Multilevel Kernel Methods for Virtual Metrology in Semiconductor Manufacturing *

Andrea Schirru∗ Simone Pampuri∗ Cristina De Luca** Giuseppe De Nicolao∗

* Department of computer and systems science, University of Pavia, Italy
** Infineon Technologies Austria AG

Abstract: In semiconductor manufacturing, Virtual Metrology (VM) methodologies aim to obtain reliable estimates of process results without actually performing measurement operations, that are cost-intensive and time-consuming. This goal is usually achieved by means of statistical models, linking (easily collectible) process data to target measurements. In this paper, we tackle two of the most important issues in VM: function regression in high-dimensional spaces and data heterogeneity caused by inhomogeneous production and equipment logistics. We propose a hierarchical framework based on kernel methods and solved by means of multitask learning strategies and mixed-effects statistical models to improve the quality of estimates. The proposed methodology is validated on actual process and measurement data from the semiconductor manufacturing.

Keywords: Semiconductors, Machine learning, Industry automation, Probabilistic models, Computer-aided manufacturing

1. INTRODUCTION

Virtual sensors are used to estimate the result of an operation (typically a measurement) when the implementation of an actual sensor would be uneconomic, troublesome or impossible (for example, in hostile environments). In the design of a virtual sensor, the main goal is to find and exploit the relationships between "easily" collectible data and actual measurements. In semiconductor manufacturing, metrology operations (usually performed by means of Scanning Electron Microscopes) are so expensive and time-consuming that only a relatively small sample of produced wafers is actually evaluated. In recent years, Virtual Metrology (VM) tools (who rely on process data to predict measurements) are receiving growing attention by semiconductor manufacturers [Weber, 2007]: intuitively, a reliable VM tool is expected to shrink overall costs and simultaneously increase the amount and readiness of metrology data. This would yield positive fallouts on metrology-related process applications such as Run-to-Run controllers, sampling tools and decision-aiding tools. Research directions include algorithm development and validation [Khan et al., 2008] [Lin et al., 2009], analysis of interactions between control system and measurement strategy [Su et al., 2008a] and performance assessment [Tsai et al., 2010] [Su et al., 2008b].

The core of VM is a mathematical model linking measurements (output) to a set of process data (inputs). When actual measurements are not available, the model is used to predict them from available process data. Given the difficulty of developing physics-based mechanistic models, VM models are most often estimated from experimental data (black-box approach). From the algorithmic point of view, some of the most challenging issues in VM are (i) high-dimensional input spaces, often combined with strong nonlinearities, and (ii) small datasets due to sampling strategies and inter-chamber variability. In fact, many processes involve multi-chamber equipments, meaning that several chambers perform in parallel the same process step. A real example is depicted in tree form in Figure 1, referring to a three-chamber equipment from the CVD area (Chemical Vapor Deposition), with two subchambers for every chamber; since the equipment performs two processes, there is a grand total of twelve possible logistic paths along the tree. Intuitively, the use of a unique model for the whole equipment would yield suboptimal predictions, ignoring the intrinsic differences between chambers and processes, while building a model for every distinct path would require an adequate amount of data for each of them (and therefore, it would be cost intensive).

In this paper, issue (i) is tackled by means of kernel-based methodologies, while a novel multilevel framework is proposed in order to deal with issue (ii). The paper is organized as follows:

• Section 2 introduces the VM problem from a learning perspective and presents a concise introduction to Kernel methods
Fig. 1. Tree representation of a CVD equipment with three chambers (A, B, C) with two subchambers each (1 and 2), involved in two processes (Process 1 and Process 2)

- Section 3 describes the proposed multilevel kernel framework
- Section 4 evaluates the proposed methodology using actual manufacturing data as benchmark

2. BASICS OF MACHINE LEARNING AND KERNEL METHODS

This section provides an introduction to machine learning and kernel-based methods in connection to Virtual Metrology applications. The reader interested in a more detailed treatment of kernel-based methods may usefully refer to Shawe-Taylor and Cristianini [2004]. The basic assumption in machine learning is that latent knowledge can be learned from data. Given a training set \( \{x_i, y_i\} \in \mathbb{R}^{n \times p} \) of \( n \) examples, let \( X \in \mathbb{R}^{n \times p} \) be a matrix of the \( p \)-variate inputs and let \( Y \in \mathbb{R}^{n} \) be the associated output vector. The goal is to find some function \( f: \mathbb{R}^p \rightarrow \mathbb{R} \) such that, given a new example \( \{x_{new}, y_{new}\} \), the prediction \( f(x_{new}) \) is close to \( y_{new} \). In this context, \( f(x) \) is called an estimator of \( y \). The basic machine learning technique, Ordinary Least Squares (OLS), can be traced back to Gauss and Legendre Gauss [1809]; it looks for a linear relationship \( f(x) = wx \) by minimizing the following sum of squared residuals with respect to \( w \in \mathbb{R}^p \):

\[
J_{OLS}(w) := \frac{1}{2}||Y - Xw||^2 = \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i w)^2 \tag{1}
\]

Under a statistical framework, this amounts to maximizing the conditional probability \( p(Y|X) \) when assuming \( Y|X \sim N(Xw, \sigma^2 I) \) or, equivalently, \( Y = Xw + \epsilon \) with \( \epsilon \sim N(0, \sigma^2 I) \) (that is, i.i.d. Gaussian noise). The optimal coefficient vector \( w_{OLS} \) is just

\[
w_{OLS} = (X'X)^{-1}X'Y \tag{2}
\]

and does not depend on \( \sigma^2 \). This scheme suffers from two main drawbacks: (i) for small datasets \( n \approx p \) the estimated \( f(x) \) may overfit the noisy data or even interpolate the training examples, and (ii) the matrix \( X'X \) may be ill-conditioned or even singular. In order to overcome these drawbacks, Ridge Regression (RR, also known as Tikhonov regularization) was proposed in the forties of the last century Tikhonov [1943]: a linear estimator is obtained by minimizing the loss function

\[
J_{RR}(w) := \frac{1}{2}||Y - Xw||^2 + \frac{\lambda}{2} w'w = J_{OLS}(w) + \frac{\lambda}{2} w'w \tag{3}
\]

where \( \lambda \in \mathbb{R}^+ \) is the regularization (hyper)parameter. Under a Bayesian framework, \( J_{RR} \) is a logposterior distribution, and the term \( \frac{\lambda}{2} w'w \) in (3) is related to the prior distribution of \( w \), \( p(w) \), assuming \( w \sim N(0, \lambda^{-1} I) \). The larger \( \lambda \), the smaller the variance of the estimator, at the cost of introducing some bias; in practical applications, \( \lambda \) is often used as a "tuning knob" controlling the bias/variance tradeoff, which is typically tuned either via crossvalidation or other statistical criteria. The optimal coefficient vector \( w_{RR} \) and the estimator \( f_{RR}(x) \) are

\[
\hat{w}_{RR} = (X'X + \lambda I)^{-1}X'Y \tag{4}
\]

\[
f_{RR}(x) = x(X'X + \lambda I)^{-1}X'Y \tag{5}
\]

The numerical stability problems of equation (2) are now avoided, because \( (X'X + \lambda I) \) has full rank for any \( \lambda > 0 \).

It is obviously desirable to extend the possible estimators \( f(x) \) to include also nonlinear relationships: instead of considering \( y = wx \) as model, we consider \( y = \phi(x)w \), where \( \phi: \mathbb{R}^p \rightarrow \mathbb{R}^p \) is a mapping function and \( w \in \mathbb{R}^p \) is a new coefficient vector. For example, let \( p = 2 \) and \( \phi \) be defined as \( \phi(x) = [x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9] \). Then, the new estimator includes all possible polynomials up to second degree. Unfortunately, the number \( p \) of coefficients of a basis expanded to the \( d \)-th polynomial order would be

\[
p = (d - 1)p + \sum_{i=0}^{d-1} \binom{d}{i} \tag{6}
\]

To give some numerical example, for \( p = 100 \) and \( d = 2 \), \( p = 5150 \); for \( d = 3 \), \( p = 166950 \). It is apparent that (4) would be computationally intractable even for relatively low degree polynomials \( (p \)-th order matrix inversion is required). In order to overcome this issue, it is possible to resort to the so-called kernel trick Aizerman et al. [1964]: by using the matrix identity

\[
(X'X + \lambda I)^{-1}X' = X'(XX' + \lambda I)^{-1} \tag{7}
\]

the optimal RR estimator (5) can then be rewritten in dual form

\[
f_{RR}(x) = \langle x, X \rangle \langle X, X + \lambda I \rangle^{-1}Y \tag{8}
\]

that is

\[
f_{RR}(x) = \langle x, X \rangle \langle X, X + \lambda I \rangle^{-1}Y \tag{9}
\]
where \( \langle u, v \rangle = u'v \) for \( u \) and \( v \) column vectors of equal size denotes the inner product between \( u \) and \( v \). Essentially, equation (9) depends only on inner products, and the Gram matrix \( K = (X, X) \) is hereafter called kernel. This result, intimately related to the Riesz representation theorem in Hilbert spaces, allows to solve a nonlinear learning problem by embedding it in a linear framework Muller et al. [2001]. Notably, if there exist a kernel function \( \phi \) it is possible to compute \( f_{RR}(x) \) without explicitly evaluating \( \phi(X) \). Indeed, it is sufficient to compute the Gram matrix \( K \in \mathbb{R}^{n \times n} \) such that the element \((i, j)\) is \( K[i, j] = K(x_i, x_j) \) and the vector \( k \in \mathbb{R}^{1 \times n} \) such that \( k[i] = K(x, x_i) \). Then, (9) is rewritten as

\[
f_{RR}(x) = k(x)(K + \lambda I)^{-1}Y = k(x)e^* \tag{11}
\]

where \( e^* \) is the solution of the following

**Problem 1 (Kernel Ridge Regression):**

\[
\arg \min_c \frac{1}{2} \left\| Y - Kc \right\|^2 + \frac{\lambda}{2} c^T Kc \tag{12}
\]

Being (12) an unconstrained quadratic form with respect to \( c \), the minimizer is

\[
e^* = (K + \lambda I)^{-1}Y \tag{13}
\]

The estimator (11) can also be written as

\[
f_{RR}(x) = \phi(x)\phi(X)'(K + \lambda I)^{-1}Y = \phi(x)\overline{c} \tag{14}
\]

where \( \overline{c} = \phi(X)'e^* \) is the coefficient vector linking \( \phi(x) \) and \( f_{RR}(x) \). It is to note that, in this form, the basis expansion function \( \phi(\cdot) \) needs to be computed explicitly. As a matter of fact, equation (14) is not used in practice, but its definition is just instrumental to subsequent derivations.

Notable examples of kernels are the polynomial (homogeneous and inhomogeneous) of degree \( d \), realized by the functions

\[
K(u, v) = ((u, v))^d
\]

\[
K(u, v) = (1 + (u, v))^d
\]

and the radial basis kernel, relying on an infinite-dimensional input space and represented by the kernel function

\[
K(u, v) = e^{-\frac{|u - v|^2}{2\sigma^2}}
\]

We conclude the section with a couple of remarks:

- Kernel ridge regression was used as an example, but any machine learning algorithm whose solution can be written in dual form is a candidate for application of the kernel trick
- In Virtual Metrology, the input matrix \( X \) relies on process-related data, such as equipment settings and measurement data (either single measurements or statistics - for example, average layer thickness)

3. **MULTILEVEL KERNEL FRAMEWORK**

This section extends the use of Kernel Ridge Regression to a multilevel framework and derives a statistical interpretation. Finally, parameter tuning strategies are explored.

3.1 **Problem statement and notation**

The tree representation of Figure 1 calls for a methodology able to handle mutually exclusive data paths and their relationships. A growing cardinal number \( \mathcal{L} \) is conventionally assigned to the \( \eta \) nodes of the tree, moving from the root (node #0) to the leaves and from left to right, the last node being numbered \#\( \eta = 1 \) (Figure 2):

Every observation \( \{x_i, y_i\} \) follows a logistic path \( \mathcal{P}_i = \{L_0, L_1, \ldots \} \). For instance, if the \( i \)-th wafer undergoes ”Process 1” and is processed by chamber ”A1”, \( \mathcal{P}_i = \{0, 1, 3\} \). If it undergoes ”Process 2” and is processed by ”B2”, \( \mathcal{P}_i = \{0, 2, 6\} \). The goal is then to extend the theory presented in Section 2 to incorporate logistic paths: we look for estimators of the form \( f(x, \mathcal{P}) \) instead of \( f(x) \).

We aim to derive the estimator of a generic \( x, f(x, \mathcal{P}) \), as an additive model:

\[
f(x) = \sum_{j \in \mathcal{P}} f_j(x(x, j)) \tag{15}
\]

The goal then is to estimate \( \eta \) functions \( f_j \) by means of a multitask learning paradigm [Caruana, 1997]. It is to note that the number \( \Gamma \) of paths grows with the product of the number of nodes per level \( (1 \times 2 \times 6 = 12 \) for the equipment in Figure 1): the number of functions to be estimated may be reduced by exploitation of data commonalities. In our example, the number of functions to be estimated is 9 (that is the number of nodes \( \eta \)).
It is worth noting that, in most cases, the functions \( f_j(x,∗) \) will not depend on the entire vector \( x,∗ \), but only on a subset of variables here named \( x_{(∗,j)} \). For example, only process-related variables within \( x,∗ \) will be assigned to "Process" nodes (♯1 and ♯2), while only equipment-related variables will be assigned to "Chamber" (♯3 – ♯8) and "Equipment" (♯0) nodes.

Hereafter, let \( x_i \) be the \( i \)-th training input and \( y_i \) the \( i \)-th training output. Let \( \phi_j : p_j \rightarrow p_j \) be a function realizing the desired basis expansion for the \( j \)-th node, and let \( K_j(u, v) = \langle \phi_j(u), \phi_j(v) \rangle \) be the kernel function associated to the \( j \)-th node.

Before introducing the proposed methodology, we focus on tuning parameters. Kernel Ridge Regression is tuned by means of a single hyperparameter, \( \lambda \). In order to account for multilevel variability, we extend that as follows:

- as a straight generalization of the single-level case, every estimator \( f_j \) is controlled by \( \lambda_j \in \mathbb{R}^+ \)
- additionally, similarity between \( f_j \) and \( f_k \) is enforced by means of \( \xi_{j(k)} \in \{ \mathbb{R}^+ \cup 0 \} \), if nodes \( j \) and \( k \) are input-compatible and share the same kernel \( K_j(u, v) = K_k(u, v) \forall u, v \)

**Definition.** The nodes \( j \) and \( k \) are input-compatible if

- \( p_j = p_k \)
- \( x_{(∗,j)} \) and \( x_{(∗,k)} \) span the same input space
- Nodes \( j \) and \( k \) are mutually exclusive (\( j \in P_i \) \( \rightarrow k \not\in P_i \))

Basically, for \( j \) and \( k \) to be input-compatible is required that the \( j \)-th node and the \( k \)-th node share the input space. The set of pairs \( (j, k) \) enjoying the input-compatible property is named \( IC \).

The vector \( \lambda = [\lambda_0, \ldots, \lambda_{n−1}] \) generalizes the single-level hyperparameter \( \lambda \). using the notation from equation (14), the goal is to use \( \lambda_j \) to penalize \( ||\varpi_j||^2 \). On the other hand, the vector parameter \( \xi \) is intended to regularize \( ||\varpi_j − \varpi_k||^2 \).

In the following, let the entries \( (i, k) \) of \( K_j \in \mathbb{R}^{n \times n} \) be

\[
K_{j}[i, k] = \begin{cases} 
K_j(x_{(i,j)}, x_{(k,j)}) & j \in (P_i \cap P_k) \\
0 & j \not\in (P_i \cap P_k)
\end{cases}
\]  

(16)

with \( K_j = \bar{X}_j^T \bar{X}_j \) and \( \bar{K} = \sum_{i=0}^{n−1} K_j \). Additionally, let the \( i \)-th element of \( \mathbf{k}_j(x,∗) \) be

\[
\mathbf{k}_j(x,∗)[i] = \begin{cases} 
K_j(x_{(∗,j)}, x_{(i,j)}) & j \in P_i \\
0 & j \not\in P_i
\end{cases}
\]  

(17)

3.2 Multilevel estimation with Kernel Ridge Regression

For Kernel Multilevel Ridge Regression the estimator \( f \) of a generic \( x,∗ \) is extended from (11) by summing over \( P_* \):

\[
f(x,∗, P_*) = \sum_{j \in P_*} f_j(x_{(∗,j)}) = \sum_{j \in P_*} \mathbf{k}_j(x,∗) c
\]  

(18)

The optimal weighting vector \( c \) in (18) is obtained by solving the following

**Problem 2 (Multilevel Kernel Ridge Regression):**

\[
\arg\min \limits_c \frac{1}{2} ||Y - \mathbf{K}c||^2 + \frac{1}{2} c^T \mathbf{G}c
\]  

(19)

where the entries \( (i, k) \) of \( \mathbf{G} \) are

\[
G[i, k] = \sum_{j=0}^{n−1} \lambda_j K_j[i, k] + \sum_{j, z \in IC} \xi_{j,z} \tilde{G}[i, k]
\]  

(20)

with

\[
\tilde{G}[j, k] = \sum_{j, z \in (P_i \cup P_k)} K_j(x_{(i,j)} - x_{(k,z)} , x_{(i,j)} - x_{(k,z)})
\]  

(21)

Problem 2 is a positive definite quadratic programming problem with respect to \( c \), and its minimizer \( c^* \) is

\[
c^* = (\mathbf{K}^2 + \mathbf{G})^{-1} \mathbf{K}Y
\]  

(22)

Combining (18) and (22), the estimator \( f(x,∗, P_*) \) is then

\[
f(x,∗, P_*) = \sum_{j \in P_*} \mathbf{k}_j(x_{(∗,j)})(\mathbf{K}^2 + \mathbf{G})^{-1} \mathbf{K}Y
\]  

(23)

Using the notation of Equation (14), it is to note that

\[
c^T \mathbf{K}c = c^T \bar{X}_j \bar{X}_j^T c = ||\varpi_j||^2
\]  

(24)

Similarly,

\[
c^T \tilde{G}c = c^T (\bar{X}_j - \bar{X}_j')(\bar{X}_j - \bar{X}_j')^T c = ||\varpi_j - \varpi_k||^2
\]  

(25)

Problem 2 can then be rewritten as follows

**Problem 2b (Multilevel Kernel Ridge Regression in primary form):**

\[
\arg\min \limits_c \frac{1}{2} ||Y - \sum_{j=0}^{n−1} \varpi_j||^2 + \frac{1}{2} \sum_{j=0}^{n−1} \lambda_j ||\varpi_j||^2 + \frac{1}{2} \sum_{j, z \in IC} \xi_{j,z} ||\varpi_j - \varpi_k||^2
\]  

(26)

In analogy with (14), this formulation requires the explicit computation of \( \phi_j \) and is not used in practice; it is useful, however, to enlighten the role of \( \bar{X} \) and \( \xi \).

3.3 Probabilistic interpretation and output error estimation

In order to produce a probabilistic output for Virtual Metrology module, we introduce the following
Problem 3 (Multilevel Kernel Ridge Regression with output error covariance matrix W):

$$\arg\min_c \frac{1}{2}||Y - Kc||_W^2 + \frac{1}{2}c^T G c$$

where $||x||_W^2 = x^T W x$ for $x$ column vector.

Problem 3 admits an interesting Bayesian interpretation: letting $c$ be a priori distributed as $c \sim N(0, G^{-1})$ and assuming $Y|c \sim N(Kc, W^{-1})$, Bayes’ theorem states

$$p(c|Y) \propto p(Y|c)p(c)$$

By exploiting the conjugation properties of the multivariate normal distribution [Barker et al., 1995],

$$c|Y \sim N(\mu_c, \Sigma_c)$$

$$\mu_c = (KWK + G)^{-1}KWY$$

$$\Sigma_c = (KWK + G)^{-1}$$

Therefore, the posterior distribution $p(c|Y)$ depends on the hypothesized error covariance matrix $W^{-1}$. We consider a generic parametrization for $W$, such that $W := W(\theta)$. The free parameters of the proposed model now include the regularization vectors $\lambda$ and $\xi$ and the parameter vector $\theta$. In order to optimize such parameters, it is convenient to resort to Restricted Maximum Likelihood (REML) estimation [Harville, 1977]. Observing that

$$Y|(\tilde{X}, \tilde{\xi}, \theta) \sim N(K\mu_c, K\Sigma_c K + W^{-1})$$

the Restricted loglikelihood of $p(Y|(\tilde{X}, \tilde{\xi}, \theta))$ is

$$l_R = \log(p(Y|(\tilde{X}, \tilde{\xi}, \theta)) =$$

$$-\frac{1}{2} (Y - K\mu_c)'(K\Sigma_c K + W^{-1})^{-1}(Y - K\mu_c)$$

$$-\frac{1}{2} \log(|K\Sigma_c K + W^{-1}|)$$

In order to find the optimal set of parameters, one has to solve the following

Problem 4 (REML optimization)

$$\arg\min_{(\tilde{X}, \tilde{\xi}, \theta)} -l_R(\tilde{X}, \tilde{\xi}, \theta)$$

whose minimizer $[\tilde{X}^{*}, \tilde{\xi}^{*}, \theta^{*}]$ can be found by means of a Newton-Raphson algorithm, as documented in [Wolfinger et al., 1994].

3.4 Implementation notes

This subsection presents a workflow for implementing the proposed algorithm, in order to facilitate practical use. Given a training set $\{X, Y\}$ and a known model hierarchy,

(1) For every node $j$,

(a) Select an input space from $X$ such that $x_{(i,j)}$ is the input to the $j$-th node for the $i$-th observation

(b) Choose a kernel function $K_j$

(2) Optimize model parameters by solving Problem 4

(3) Find the optimal posterior distribution of $c$ by solving Problem 3

(4) Make predictions using Equation (18)

It is also possible to use Generalized Cross Validation (GCV) instead of REML to tune the model parameters.

4. EXPERIMENTAL RESULTS

The proposed multilevel VM architecture has been tested on an extensive dataset made of 3000 examples (wafers) collected over 29 weeks of production; the multilevel structure of the data is represented in Figure 1. The wafers, all processed by the same CVD equipment (“Equipment 1”) at the Infineon facility in Villach, underwent two distinct production processes (“Process 1” and “Process 2”). Moreover, the equipment is composed of three chambers (“A,” “B” and “C”) with two subchambers each (e.g., “A1” and “A2”). Therefore, associated with each wafer there is a path $P$ identifying process, chamber and subchamber. In the dataset, all 12 possible paths are represented.

The goal is to develop a VM algorithm that predicts average thickness measurements using process data. The explanatory variables, summarized in matrix $X$, include time series profiles from equipped sensors, process and controller settings, for a total of about 600 regressors. The output vector $Y$ is given by scalar measurements of average layer thickness.

4.1 Design of experiment

Production data are characterized by drifts and jumps, possibly due to maintenance operations, variations in equipment health, and so on: therefore, some form of adaptivity must be incorporated in the VM algorithm in order to be able to track changes in the input-output relationship [Kang et al., 2010]. Our adaptive implementation relies on a moving window, whose length is optimized in each subexperiment as described below. The proposed methodology was tested against two competitors:

- A simple mean propagation, serving as baseline or "naive" strategy (completely ignoring information from explanatory variables)
- Single-level Kernel Ridge Regression as described in Section 2

Data covering 29 weeks were divided into 9 nested subexperiments organized as follows: the first one includes data collected during weeks 1-5, the second one refers to weeks 1-8, the $i$-th one refers to weeks 1 to $3i + 2$, the last one ($i = 9$) covers weeks 1-29. In each subexperiment, the last two weeks were reserved for testing. This allows to test the predictive capabilities of the proposed methodology with training sets of different sizes.

The length of the moving window was optimized for each subexperiment: for subexperiment 1, weeks 1-2 were used for training predictors based on different window lengths, whereas week 3 played the role of validation set in order to decide the optimal window length. More in general, for the $i$-th subexperiment, weeks 1 to $3i - 1$ were used for training and week $3i$ was used for validation in order
Fig. 3. VM test on 50 wafers: the proposed methodology is able to produce good estimates of actual metrology to decide the optimal window length. Once the window length has been decided, the estimators are recomputed using weeks 1 to 3 and RMSE (Root Mean Square Error) is computed on test data from weeks 3$\lambda + 1$ and 3$\lambda + 2$. An example of VM predictions from path "Process , A2", week 27, is reported in Figure 3. The panels in Figure 5 show a graphical representation of the comparative results for every path $P$, while average results (across paths) are reported in Figure 4.

Additionally, vectors $\overline{X}$, $\overline{\xi}$ and $\theta$ presented in Section 3 were reparametrized as follows:

- Three $\lambda$ parameters were used: one for the root node, one for the process nodes and one for the chamber nodes.
- A single shared $\xi$ parameter was used to enforce similarities among subchambers (A1-A2, B1-B2, C1-C2).

As for $W$, it was assumed that

$$W^{-1} = \text{diag}\{\sum_{j \in P_i} \sigma_j^2\}$$  \hspace{1cm} (33)

so that $\eta$ error variances $\sigma_j^2$, $j = 0, \ldots, \eta - 1$ are to be estimated. This way, optimization was run over 13 parameters instead of 24 from the single-level case.

4.2 Comments

As expected the most trivial strategy, propagated mean, yields the worst performances. A first improvement is obtained by using single-level KRR. Rather interestingly, from Table 2, that reports RMSE ratios between multi- and single-level KRR, it is seen that the multilevel KRR performs systematically better than single-level one (90 out of 94 test points). Best performance per path is highlighted in boldface, while worst performances are in red. Therefore, these experimental results indicate that full exploitation of commonalities in logistic paths can play a key role in improving VM prediction capabilities. Remarkably, the selected window length varies between 4 and 8 weeks: this observation confirms the drifting nature of the experimental setting.

5. CONCLUSIONS

In this paper, a new Kernel-based Virtual Metrology algorithm, based on multilevel learning, has been proposed and validated on experimental data from the CVD area. The key point is the exploitation of commonalities in logistic paths of different wafers in order to learn predictive models also from heterogeneous datasets. In this way, it is possible to obtain reliable VM modules also when the number of wafers sharing the same logistic paths is relatively small. The proposed solution is especially relevant to production sites oriented to low-volume processes. Through extensive experimental testing, the advantage with respect to the (traditional) single-level approach was demonstrated and assessed.

REFERENCES


Table 1. Path-by-path RMSE results: the proposed methodology performs significantly better than its competitors.

<table>
<thead>
<tr>
<th>Path - Week</th>
<th>3</th>
<th>6</th>
<th>9</th>
<th>12</th>
<th>15</th>
<th>18</th>
<th>21</th>
<th>24</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process 1, A1</td>
<td>0.5015</td>
<td>0.6097</td>
<td>0.4776</td>
<td>0.6501</td>
<td>0.6051</td>
<td>0.8745</td>
<td>0.6542</td>
<td>0.7821</td>
<td>0.7113</td>
</tr>
<tr>
<td>Process 2, A1</td>
<td>0.4511</td>
<td>0.7399</td>
<td>/</td>
<td>/</td>
<td>0.6228</td>
<td>0.7143</td>
<td>1.0228</td>
<td>0.6769</td>
<td>0.6366</td>
</tr>
<tr>
<td>Process 1, A2</td>
<td>0.5865</td>
<td>0.7849</td>
<td>0.8110</td>
<td>0.6700</td>
<td>0.7153</td>
<td>0.8938</td>
<td>0.5660</td>
<td>0.6045</td>
<td>0.5551</td>
</tr>
<tr>
<td>Process 2, A2</td>
<td>0.6536</td>
<td>0.7870</td>
<td>/</td>
<td>/</td>
<td>0.8094</td>
<td>0.6123</td>
<td>1.1416</td>
<td>0.7511</td>
<td>0.4906</td>
</tr>
<tr>
<td>Process 1, B1</td>
<td>0.5166</td>
<td>0.7489</td>
<td>1.0819</td>
<td>0.6614</td>
<td>0.4579</td>
<td>0.6470</td>
<td>0.9099</td>
<td>0.5870</td>
<td>0.6565</td>
</tr>
<tr>
<td>Process 2, B1</td>
<td>0.6669</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>0.6262</td>
<td>0.7394</td>
<td>0.7675</td>
<td>0.8255</td>
<td>0.4129</td>
</tr>
<tr>
<td>Process 1, B2</td>
<td>0.6140</td>
<td>0.9906</td>
<td>0.4333</td>
<td>0.8267</td>
<td>0.7115</td>
<td>0.6320</td>
<td>0.6425</td>
<td>0.8266</td>
<td>0.4371</td>
</tr>
<tr>
<td>Process 2, B2</td>
<td>0.5521</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>0.5181</td>
<td>0.7616</td>
<td>1.0674</td>
<td>0.7151</td>
<td>0.4804</td>
</tr>
<tr>
<td>Process 1, C1</td>
<td>/</td>
<td>0.8618</td>
<td>0.7639</td>
<td>0.7529</td>
<td>0.7519</td>
<td>0.6956</td>
<td>0.7266</td>
<td>0.5771</td>
<td>0.5625</td>
</tr>
<tr>
<td>Process 2, C1</td>
<td>/</td>
<td>0.6331</td>
<td>0.7774</td>
<td>0.6359</td>
<td>0.6215</td>
<td>0.5355</td>
<td>0.5101</td>
<td>0.6436</td>
<td>0.7288</td>
</tr>
<tr>
<td>Process 1, C2</td>
<td>/</td>
<td>0.5977</td>
<td>0.7073</td>
<td>0.5470</td>
<td>0.5652</td>
<td>0.5605</td>
<td>0.5818</td>
<td>0.7013</td>
<td>0.3675</td>
</tr>
<tr>
<td>Process 2, C2</td>
<td>/</td>
<td>0.4128</td>
<td>0.6436</td>
<td>0.6510</td>
<td>0.8394</td>
<td>0.5678</td>
<td>0.4378</td>
<td>0.6885</td>
<td>0.5570</td>
</tr>
</tbody>
</table>

Table 2. RMSE ratios multilevel/single-level: multilevel RMSE is lower with respect to single-level RMSE in 90 out of 94 experiments.