Multi-step Time Series Prediction in Complex Instrumented Domains

Amit Dhurandhar
IBM T.J. Watson, Yorktown Heights, NY, USA, 10598
adhuran@us.ibm.com

***Abstract*—Time series prediction algorithms are widely used for applications such as demand forecasting, weather forecasting and many others to make well informed decisions. In this paper, we compare the most prevalent of these methods as well as suggest our own, where the time series are generated from highly complex industrial processes. These time series are non-stationary and the relationships between the various time series vary with time. Given a set of time series from an industrial process, the challenge is to keep predicting a chosen one as far ahead as possible, with the knowledge of the other time series at those instants in time. This scenario occurs, since the chosen time series is usually very expensive to measure or extremely difficult to obtain compared to the rest. Our studies on real data suggest, that our method is substantially more robust to predicting multiple steps ahead than the existing methods in these complex domains.**

***Keywords*-time series, non-stationary**

I. Introduction

Time series forecasting has been an active research area in various disciplines such as econometrics [1], [2], [3], [4] and machine learning [5], [6], [7], [8], [9] for decades. The reason being that effective solutions to this problem have wide spread use in numerous domains such as business intelligence, weather forecasting, stock market prediction etc. For example, demand forecasting models predict the quantity of an item that will be sold the following week given the sales up until the current week. In the stock market, given the stock prices up until today one can predict the stock prices for tomorrow. Using a weather forecasting engine one can predict the weather for the next day or may be for the entire week. The farther we predict however, the more likely we are to err. In any case, in all these applications predicting the next time step which may be the next day or the next week is still quite useful from a strategic point of view or a decision making point of view. When considering complex industrial processes however, this may not hold.

In complex instrumented domains certain parameters may be extremely expensive to obtain as compared to others and hence, we want to measure these parameters as infrequently as possible. From a time series forecasting perspective this amounts to predicting these parameters as far ahead as possible using information from the other less critical parameters that may be measured at every point in time. For example, in the electronic chip manufacturing industry measuring the deposition rate of nitrogen dioxide on a wafer is expensive, however, parameters such as temperature, pressure etc. can be readily measured without much overhead. Another example, is the oil production industry, where accurately predicting daily production can save companies loads of money in comparison with the alternative, which involves launching expensive production logging campaigns.

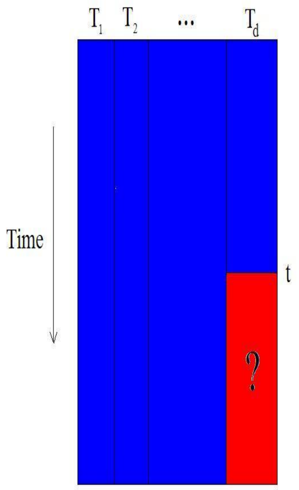

Figure 1. The red region with the question mark is to be predicted given the information in the blue region i.e. time series $T_1, T_2, ..., T_{d-1}$ at every instant and T_d till time t.

In this paper, we present a methodology that outperforms other state-of-the-art methods for multi-step time series prediction in non-stationary domains. The critical aspects of the methodology include a novel way of identifying distinct regimes in the data and if needed, a unique way of increasing the dynamic range of the predictions.

The rest of the paper is organized as follows. In Section II, we provide a formal definition of the problem we are trying to solve. In Section III, we review prior work that is related with time series forecasting. In Section IV, we describe our methodology in detail. In Section V, we compare our model with other state-of-the-art models on real industrial datasets. We then discuss directions of future research and conclude in section VI.

II. Problem Definition

Formally, the problem we are tackling can be stated as follows: *With d time series T_1, T_2, ...,T_d we want to predict T_d as far ahead as possible from some time instant t, given the other $d-1$ time series and values of T_d up until time t.*

An illustration of this is shown in figure 1, where the region with the question mark is to be predicted given the rest of the information.

III. Related Work

Time series forecasting models can primarily be partitioned into two groups namely; Auto-correlation models and Markovian models where the states may be explicit or as in most cases hidden.

Auto-correlation models: Given a time series $x_1, x_2, ..., x_t$ these models try to predict the next observation x_{t+1}. There are many models that fall under this category and are based on different assumptions. The Moving average model [10] is one of the simplest in this category which computes the average of the past m observations and reports this as an estimate of the current value. Exponential smoothing [10] is a little more sophisticated in that it exponentially weights down observations that are farther away in time and reports the result as an estimate. The more sophisticated models in this category are the auto-regressive models and combinations of auto-regressive and moving average models. These models compute the current value in the time series as a function of a finite past along with some white noise. Auto regressive moving average model (ARMA) and Auto regressive integrated moving average model (ARIMA) [11] are the most common examples in this category. Other less common but effective models in this category are the phase space reconstruction methods which assume that the time series of interest is a one dimensional projection of a system whose true phase space is higher dimensional [12]. These methods try to reconstruct the original phase space and predict current values of the time series by building statistical models in this reconstructed space. There are extensions of most of these models to incorporate exogenous variables as a linear combination.

Markovian models: These models learn a stationary distribution over a predefined or automatically deciphered state space. If the state space is not defined, Hidden markov models [13] and their extensions [5] decipher the hidden states. Each of the states in these models corresponds to a unique regression function learned from the training data. There are variants of these models which learn the state transitions as a function of the exogenous variables or input vector [2], however the transitions learned are still stationary in time.

Given the challenges we face in the problem we are trying to solve each of these groups suffers from at least a major limitation. The auto-correlation models do not provide a natural or powerful way of partitioning the state space and modeling non-linear dependencies of exogenous variables, while the markovian models learn stationary distributions over the state space which may not be applicable in drifting systems when trying to predict multiple steps ahead. Taking these facts into consideration, we now decribe a novel approach which as we will see in the experimental section outperforms the aforementioned methods.

IV. Modeling

As we discussed in the previous section the Markovian models learn stationary state transition distributions, which is their achille's heel, however, the idea of modeling different (hidden) states and learning independent regressors is important for our problem. The reason for this is that the time series we consider usually have different regimes, which need to be deciphered for better modeling. We thus borrow this idea for our approach. However, the critical question now is, on what basis do we identify different states which correspond to different regimes in the data?

To answer the above question it is important to distinguish when individual time series change from what matters in terms of prediction. For example, consider two time series T_1 and T_2 where we want to predict T_2. Say the values for T_1 are $(1, 1, 1, 2, 2, 2)$ and the corresponding values for T_2 are $(2, 2, 2, 4, 4, 4)$. If we look at each of the time series in isolation there is a change at time step 4. However, if we were to fit a linear model the relationship between the two time series has remained unchanged i.e. the relationship is still $T_2 = 2T_1$. Hence, though the time series have changed in isolation, from a prediction point of view there is no change. We would thus want to conclude that our model should have 1 state rather than 2 since, from the point of view of prediction there is just a single regime. On the other hand if T_2 was $(2, 2, 2, 10, 10, 10)$, we would want to split the data into 2 regimes and our model to have 2 states.

This intuition is captured in algorithm 1, where different states are based on changes in relationship between pairs of input time series. The algorithm takes as input a minimum cluster size (δ), which specifies a lower bound on the amount of data in each regime and ϵ which is an error threshold that we will describe shortly. The idea is to first consider the entire dataset and fit linear regressors between all pairs of inputs and compute the cumulative mean squared error (CMSE). Then the dataset is split into 2 equal parts and linear regressors are fit in each part for all pairs of inputs with each reporting its corresponding CMSE. This process continues until a further split of a particular part leads to a part that is smaller than δ. Hence, we now have a binary tree-like structure where every node in this tree is associated with a CMSE value and a set of

Algorithm 1 Regression algorithm where we identify different states and learn a regression function for each state (Training phase).

```
Input: $d$ time series $T_1, T_2, ..., T_d$ of length $N$, where $T_d$ is the target time series, minimum cluster size $\delta$ and CMSE error threshold $\epsilon$.
Output: Regression functions predicting the target and pairwise least square coefficients for every pair of input time series corresponding to each state/regime.
Initialise currentSize = $N$, currentChildren = 1, Array States = [], h = 1 {h denotes depth of the tree}
while currentSize $\geq \delta$ do
  {Compute CMSE for each node in the tree.}
  for $i = 1$ to currentChildren do
    Perform a least squares fit between $T_j^{(i)}$ and $T_k^{(i)}$ $\forall j \in \{1, ..., d-2\}, k \in \{j+1, ..., d-1\}$ storing the corresponding $MSE_{jk}$ {$T_j^{(i)}$ corresponds to values in time series $T_j$ restricted to the $i^{th}$ child at depth $h$.}
    $CMSE^{(hi)} = \sum_{j=1,k=j+1}^{d-2,d-1} MSE_{jk}$
  end for
  $h = h + 1$
  $currentSize = \lfloor \frac{currentSize}{2} \rfloor$
  $currentChildren = 2currentChildren$
end while
$currentChildren = \frac{currentChildren}{2}$
count = 0
h=h-1
if h==1 then
  {Identifying states.}
  count = count + 1
  States[count] = 11
else
  for $i = 1$ to currentChildren do
    p = i
    for $j = h$ to 2 do
      if $CMSE^{(j\lceil \frac{p}{2} \rceil)} - CMSE^{(jp)} \geq \epsilon$ then
        count = count+1
        States[count] = jp
        break
      else if $j == 2$ then
        count = count + 1
        States[count] = 11
      end if
      $p = \lceil \frac{p}{2} \rceil$
    end for
  end for
end if
From the array States note the distinct states and learn regression functions for each state.
```

coefficients of the $\frac{(d-1)(d-2)}{2}$ linear regressors. It can be shown that when a particular part is split into 2 child parts there always exists a child part with mean squared error (MSE) for any pair of time series less than or equal to the parent part. In fact, a more general version of this statement is stated in the following lemma[1],

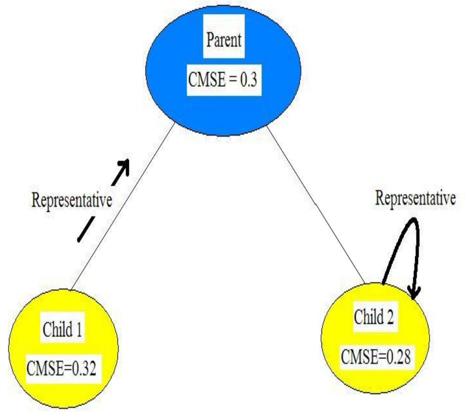

Figure 2. We see the CMSE of a parent part and its child parts. If the threshold ϵ is set to 0.01, the parent would be representative of child 1, while child 2 would represent a different state.

Lemma 1: Consider two real vectors T_1 and T_2 of length N, given by $t_1^{(1)}, t_1^{(2)}, ..., t_1^{(N)}$ and $t_2^{(1)}, t_2^{(2)}, ..., t_2^{(N)}$ respectively. Let m equisize sequential partitions of these vectors be denoted by $P_1, P_2, ..., P_m$. Each P_i $(1 \leq i \leq m)$ corresponds to a segment of m consecutive observations from the two time series given by $\{(t_1^{((i-1)m+1)}, t_1^{((i-1)m+2)}, ..., t_1^{((i)m)}), (t_2^{((i-1)m+1)}, t_2^{((i-1)m+2)}, ..., t_2^{((i)m)})\}$. Without loss of generality with T_1 as the input time series and T_2 as the ouput, if we build $m+1$ least square regression models, one for each of the m partitions P_i and one on the entire time series denoted by $R_1, R_2, ..., R_m$ and R_T respectively; then there always exists a partition P_i with the corresponding model R_i whose mean squared error is less than or equal to the mean squared error of R_T.

It is thus advisable to use a > 0 threshold to decide if to use the child part as a state or the parent part i.e. only if the reduction in CMSE from the parent to the child is greater than or equal to the threshold, do we use the child part to

[1]Proof in appendix.

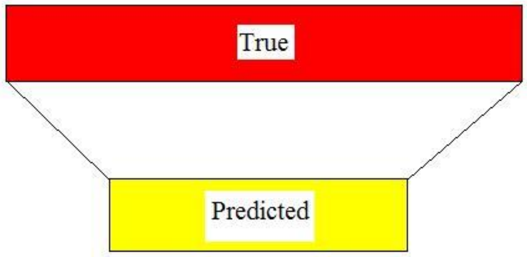

Figure 3. Predicted values have a smaller dynamic range than the true values and we want to ideally, map the predicted to the true range.

represent a state else we use the parent. This is shown figure 2. In algorithm 1, this threshold is given by ϵ.

After we have identified these different regimes/states we can train our favorite regression algorithm for each of these regimes. When testing, a test point is assigned to the regime whose coefficients when applied to the test inputs lead to the lowest CMSE.

Hence, our method to identify different states is based on the assumption that since, the various time series are generated from the same physical system, the time at which relationships change significantly between any pair of them should roughly correspond to changes in relationship between them and the target time series. Our method does *not* assume that this change is of the same amount as that of the involved time series or is in any way correlated to the respective time series in quantitative terms. It only assumes that the change is correlated in time with those time series.[2]

A. *Increasing Dynamic Range*

We just described a method to decipher regimes that are highly coherant from a predictive standpoint. However, it does occur sometimes that this is not sufficient to capture the dynamic range of the true values as shown in figure 3. There could be multiple reasons for this to happen but, one of the primary reasons is that the target may be drifting with a certain bias. What we mean by this is that up until the current time period we may have very few target values above k say, however this count drastically changes as we try to predict multiple steps ahead. Since, our regression algorithms try to minimize some cumulative error, a mediocre prediction on observations with true value above k does not increase the cumulative error by much in the current time period. However, as the proportion of such observations increases in the future, the impact on the cumulative error becomes significant.

[2]It is also important to note that one can perform feature selection after the states have been identified if needed.

Algorithm 2 Learning scaling factors to increase the dynamic range of predictions (Training phase).

```
Input: $d$ time series $T_1, T_2, ..., T_d$ of length $N$, where $T_d$ is the target time series, $l$ and $u$. $\{l$ and $u$ are limits to determine the 3 classes.$\}$
Output: Scaling factors for 3 classes.
Initialize $S_1 = 0, S_2 = 0, S_3 = 0, c_1 = 0, c_2 = 0, c_3 = 0$ $\{S_1, S_2, S_3$ are the scaling factors.$\}$
for $i = 1$ to $N$ do
    $\{$Create new time series $T_{d+1}$ of class labels depending on $l$, $u$ and $T_d$.$\}$if $t_d^{(i)} \leq l$ then $\{T_i = (t_i^{(1)}, t_i^{(2)}, ..., t_i^{(N)})$.$\}$
    $t_{d+1} = 1$
  else if $t_d^{(i)} \geq u$ then
    $t_{d+1} = 3$
  else
    $t_{d+1} = 2$
  end if
end for
Train the classification algorithm chosen, with $T_1, ..., T_{d-1}$ as input and $T_{d+1}$ as output.
Ouput a classification function $\zeta(.)$. $\{\zeta$ takes a datapoint as input and outputs a class label.$\}$
for $i = 1$ to $N$ do
  if $\zeta(x_i) == 1$ then
    $\{x_i = (t_1^{(i)}, ..., t_{d-1}^{(i)})\}$
    $S_1 = S_1 + \frac{t_d^{(i)}}{\tau(x_i)}$ $\{\tau(.)$ is the regression function built by algorithm 1.$\}$
    $c_1 = c_1 + 1$
  else if $\zeta(x_i) == 2$ then
    $S_2 = S_2 + \frac{t_d^{(i)}}{\tau(x_i)}$
    $c_2 = c_2 + 1$
  else
    $S_3 = S_3 + \frac{t_d^{(i)}}{\tau(x_i)}$
    $c_3 = c_3 + 1$
  end if
end for
if $c_1 \geq 1$ then
  $S_1 = \frac{S_1}{c_1}$
else
  $S_1 = 1$
end if
if $c_2 \geq 1$ then
  $S_2 = \frac{S_2}{c_2}$
else
  $S_2 = 1$
end if
if $c_3 \geq 1$ then
  $S_3 = \frac{S_3}{c_3}$
else
  $S_3 = 1$
end if
```

In such a scenario, we want to scale our predictions from the regression algorithm in the appropriate direction depending on if they are at the lower end of the spectrum, in the middle or at the higher end of the spectrum. Algorithm 2 describes such a technique. The manner in which one can decide what threshold to use for the lower end of the spectrum and analogously the higher end of the spectrum can be figured out from domain knowledge or by some standard model selection technique such as cross-validation. The algorithm outputs scaling factors for each class i.e. the amount by which the output from an regression algorithm should be scaled depending on the class in which the input lies. When testing, the output from the regression algorithm is multiplied to a convex combination of scaling factors, only if the probability of being in that class is high (> 0.5) for that input, where the weights in the convex combination are probabilities of the particular input belonging to each of the classes.

The intuition behind such an approach is that, the errors that the classification and regression algorithms make are hopefully as uncorrelated as possible and hence, even if one of them is accurate with the other being moderately erroneous on certain inputs the cumulative error should most likely reduce. Loosely speaking, one can think of this as a logical *or* i.e. if at least one of the methods does well (and the other is not great but reasonable) on each of the inputs the cumulative error should in all likelihood reduce.

A key aspect for the above methodology to work is to try to get as uncorrelated errors as possible. One way of maximizing the chances of this happening is by using models for regression and classification that cover different parts of the hypothesis space. In other words, using the same model such as decision trees or some other model for both classification and regression would not be recommended.

B. Time Complexity

We now analyze the time complexity of the regression methodology alone and regression along with classification. If N is the length of the time series, δ is the minimum cluster size, d is the number of time series and $O(\mathcal{R})$ is the time complexity of the regression algorithm used in each state, then the time complexity of algorithm 1 is $O((\frac{N}{\delta}d)^2 + \mathcal{R})$. If the $O(\mathcal{C})$ is the time complexity of the classification algorithm then the time complexity of algorithm 1 and algorithm 2 used together is $O((\frac{N}{\delta}d)^2 + \mathcal{R} + \mathcal{C})$. Usually δ isn't too small since, one needs enough data in each state to train a regression algorithm and hence, $\frac{N}{\delta} << N$.

V. Experiments

In this section we compare our methodology with other state-of-the-art methods. We do this by performing experiments on 3 real industrial dataset. On each of the datasets we perform a sequential 90%/10% train/test split (since, some of the datasets have less number of samples i.e. < 1000) and

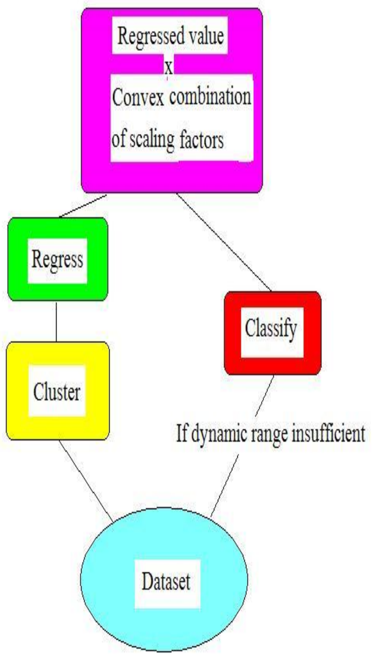

Figure 4. Overview of our methodology to predict in complex instrumented domains.

compare our method with the HMM based model from [5] and with the best results from ARIMA, ARMA, exponential smoothing and phase space reconstruction model (i.e. the auto-correlation models). We report the performance in terms of R^2 as a function of the number of steps ahead we need to predict. R^2 is a standard evaluation criteria in statistics, which measures how good a given model is compared to a baseline model that is the mean of target variable on the training set. Formally,

$$R^2 = 1 - \frac{MSE(M)}{MSE(\mu)}$$

where $MSE(M)$ is the mean squared error of your model M and $MSE(\mu)$ is the mean squared error of the mean of the target variable on the training set. Hence, a R^2 close to 1 and away (in the positive direction) from zero indicates that the model is performing much better than the baseline. While a negative R^2 indicates that the model is doing worse than the naive baseline. In our experiments, we indicate negative R^2 as zero since, using the model in this case is anyway useless.

A. Oil production dataset

Oil companies periodically launch production logging campaigns to get an idea of the overall performance as well

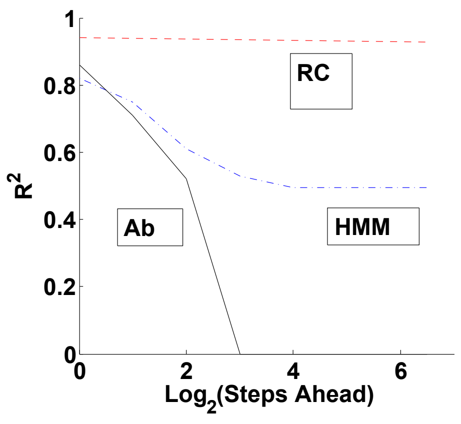

Figure 5. These are the results of the various methods on the oil production dataset. RC (acronym for regression+clustering) denotes our method described in algorithm 1. Ab is the best values taken from the previously mentioned auto-correlation methods and HMM is the hidden markov model based algorithm described in [5].

as to assess their individual performance at particular oil wells and reservoirs. These campaigns are usually expensive and laden with danger for the people involved in the campaign. Automated monitoring of oil production equipment is an efficient, risk free and economical alternative to the above solution.

The dataset we perform experiments on, is obtained from a major oil corporation. There are a total of 9 measures in the dataset. These measures are obtained from the sensors of a 3-stage separator which separates oil, water and gas. The 9 measures are composed of 2 measured levels of the oil water interface at each of the 3 stages and 3 overall measures. In particular, the measures are as follows:

1) 20LT_0017_mean (stage 1)
2) 20LT_0021_mean (stage 1)
3) 20LT_0081_mean (stage 2)
4) 20LT_0085_mean (stage 2)
5) 20LT_0116_mean (stage 3)
6) 20LT_0120_mean (stage 3)
7) Daily water in oil (Daily WIO)
8) Daily oil in water (Daily OIW)
9) Daily production

Our target measure is the last listed measure which is Daily production. The dataset size is 992. δ is set to 200 and ϵ is set to 0.1 in algorithm 1. We use a 2-layer radial basis neural network in each of the states. Since, algorithm 1 gives us sufficient dynamic range on the training set we do not use the classification part i.e. algorithm 2 in order to expand it.

From figure 5 we see that our method does significantly better than the other methods for all step sizes. RC and HMM plateau out to a positive R^2 while the auto-correlation methods do not since, the prior methods use also the 8 time series other than the target to make their predictions, which is not the case with the auto-correlation models. Hence, when we try to predict multiple steps ahead the current target value is not that informative and the auto-correlation models suffer since, they have no other source of information to rely on.

B. Semiconductor etching dataset

In a semi-condutor fabrication plant, etching is one of the important processes that takes place. The acronym for this process is EPI. This is a key step in chip manufacturing where a wafer etched out to deposit other material further downstream. To etch out a wafer a certain compound is deposited and it is crucial that this compound deposited in the right quantity. Hence, a key parameter in this process is deposition rate of that compound, which is expensive to measure as it requires sophisticated equipment.

The dataset we have has 2026 features where, deposition rate is one of them. The other features are a combination of physical measurements (temperature, pressure etc.) and equipment settings (gas flow offset, temparature flow offset etc.). The dataset size is 9000. δ is set to 2000 given the high dimensionality and ϵ is set to 0.1 in algorithm 1. We use ridge regression in each of the states. Since, algorithm 1 gives us sufficient dynamic range on the training set we do not use the classification part i.e. algorithm 2 in order to expand it.

From figure 6 we see that our method does significantly better than the other methods for step sizes beyond 1. For step size 1 the performance is more or less comparable with the auto-correlation methods but slightly worse. RC and HMM plateau out to a positive R^2 while the auto-correlation methods do not since, the prior methods use also the 2025 time series other than the target to make their predictions, which is not the case with the auto-correlation models. Hence, again, when we try to predict multiple steps ahead the current target value is not that informative and the auto-correlation models suffer since, they have no other source of information to rely on.

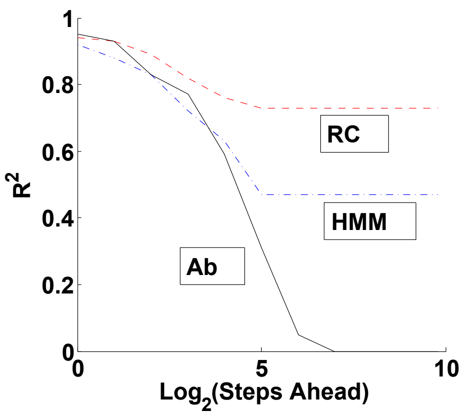

Figure 6. These are the results of the various methods on the EPI dataset. RC (acronym for regression+clustering) denotes our method described in algorithm 1. Ab is the best values taken from the previously mentioned auto-correlation methods and HMM is the hidden markov model based algorithm described in [5].

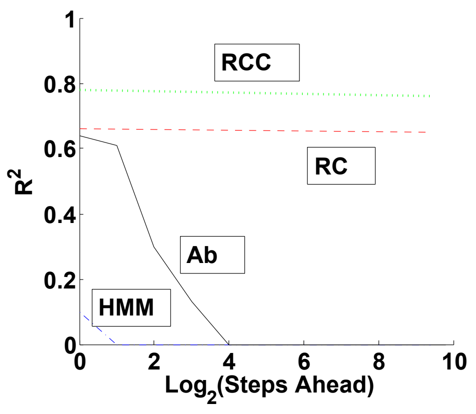

Figure 7. These are the results of the various methods on the speed dataset. RCC (acronym for regression+clustering+classification) denotes our method which is a combination of algorithm 1 and algorithm 2. RC (acronym for regression+clustering) denotes our method described in algorithm 1 alone. Ab is the best values taken from the previously mentioned auto-correlation methods and HMM is the hidden markov model based algorithm described in [5].

C. Wafer speed prediction dataset

In the chip manufacturing industry predicting speed of the wafers (which is a collection of chips) accurately ahead of time can be crucial in choosing the appropriate set of wafers to send forward for further processing. Eliminating faulty wafers can save the industry a huge amount of resources in terms of time and money.

The dataset we have has 175 features where, the wafer speed is one of them. The other features are a combination of physical measurements and electrical measurements made on the wafer. The dataset size is 2361. δ is set to 1000 given the dimensionality and ϵ is set to 0.1 in algorithm 1. We use ridge regression in each of the states. In this case, algorithm 1 does not give us a sufficient dynamic range on the training set so we do use the classification part i.e. algorithm 2 in order to expand it. The class ranges are deciphered using 10-fold cross-validation in this case, which are inputs to algorithm 2.

Here again both our methods are significantly better than the other methods with RCC being the best. The classification part provides the necessary boost to the regression part yielding much better estimates.

VI. Discussion

In the future, multiple extensions can be made to the methodology described in this paper. A simple yet useful extension would be to combine the various states based on the *closeness* of their pairwise regression coefficients. By *closeness* we mean that if the coefficients for every pair of input time series in a particular state are represented as a single datapoint in the corresponding parameter space, then the distance between this point and another point representing a different state lies below a threshold. Another extension would be to update these coefficients for the respective states during testing, based on the error[3] of the assigned test points. This may help in better capturing drifts that may occur in the data.

Another interesting direction would be to frame this problem as a relational learning problem [14]. One way to accomplish this, is to consider observations up to say k steps behind as relational neighbors and use the machinery developed in statistical relational learning to perform collective inference. It would be interesting to see the implications of combining methods from these two seemingly distinct domains.

To summarize, we have described a novel methodology to effectively tackle multi-step prediction problems in highly complex non-stationary domains. Key aspects of this methodology are, (1) an effective way of identifying states from the predictive standpoint and (2) an innovative classification scheme to improve the dynamic range of the predictions. We have empirically compared our method on real industrial datasets with other state-of-the-art methods and it seems to perform significantly better. In addition, we have described certain promising directions for the future. It remains to be seen, how our method and the suggested extensions would fair, as we apply them to data from other industrial domains.

Acknowledgement

Special thanks to Yan Liu for providing the HMM code and helpful discussions. In addition, we would like to thank Jayant Kalagnanam and Robert Baseman for providing the real datasets.

VII. Appendix

Proof: Consider two real vectors T_1 and T_2 of length N, given by $t_1^{(1)}, t_1^{(2)}, ..., t_1^{(N)}$ and $t_2^{(1)}, t_2^{(2)}, ..., t_2^{(N)}$ respectively. Let m equisize sequential partitions of these vectors be denoted by $P_1, P_2, ..., P_m$. Each P_i $(1 \leq i \leq m)$ corresponds to a segment of m consecutive observations from the two time series given by $\{(t_1^{((i-1)m+1)}, t_1^{((i-1)m+2)}, ..., t_1^{((i)m)}), (t_2^{((i-1)m+1)}, t_2^{((i-1)m+2)}, ..., t_2^{((i)m)})\}$. Without loss of generality (w.l.o.g.) let T_1 be the input time series and T_2 the ouput. We then build $m+1$ least square regression models, one for each of the m partitions P_i and one on the entire time series denoted by $R_1, R_2, ..., R_m$ and R_T respectively. Let their corresponding mean squared errors be $\Delta_1, \Delta_2, ..., \Delta_m$ and Δ_T. W.l.o.g. let us assume that R_1 has the least MSE i.e. $\Delta_1 \leq \Delta_i \ \forall i \in \{2, ..., m\}$. With this we have,

$$\begin{aligned}\Delta_T &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N}(R_T(t_1^{(i)}) - t_2^{(i)})^2\\
&= \tfrac{1}{m}[\tfrac{1}{\frac{N}{m}}\textstyle\sum_{i=1}^{m}(R_T(t_1^{(i)}) - t_2^{(i)})^2 + ...\\
&+ \tfrac{1}{\frac{N}{m}}\textstyle\sum_{i=N-m+1}^{N}(R_T(t_1^{(i)}) - t_2^{(i)})^2]\\
&\geq \tfrac{1}{m}[\tfrac{1}{\frac{N}{m}}\textstyle\sum_{i=1}^{m}(R_1(t_1^{(i)}) - t_2^{(i)})^2 + ...\\
&+ \tfrac{1}{\frac{N}{m}}\textstyle\sum_{i=N-m+1}^{N}(R_m(t_1^{(i)}) - t_2^{(i)})^2]\\
&= \tfrac{1}{m}\textstyle\sum_{i=1}^{m}\Delta_i \geq \tfrac{1}{m}m\Delta_1 = \Delta_1\end{aligned}$$

We have thus proved our result. The first inequality in the above equation comes from the fact that the individual models have the lowest MSE by definition. The second inequality is valid since, Δ_1 is the least amongst all the Δ_i $i \in \{1, 2, ..., m\}$. ■

[3]This not the prediction error but error w.r.t. the pairwise regression coefficients which related only to the input time series.

References

[1] S. W. S. Makridakis and J. Rob, 1998.

[2] J. L. F. Diebold and G. Weinbach, "Regime switching with time-varying transition probabilities," no. 283-302, 1993.

[3] W. Enders, *Applied Econometric Time Series*, 2nd ed. Wiley and Sons, 2003.

[4] C. Granger, "Testing for causality: a personal viewpoint," 2001.

[5] J. K. Y. Liu and O. Johnsen, "Learning dynamic temporal graphs for oil-production equipment monitoring system," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2009, pp. 1225–1234.

[6] Y. L. A. Lozano, N. Abe and S. Rosset, "Grouped graphical granger modeling methods for temporal causal modeling," in *KDD*. ACM, 2009.

[7] Y. L. A. Arnold and N. Abe, "Temporal causal modeling with graphical granger methods," in *KDD*. ACM, 2007.

[8] I. N. N. Friedman and D. Peer, "Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm," in *UAI*, 1999.

[9] D. Heckerman, "A tutorial on learning with bayesian networks," MIT Press, Tech. Rep., 1996.

[10] G. J. G. Box and G. Reinsel, in *Time Series Analysis:Forecasting and Control.* Prentice-Hall, 1994.

[11] T. Mills, in *Time Series Techniques for Economists.* Cambridge University Press, 1990.

[12] A. M. L. Cao and K. Judd, "Dynamic from multivariate time series," *Physica*, pp. 75–88, 1998.

[13] P. H. R. Duda and D. Stork, in *Pattern Classification*. Wiley-Interscience, 2000.

[14] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning*. MIT Press, 2007.