Classification Algorithms for Virtual Metrology

I. INTRODUCTION

Semiconductor manufacturing production line includes many steps and requires a high level of accuracy in each step. Several sensors are placed in different locations in order to detect the defects to stop following the manufacturing process for already defected items. Information collected from these sensors are, then, used in quality control. The control is performed in many levels to ensure the stability and the performance of the production process. [2] enumerates some levels that should be taken into account during the control process. These levels are mentioned as layers of control from tools to product. At the product level, electric tests verify proper functionality of chips. These tests are performed at the end of the manufacturing stage [13] [18]. Some other integration tests are applied during the fabrication to verify the properties of technological modules. Tests like verifying that transistor’s shape is correctly processed, or examining that electrical properties of several layers of materials are in the desired specifications. After each step, usually it is possible to perform some tests to monitor any abnormal variations of tools that operate a particular process. Finally, at the tool level, numerous sensors are employed to regulate and to monitor the process.

All the tests and the monitoring steps described above are based on few samples of products, except for the electrical wafer sort being the final product. As a consequence, a drift in product quality that occur during non-sampling periods. This approach enables a hundred percent control and improves the precision of statistical control, specially while there is no sampling activity in manufacturing process. The main challenge in virtual metrology is inaccurate predictions. As such, the choice of an appropriate algorithm for prediction is crucial. We compare several algorithms that can be used for prediction in virtual metrology. The comparison over different prediction algorithms is made on a simulated data inspired from virtual metrology application.

II. VIRTUAL METROLOGY

Virtual metrology has been introduced to employ mathematical models on accessible measurements from an operating process with the aim of predicting some variables of interest. This methodology allows to predict relevant variables using equipment measurement, without physically conducting quality measurement [21] [14]. [6] observed a strong correlation between the tool history and the wafer measurement. They found the coefficient of determination $R^2 > 0.97$ can be achieved with more than 500 wafers on deposition as the response and equipments as the predicting variables. This strong correlation suggests that a wafer-to-wafer control can be quickly enabled using an existing lot-to-lot control system. This indirect technique, called virtual metrology, provides an efficient and economical alternative to wafer-to-wafer control.

Several algorithms have been proposed in the VM literature. [14] suggested a linear regression model to predict the state of the wafer in real time. They proposed to find the coefficients using least squares. The univariate output of their model was the wafer measurement and the inputs were the equipments’ measurement. The model was updated as new output measurement was available. [3] studied a virtual metrology model using partial least squares. This model predicts chemical vapor deposition oxide thickness for an Inter Metal Dielectric deposition process.

Many VM algorithms have been developed based on neural networks [22]. Neural network is an implicit nonlinear model fitting. Different versions of neural networks have been considered in the VM literature. [15] established a VM model using radial basis network. The effectiveness of the proposed VM system was tested on chemical vapor deposition processes in a practical semiconductor manufacturing. Their result confirmed that neural network can be used effectively to construct a predictive model. [14] adopted a system using back propagation neural network for establishing a model for the etching process in semiconductor manufacturing. [7] compared the performance of the radial basis network and the back propagation neural network on the thin-film transistor liquid crystal display industry. The radial basis function network and the back propagation neural network produced quite similar

Abstract—Virtual metrology in quality control deals with drifts in product quality that occur during non-sampling periods. This approach enables a hundred percent control and improves the precision of statistical control, specially while there is no sampling activity in manufacturing process. The main challenge in virtual metrology is inaccurate predictions. As such, the choice of an appropriate algorithm for prediction is crucial. We compare several algorithms that can be used for prediction in virtual metrology. The comparison over different prediction algorithms is made on a simulated data inspired from virtual metrology application.
results. Some other versions of neural models are proposed to detect wafer anomaly such as polynomial neural network, piecewise linear neural network, and fuzzy neural network, see for instance [5], [4], [11], and [20].

Kernel approaches, specifically support vector machines, are a powerful tool to predict wafer drifts based on equipment measurements [16]. [7] reports that the support vector machines approach gives a better prediction accuracy compared with the radial basis function network and also compared with the back-propagation approach, see also [1].

Genetic algorithm is a powerful optimization tool for model fitting [8]. A kernel adjustment is proposed to deal with overfitting problems, [7] combines the support vector machines and the genetic algorithm to construct a virtual metrology system for the chemical vapor deposition process. [23] suggests principal components axes to reduce the dimensionality of plasma dimensions after etching process. It is well-known that uncorrelated features improve the estimation and the prediction of statistical models. The principal components are linear combinations of the inputs and are mutually uncorrelated.

Existence of a large list of different classification algorithms makes the choice of an appropriate algorithm difficult. We aim to compare different classification algorithms proposed in the literature on a simple simulated example motivated from a VM problem.

III. DATA SIMULATION

We tried to simulate the data according realistic conditions appearing in VM. The inputs, say \( x \), represent the equipment measurements. In practice, the inputs can be power, pressure, temperature, etc. Some of such inputs are intercorrelated and some others are independent from each other. We simulated the total of 10 inputs. The output variable, say \( y \), is a binary variable that represents the final product’s state, being correct or defected. The following matrix describes the data structure

\[
\begin{bmatrix}
 x_{11} & x_{12} & \cdots & x_{110} & y_1 \\
 x_{21} & x_{22} & \cdots & x_{210} & y_2 \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 x_{n1} & x_{n2} & \cdots & x_{n10} & y_n
\end{bmatrix}
\]

where each row corresponds to a measurement of on a wafer. Each column \( x_j \) is a continuous value of an input variable, say equipment \( j \), and the last column \( y \) shows the binary output. The number of rows, \( n \), is the total number of wafers. The matrix entries \( x_{ij} \) represents the measurement of wafer \( i \) on equipment \( j \) and \( y_i \) is the final state of wafer \( i \).

We simulated the data with the following structure. Input variables \( x_1 \) and \( x_2 \) are a block of intercorrelated variables; another block of correlated variables contains \( x_3, x_4, \) and \( x_5 \). The other inputs \( x_6, x_7, x_8, x_9 \) and \( x_{10} \) are generated independently, all irrelevant to classify the output. The latter block does not affect the output variable, but they contribute in the classification error generated by the measurement system. Table I illustrates the dependence structure of the simulated input variables in which \( N_p(\mu, \Sigma) \) denotes a \( p \)-variate Gaussian distribution with mean \( \mu \) and variance-covariance \( \Sigma \).

<table>
<thead>
<tr>
<th>Block</th>
<th>Inputs</th>
<th>Correlation</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((x_1, x_2))</td>
<td>yes</td>
<td>( N_2 \left[ \begin{bmatrix} 0 \ 0 \end{bmatrix}, \begin{bmatrix} 1 &amp; 0.9 \ 0.9 &amp; 1 \end{bmatrix} \right] )</td>
</tr>
<tr>
<td>2</td>
<td>((x_3, x_4, x_5))</td>
<td>yes</td>
<td>( N_3 \left[ \begin{bmatrix} 0 \ 0 \end{bmatrix}, \begin{bmatrix} 1 &amp; 0.9 \ 0.9 &amp; 1 \end{bmatrix} \right] )</td>
</tr>
<tr>
<td>3</td>
<td>(x_6, \ldots, x_{10})</td>
<td>no</td>
<td>( N_1(0, 1) )</td>
</tr>
</tbody>
</table>

TABLE I: The generated input data structure. Three blocks of input variables are generated of size \( n = 100 \) observations.

First, we simulated a binary output \( y_i \) being generated as a function of only three input variables \( x_{11}, x_{13}, \) and \( x_{14} \)

\[
\begin{cases}
 y_i = 1 & \text{if } a^T z_i \geq 0, \\
 y_i = 0 & \text{if } a^T z_i < 0,
\end{cases}
\]

where \( a = (a_1, a_2, a_3) \) and \( z_i = (x_{i1}, x_{i3}, x_{i4}) \). This model produces observations that only \( x_1, x_3, \) and \( x_4 \) are useful for classification, and the other variables are noise. Second, we simulated a binary output using a quadratic function of \( x_1, x_2, \) and \( x_4 \) is generated

\[
\begin{cases}
 y_i = 1 & \text{if } a^T z_i + z_i^T A z_i \geq 0, \\
 y_i = 0 & \text{if } a^T z_i + z_i^T A z_i < 0.
\end{cases}
\]

The elements of the vector \( a \) are sampled independently and uniformly from \([-6, -3, 3, 6]\), and the elements of the symmetric matrix \( A \) are sampled from \([-6, -3, 0, 3, 6]\). We generated \( n = 100 \) observations as the training set. A dataset of the same size is generated as the validation set. The model is fitted on the training set, and the precision of the resulting classification is evaluated on the validation set. The total of 20 Monte Carlo simulations have been run.

IV. CLASSIFICATION ALGORITHMS

Several classification algorithms listed below are used to predict the output as a function of the inputs.

A. \( k \)-Nearest-Neighbours

The \( k \)-nearest-neighbours is a model-free algorithm that predicts the output based on its \( k \) nearest neighbours. The nearest neighbours are found using a distance, often the Euclidean distance, computed over the corresponding input variables. Suppose \( N(x) \) is the neighbourhood at point \( x \) where the \( k \) data fall into, then

\[
\hat{y}(x) = \frac{1}{k} \sum_{x_i \in N(x)} y_i.
\]

This technique gives a step function approximation to the classification function, see Fig. 1. The tuning parameter \( k \) is chosen manually or is estimated using cross-validation.

B. Logistic Regression

The logistic regression is a generalization of the linear regression where the output variable is binary. This technique is used to predict a binary outcome based on one or more continuous predictor variables. The logistic regression estimates the coefficients of a linear classifier using the conditional
distribution of \( y_i \mid x_i \). Since the logistic regression uses a probabilistic model to estimate the classification function, the probability of \( (y = 1 \mid x) \) can be extracted after the fitting for a given \( x \). In order to produce a binary predict, this estimated probability is cut at a certain point, usually 0.5. The probability of \( y_i \mid x_i \) is expressed as

\[
\Pr(y_i = 1|x_i) = \frac{\exp(\beta_0 + x_i'\beta)}{1 + \exp(\beta_0 + x_i'\beta)},
\]

where \( x'_i = (x_{i1}, x_{i2}, \ldots, x_{i10}) \) and \( \beta = (\beta_1, \ldots, \beta_{10})' \). The regression coefficients are estimated using maximum likelihood. The log likelihood function of the Bernoulli distribution is maximized using iterative reweighted least squares. The Bernoulli log likelihood, say \( \ell(\beta) \), is expressed as

\[
\ell(\beta) = \sum_{i=1}^{n} \left\{ \log \left( \frac{\exp(\beta_0 + x_i'\beta)}{1 + \exp(\beta_0 + x_i'\beta)} \right)^{y_i} \left[ 1 - \frac{\exp(\beta_0 + x_i'\beta)}{1 + \exp(\beta_0 + x_i'\beta)} \right]^{1-y_i} \right\}
\]

Like linear regression, logistic regression suffers from overfitting and produces unstable estimation of coefficients while a many of noise variables is added in the model. As a remedy the penalized logistic regression is fitted. The penalty term, penalizes large absolute values of model coefficients. The maximizing function is

\[
\ell(\beta) - \lambda||\beta||_2^2
\]

where \( ||\beta||_2^2 = \sum_{j=1}^{10} \beta_j^2 \) is the squared Euclidean norm of the regression coefficients. Here \( \lambda \) is a positive tuning parameter, usually estimated by cross-validation.

### C. Neural Network

A neural network is a set of simple but highly interconnected processing elements, called neurons, to fit a highly nonlinear model, see Fig. 2. Neural network has been evaluated for different number of hidden layers with different weights. The most predictive number of layers is chosen. This approach help regularizing this algorithm and avoids overfitting.

### D. Linear Discriminant

Linear discriminant analysis separates data into different classes (two classes for a binary output) using a linear hyperplane, see Fig. ?? (left panel). However, linear discriminant coefficients are sensitive to the correlation between the input variables. In order to improve the classification performance, it is proposed to perform the classification on the principal components of data [17]. We applied linear discriminant on four principal components.

An alternative to improve the performance in the presence of correlated variables is penalization. A penalized discriminant analysis is suggested in [9]. Absolute norm penalty, also called the lasso penalty, is applied to the discriminant vectors to encourage variable selection simultaneously.

### E. Quadratic Discriminant

Quadratic discriminant analysis, as its name indicates, proposes quadratic boundaries to separate data. This algorithm is similar to the linear discriminant, except it allows for quadratic coefficients as well, see Fig. ?? (right panel). We applied this algorithm on the principal components of data also.

### F. Mixture Discriminant

Polynomial boundaries such as linear and quadratic functions are too restrictive for complex data. Mixture of discriminant functions covers a flexible class of classification functions. This algorithm is called mixture discriminant analysis [10]. A Gaussian mixture model for the kth class has density

\[
\Pr(X \in G = k) = \sum_{i=1}^{R_k} \pi_{kr} \phi(X, \mu_{kr}, \Sigma_{kr}),
\]

where the mixing proportions \( \pi_{kr} \) sum to one. This has \( R_k \) prototypes for the kth class, and in our specification, the covariance matrix \( \Sigma_{kr} \) is used as the metric throughout. Given such a model for each class, the class posterior probabilities...
where π function transforms the classification problem into a new space
hyperplane, the data are categorized into 2 classes. Instead of
training data point of any class, so-called functional margins,
G. Kernelized Support Vector Machines

A support vector machine constructs a hyperplane in a high
dimensional space. Intuitively, a good separation is achieved by
the hyperplane that has the largest distance to the nearest
training data point of any class, so-called functional margins,
see Fig. 4. Like the other methods, after computing the
hyperplane, the data are categorized into 2 classes. Instead of
the linear support vector machines, we tested a more flexible
version called kernelized support vector machines. The kernel
function transforms the classification problem into a new space
defined by the kernel (inner product) on the input variables.
We used the radial basis kernel also called Gaussian kernel.

V. NUMERICAL RESULTS

A Monte Carlo simulation study over the linear and the
quadratic models (1) and (2) is summarized in Table II.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Linear</th>
<th>Quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network</td>
<td>94</td>
<td>94</td>
</tr>
<tr>
<td>Kernel SVM</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>MDA-PCA</td>
<td>86</td>
<td>87</td>
</tr>
<tr>
<td>QDA-PCA</td>
<td>86</td>
<td>87</td>
</tr>
<tr>
<td>KNN</td>
<td>85</td>
<td>86</td>
</tr>
<tr>
<td>LDA-PCA</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>Penalized LDA</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>Penalized LR</td>
<td>90</td>
<td>91</td>
</tr>
<tr>
<td>LR</td>
<td>88</td>
<td>89</td>
</tr>
</tbody>
</table>

The simulation codes are written the statistical programming
language R [19]. Simulations are performed using a 2.30 GHz
Intel core i5-2410m processor and 6.00 Go RAM, taking
around 2 minutes to run all algorithms. Datasets and the R
codes are available and will be provided upon request. The
correct classification rates of the output variable is summarized
in Table II.

Neural network outperforms all other algorithms for both
linear and quadratic data. The logistic regression (LR) and
the penalized logistic regression are, also, good classifiers
for the linear data. However, they give significantly inferior
classification rates for the quadratic data. The penalized
logistic regression (Penalized LR) improves the rate of correct
classification compared to the logistic regression, particularly
for the quadratic output. It achieves an increase of 3% (from
64% to 67%). The quadratic discriminant combined with the
principal components (QDA-PCA) shows better results than
the linear discriminant (LDA-PCA). The QDA-PC on the
quadratic output shows 6% increase of the correct classification
rate compared to LDA. The mixture discriminant method
(MDA) gives results similar to the quadratic discriminant, but
better than the LDA-PCA and the Penalized LDA, for the linear
output. The kernelized support vector machines (Kernel SVM)
gives accurate predictions for both linear and quadratic outputs.
VI. CONCLUSION

We briefly reviewed the existing literature on quality control with an emphasis on virtual metrology. We insist that the choice of a proper classification algorithm is of great importance in this area. Therefore, we studied several algorithms that could be used for VM on some simulated data. These algorithms perform differently depending on the outputs (linear or quadratic). However, neural network outperforms all others in both cases. This suggests to keep neural network method as a strong potential candidate for modelling in VM.

REFERENCES


