MORES: Online Incremental Multiple-Output Regression for Data Streams

Changsheng Li, Weishan Dong, Qingshan Liu, *Senior Member, IEEE,* and Xin Zhang

***Abstract*—Online multiple-output regression is an important machine learning technique for modeling, predicting, and compressing multi-dimensional correlated data streams. In this paper, we propose a novel online multiple-output regression method, called MORES, for streaming data. MORES can *dynamically* learn the structure of the regression coefficients to facilitate the model's continuous refinement. We observe that limited expressive ability of the regression model, especially in the preliminary stage of online update, often leads to the variables in the residual errors being dependent. In light of this point, MORES intends to *dynamically* learn and leverage the structure of the residual errors to improve the prediction accuracy. Moreover, we define three statistical variables to *exactly* represent all the seen samples for *incrementally* calculating prediction loss in each online update round, which can avoid loading all the training data into memory for updating model, and also effectively prevent drastic fluctuation of the model in the presence of noise. Furthermore, we introduce a forgetting factor to set different weights on samples so as to track the data streams' evolving characteristics quickly from the latest samples. Experiments on three real-world datasets validate the effectiveness and efficiency of the proposed method.**

***Index Terms*—Online multiple-output regression, dynamic relationship learning, forgetting factor, lossless compression**

I. Introduction

Data streams arise in many scenarios, such as online transactions in the financial market, Internet traffic and so on [1]. Unlike traditional datasets in batch mode, a data stream should be viewed as a potentially infinite process collecting data with varying update rates, as well as continuously evolving over time. In the context of data streams, although many research issues, such as classification [2], [3], [4], clustering [5], [6], [7], active learning [8], [9], [10], online feature selection [11], [12], multi-task learning [13], [14], change point detection [15], [16], etc., have been extensively studied over the last decade, little attention is paid to multiple-output regression. However, multiple-output regression also has a great variety of potential applications on data streams, including weather forecast, air quality prediction, etc.

In batch data processing, the purpose of multiple-output regression is to learn a mapping Φ from an input space $\mathbb{R}^d$ to an output space $\mathbb{R}^m$ on the whole training dataset. Based on the learned Φ, we can simultaneously predict multiple output variables $\mathbf{y} \in \mathbb{R}^m$ to a given new input vector $\mathbf{x} \in \mathbb{R}^d$ by: $\mathbf{y} = \Phi(\mathbf{x})$. According to the property of the learned mapping Φ, multiple-output regression techniques can be divided into linear and nonlinear ones. Since linear methods usually have low complexities and reliable prediction performance, they have attracted more attention in the past. Lots of batch multiple-output regression algorithms have been proposed [17], [18], [19], [20], [21], [22], [23], [24]. However, in streaming environments, the data is not stored in a batch mode, but it arrives sequentially and continuously. If using these batch methods to re-train the models for streaming data, the computational complexity and memory complexity will increase sharply. Moreover, when the size of the arrived data becomes large, it is also impractical to load all the historical data into memory for model training. Therefore, it is necessary to develop an online regression algorithm for simultaneously predicting multiple outputs.

So far, many online regression algorithms for predicting single output variable have been proposed [25], [26], [27]. The representative method is online passive-aggressive (PA) algorithm [26]. PA is a margin based online learning algorithm, and it has an analytical solution to update model parameters as the new data sample(s) arrives. Since there are often correlations among outputs, mining the correlation relationships can improve the prediction accuracy of the model [19]. However, PA only treats each of multiple outputs as an independent task, and thus can not capture any correlations among outputs. Recently, McWilliams and Montana take advantage of partial least squares (PLS) to build a recursive regression model for online predicting multiple outputs, called iS-PLS [28]. iS-PLS aims at finding a low-dimensional subspace to make the correlation between inputs and outputs maximized. iS-PLS focuses on the correlation between inputs and outputs, while it does not consider the correlations among outputs.

In this paper, we propose a novel Multiple-Output REgression method for Streaming data, named as MORES. MORES works in an incremental fashion. It aims at *dynamically* learning the structures of both the regression coefficients and the residual errors to continuously update the model. Specifically, when a new training sample arrives, we transform the update of the regression coefficients into an optimization problem. In the objective function, we highlight the following three aspects:

First, we take advantage of the matrix *Mahalanobis norm* to measure the divergence between updated regression coefficient matrix and current regression coefficient matrix, which can learn the structure of the regression coefficients to facilitate model's update.

Second, due to limited expressive power of the regression coefficient matrix, especially in the early stage of online

C. Li, W. Dong, and X. Zhang are with IBM Research-China, Beijing 100093. E-mail: {lcsheng, dongweis, zxin}@cn.ibm.com

Q. Liu is with the B-DAT Laboratory at the school of Information and Control, Nanjing University of Information Science and Technology, Nanjing 210014, China. E-mail: qsliu@nuist.edu.cn

update, there are often correlations between the residual errors. MORES thus utilizes the *Mahalanobis* distance instead of the Euclidean distance to measure the prediction error between the true values and the predicted values, which can learn the structure of the residual errors to improve the prediction accuracy.

Third, we define three statistical variables to store necessary information of both the historical data and the current data for exactly measuring the prediction error, such that MORES can avoid loading all the data into memory and visiting data multiple times. This also effectively prevents the updated regression coefficient matrix gradually deviating from the true coefficient matrix due to noise and outliers in the data streams. Meanwhile, we introduce a forgetting factor to set different weights on samples for adapting to data streams' evolvement.

Extensive experiments are conducted on three real-world datasets, and the experimental results demonstrate the effectiveness and efficiency of MORES.

II. Online Multiple-Output Regression for Streaming Data

Following the general setting of online learning [26], we assume that the learner first observes an instance $\mathbf{x}_t \in \mathbb{R}^d$ on the t-th round, and it simultaneously predicts multiple outputs $\widehat{\mathbf{y}}_t \in \mathbb{R}^m$ based on the current model $\mathbf{P}_{t-1} \in \mathbb{R}^{m\times d}$. After that, the learner receives the true responses $\mathbf{y}_t \in \mathbb{R}^m$ for this instance. Finally, the learner updates the current model $\mathbf{P}_{t-1}$ based on the new data point $(\mathbf{x}_t, \mathbf{y}_t)$. In this paper, our goal is to *online* update $\mathbf{P}_{t-1}$, such that the updated $\mathbf{P}_t$ can predict the outputs for the incoming instance $\mathbf{x}_{t+1}$ as accurately as possible. The prediction can be expressed in the following form:

$$\mathbf{y}_{t+1} = \mathbf{P}_t\mathbf{x}_{t+1} + \epsilon_{t+1}, \tag{1}$$

where $\mathbf{P}_t = [\mathbf{p}_{t,1}, \ldots, \mathbf{p}_{t,m}]^T$ denotes the learned regression coefficient matrix on the t-th round, and $\mathbf{p}_{t,i}$ is the regression coefficient vector of the i-th output. $\epsilon_{t+1} = [\epsilon_{t+1,1}, \ldots, \epsilon_{t+1,m}]^T$ is a vector consisting of m residual errors.

A. Objective Function

In order to obtain $\mathbf{P}_t$ on the t-th round, we first propose a simple formulation as:

$$\begin{aligned} \mathbf{P}_t = &\arg\min_{\mathbf{P}\in\mathbb{R}^{m\times d}} \|\mathbf{P} - \mathbf{P}_{t-1}\|_{\mathbf{F}}^2 \\ s.t.\ \ &\|\mathbf{y}_t - \mathbf{P}\mathbf{x}_t\|_2^2 \le \xi, \end{aligned} \tag{2}$$

where ξ is a positive parameter that controls the sensitivity to the prediction error. $\|\cdot\|_{\mathbf{F}}$ denotes the matrix *Frobenius norm*, and $\|\cdot\|_2$ denotes the l_2 *norm* of a vector.

The core idea of objective function (2) is as follows: On one hand, it intends to minimize the distance between $\mathbf{P}_t$ and $\mathbf{P}_{t-1}$ to make $\mathbf{P}_t$ close to $\mathbf{P}_{t-1}$ as much as possible, which can retain the information learned on previous rounds. On the other hand, it requires $\mathbf{P}_t$ to meet the condition: The total prediction error on the current data point $(\mathbf{x}_t, \mathbf{y}_t)$ is less than or equal to ξ.

Following [26], the optimization problem defined by (2) can be easily solved by the Lagrange multiplier method.

Although (2) has some merits for online regression prediction of streaming data, it still has the following limitations:

(i) Because of $\|\mathbf{P} - \mathbf{P}_{t-1}\|_{\mathbf{F}}^2 = tr((\mathbf{P} - \mathbf{P}_{t-1})(\mathbf{P} - \mathbf{P}_{t-1})^T) = \sum_{i=1}^m((\mathbf{p}_i - \mathbf{p}_{t-1,i})^T(\mathbf{p}_i - \mathbf{p}_{t-1,i}))$, we can see that the objective function in (2) treats the update of regression coefficients as m independent tasks. However, in streaming environments, outputs are often dependent, i.e., there are some positive or negative correlations among outputs, so updating regression coefficient vectors for all the outputs should not be regarded as completely independent tasks.

(ii) In order to acquire the updater $\mathbf{P}_t$, (2) imposes a constraint on $\mathbf{P}$, i.e., $\|\mathbf{y}_t - \mathbf{P}\mathbf{x}_t\|_2^2 \le \xi$. This constraint just refers to the current data point $(\mathbf{x}_t, \mathbf{y}_t)$ but ignores the historical data points $\mathcal{S}_{t-1} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{t-1}$. This may lead to the updated coefficient matrix gradually deviating from the true coefficient matrix because of noise and outliers in many practical streaming data applications.

(iii) The constraint of (2) takes advantage of the l_2 norm to measure the total prediction error. As we know, the l_2 norm of a vector assumes that all the variables in the vector are independent. However, due to limited expressive power of $\mathbf{P}_t$, especially when the round t is small, there are often correlations between the residual errors. Therefore, the l_2 norm is often the suboptimal choice for measuring the total prediction error.

In light of above three limitations, we propose to minimize the following objective function, in order to update the model on round t:

$$\begin{aligned} \min\ \ J(\mathbf{P},\Omega,\Gamma) =& \|\mathbf{P} - \mathbf{P}_{t-1}\|_{\Omega}^2 + \alpha\ell(\mathbf{P},\Gamma;\mathcal{S}_t) \\ &+ \beta\Delta_\phi(\Omega,\Omega_{t-1}) - \eta\log|\Gamma| \\ s.t.\ \ \ &\Omega \succeq 0,\ \ \Gamma \succeq 0, \end{aligned} \tag{3}$$

where α, β, and η are three trade-off parameters. $\|\cdot\|_\Omega$ denotes the matrix *Mahalanobis norm*. $\ell(\mathbf{P},\Gamma;\mathcal{S}_t)$ is the total prediction error on the data points $\mathcal{S}_t = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^t$. $\Delta_\phi(\Omega,\Omega_{t-1})$ denotes the Bregman divergence [29] that measures the distance between the matrix Ω and the matrix Ω_{t-1}. $\Omega \succeq 0$ and $\Gamma \succeq 0$ means that they are positive semi-definite.

In the objective function (3), the first term aims to learn the structure Ω of the regression coefficient matrix, and leverage Ω to measure the divergence between the updated matrix $\mathbf{P}$ and the current matrix $\mathbf{P}_{t-1}$ on round t. The second term intends to mine the underlying structure Γ of the residual errors, and take advantage of Γ to measure the total prediction error on the current data and the historical data. Here we define three statistical variables to store necessary information of data for lowering memory complexity, and introduce a forgetting factor to adapt to the evolving data streams (See (5), (6) for details). The third term is a regularization term, which aims at keeping Ω updated in a conservative strategy to reduce the influence of noise. The last term is used to control the complexity of Γ. Next, we will respectively explain the first three terms in detail.

The first term: In order to capture the structure of the regression coefficient matrix, we introduce *Mahalanobis norm*

of the matrix $(\mathbf{P}-\mathbf{P}_{t-1})$ to measure the divergence between $\mathbf{P}$ and $\mathbf{P}_{t-1}$. The *Mahalanobis norm* is expressed as:

$$\|\mathbf{P}-\mathbf{P}_{t-1}\|_{\Omega}=\sqrt{tr((\mathbf{P}-\mathbf{P}_{t-1})^T\Omega(\mathbf{P}-\mathbf{P}_{t-1}))}$$
$$=\sqrt{\sum_{i=1}^{d}(\mathbf{P}(i)-\mathbf{P}_{t-1}(i))^T\Omega(\mathbf{P}(i)-\mathbf{P}_{t-1}(i))}, \quad (4)$$

where $\mathbf{P}(i)$ denotes the i-th column of $\mathbf{P}$. When Ω is set to the identity matrix, the *Mahalanobis norm* of the matrix is reduced to the *Frobenius norm*. In (4), the term $(\mathbf{P}(i)-\mathbf{P}_{t-1}(i))^T\Omega(\mathbf{P}(i)-\mathbf{P}_{t-1}(i))$ is actually the *Mahalanobis* distance between $\mathbf{P}(i)$ and $\mathbf{P}_{t-1}(i)$, where Ω encodes the correlations between the variables of the i-th column of the regression coefficient matrix on round t [30]. Therefore, $\|\mathbf{P}-\mathbf{P}_{t-1}\|_{\Omega}^2$ can be viewed as a summation of d Mahalanobis distances, of which each measures the distance between the corresponding column vectors of $\mathbf{P}$ and $\mathbf{P}_{t-1}$.

The second term: The loss function $\ell(\mathbf{P},\Gamma;\mathcal{S})$ measures the prediction error on $\mathcal{S}$, which is defined as:

$$\ell(\mathbf{P},\Gamma;\mathcal{S})=\sum_{i=1}^{t}\mu^{t-i}(\mathbf{y}_i-\mathbf{P}\mathbf{x}_i)^T\Gamma(\mathbf{y}_i-\mathbf{P}\mathbf{x}_i), \quad (5)$$

where $0\leq\mu\leq 1$ is a forgetting factor[1]. When $\mu=0$, the prediction loss is only measured on the current sample without any historical samples. When $\mu=1$, all the samples have equal weights to contribute to the prediction loss. When $0<\mu<1$, all the samples have different contributions to the prediction loss. As a matter of fact, the function of μ is similar to a new form of time window on samples. The farther the historical sample is from the current sample in the time domain, the lower its importance is, which will fit in the evolving characteristic of data streams well. The matrix Γ embeds the correlation relationships among the residual errors on round t. The term $(\mathbf{y}_i-\mathbf{P}\mathbf{x}_i)^T\Gamma(\mathbf{y}_i-\mathbf{P}\mathbf{x}_i)$ measures the Mahalnobis distance between the true value $\mathbf{y}_i$ and the predicted value $\mathbf{P}\mathbf{x}_i$, which can remove the influence of the residual errors' correlations on distance calculation [31].

In streaming environments, it is impractical to load all the historical data into memory or scan a sample multiple times. An effective way to handle this issue is to define some statistical variables to store necessary information of the samples. In this paper, we introduce three statistical variables to realize lossless compression of the data. To do this, we will make use of the following property and lemma.

Property 1: Given a set of arbitrary sequence vectors, $\mathbf{x}_1,\ldots,\mathbf{x}_t$, and a constant μ, if $\mathbf{C}_t=\sum_{i=1}^{t}\mu^{t-i}\mathbf{x}_i\mathbf{x}_i^T$, then $\mathbf{C}_t=\mu\mathbf{C}_{t-1}+\mathbf{x}_t\mathbf{x}_t^T$, where t is the timestamp.

Lemma 1: The loss function (5) can be expressed as

$$\ell=tr(\Gamma\mathbf{C}_{t,\mathbf{YY}})+tr(\mathbf{P}^T\Gamma\mathbf{P}\mathbf{C}_{t,\mathbf{XX}})-2tr(\Gamma\mathbf{P}\mathbf{C}_{t,\mathbf{XY}}). \quad (6)$$

where $\mathbf{C}_{t,\mathbf{YY}}$, $\mathbf{C}_{t,\mathbf{XY}}$, and $\mathbf{C}_{t,\mathbf{XX}}$ are three variables, which are respectively defined as:

$$\mathbf{C}_{t,\mathbf{YY}}=\mu\mathbf{C}_{t-1,\mathbf{YY}}+\mathbf{y}_t\mathbf{y}_t^T. \quad (7)$$
$$\mathbf{C}_{t,\mathbf{XY}}=\mu\mathbf{C}_{t-1,\mathbf{XY}}+\mathbf{x}_t\mathbf{y}_t^T. \quad (8)$$
$$\mathbf{C}_{t,\mathbf{XX}}=\mu\mathbf{C}_{t-1,\mathbf{XX}}+\mathbf{x}_t\mathbf{x}_t^T. \quad (9)$$

[1]When $\mu=0$, and $t-i=0$, we define $\mu^{t-i}=1$ to ensure consistency.

Proof: Based on (5), we have

$$\begin{aligned}\ell(\mathbf{P},\Gamma;\mathcal{S})&=\sum_{i=1}^{t}\mu^{t-i}(\mathbf{y}_i-\mathbf{P}\mathbf{x}_i)^T\Gamma(\mathbf{y}_i-\mathbf{P}\mathbf{x}_i)\\&=\sum_{i=1}^{t}\mu^{t-i}(tr(\mathbf{y}_i^T\Gamma\mathbf{y}_i)+tr(\mathbf{x}_i^T\mathbf{P}^T\Gamma\mathbf{P}\mathbf{x}_i))\\&\quad-2\sum_{i=1}^{t}\mu^{t-i}tr(\mathbf{y}_i^T\Gamma\mathbf{P}\mathbf{x}_i)\\&=\sum_{i=1}^{t}\mu^{t-i}(tr(\Gamma\mathbf{y}_i\mathbf{y}_i^T)+tr(\mathbf{P}^T\Gamma\mathbf{P}\mathbf{x}_i\mathbf{x}_i^T))\\&\quad-2\sum_{i=1}^{t}\mu^{t-i}tr(\Gamma\mathbf{P}\mathbf{x}_i\mathbf{y}_i^T)\\&=tr(\Gamma\sum_{i=1}^{t}\mu^{t-i}\mathbf{y}_i\mathbf{y}_i^T)+tr(\mathbf{P}^T\Gamma\mathbf{P}\sum_{i=1}^{t}\mu^{t-i}\mathbf{x}_i\mathbf{x}_i^T)\\&\quad-2tr(\Gamma\mathbf{P}\sum_{i=1}^{t}\mu^{t-i}\mathbf{x}_i\mathbf{y}_i^T). \quad (10)\end{aligned}$$

Defining the matrix variables $\mathbf{C}_{t,\mathbf{YY}},\mathbf{C}_{t,\mathbf{XY}}$, and $\mathbf{C}_{t,\mathbf{XX}}$ as:

$$\mathbf{C}_{t,\mathbf{YY}}=\sum_{i=1}^{t}\mu^{t-i}\mathbf{y}_i\mathbf{y}_i^T \quad (11)$$
$$\mathbf{C}_{t,\mathbf{XY}}=\sum_{i=1}^{t}\mu^{t-i}\mathbf{x}_i\mathbf{y}_i^T \quad (12)$$
$$\mathbf{C}_{t,\mathbf{XX}}=\sum_{i=1}^{t}\mu^{t-i}\mathbf{x}_i\mathbf{x}_i^T \quad (13)$$

Substituting $\mathbf{C}_{t,\mathbf{YY}},\mathbf{C}_{t,\mathbf{XY}}$, and $\mathbf{C}_{t,\mathbf{XX}}$ into (10), the loss function (5) becomes (6). Based on the Property 1, (11), (12), and (13) can be expressed by (7), (8), and (9), respectively. ■

When a new data point $(\mathbf{x}_{t+1},\mathbf{y}_{t+1})$ arrives on round $t+1$, we first update the statistical variables $\mathbf{C}_{t+1,\mathbf{YY}}$, $\mathbf{C}_{t+1,\mathbf{XY}}$, $\mathbf{C}_{t+1,\mathbf{XX}}$ based on (7), (8), and (9) respectively, whose memory complexity is a constant: $O(m^2+md+d^2)$. After that, the prediction loss ℓ can be measured by (6), so it is no longer necessary to load all the training samples into memory or visit a training sample multiple times.

In summary, the loss function (5) has the following merits: First, it can dynamically learn the structure of the residual errors as the samples continuously arrive, and leverage the structure information to effectively measure the true distance between the predicted value and the ground truth. Second, the loss is measured based on all the seen samples not just the current sample, which can cut down on the influence of noise on model's update. Third, on each round, instead of loading all the samples into memory, the loss can be measured just relying on three defined statistical variables without information loss, as expressed in (6). Furthermore, by introducing the factor μ to weight the samples, MORES can fit in streaming data's evolvement well.

The third term: In order to restrain ϕ fluctuating drastically on each round, we hope that the divergence $\Delta_{\phi}(\Omega,\Omega_{t-1})$ is as small as possible. $\Delta_{\phi}(\Omega,\Omega_{t-1})$ is defined as:

$$\Delta_{\phi}(\Omega,\Omega_{t-1})=\phi(\Omega)-\phi(\Omega_{t-1})-tr(g(\Omega_{t-1})(\Omega-\Omega_{t-1})),$$

where ϕ is a real-valued strictly convex differentiable function on the parameter domain $\mathbb{R}^{m\times m}$. $g(\Omega_{t-1})=\nabla_{\Omega}\phi(\Omega)|_{\Omega_{t-1}}$. In this paper, we employ two matrix divergence metrics, *quantum relative entropy* and *LogDet* divergence, because of their good properties stated in [32].

(i) When $\phi(\Omega)=tr(\Omega\log\Omega-\Omega)$, the *quantum relative entropy* is defined as:

$$\Delta_{\phi}(\Omega,\Omega_{t-1})=tr(\Omega\log\Omega-\Omega\log\Omega_{t-1}-\Omega+\Omega_{t-1}),$$

where $\log\Omega = \mathbf{V}(\log\Lambda)\mathbf{V}^T$, and $(\log\Lambda)_{i,i} = \log(\Lambda_{i,i})$. Ω is a strictly positive definite matrix, and $\Omega = \mathbf{V}\Lambda\mathbf{V}^T$.

(ii) When $\phi(\Omega) = -\text{logdet}(\Omega)$, *LogDet* is defined as:

$$\Delta_\phi(\Omega, \Omega_{t-1}) = \log\frac{\det(\Omega_{t-1})}{\det(\Omega)} + tr(\Omega_{t-1}^{-1}\Omega) - m,$$

where $\det(\cdot)$ denotes the determinant of a matrix.

B. Optimization Procedure

The objective function (3) is not convex with respect to all variables, but it is convex with each variable when others are fixed. We adopt an alternating optimization strategy to solve (3), which can find local minima.

Optimizing $\mathbf{P}$, given Ω and Γ: When Ω and Γ are fixed, (3) is then unconstrained and convex. $\mathbf{P}$ can be obtained by minimizing the following objective function:

$$\min\ J_1(\mathbf{P}) = \|\mathbf{P} - \mathbf{P}_{t-1}\|_\Omega^2 + \alpha\ell(\mathbf{P}, \Gamma; \mathcal{S}_t) \qquad (14)$$

The necessary condition for the optimality is:

$$\begin{aligned}&\frac{\partial J_1(\mathbf{P})}{\partial \mathbf{P}} = 0\\ &\Rightarrow \Omega\mathbf{P} + \alpha\Gamma\mathbf{P}\mathbf{C}_{t,\mathbf{XX}} = \Omega\mathbf{P}_{t-1} + \alpha\Gamma\mathbf{C}_{t,\mathbf{XY}}^T\\ &\Rightarrow \text{vec}(\Omega\mathbf{P} + \alpha\Gamma\mathbf{P}\mathbf{C}_{t,\mathbf{XX}}) = \text{vec}(\Omega\mathbf{P}_{t-1} + \alpha\Gamma\mathbf{C}_{t,\mathbf{XY}}^T)\end{aligned} \qquad (15)$$

where $\text{vec}(\cdot)$ is an operator that reshapes a matrix of size $m\times n$ into a vector of size $mn \times 1$.

Noticing that $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$ and $\text{vec}(\mathbf{A} + \mathbf{B})$=$\text{vec}(\mathbf{A})$+$\text{vec}(\mathbf{B})$, (15) can become:

$$\begin{aligned}&(\mathbf{I}\otimes\Omega + \alpha\mathbf{C}_{t,\mathbf{XX}}\otimes\Gamma)\text{vec}(\mathbf{P}) = \text{vec}(\Omega\mathbf{P}_{t-1} + \alpha\Gamma\mathbf{C}_{t,\mathbf{XY}}^T)\\ &\Rightarrow \text{vec}(\mathbf{P}) = (\mathbf{I}\otimes\Omega + \alpha\mathbf{C}_{t,\mathbf{XX}}\otimes\Gamma)^\dagger\text{vec}(\Omega\mathbf{P}_{t-1} + \alpha\Gamma\mathbf{C}_{t,\mathbf{XY}}^T)\end{aligned}$$

where $\otimes$ denotes Kronecker product. $(\mathbf{I}\otimes\Omega + \alpha\mathbf{C}_{t,\mathbf{XX}}\otimes\Gamma)^\dagger$ is the Moore-Penrose pseudoinverse of the matrix $(\mathbf{I}\otimes\Omega + \alpha\mathbf{C}_{t,\mathbf{XX}}\otimes\Gamma)$. We know that when a matrix is invertible, its pseudoinverse is equal to its inverse. After obtaining $\text{vec}(\mathbf{P})$, $\mathbf{P}$ can be easily found by the operator $\text{unvec}(\cdot)$ that reshapes a vector into a matrix.

Optimizing Ω, given $\mathbf{P}$ and Γ: When $\mathbf{P}$ and Γ are fixed, solving Ω becomes a convex optimization problem as:

$$\min_{\Omega\succeq 0} J_2(\Omega) = \|\mathbf{P} - \mathbf{P}_{t-1}\|_\Omega^2 + \beta\Delta_\phi(\Omega, \Omega_{t-1}). \qquad (16)$$

(i) When $\Delta_\phi(\cdot)$ is *quantum relative entropy*, (16) becomes

$$\begin{aligned}\min_{\Omega\succ 0} J_2(\Omega) = &tr((\mathbf{P} - \mathbf{P}_{t-1})^T\Omega(\mathbf{P} - \mathbf{P}_{t-1}))\\ &+\beta tr(\Omega\log\Omega - \Omega\log\Omega_{t-1} - \Omega + \Omega_{t-1}).\end{aligned} \qquad (17)$$

Taking the derivative of $J_2(\Omega)$ with respect to Ω, and setting it to zero, we can obtain a closed form of Ω:

$$\Omega = \exp(\log\Omega_{t-1} - \frac{1}{\beta}(\mathbf{P} - \mathbf{P}_{t-1})(\mathbf{P} - \mathbf{P}_{t-1})^T),$$

where $\exp(\cdot)$ is the matrix exponential. According to [32], it is easily inferred that Ω is positive definite.

Algorithm 1 Multiple-Output Regression for Streaming Data(MORES)

Input: Data streams $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \cdots\}$ that arrive one sample each time;
Parameters: α, β, η, and the forgetting factor μ;
Initialize $\mathbf{P}_0 = \mathbf{0}_{d\times m}$, $\mathbf{C}_{0,\mathbf{XX}} = \mathbf{0}_{d\times d}$, $\mathbf{C}_{0,\mathbf{XY}} = \mathbf{0}_{d\times m}$, $\mathbf{C}_{0,\mathbf{YY}} = \mathbf{0}_{m\times m}$, and $\Omega_0 = \Gamma_0 = \frac{1}{m}\times\mathbf{I}_{m\times m}$;
Method

```
for t = 1, 2, ···
    C_{t,YY} = μC_{t-1,YY} + y_t y_t^T;
    C_{t,XY} = μC_{t-1,XY} + x_t y_t^T;
    C_{t,XX} = μC_{t-1,XX} + x_t x_t^T;
    if t < T_min
        Find the optimal P* by solving (14);
        P_t ← P*; Ω_t ← Ω_0; Γ_t ← Γ_0;
    else
        Find the local optimal P*, Ω*, and Γ* by solving (14),
          (17) or (18), and (19), respectively;
        P_t ← P*; Ω_t ← Ω*; Γ_t ← Γ*;
    end
end
```

end Method
Output: Regression coefficient matrix $\mathbf{P}_t \in \mathbb{R}^{m\times d}$.

(ii) When using *LogDet* to represent $\Delta_\phi(\cdot)$, (16) becomes

$$\begin{aligned}\min_{\Omega\succeq 0} J_3(\Omega) = &tr((\mathbf{P} - \mathbf{P}_{t-1})^T\Omega(\mathbf{P} - \mathbf{P}_{t-1}))\\ &+\beta(\log\frac{\det(\Omega_{t-1})}{\det(\Omega)} + tr(\Omega_{t-1}^{-1}\Omega) - m).\end{aligned} \qquad (18)$$

Similarly, solving $\frac{\partial J_3(\Omega)}{\partial(\Omega)} = 0$, we have

$$\Omega = (\Omega_{t-1}^{-1} + \frac{1}{\beta}(\mathbf{P} - \mathbf{P}_{t-1})(\mathbf{P} - \mathbf{P}_{t-1})^T)^{-1}.$$

For simplicity, we use the matrix $\mathbf{M}$ to represent $(\mathbf{P} - \mathbf{P}_{t-1})(\mathbf{P} - \mathbf{P}_{t-1})^T$. To guarantee Ω to be positive semi-definite, the matrix $\mathbf{M}$ should be positive semi-definite. We adopt the following strategy to make $\mathbf{M}$ positive semi-definite: first, we calculate its spectral decomposition: $\mathbf{M} = \mathbf{U}\Lambda\mathbf{U}^T$. Then $\mathbf{M}$ is updated by thresholding the corresponding eigenvalues as: $\mathbf{M} = \mathbf{U}\max(\Lambda, 0)\mathbf{U}^T$.

Optimizing Γ, given $\mathbf{P}$ and Ω: When $\mathbf{P}$ and Ω are fixed, Γ can be obtained by solving the following convex optimization problem:

$$\min_{\Gamma\succeq 0} J_4(\Gamma) = \alpha\ell(\Gamma; \mathbf{P}, \mathcal{S}) - \eta\log|\Gamma|. \qquad (19)$$

The necessary condition for the optimality is $\frac{\partial J_4(\Gamma)}{\partial\Gamma} = 0$. Therefore, we obtain the following closed form solution:

$$\Gamma = \frac{\eta}{\alpha}(\mathbf{C}_{t,\mathbf{YY}} - \mathbf{C}_{t,\mathbf{XY}}^T\mathbf{P}^T - \mathbf{P}\mathbf{C}_{t,\mathbf{XY}} + \mathbf{P}\mathbf{C}_{t,\mathbf{XX}}\mathbf{P}^T)^{-1}.$$

After obtaining the solution Γ, we employ the same strategy to make Γ positive semi-definite as in (18).

Finally, we summarize the procedure of MORES in Algorithm 1. During the preliminary stage of online update, the regression coefficient matrix $\mathbf{P}$ are not well formed, and poor initial estimations of $\mathbf{P}$ may result in poor estimations of Ω and Γ. To avoid this case, we do not update Ω or Γ until the round t is greater than or equal to a given threshold T_{min}.

TABLE I
MAEs OF DIFFERENT METHODS ON THE BARRETT WAM DATASET. THE LAST COLUMN IS THE AVERAGE MAE. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONTS.

Method	1st DOF	2nd DOF	3rd DOF	4th DOF	5th DOF	6th DOF	7th DOF	Average
PA-I	1.196	0.753	0.350	0.382	0.101	0.089	0.045	0.417
PA-II	1.179	0.772	0.332	0.391	0.095	0.084	0.043	0.414
iS-PLS	1.026	6.510	0.398	2.162	0.087	0.089	0.036	1.473
SOMOR	1.176	0.756	0.339	0.384	0.098	0.086	0.044	0.412
MORES-LD	0.547	**0.375**	**0.176**	0.184	0.047	**0.047**	**0.025**	**0.200**
MORES-QE	**0.526**	0.391	0.182	**0.180**	**0.044**	0.049	0.026	**0.200**

C. Time Complexity Analysis

In Algorithm 1, the most time-consuming part of MORES is to update $\mathbf{P}_t$, Ω_t, and Γ_t, and the time cost of other parts can be ignored. Here we focus on analyzing the complexity of the case where t is greater than or equal to T_{min}. For updating $\mathbf{P}_t$, the complexity is $O(m^2d^2+d^3m^3) = O(d^3m^3)$. Updating Γ needs $O(m^3 + d^2m)$. In order to update Ω_t, we utilize two kinds of divergence metric(*quantum relative entropy* and *LogDet* divergence) in this paper. When using *quantum relative entropy* as the divergence metric, updating Ω_t involves the spectral decomposition, whose complexity is the same as the eigen-decomposition, typically, $O(m^3)$ in practice. Therefore, updating Ω_t needs $O(m^3 + dm^2)$. When using *LogDet*, it also costs $O(m^3 + dm^2)$ to update Ω_t. Therefore, the total time complexity of MORES is of order $O(d^3m^3 + m^3 + d^2m + m^3 + dm^2) = O(d^3m^3)$, which is dominated by the update of $\mathbf{P}_t$.

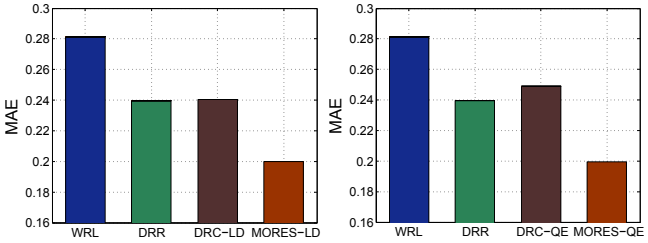
III. EXPERIMENTS

To evaluate the performance of MORES, we perform the experiments on three real-world datasets: the Barrett WAM dataset [33], the stock price dataset, and the weather dataset [34].

A. Experimental Setups

We compare our method with iS-PLS [19] that is the most relevant work to ours. We also compare with two variants of PA algorithm in [26] called PA-I and PA-II, which are two classical online learning approaches for single regression tasks. In the experiment, we use PA to train a regression model for each output. We name our simple formulation of online multiple-output regression proposed at the beginning of Sect. 2 as SOMOR for short. In addition, our proposed MORES has two variants: one using *quantum relative entropy* to measure the divergence of matrices, which is named as MORES-QE. The other using *LogDet* is called MORES-LD.

There are some parameters to be set in advance. In order to lower the cost of parameter tuning, we tune the parameters α, β, and η in our algorithms by a grid-search strategy from $\{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$ on the Concrete Slump dataset [35], and choose the optimal parameters to directly apply to the above three datasets. The threshold T_{min} is set to 50, and the forgetting factor μ is set to 0.8 throughout the experiments, unless otherwise stated. To fairly conduct experimental comparisons, the parameters in the other methods are also searched from $\{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$.



(a) MORES-LD (b) MORES-QE

Fig. 1. Verify the effectiveness of the components of our algorithms on the Barrett WAM dataset.

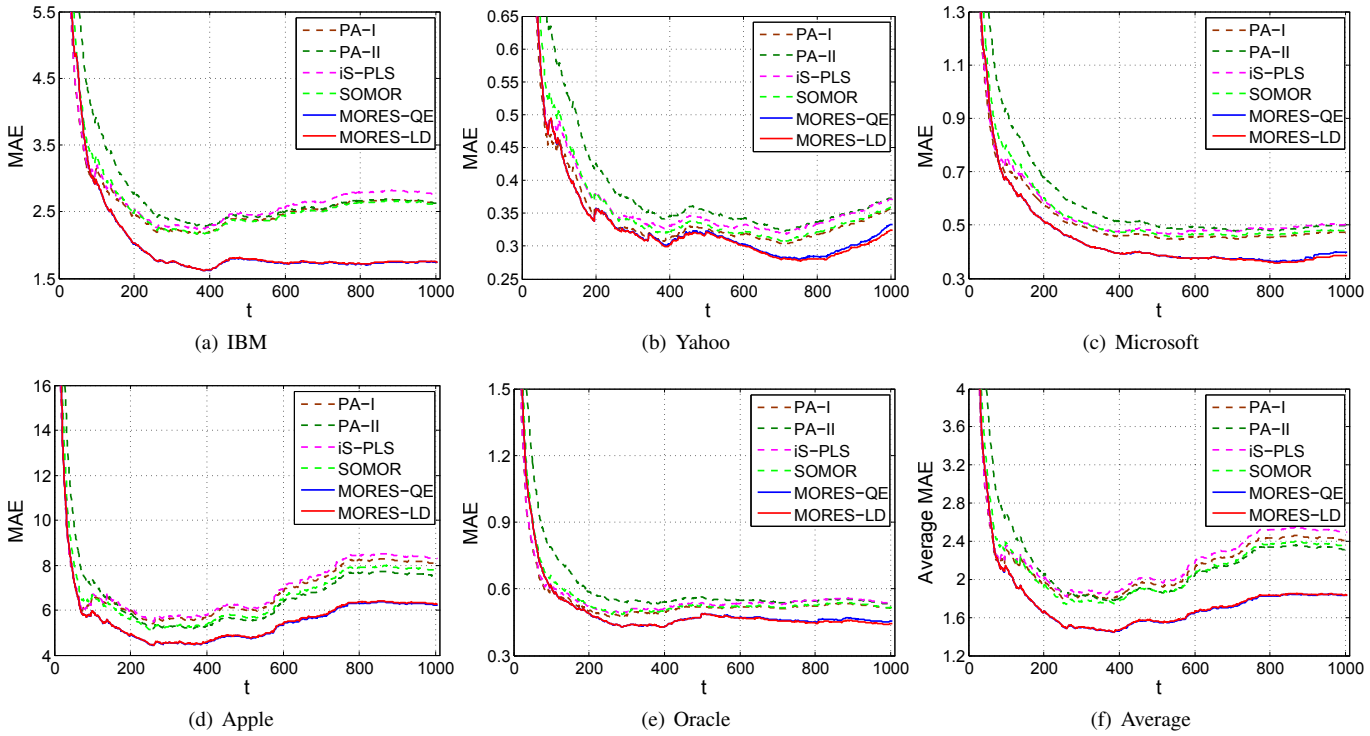
In the experimental study, we focus on the accuracy of online regression prediction to evaluate our models' quality. The popular metric, Mean Absolute Error (MAE), is used to measure the prediction quality. MAE is defined as: MAE $= \frac{1}{t}\sum_{i=1}^{t}|y_i - \widehat{y}_i|$, where $\widehat{y}_i$ denotes the estimated values of the i-th instance, and y_i is the true response values.

B. Robot Inverse Dynamics

We first study the problem of online learning the inverse dynamics of a 7 degrees of freedom of robotic arms on the Barrett WAM dataset. This dataset consists of a total of 16,200 samples, where each sample is represented by 21 input features, corresponding to seven joint positions, seven joint velocities and seven joint accelerations. Seven joint torques for the seven degrees of freedom (DOF) are used as the outputs.

We summarize the results of different methods in Table I. For each output, MORES-QE and MORES-LD attain better prediction performances than all the other methods. Meanwhile, they both achieve 51.5% relative error deduction in terms of the average MAE over SOMOR. The performance of MORES-QE is comparable to that of MORES-LD. It indicates that the *quantum relative entropy* divergence metric has similar effect on the prediction performance with the *LogDet* divergence metric. In addition, iS-PLS has a poor performance on the second and the fourth outputs. The reason may be that iS-PLS tries to find a low-dimensional subspace, where the covariance between the inputs and the outputs is maximized, while the found subspace does not preserve the structures of both the second and the fourth outputs well.

In our method, there are three main components: dynamically learning the structures of both the regression coefficient matrix and the residual error vector, and introducing a forgetting factor to measure the prediction error in an incremental fashion. We verify the effectiveness of the three components



(a) IBM (b) Yahoo (c) Microsoft

(d) Apple (e) Oracle (f) Average

Fig. 2. MAE as a function of sequence length t on different companies. In (a), (d), and (f), the curves of MORES-QE are almost overlapped with those of MORES-LD, respectively.

TABLE II
MAEs OF OUR METHODS WITH DIFFERENT VALUES OF THE FORGETTING FACTOR μ.

μ	0	0.6	0.7	0.8	0.9	1
MORES-LD	0.416	0.207	**0.194**	0.200	0.278	0.591
MORES-QE	0.416	0.278	0.254	0.200	**0.197**	0.590

in terms of the average MAE of all the outputs on this dataset. The experimental setting is as follows: When both Ω and Γ are set to the identity matrix $\mathbf{I}$ in (3), the model is updated Without Relationship Learning on each round. We name it WRL for short. When Ω is set to $\mathbf{I}$ and Γ is updated on each round, it indicates that we only Dynamically learn the Relationships of the Residual errors in the process of model's update. We call it DRR. When Ω is updated on each round and Γ is set to $\mathbf{I}$, we only Dynamically learn the Relationships of the regression Coefficient matrix. Because we utilize *quantum relative entropy* and *LogDet* in updating Ω, we call them DRC-QE and DRC-LD, respectively. The results are shown in Fig. 1. Taking Fig. 1(a) as an example, both DRR and DRC-LD are better than WRL. It shows that dynamically learning the structures of the regression coefficient matrix and the residual errors are both beneficial to online regression. MORES-LD achieves the best performance. This indicates that the combination of the two components is effective for online multiple-output regression.

In order to verify the effectiveness of the forgetting factor μ, we conduct the experiments with different values of μ. Table II lists the results. Based on previous analysis in Sect. 2, we know no historical data is utilized to update the model on each round if $\mu = 0$. When $\mu = 1$, all the samples have the same weight for calculating the prediction loss. As can be seen in Table II, when $0.6 \leq \mu \leq 0.9$, the performance is improved compared to that of $\mu = 0$. This shows that taking advantage of the historical samples is good for online regression. Moreover, we observe that when $\mu = 0.7$ and $\mu = 0.9$, MORES-LD and MORES-QE respectively achieve the best performances. It implies that the data in this dataset is indeed evolving. By introducing μ to set higher weights on the newer training samples, the model can adapt to the data stream's evolvement, and the prediction accuracy can be improved.

C. Stock Price Prediction

Following previous studies in [36] and [20], we also apply our algorithms to the stock data of companies for price prediction. We choose the daily stock price data of five companies including IBM, Yahoo, Microsoft, Apple, and Oracle in the period from 2010 to 2013. The learned model can predict the stock prices in the future by using the stock prices in the past as inputs. We use the autoregressive 1, aka AR(1) model $\mathbf{y}_{t+1} = \mathbf{P}_t\mathbf{y}_t + \epsilon_t$, where $\mathbf{y}_{t+1}$ represents the real stock prices of the five companies at time $t+1$, and $\mathbf{P}_t$ denotes the learned regression coefficient matrix at time t.

The experimental results are reported in Table III. MORES-LD and MORES-QE achieve better performances compared to the other methods. Taking MORES-LD and PA-II as an example, MORES-LD gains 33.6%, 13.1%, 22.3%, 16.6%, and 17.0% relative accuracy improvement over PA-II for IBM, Yahoo, Microsoft, Apple, and Oracle, respectively. Meanwhile, MORES-LD obtains 20.6% relative improvement in terms

TABLE III
MAEs OF DIFFERENT METHODS ON THE STOCK PRICE DATASET. THE LAST COLUMN IS THE AVERAGE MAE. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONTS.

Method	IBM	Yahoo	Microsoft	Apple	Oracle	Average
PA-I	2.626	0.356	0.473	8.082	0.515	2.410
PA-II	2.630	0.373	0.497	7.539	0.534	2.315
iS-PLS	2.762	0.371	0.503	8.293	0.538	2.493
SOMOR	2.606	0.358	0.481	7.783	0.516	2.349
MORES-LD	1.746	**0.324**	**0.386**	6.286	**0.443**	1.837
MORES-QE	**1.743**	0.333	0.399	**6.240**	0.454	**1.834**

of the average MAE over PA-II. These results show that dynamically learning the structures of both the regression coefficient matrix and the residual error vector, as well as utilizing the historical data in an appropriate way, is good for online multiple-output regression.

We also investigate the performances of different methods as a function of sequence length (t). At the end of each online round, we calculate the MAE for each output attained so far. Fig. 2 shows the results. The performances of MORES-LD and MORES-QE are superior to those of the other methods, especially when the sequence length t is larger. In addition, the MAE curves of Fig. 2 (a), (b), and (d) rise after falling as t increases. This is because the stock price is severely evolving at the inflection point, such that the current model can not predict the future price well. Although the data is evolving, our algorithms are still better than the other methods under this circumstance. From Fig. 2 (a), we can see MORES-LD and MORES-QE can quickly adjust the model to fit in the data's evolvement.

D. Weather Forecast

We also evaluate our algorithms on the weather dataset for weather forecast [34]. This dataset consists of wind speed, wind direction, barometric pressure, water depth, maximum gust, maximum wave height, air temperature, water temperature and average wave height, which is collected every five minutes by a sensor network located on the south coast of England. One and a half years' data containing 143,034 samples are used in the experiments. The first five variables are used as the predictors, and the rest are the response variables.

The experimental results are reported in Table IV. MORES-LD and MORES-QE obtain better prediction performances than the other methods for all the response variables. Meanwhile, the results of our algorithms are superior to those of the other methods in terms of the average MAE.

We further test all the methods with different model update frequencies. The experimental setting is as follows: The model is updated when accumulatively receiving N (=1, ..., 10) training data points, while the test is still performed on all the data points. As reported in Table V, the prediction accuracies of all the methods are gradually reduced as N increases. Moreover, for various values of N, MORES-LD and MORES-QE perform better than the other methods because of dynamic learning of the output structures and the utilization of the historical data. In addition, we can see that the performance of MORES-LD with $N = 10$ is comparable to that of PA-II with $N = 1$.

TABLE IV
MAEs OF DIFFERENT METHODS ON THE WEATHER DATASET. 'MWH', 'AT', 'WT', AND 'AWH' DENOTE MAXIMUM WAVE HEIGHT, AIR TEMPERATURE, WATER TEMPERATURE, AND AVERAGE WAVE HEIGHT, RESPECTIVELY. THE LAST COLUMN IS THE AVERAGE MAE. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONTS.

Method	MWH	AT	WT	AWH	Average
PA-I	0.771	0.242	0.177	0.007	0.299
PA-II	0.766	0.240	0.174	0.006	0.297
iS-PLS	0.784	0.653	0.721	0.007	0.541
SOMOR	0.772	0.239	0.173	0.006	0.298
MORES-LD	**0.671**	**0.145**	**0.034**	**0.005**	**0.214**
MORES-QE	0.672	**0.145**	**0.034**	**0.005**	**0.214**

TABLE V
MAEs OF DIFFERENT METHODS ON THE WEATHER DATASET WITH VARIOUS VALUES N. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONTS.

Method	N					
	1	2	4	6	8	10
PA-I	0.299	0.336	0.389	0.430	0.470	0.502
PA-II	0.297	0.333	0.385	0.423	0.462	0.491
iS-PLS	0.541	0.634	0.759	0.850	0.924	0.988
SOMOR	0.298	0.334	0.386	0.424	0.462	0.490
MORES-LD	**0.214**	**0.226**	**0.247**	**0.274**	**0.291**	**0.297**
MORES-QE	**0.214**	0.227	0.248	0.279	0.399	0.432

E. Sensitivity Analysis

We also study the sensitivity of parameters α, β, and η in our algorithm on the larger dataset, the weather dataset. As shown in Fig. 3, with the fixed η, our method is not sensitive to α and β with wide ranges. As for parameter η, when α and β are fixed, the performance is gradually improved as η increases. When $\eta > 0.1$, the performance is gradually degraded with η increasing. When η is set to 0.1, the performance is the best.

F. Efficiency

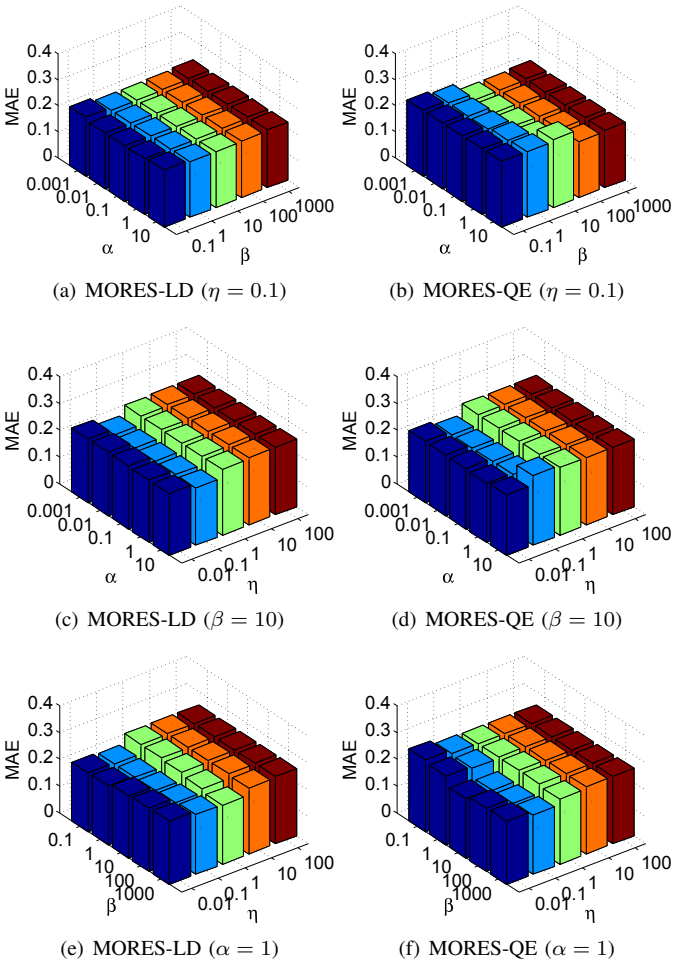
We test the update speeds of our algorithms on the above three datasets. The experiments are conducted on a desktop with Inter(R) Core(TM) i7-3770 CPU, and MORES are implemented using MATLAB R2012b 64bit edition without parallel operation. The update speeds of our algorithms are reported in Table VI. On the weather dataset, MORES-QE achieves 3566 updates per second. And on the Inverse Dynamics dataset having the largest dimensions, our algorithms perform more than 1000 updates per second. If we apply some parallel implementations or use more efficient programming language, the update speed of MORES can be further improved.

TABLE VI
UPDATE SPEED OF MORES (UNIT: # SAMPLES PER SECOND)

Datasets	Inverse Dynamics	Stock	Weather
MORES-LD	1036	2532	2785
MORES-QE	1160	3284	3566

IV. RELATED WORK

In this section, we review the related works from two aspects: online single-output regression and batch multiple-output regression.



(a) MORES-LD ($\eta = 0.1$) (b) MORES-QE ($\eta = 0.1$)

(c) MORES-LD ($\beta = 10$) (d) MORES-QE ($\beta = 10$)

(e) MORES-LD ($\alpha = 1$) (f) MORES-QE ($\alpha = 1$)

Fig. 3. MORES-LD and MORES-QE with different α, β, and η on the weather dataset.

Online single-output regression: [25] presented an online version of support vector regression algorithm, called (AOSVR). AOSVR classified all training samples into three distinct auxiliary sets according to the KKT conditions that define the optimal solution. After that, the rules for recursively updating the optimal solution were devised based on this classification. [26] proposed a margin based online regression algorithm, called passive-aggressive (PA). PA incrementally updated the model by formalizing the trade-off between the amount of progress made on each round and the amount of information retained from previous rounds. [27] proposed an incremental support vector regression algorithm, which evolved a pool of online SVR experts and learned to trade by dynamically weighting the experts' opinions.

Batch multiple-output regression: Many batch multiple-out-put regression algorithms have been proposed, which tried to mine the structure among outputs. Rothman et al. [19] presented MRCE, which jointly learned the output structure in the form of the noise covariance matrix and the regression coefficients for predicting each output. Sohn and Kim [20] designed an algorithm to simultaneously estimate the regression coefficient vector for each output along with the covariance structure of the outputs with a shared sparsity assumption on the regression coefficient vectors. Rai et al. [21] proposed an approach that leveraged the covariance structure of the regression coefficient matrix and the conditional covariance structure of the outputs for learning the model parameters. [22] proposed a tree-guided group lasso, or tree lasso, that directly combined statistical strength across multiple related outputs. They estimated the structured sparsity under multi-output regression by employing a novel penalty function constructed from the tree. Since these methods are trained in the batch mode, they are not suitable for online multiple-output prediction.

V. Conclusions

In this paper, we proposed a novel online multiple-output regression method for streaming data. The proposed method can simultaneously and dynamically learn the structures of both the regression coefficients and the residual errors, and leverage the learned structure information to continuously update the model. Meanwhile, we accumulated the prediction error on all the seen samples in an incremental way without information loss, and introduced a forgetting factor to weight the samples so as to fit in data streams' evolvement. The experiments were conducted on three real-world datasets, and the experimental results demonstrated the effectiveness and efficiency of the proposed method.

References

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Second Edition: Diane Cerra, 2006.

[2] P. Domingos and G. Hulten, "Mining high-speed data streams," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2000, pp. 71–80.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for on-demand classification of evolving data streams," *IEEE Trans. on Knowledge and Data Engineering*, vol. 18, no. 5, pp. 577–589, 2006.

[4] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn ++.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. on Neural Networks*, vol. 20, no. 1, pp. 152–168, 2009.

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *International Conference on Very Large Data Bases (VLDB)*, 2003, pp. 81–92.

[6] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. ACM, 2007, pp. 133–142.

[7] C.-D. Wang, J.-H. Lai, D. Huang, and W.-S. Zheng, "Svstream: A support vector-based algorithm for clustering data streams," *IEEE Trans. on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1410–1424, 2013.

[8] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from data streams," in *IEEE International Conference on Data Mining (ICDM)*, 2007, pp. 757–762.

[9] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng, "Unbiased online active learning in data streams," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 195–203.

[10] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data." *IEEE Trans. on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 27–39, 2014.

[11] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1178–1192, 2013.

[12] J. Wang, P. Zhao, S. Hoi, and R. Jin, "Online feature selection and its applications," *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 698–710, 2014.

[13] A. Saha, P. Rai, H. Daumé III, and S. Venkatasubramanian, "Online learning of multiple tasks and their relationships," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 643–651.

[14] O. Dekel, P. M. Long, and Y. Singer, "Online learning of multiple tasks with a shared loss." *Journal of Machine Learning Research*, vol. 8, no. 10, 2007.

[15] S.-S. Ho and H. Wechsler, "A martingale framework for detecting changes in data streams by testing exchangeability," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2113–2127, 2010.

[16] M. Yamada, A. Kimura, F. Naya, and H. Sawada, "Change-point detection with feature selection in high-dimensional time-series data," in *International joint conference on Artificial Intelligence (AAAI)*. AAAI Press, 2013, pp. 1827–1833.

[17] S. Kim, K.-A. Sohn, and E. P. Xing, "A multivariate regression approach to association analysis of a quantitative trait network," *Bioinformatics*, vol. 25, no. 12, pp. i204–i212, 2009.

[18] M. Sánchez-Fernández, M. de Prado-Cumplido, J. Arenas-García, and F. Pérez-Cruz, "Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2298–2307, 2004.

[19] A. J. Rothman, E. Levina, and J. Zhu, "Sparse multivariate regression with covariance estimation," *Journal of Computational and Graphical Statistics*, vol. 19, no. 4, pp. 947–962, 2010.

[20] K.-A. Sohn and S. Kim, "Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 1081–1089.

[21] P. Rai, A. Kumar, and H. Daumé III, "Simultaneously leveraging output and task structures for multiple-output regression." in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 3194–3202.

[22] S. Kim, E. P. Xing *et al.*, "Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eqtl mapping," *The Annals of Applied Statistics*, vol. 6, no. 3, pp. 1095–1117, 2012.

[23] M. Gonen and S. Kaski, "Kernelized bayesian matrix factorization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 2047–2060, 2014.

[24] H. Liu, L. Wang, and T. Zhao, "Multivariate regression with calibration," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 127–135.

[25] J. Ma, J. Theiler, and S. Perkins, "Accurate on-line support vector regression," *Neural Computation*, vol. 15, no. 11, pp. 2683–2703, 2003.

[26] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[27] G. Montana and F. Parrella, "Learning to trade with incremental support vector regression experts," in *Hybrid Artificial Intelligence Systems*. Springer, 2008, pp. 591–598.

[28] B. McWilliams and G. Montana, "Sparse partial least squares regression for on-line variable selection with multivariate data streams," *Statistical Analysis and Data Mining*, vol. 3, no. 3, pp. 170–193, 2010.

[29] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132, no. 1, pp. 1–63, 1997.

[30] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010, pp. 733–742.

[31] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems (NIPS)*, 2005, pp. 1473–1480.

[32] K. Tsuda, G. Rätsch, M. K. Warmuth, and Y. Singer, "Matrix exponentiated gradient updates for on-line learning and bregman projection." *Journal of Machine Learning Research*, vol. 6, no. 6, pp. 995–1018, 2005.

[33] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," no. 15, pp. 2015–2034, 2009.

[34] M. A. Alvarez and N. D. Lawrence, "Sparse convolved gaussian processes for multi-output regression." in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 57–64.

[35] I. Yeh *et al.*, "Modeling slump flow of concrete using second-order regressions and artificial neural networks," *Cement and Concrete Composites*, vol. 29, no. 6, pp. 474–480, 2007.

[36] A. C. Lozano, H. Jiang, and X. Deng, "Robust sparse estimation of multiresponse regression and inverse covariance matrix via the l2 distance," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2013, pp. 293–301.