7)
Eunite03	8.7 (4)	8.6 (3)	8.0 (2)	9.4 (5)	**7.0 (1)**	25.6 (7)	22.2 (6)
RF1	11.5 (4)	11.4 (2)	11.5 (3)	18.0 (5)	**11.0 (1)**	53.2 (6)	60.4 (7)
RF2	37.2 (2)	37.6 (3)	38.7 (4)	39.7 (5)	**33.9 (1)**	123.1 (7)	117.9 (6)
SCM1d	665.4 (3)	667.0 (4)	664.2 (2)	707.0 (5)	**649.1 (1)**	917.4 (7)	915.7 (6)
SCM20d	275.0 (2)	278.6 (3)	283.9 (4)	307.0 (5)	**273.1 (1)**	447.3 (6)	470.4 (7)
Traffic flow	23.8 (3)	25.9 (5)	20.2 (2)	24.6 (4)	**17.7 (1)**	94.2 (6)	98.5 (7)
Trains	105.6 (2)	110.7 (3)	125.9 (5)	114.3 (4)	**102.3 (1)**	453.1 (6)	462.7 (7)
AVE Rank	2.75	3.25	3.25	4.75	**1.00**	6.38	6.63

16 outputs which correspond to the next day's mean price (SCM1d) or mean price for 20 days in the future (SCM20d), regarding each product in the simulation.

5) *Traffic Flow:* The dataset [10] was collected by us to build an experimental dataset. In this, there are 12 590 traffic data points were collected as a time-series data set and each data point has four outputs. Furthermore, each output has been processed to present the amount of traffic in bytes per unit of time. However, in this dataset, the first output attribute has a clear relationship with the second output attribute, and the third output attribute has a clear relationship with the fourth output attribute.
6) *Sydney Trains:* This dataset was collected from the TfNSW Open Data Hub (https://opendata.transport.nsw.gov.au/search/type/dataset). This dataset records the passenger load on the Waratah train. The Waratah train has eight carriages, so predicting the passenger load of the Waratah train involves predicting the number of passengers in each carriage when it leaves a platform. Usually, the number of passengers in carriage 1 and carriage 8 are very similar; and the remaining carriages have a similar passenger load.

First, with respect to predictive performance, Table VII summarizes the ARMSE. As shown in Table VII, compared with the structured-outputs algorithms, although our proposed MORStreaming does not obtain the best accuracy in all datasets, it was the most accurate system because it obtained the best average rank (2), particularly for those datasets whose outputs have a clear relationship. For example, for the Traffic Flow and Sydney Trains datasets, the accuracy of MORStreaming is much higher than the other regressors. The iSOUP-Tree was the second-best performer, achieving the second-best average rank (2.75), and MTR-HT$_{\text{Perception}}$ was the worst predictor. Compared with the single-output algorithms, the accuracy of MORStreaming was also better than MTR-IBL and MTR-SVR. In summary, our MORStreaming improved the accuracy of a prediction considering the relationship of the output attributes. To more completely show the advantage of MORStreaming, we further compare each algorithm in terms of ARMAE [19] (see Table VIII) and reach the same conclusion as the former experiment, so these results also illustrate the advantage of MORStreaming.

With respect to the running times, the single-output algorithms, especially IBLStreams achieves a lower time in the learning model, but the time needs to double if parallel processing is not implemented. Specific to the structured-outputs algorithms, the simplest alternative to the Hoeffding tree-based method, MTR-HT$_{\text{Mean}}$, is the fastest predictor for all datasets, as shown in Table IX. Our proposed MORStreaming needs more running time than MTR-HT$_{\text{Mean}}$, but it has a faster running time than the other algorithms, and it achieved the second smallest average rank (3) in terms of running time.

Finally, the model size (MB) of each method is shown in Table X. From Table X, we can see the biggest advantage of the structured-outputs algorithms over the single-output

algorithms is that the model size obtained by the structured-outputs algorithms is less than the single-output algorithms. The reason for this advantage is the structured-outputs algorithms only need to save a model, but the single-output algorithms need to save multiple models. Further, compared with other structured-outputs algorithms, our MORStreaming also used less memory resources in all datasets (rank 1). In contrast, iSOUP-HT used more memory resources than its other structured-outputs competitors (rank 4.5), but still achieves better performance than single-output algorithms. The remaining structured-outputs methods only differed by a small amount, regardless of the compared dataset.

V. Conclusion and Further Study

In this article, we propose a new multioutput regression system called MORStreaming. Furthermore, the establishment of the MORStreaming system solves two additional challenges. The first challenge is that there is an existing structure in the outputs due to the number of outputs which are greater than one, and the structured outputs can change due to concept drift. The second challenge is the need for online learning instances. The algorithm is built on adaptive rules, so the rules can not only be learned in an online manner but can also fit the drift of the structured outputs. In addition, an online instances learning algorithm is proposed to sequentially handle input data by following the rules that we learned, and saving a topology network, which is significantly smaller than the size of the original streaming data. Based on this topology network, noise and concept drift can easily be solved. In future works, to further improve the performance of the MORStreaming system, a mechanism for overfitting prevention should be introduced into MORStreaming. In addition, an outlier detection method should be introduced to find unusual samples.

References

[1] L. Zhang, J. Lin, and R. Karim, "Sliding window-based fault detection from high-dimensional data streams," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 289–303, Feb. 2017.

[2] L. Carafoli, F. Mandreoli, R. Martoglia, and W. Penzo, "Streaming tables: Native support to streaming data in DBMSs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 10, pp. 2768–2782, Oct. 2017.

[3] J. Shao, F. Huang, Q. Yang, and G. Luo, "Robust prototype-based learning on data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 978–991, May 2018.

[4] G. Krempl *et al.*, "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newslett.*, vol. 16, no. 1, pp. 1–10, 2014.

[5] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[6] Y. Song, J. Lu, H. Lu, and G. Zhang, "Fuzzy clustering-based adaptive regression for drifting data streams," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 3, pp. 544–557, Mar. 2020.

[7] S. Deng, B. Wang, S. Huang, C. Yue, J. Zhou, and G. Wang, "Self-adaptive framework for efficient stream data classification on storm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 123–136, Jan. 2020.

[8] J. Shao, Y. Tan, L. Gao, Q. Yang, C. Plant, and I. Assent, "Synchronization-based clustering on evolving data stream," *Inf. Sci.*, vol. 501, pp. 573–587, Oct. 2019.

[9] E. Ikonomovska, J. Gama, and S. Dzeroski, "Learning model trees from evolving data streams," *Data Min. Knowl. Disc.*, vol. 23, no. 1, pp. 128–168, 2011.

[10] H. Yu, J. Lu, J. Xu, and G. Zhang, "A hybrid incremental regression neural network for uncertain data streams," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–8, doi: 10.1109/IJCNN.2019.8852364.

[11] C. Li, F. Wei, W. Dong, Q. Liu, S. Member, and X. Wang, "Dynamic structure embedded online multiple-output regression for stream data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 323–336, Feb. 2019.

[12] X. Zhen, M. Yu, X. He, and S. Li, "Multi-target regression via robust low-rank learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 497–504, Feb. 2018.

[13] H. Yu, A. Liu, B. Wang, R. Li, G. Zhang, and J. Lu, *Real-Time Decision Making for Train Carriage Load Prediction via Multi-stream Learning* [Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)], vol. 12576, 2020, pp. 29–41, doi: 10.1007/978-3-030-64984-5_3.

[14] W. Chung, J. Kim, H. Lee, and E. Kim, "General dimensional multiple-output support vector regressions and their multiple kernel learning," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2572–2584, Nov. 2015.

[15] D. Kocev, C. Vens, and J. Struyf, "Tree ensembles for predicting structured outputs," *Pattern Recognit.*, vol. 46, no. 3, pp. 817–833, 2013.

[16] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Data Min. Knowl. Discov.*, vol. 5, no. 5, pp. 216–233, 2015.

[17] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 66, no. 6, pp. 37–66, 1991.

[18] J. R. Quinlan, "Combining instance-based and model-based learning," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 236–243.

[19] J. Duarte and J. Gama, "Multi-target regression from high-speed data streams with adaptive model rules," *Proc. IEEE Int. Conf. Data Sci. Adv. Anal.*, 2015, pp. 1–10.

[20] H. Yu, J. Lu, and G. Zhang, "Online topology learning by a Gaussian membership-based self-organizing incremental neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 3947–3961, Oct. 2020.

[21] B. Pfahringer, G. Holmes, and R. Kirkby, "New options for Hoeffding trees," in *Advances in Artificial Intelligence* (Lecture Notes in Artificial Intelligence 4830), M. A. Orgun and J. Thornton, Eds. Heidelberg, Germany: Springer, 2007, pp. 90–99.

[22] A. Wibisono, W. Jatmiko, H. A. Wisesa, B. Hardjono, and P. Mursanto, "Traffic big data prediction and visualization using Fast Incremental Model Trees-Drift Detection (FIMT-DD)," *Knowl. Based Syst.*, vol. 93, pp. 33–46, Feb. 2016.

[23] H. M. Gomes, J. P. Barddal, L. E. B. Ferreira, and A. Bifet, "Adaptive random forests for data stream regression," in *Proc. 26th Eur. Symp. Artif. Neural Netw.*, Bruges, Belgium, Apr. 2018, pp. 267–272.

[24] E. Ikonomovska, J. Gama, and S. Džeroski, "Online tree-based ensembles and option trees for regression on evolving data streams," *Neurocomputing*, vol. 150, pp. 458–470, Feb. 2015.

[25] E. Almeida, C. Ferreira, and J. Gama, "Adaptive model rules from data streams," in *Proc. Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2013, pp. 480–492.

[26] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A Survey," *Inf. Sci.*, vol. 490, pp. 344–368, Jul. 2019.

[27] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1403–1416, Jul. 2015.

[28] H. Yu, J. Lu, and G. Zhang, "Continuous support vector regression for nonstationary streaming data," *IEEE Trans. Cybern.*, early access, Sep. 10, 2020, doi: 10.1109/TCYB.2020.3015266.

[29] H. Yu, J. Lu, and G. Zhang, "An online robust support vector regression for data streams," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 12, 2020, doi: 10.1109/TKDE.2020.2979967.

[30] A. Shaker and E. Hullermeier, "IBLStreams: A system for instance-based classification and regression on data streams," *Evol. Syst.*, vol. 3, no. 4, pp. 235–249, 2012.

[31] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 3, pp. 55–67, 1970.

[32] G. Shantanu and S. Sarawagi, "Discriminative methods for multilabeled classification," in *Proc. Pacific–Asia Conf. Knowl. Disc. Data Mining*, Sydney, NSW, Australia, 2004, pp. 22–30.

[33] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[34] W. Zhang, X. Liu, Y. Ding, and D. Shi, "Multi-output LS-SVR machine in extended feature space," in *Proc. IEEE Int. Conf. Comput. Intell. Meas. Syst. Appl.*, Jul. 2012, pp. 130–134
[35] A. J. Izenman, "Reduced-rank regression for the multivariate linear model," *J. Multivariate Anal.*, vol. 5, no. 2, pp. 248–264, 1975.
[36] D. Tuia, J. Verrelst, L. Alonso, F. Perez-Cruz, and G. Camps-Valls, "Multioutput support vector regression for remote sensing biophysical parameter estimation," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 804–808, Jul. 2011.
[37] J. Struyf and S. Dzeroski, "Constraint based induction of multi-objective? Regression trees," in *Proc. Int. Workshop Knowl. Disc. Inductive Databases*, 2005, pp. 222–233.
[38] T. Aho, B. Ženko, and S. Džeroski, "Rule ensembles for multi-target regression," in *Proc. 9th IEEE Int. Conf. Data Min.*, Dec. 2009, pp. 21–30.
[39] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, pp. 257–286, Mar. 2000.
[40] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett*. vol. 31, no. 8, pp. 651–666, 2010.
[41] S. Furao and O. Hasegawa, "An incremental network for online unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, 2006.
[42] A. Liu, J. Lu, and G. Zhang, "Concept drift detection via equal intensity k-means space partitioning," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3198–3211, Jun. 2021.
[43] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. 17th Braz. Symp. Artif. Intell.*, 2004, pp. 286–295.
[44] P. Li, X. Hu, Q. Liang, and Y. Gao, "Concept drifting detection on noisy streaming data in random ensemble decision trees," in *Machine Learning and Data Mining in Pattern Recognition* (Lecture Notes in Computer Science 5632), P. Perner, Ed. Heidelberg, Germany: Springer, 2009, pp. 236–250.
[45] J. Gama, *Knowledge Discovery from Data Streams*. Boca Raton, FL, USA: Chapman Hall/CRC, 2010.
[46] G. D. F. Morales and A. Bifet, "SAMOA: Scalable advanced massive online analysis," *J. Mach. Learn. Res.*, vol. 16, no. 5, pp. 149–153, 2015.
[47] A. Osojnik, P. Panov, and S. Džeroski, "Tree-based methods for online multi-target regression," *J. Intell. Inf. Syst.*, vol. 50, no. 2, pp. 315–339, 2018.
[48] H. Yu, J. Lu, and G. Zhang, "Topology learning-based fuzzy random neural network for streaming data regression," *IEEE Trans. Fuzzy Syst.*, early access, Nov. 20, 2020, doi: 10.1109/TFUZZ.2020.3039681.
[49] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

Hang Yu (Member, IEEE) received the Ph.D. degree in software engineering from the University of Technology Sydney, Sydney, NSW, Australia, in 2020.

He is a member of the Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney. He is also an Lecturer with the Faculty of Computer Engineering and Science, Shanghai University, Shanghai, China. He has published over ten research papers in IEEE TRANSACTIONS and other refereed journals and conference proceedings. His research interests include streaming data mining, concept drift, and fuzzy systems.

Jie Lu (Fellow, IEEE) received the Ph.D. degree in applied mathematics from Curtin University, Perth, WA, Australia, in 2000.

She is a Distinguished Professor and the Director of the Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW, Australia. She has been awarded ten Australian Research Council discovery grants and led 15 industry projects. She has published over 450 papers in IEEE TRANSACTIONS and other journals and conferences, supervised 40 Ph.D. students to completion. Her main research expertise is in transfer learning, concept drift, decision support systems, and recommender systems.

Dr. Lu has received the UTS Medal for Research and Teaching Integration in 2010, the UTS Medal for Research Excellence in 2019, the IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2019, the *Computer Journal* Wilkes Award in 2018, and the Australian Most Innovative Engineer Award in 2019. She serves as the Editor-in-Chief for *Knowledge-Based Systems* (Elsevier) and the *International Journal on Computational Intelligence Systems* (Atlantis). She has delivered 27 keynote speeches at IEEE and other international conferences and chaired 15 international conferences. She is also an IFSA Fellow and an Australian Laureate Fellow.

Guangquan Zhang received the Ph.D. degree in applied mathematics from Curtin University, Perth, WA, Australia, in 2001.

He is an Australian Research Council (ARC) QEII Fellow, an Associate Professor, and the Director of the Decision Systems and e-Service Intelligent Research Laboratory, Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW, Australia. From 1993 to 1997, he was a Full Professor with the Department of Mathematics, Hebei University, Baoding, China. He has published six authored monographs, five edited research books, and over 450 papers including 240 refereed journal articles. He has won nine ARC Discovery Project grants and many other research grants, supervised 30 Ph.D. students to completion. His main research interests lie in the area of fuzzy multiobjective, bilevel, and group decision making, fuzzy measure, and machine learning.

Dr. Zhang has served as a guest editor for five special issues of IEEE TRANSACTIONS and other international journals.