Novel Machine Learning Approaches for Modeling Variations in Semiconductor Manufacturing

by

Hongge Chen

B.E. in Electrical Engineering
Tsinghua University, 2015

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author ..... Signature redacted .....
Department of Electrical Engineering and Computer Science
May 19, 2017

Certified by ..... Signature redacted
Duane S. Boning
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ..... Signature redacted
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Novel Machine Learning Approaches for Modeling Variations in Semiconductor Manufacturing

by

Hongge Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

In the competitive semiconductor manufacturing industry where large amounts of data are generated, data driven quality control technologies are gaining increasing importance. The primary goal of this thesis is to build machine learning models for variation analysis and yield improvement. In this thesis, we first propose a novel method to estimate and characterize spatial variations on dies or wafers. This new technique exploits recent developments in matrix completion, enabling estimation of spatial variation across wafers or dies with a small number of randomly picked sampling points while still achieving fairly high accuracy. This new approach can also be easily generalized, including for estimation of mixed spatial and structure or device type information.

Then machine learning models for high yield and time varying semiconductor manufacturing processes are developed. Challenges include class imbalance, concept drift (temporal variation) and feature selection. Batch and online learning methods are introduced to overcome the class imbalance. Incremental learning frameworks are developed to handle concept drift and class imbalance simultaneously. We study the packaging and testing process in chip stack flash memory manufacturing as an application, and show the possibility of yield improvement with machine learning based classifiers detecting bad dies before packaging. Experimental results demonstrate significant yield improvement potential using real data from industry. Without concept drift, for stacks of eight dies, approximately 9% yield improvement can be achieved. In a longer period of time with realistic concept drift, our incremental learning approach achieves approximately 1.4% yield improvement in the eight die stack case and 3.4% in the sixteen die stack case.

Thesis Supervisor: Duane S. Boning
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank my advisor, Professor Duane Boning, for helping me, encouraging me and challenging me to do my best in this thesis work. He gave me a great amount of freedom in research to explore new ideas that intrigue me, and provided guidance and mentorship over the past two years. His passion, engineering intuition and dedication will always be my sources of inspiration. I am also fortunate to be part of Professor Boning's research group. I have overlapped with John Lee, Christopher Lang, Sally El-Henawy, Germain Martinez, Daniel Moon, Seyed Yahya Hosseini and Masahiro Sakuta. I am grateful for the time we have spent together and your support. I also want to thank Zheng Zhang, Professor Roy Welsch and Professor Jacob White for all the interesting ideas they shared with me. Finally, Mom and Dad, thank you so much for your love and selfness support in all these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter covers the general overview of the project. Section 1.1 is an introduction to the motivation of the project. We discuss the need for "virtual metrology" approaches to predict measurement outcome to facilitate operational improvement in testing or in the use of test information to improve yield. Section 1.2 states the contributions of this thesis. First, we develop a novel spatial variation recovery and modeling method based on matrix completion. Second, we build machine learning models to predict the outcomes of subsequent process steps and tests based on early data. Section 1.3 previews the structure of the thesis.

1.1 Motivation

The process of complex semiconductor manufacturing faces multiple sources of uncertainty. Integrated circuit fabrication at the nanoscale is not entirely deterministic and tiny uncontrollable variations or defects in manufacturing lead to significant uncertainty in the behavior of individual electronic devices and of circuits as a whole.

In order to predict the variations in fabrication, various silicon integrated circuit optimization and analysis approaches have been proposed, and they are playing important roles in the semiconductor industry. To accurately model circuits, these methods rely on testing data from both devices and circuits. However, integrated circuit testing is not free. The cost of semiconductor characterization can be divided into two groups:

- hundreds of test structures, such as ring oscillators, must be carefully designed, fabricated, and measured;

- complex test process are designed, requiring long test time and expensive test machines.

The first cost is due to the variation, especially spatial variation on wafers or dies[13]. To reduce the cost of measurement, the idea of virtual metrology and related methods has been proposed [14] [15]. With virtual metrology, instead of measuring some "expensive" variables, mathematical and statistical models are constructed to predict them from more readily available fabrication parameters or sensor data. In this project, the idea of virtual metrology can be further refined. In the semiconductor industry, devices of the same type are produced many times. Dies of the same type are aligned on a wafer and those wafers are fabricated every workday. This kind of repeatability in the semiconductor industry leads to our question: for a specific parameter, can we propose a virtual metrology system to sample some of the wafers or dies and measure the parameters on them, and use these to predict the measurement outcome of other key parameters?

For the second cost, a similar question is proposed: can we predict the result of the expensive test process based on previous manufacturing and early stage data to facilitate operational improvements?

1.2 Contributions

The contributions of this thesis are twofold. The first contribution is to develop a new spatial variation estimation and modeling method based on recent progress in compressed sensing. A matrix completion method is used to recover the pattern of spatial variation from a limited number of samples. Compared with the previous methods in this area, the matrix completion technique can handle large scale problems faster and more efficiently. This method is tested on contact resistance measurement data from [7].

The second contribution is a method for predicting the result of expensive semiconductor tests based on early data. More specifically, the problem in this part is to

make prediction using data stream that suffers from concept drift and class imbalance. Both batch and online learning methods are studied. The research outcomes in this part are demonstrated based on data from manufacturing process at SanDisk Shanghai.

1.3 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 presents a high level literature review on the relevant publications from both academia and industry under the topic of variation modeling in semiconductor manufacturing. Chapter 3 provides the details of our matrix completion method for spatial variation estimation. The method proposed in this chapter is tested on a set of contact resistance spatial variation data. In Chapter 4, the problems from SanDisk Shanghai are discussed. In this section, novel machine learning techniques to handle semiconductor manufacturing data with both class imbalance and concept drift are presented. Experimental results demonstrate significant yield improvement potential using real data from industry. Finally, Chapter 5 summarizes the thesis and suggests next steps for future research.

Chapter 2

Related Work

This chapter presents a brief review of the literature on machine learning approaches in the semiconductor industry. Section 2.1 covers the topic of virtual metrology. The modeling of spatial variation and the virtual probe approach are discussed. Section 2.2 surveys previous work on machine learning methods as applied to yield modeling in semiconductor manufacturing.

2.1 Virtual Metrology

Advanced semiconductor manufacturing is subject to multiple sources of uncertainty. Integrated circuit fabrication at the nanoscale is not entirely deterministic, and variations in manufacturing lead to significant uncertainty in the behavior of individual electronic devices and of circuits as a whole. In order to predict and reduce the impact of variations in fabrication, various silicon integrated circuit analysis and optimization approaches have been proposed, and they are playing important roles in the semiconductor industry. To accurately model circuit performance and yield, these methods rely on testing data from fabricated devices and circuits. However, integrated circuit testing is not free. Large numbers of test structures, such as ring oscillators, must be carefully designed, fabricated, and measured and thus the cost is considerable. To reduce the cost of measurement, the idea of virtual metrology [16][17][18] and related methods have been proposed. With virtual metrology, instead of actually measuring some "expensive" variables, mathematical or statistical models are constructed to

predict them from fabrication parameters or sensor data.

An efficient virtual metrology system requires [17] high prediction accuracy, low computational time and cost for fast process control and interpretability for fault detection. The challenges in developing virtual metrology systems for semiconductor manufacturing process are [17] high dimensionality, data fragmentation, time series data and multiprocess modeling.

2.1.1 Spatial Variation Modeling

This section focuses on the previous research on virtual metrology systems for spatial variation modeling, which is related to the research in Chapter 3.

Spatial Variation in Semiconductor Manufacturing

Semiconductor manufacturing is a highly cost-sensitive high-tech industry sector. With the size or complexity of the wafers or dies increasing and scale of the devices shrinking, it becomes more and more important to maintain the uniform performance of devices across a wafer or die. Variation, especially spatial variation, is a critical issue to be analyzed in semiconductor manufacturing [13][19][2][20].

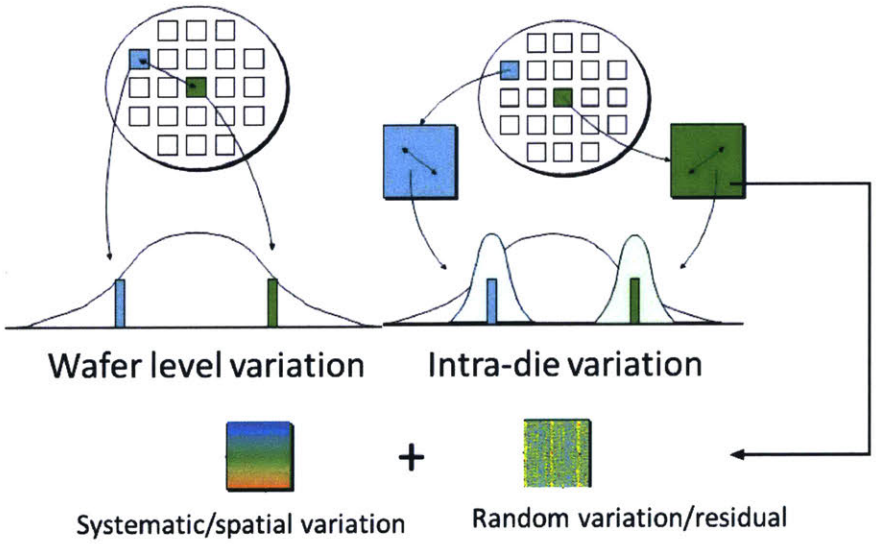


Figure 2-1: Wafer level variation and intra-die variation, extended from [1].

The variation in semiconductor manufacturing can be roughly divided into four

categories with different scales: lot-to-lot variation, wafer-to-wafer variation, intra-wafer (die-to-die) variation and intra-die variation [2]. The lot-to-lot variation has a temporal nature and can be compensated by statistical process control approaches. Wafer-to-wafer variation can be temporal or spatial. The temporal part is due to the drift in the process. Spatial inter-wafer variation may occur due to temporal drift in sequential single-wafer processes, or due to spatial non-uniformity in batch processing for example when a batch of wafers are placed at different positions within a batch furnace having non-ideal uniformity.

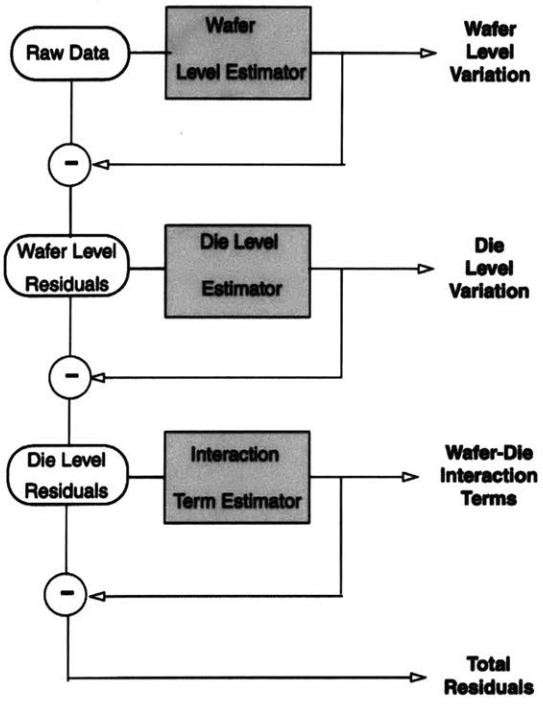


Figure 2-2: Variation decomposition flow in [2].

Wafer level die-to-die variation and intra-die variation, however, are often strongly spatial in nature. The spatial variation in this project refers to wafer level die-to-die and intra-die variations. The source of the spatial variation is complicated. Typical sources of wafer level inter-die variation include thermal gradients in etching or post exposure baking [21], and deposition variation in chemical vapor deposition [22]. The wafer level variation often shows some spatial structure such as a stripe or radial pattern. Intra-die variations are mainly due to the layout's interaction with the fabrication process [2] [23] [24] [25]. Due to the different nature of wafer level and

die level variation, it is important to separate the variations and to analyze them individually. Figure 2-2 is a diagram showing the general framework of a decomposition algorithm [2]. When the raw data is received, the algorithm decomposes the variation into wafer level variation, die level variation and wafer-die interaction terms sequentially using a hierarchical estimator.

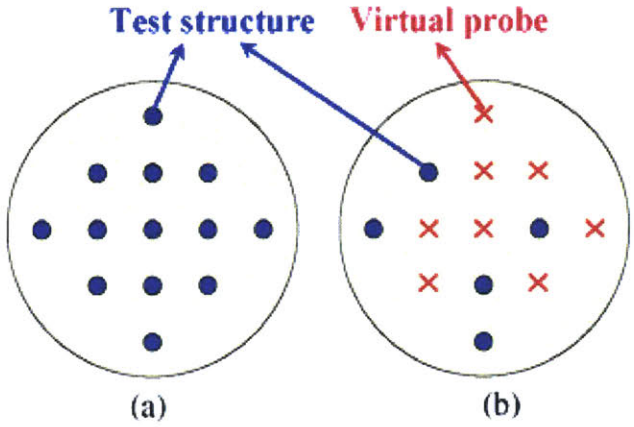


Figure 2-3: (a) Test structures are built at each point to capture the variation; (b) With virtual probe, only part of the points are measure while rest of them are estimated [3] [4].

Virtual Probe

To monitor the spatial variation, virtual metrology techniques are proposed in [3] and [4] as a "Virtual Probe".The Virtual Probe is a statistical method that recovers the pattern of spatial variation from limited number of random samples. The key idea of Virtual Probe is to treat the spatial variation pattern on wafers or dies as a 2D signal $g(x,y), x \in \{1,2,...,P\}, y \in \{1,2,...,Q\}$ and estimate its 2D discrete cosine transform $G(u,v)$. Suppose we have M observations at $(x_1,y_1),(x_2,y_2),...,(x_M,y_M)$. The relationship between any $g(x_m,y_m)$ and $G(u,v)$ is given by the inverse 2D discrete cosine transform:

$$g(x_m,y_m)=\sum_{u=1}^{P}\sum_{v=1}^{Q}\alpha_u\beta_v G(u,v)\cos(\frac{\pi(u-1)(2x_m-1)}{2P})\cos(\frac{\pi(v-1)(2y_m-1)}{2Q}), \tag{2.1}$$

where

$$\alpha_1 = \sqrt{\frac{1}{P}}, \beta_1 = \sqrt{\frac{1}{Q}}, \text{and } \alpha_u = \sqrt{\frac{2}{P}}, \beta_v = \sqrt{\frac{2}{Q}} \text{ for } u, v \geq 2.$$

If we vectorize the matrix $G(u, v)$ into η,

$$\eta = [G(1,1), G(1,2), ..., G(P,Q)]^T, \tag{2.2}$$

Then equation 2.1 can be written in a matrix form

$$A\eta = B, \tag{2.3}$$

where

$$B = [g(x_1, y_1), g(x_2, y_2), ..., g(x_m, y_m)]^T,$$

$$A = \begin{bmatrix} A_{1,1,1} & x_{1,1,2} & x_{1,1,3} & \dots & x_{1,P,Q} \\ x_{2,1,1} & x_{2,1,2} & x_{2,1,3} & \dots & x_{2,P,Q} \\ \dots & \dots & \dots & \dots & \dots \\ x_{M,1,1} & x_{M,1,2} & x_{M,1,3} & \dots & x_{M,P,Q} \end{bmatrix},$$

$$A_{m,u,v} = \alpha_u \beta_v G(u,v) \cos(\frac{\pi(u-1)(2x_m - 1)}{2P}) \cos(\frac{\pi(v-1)(2y_m - 1)}{2Q}).$$

Then an L_1 condition is applied to η to ensure sparsity

$$\begin{aligned} &\min_{\eta} ||\eta||_1 \\ &\text{such that } A\eta = B \end{aligned} \tag{2.4}$$

where the L_1 norm minimization can be interpreted as maximum a posteriori estimation under a prior of η with Laplace distribution.

This method is shown to be very effective on wafer level variation data [3]. However, the main drawback is its complexity, which makes it inefficient on large scale data. Note that the dimension of η is $P \times G$. For example, as in Chapter 3, when we have an intra-die variation pattern with $P = 256$, $Q = 144$, the dimension of η is over 30000. It is time consuming to solve the optimization problem at this scale even with the fastest L_1 minimization solver.

2.1.2 Other Virtual Metrology Systems

Other virtual metrology (VM) systems have been proposed to predict and estimate costly-to-measurable or non-measurable metrology variables based on data and parameters from the manufacturing process [16][17]. Virtual metrology provides instant information of the wafers' or devices' quality to achieve real time run-to-run control. Various virtual metrology methods have been explored different processes in semiconductor manufacturing, as summarized in Table 2.1.

For example, in [18] the authors propose a virtual metrology system based on a neural network to predict the quality of production wafers in chemical vapor deposition (CVD) using parametric data from production equipment. Similar problems in CVD are also studied in [26]. An automatic virtual metrology system with a model-creation server and many automatic virtual metrology servers is proposed in [27], and the system is applied in CVD process.

In [28], various data mining techniques are implemented in virtual metrology systems and tested on etching process data. The experiments show the potential of accurate metrology measurement prediction and fault detection. Reference [29] implements principle component analysis to select the variables and build multiple linear regression and artificial neural networks to predict wafer etch rate in plasma etch processing. Neural network based feature selection methods are studied in [30] to enhance virtual metrology accuracy.

Process	Virtual Metrology in Literature
Chemical Vapor Deposition (CVD)	[18], [26] and [27]
Etching	[28], [29] and [30]
Photolithography	[31]

Table 2.1: A brief summary of virtual metrology systems for typical processes in literature.

2.2 Machine Learning Methods in Semiconductor Manufacturing

With the rapid progress in artificial intelligence, machine learning and statistical techniques are widely applied in predicting the outcomes of manufacturing process and measurement tests. The large amounts of data generated by electrical tests, measurement of physical parameters, and collection of equipment and process parameters are used to forecast the yield and defects in fabrication. In [32], Weiss, Dhurandhar and Baseman predict an indicator of microprocessor speed by physical measurements, lithographic metrology, and electrical measurements in the process. State-of-the-art models, including boosted tree, SVM and hidden Markov models, are applied and compared in this research. This research shows that predictions from early test results enable engineers to take corrective compensation actions for better yield.

Neural networks and deep learning are the highly popular methods in recent machine learning research, and these methods have also been applied to semiconductor manufacturing. For example, fuzzy neural networks are used to predict the die yield of wafer fabrication in [33]. Deep learning frameworks are proposed in [34] to extract key underlying features from test results and leverage the features obtained to achieve partial and adaptive testing.

Another important application of machine learning techniques in semiconductor manufacturing is fault detection, which identifies the cause of failure. Fault detection helps process engineers to identify the root cause of anomalies in a complex manufacturing process [17]. He and Wang in [35] develop a k-nearest neighbors (kNN) rules-based fault detection method. The performance of the proposed method is demonstrated by detecting faults in plasma etch. This method is further improved by combining with principle component analysis in [36] to handle large-scale data. A semi-supervised one-class SVM is used in [37] to detect faults in reactive ion etching (RIE) systems.

Reference [5] is a work related to Chapter 4. The author discusses machine learning approaches for yield modeling in semiconductor manufacturing. This work presents generalized linear model (GLM) strategies for yield forecasting using defect metrology data. The GLM models in this work spans wafer and die level data and outperforms

existing model such as Seeds' yield m dodel. A three-stage (lots, wafers and dies) nested structure is constructed to improve capabilities of prediction, as shown in Figure 2-4. This research also describes a methodology to combine GLM models with classification and regression trees (CART).

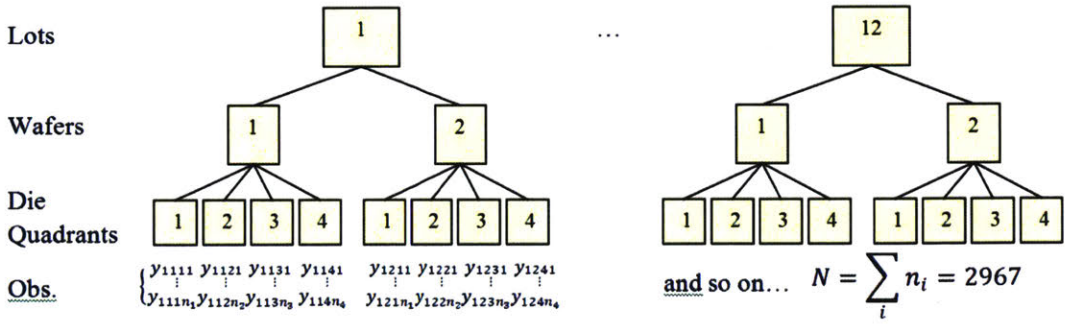


Figure 2-4: A three-stage nested structure for dies within wafers and lots in [5].

There is also considerable research on yield prediction based on machine learning approaches, which is related to our work in Chapter 4. Reference [38] is a high level review of this topic. In [39], Skinner et al. compare many methods, including PCA, cluster analysis, regressions and CART, for application in using the probe test data to determine the cause of low yield wafers. A machine learning based system for earlier detection of faulty wafers is proposed in [40]. Tong and Chao demonstrate a General Regression Neural Network (GRNN) based yield model to predict wafer yield with clustered defects [41]. For final yield prediction, a stepwise support vector machine based model is presented in [42] to classify high-yield and low-yield lots. Park et al. [43] develop a three-stage data mining framework for packaging yield prediction including data preprocessing, feature selection using random forest and classification by nonlinear SVM. A die-level machine learning model is proposed in [6] to predict the failure in the final test using four wafer map features: (a) the distance between the die and the wafer center; (b) yield at the position of the die; (c) adjacent dies' wafer test failure rate and (d) abnormal wafer map patterns. Figure 2-5 in [6] shows the final yield at different positions on a wafer associated with the die-level learning model.

Machine learning methods as reviewed here have shown benefits to the semiconductor industry. However, these existing methods have not addressed important

Figure 2-5: Final yield at each die position on a wafer in [6].

challenges including class imbalance and concept drift that are common in semiconductor manufacturing applications. Methods developed in this thesis to overcome these challenges are presented in Chapter 4.

Chapter 3

Spatial Variation Modeling

In this chapter, a matrix completion approach to spatial variation estimation is discussed. The problem is stated in Section 3.1. Then the concept of matrix completion is presented in Section 3.2. Section 3.3 provides the details of the algorithm used to solve the spatial variation problem. Numerical test results are displayed and discussed in Section 3.4 and generalization of the approach is discussed in Section 3.5. Finally, Section 3.6 is a brief summary of this chapter.

3.1 Problem Set Up

As discussed in Section 2.1.1, the method in this section is proposed to reduce the cost of spatial variation characterization in semiconductor manufacturing. More specifically, a method is desired to estimate the spatial variation pattern on a die or wafer from limited random samples on it (see Figure 2-3).

3.2 Matrix Completion

Matrix completion is an intriguing problem whose goal is to fill the missing entries in a matrix that is partially observed. Obviously, this problem is underdetermined if there are no restrictions on the matrix. To impose sparsity on the matrix, its rank is always used as a minimization objective function. An intuitive analogy of rank is the L_0 norm for a vector, which is the number of non-zero entries in a vector. To solve L_0 norm optimization problems on a vector space, the state-of-the-art methods are

based on L_1 norm (sum of the absolute values of the vector) relaxation to achieve both convexity and sparsity. Similarly, the classical way to solve matrix rank minimization problem is to use the "L_1 norm relaxation" of the rank. This norm is known as the nuclear norm of the matrix. For any $m \times n$ matrix X, a singular value decomposition exists:

$$X = U\Sigma V^T, \tag{3.1}$$

where U and V are orthogonal matrices and Σ is a diagonal matrix

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & \\ & \ddots & & \\ \vdots & & \sigma_k & \vdots \\ & & & 0 \\ & & & \ddots \\ 0 & & \cdots & 0 \end{bmatrix}.$$

Each $\sigma_i > 0$ is defined as the singular value of the matrix and the number of the strictly positive singular values, k, is the rank of the matrix. The nuclear norm, the relaxation of the rank, is defined as the sum of the singular values

$$||X||_* = \sum_{i=1}^{k} \sigma_i. \tag{3.2}$$

Similar to the L_1 norm for vectors, the nuclear norm is known to be convex for matrices and thus can serve as a convex relaxation of the rank.

3.3 Algorithm

Instead of using the discrete cosine transform and determining the frequency coefficients by solving an constrained L_1 norm minimization problem as mentioned in Section 2.1.1, another compressed sensing approach to recover a 2D signal is matrix completion (since we can treat the variation on a wafer or die as a matrix). More specifically, if we suppose the true matrix of full measurement is M, then given the

observations of some of its entries $M_{i,j}$, $(i,j) \in \Omega$ we estimate the whole matrix by finding the matrix with lowest rank under the constraint

$$\begin{aligned} &\min_X \text{rank}(X) \\ &\text{Subject to } X_{i,j} = M_{i,j}, (i,j) \in \Omega. \end{aligned} \tag{3.3}$$

Though this might seem to be straight-forward, this problem is known to be NP-hard. To make the problem tractable, an approximation is to focus on the convex relaxation of rank(X), the nuclear norm. The nuclear norm of matrix X , $||X||_*$, is defined as the sum of the singular values of X. The convex relaxed version of the problem is thus

$$\begin{aligned} &\min_X ||X||_* \\ &\text{Subject to } X_{i,j} = M_{i,j}, (i,j) \in \Omega. \end{aligned} \tag{3.4}$$

Candès and Recht [44] prove that when M is under certain conditions, if over $Cn^{5/4}r\log n$ randomly picked entries of M are observed, the above problem's solution X^* will be equal to M, with very high probability. Here we assume $n \geq m$ and the rank of M is r . If we can assume that there exists noise in the measurement, this problem can be further relaxed to a Lagrangian form

$$\min_X ||X||_* + \frac{\lambda}{2} \sum_{(i,j)\in\Omega} |X_{i,j} - M_{i,j}|^2 \tag{3.5}$$

with $\lambda \to \infty$.

Ma, Goldfarb and Chen [45] proposed Algorithm 1, a fast fixed point continuation algorithm to solve (3.6), a more general version of (3.5), based on singular value thresholding:

$$\min_X \mu||X||_* + \frac{1}{2}||\mathscr{A}(X) - b||_2^2, \ \mu \to 0. \tag{3.6}$$

Here $\mathscr{A}(X)$ is a linear transform mapping X to a vector. It is easy to see that (3.5) is a special case of (3.6) with $\mu = 1/\lambda$. Generally, with smaller μ_{final}, we have higher accuracy due to harder constraints, but we also need more iterations.

Algorithm 1 Solving the minimization in (3.6)[45]

Initialized$X = X_0, \mu_1 > \mu_2 > ... > \mu_{\text{final}}$
for $\mu = \mu_1, \mu_2, ..., \mu_{\text{final}}$ **do**
  while not converge **do**
    Pick $\tau > 0$
    $Y = X - \tau \mathscr{A}^*(\mathscr{A}(X) - b)$
    Do SVD: $Y = U\text{diag}\{\sigma_i\}V^T$
    For each σ_i, $s_{\tau\mu}(\sigma_i) = \max\{\sigma_i - \tau\mu, 0\}$
    $X = U\text{diag}\{s_{\tau\mu}(\sigma_i)\}V^T$
  end while
end for

3.4 Numerical Experiments

In this section, we present numerical experiments of this matrix completion method on real silicon measurement data. Our data are transistor contact resistance measurements from 24 dies fabricated in a 90 nm CMOS process [7]. Figure 3-1 [7] is the structure of the test device, where the yellow contact is to be characterized from I_F, V_{OUTL} and V_{OUTH}. The test chips are specifically designed to study the characteristics of contact plug resistance variability. We have $256 \times 144 = 36864$ measurements on each chip.

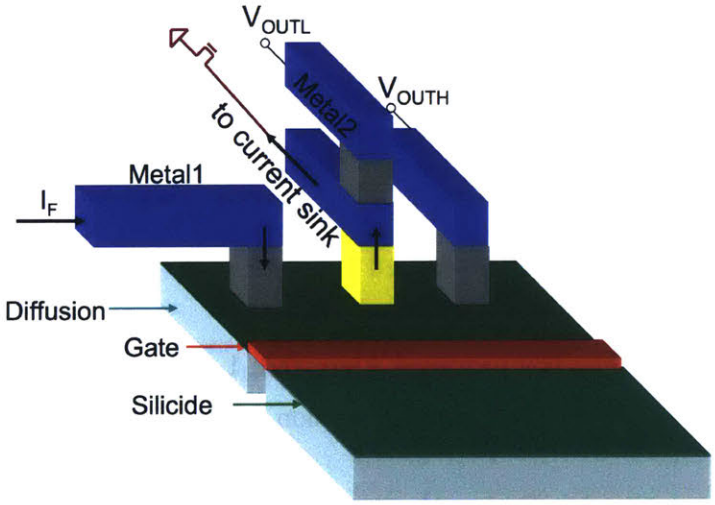


Figure 3-1: The structure of the test device [7].

Figure 3-2 shows the measured contact resistance matrix M, and the recovered contact resistance X. Here we obtain X by low-rank matrix completion with 40% of

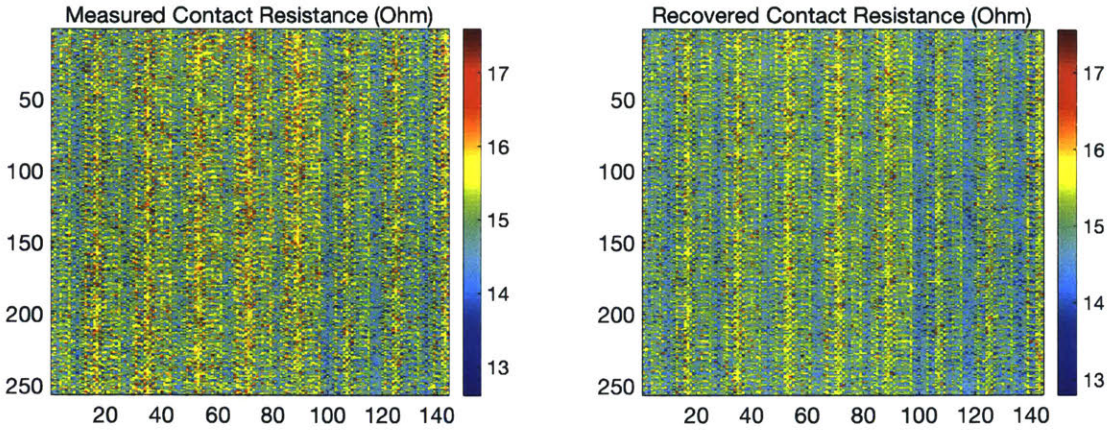


Figure 3-2: Measured contact resistance and recovered contact resistance using matrix completion with 40% (14746 out of 36864) entries observed.

the entries selected uniformly at random from M, and $\mu_{\text{final}} = 0.001$. The relative error is $||X - M||_F / ||M||_F = 0.48\%$, where the Frobenius norm of matrix A is defined as

$$||A||_F = (\sum_{i,j} A_{i,j}^2)^{1/2}.$$

As expected, the relative error decreases as we sample more entries on the matrix.

Figure 3-3 displays the relative error $||X - M||_F / ||M||_F$ obtained with different percentage of entries observed. Clearly, with more entries observed, the relative error of recovered matrix decreases and the error is 0 if all entries are observed.

3.5 Generalization

Additionally, with the algorithm solving (3.6), we can generalize (3.5) and add regularization terms when more information is available. For example, in our data we have 55 layout patterns or device types. These 55 devices types are combinations of five key layout design parameters: contact-to-gate distance (d_{cg}), contact-to-diffusion edge distance (d_{cd}), metallization layer to contact overlap for the y-dimension (d_o), the number of contacts in the source diffusion region (N_s), and the number of contacts in the drain diffusion region (N_d) [7]. We can add a small regularization term to (3.5) to exploit such additional information. Among the un-observed entries, we can make approximate estimates if there exist entries of the same layout pattern or device type

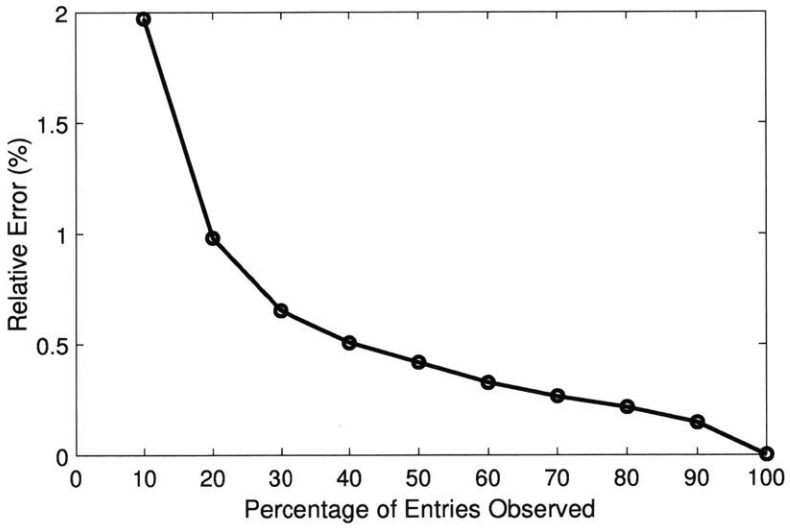


Figure 3-3: Relative error $||X - M||_F/||M||_F$ obtained with different percentage of entries observed.

that are observed. We denote these entries Ω'. The approximate estimation for the device type component (complementing the spatial component) can be determined by the average of the observed entries with the same type. The resulting minimization problem is shown in (3.7) and we can easily transform it into (3.6)

$$\min_X ||X||_* + \frac{\lambda}{2} \sum_{(i,j)\in\Omega} |X_{i,j} - M_{i,j}|^2 + \frac{\eta}{2} \sum_{(i',j')\in\Omega'} |X_{i',j'} - \hat{X}_{type(i',j')}|^2. \quad (3.7)$$

Here $\hat{X}_{type(i',j')}$ is the average of the observed entries with the same type with (i', j').

In Figure 3-4, we fix λ at 100, which means $\mu_{final} = 0.01$, and sample rate at 40%, while changing η. It is seen that with information about layout pattern types and with η properly chosen, we can further reduce the error.

3.6 Summary

In this chapter, a novel technique is proposed to handle the large scale spatial variation recovery problem. Based on recent progress in matrix completion, this new method recovers the spatial variation pattern from a limited number of samples by finding a low rank matrix consistent with available entries. The optimization problem is

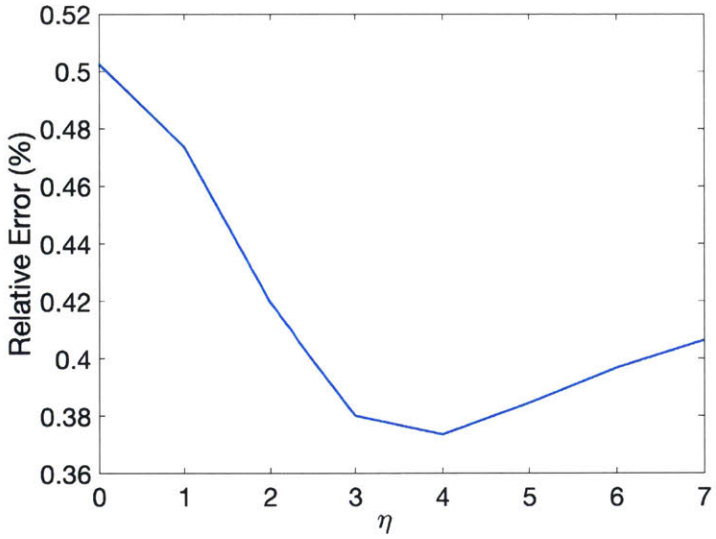


Figure 3-4: Relative error obtained by different η. Here λ is fixed at 100 ($\mu_{final} = 0.01$) and sample rate at 40%.

relaxed to a nuclear norm version and solved by an iterative algorithm. Numerical experiments show 0.48% relative error recovery of the spatial variation with only 40% of the entries observed. The method is further generalized to include type information of the devices. The use of matrix completion opens the door to new approaches for virtual sensing of spatial and device type variation. Future work will further explore and compare efficiency of our results with existing methods, and seek to understand limitations of this approach.

Chapter 4

Temporal Variations, Online Learning and Incremental Learning

This chapter studies temporal variation in semiconductor manufacturing. The data used in this chapter comes from an industrial integrated circuit manufacturer with state-of-the-art fabrication and packaging technologies. The goal of this study is to predict the performances of the flash memory dies from fabrication and early stage testing data. In this chapter, Section 4.1 provides background on the manufacturing process and Section 4.2 gives an overview of the data used. Then we have three sections discussing the challenges in this project as well as the methods proposed. The last section gives a summary.

4.1 Manufacturing Process

SanDisk is a world leading flash memory data storage products supplier, and SanDisk Shanghai (SDSS) is a state-of-the-art assembly, test and packaging facility opened in 2006. The current manufacturing process is illustrated in Figure 4-1 [8]. There are six high-level steps in the manufacturing process, and SDSS is focuses on step 2 (cherry pick) through step 6 (memory test).

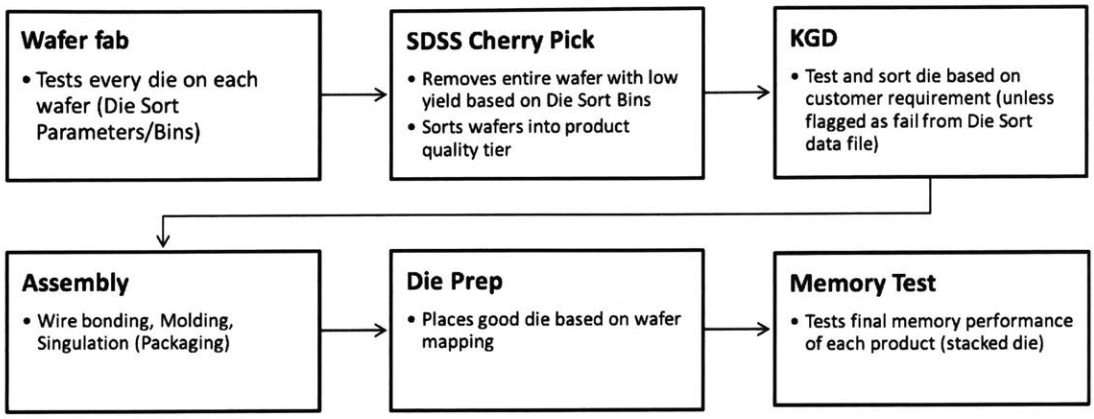


Figure 4-1: SanDisk's manufacturing process as discussed in [8].

4.1.1 Fabrication

The wafers are precessed in wafer fab facilities, after which dies on each wafer go through a standard electrical test process. Based on the results of these tests, wafers and dies are marked with different parameters and bins. The two main tests in the fabrication facility are SME1 and SME2. These two tests are basically the same type of tests at different temperature. Note that in order to reduce cost, SME1 and SME2 are performed in a sequential manner, meaning that dies that fail SME1 will not be tested in SME2. Dies that fail either SME1 or SME2 are marked on the wafer by the fab.

4.1.2 Cherry Pick and Known Good Die (KGD) Test

After fabrication the whole wafers are shipped to SDSS. The first thing in SDSS seeks to do is "cherry pick" in which the wafers are sorted into different quality tiers according to the test data from the fab. Then a further step called Known Good Die (KGD) test is carried out. KGD is product-oriented, and dies from wafers in different tiers undergo different KGD procedures. Those dies on the "prime" wafers identified in the cherry pick go through a strict KGD test and will be packaged into top tier products.

4.1.3 Packaging

After KGD test, wafers are taped, back-ground and diced into singular dies. Then wire bonding attaches the dies to the substrate circuits with other passive components mounted on. Finally, multiple dies are stacked and packaged with mold compound.

4.1.4 Memory Test

After packaging, a final memory test verifies the high performance functionality of the product in the extreme environment of the real world. Though the dies are packaged into products, memory test also still assesses the performances of each die. The memory test assigns dies into different bins (categories). Hard bins refer to the results of pass or failure in the high performance tests, while soft bins refer to the categories of failure types. In this work we only consider hard bins, which means that for each die, the output of the memory test is pass (good) or fail (bad). Some of the failing dies may be re-screened. Note that all dies in the memory test pass earlier electrical tests already but some of them still fail the high-performance memory test after packaging. In other words, there exists "actually" bad or low-performance dies despite being judged by the electrical test as good or high-performance, and with the limitations of the existing electrical tests decision process they are only detected later by the final memory test.

Because the dies are sequentially connected in the package, if one die in the package fails the memory test, the whole package is considered to fail. The company does not necessarily discard such "failed" packages, but rather downgrades them to low-end products with less profit. Since the total number of packages is fixed, to achieve high profit, the company desires more high-end products. Because a single fail die will lead to the failure and downgrading of the whole package, good or high performance dies stacked in that package may be wasted. If we know that the failure probability of a single die is $p_{\text{die fail}}$, which is usually very small, the failure rate of the final packages will be

$$p_{\text{package fail}} = 1 - (1 - p_{\text{die fail}})^s \approx s p_{\text{die fail}}, \tag{4.1}$$

where s (= 4, 8 or even 16) is the number of dies in a package. Thus, the failure rate of the product would be much higher than the failure rate of dies.

Also, the packaging process of high end products and low end products are different. Though it is possible to downgrade a packaged high end product to a low end product with additional cost, it is impossible to upgrade a low end product back to the high end category. Since the packaging process is irreversible, if we can predict the result of the memory test, good dies will be stacked together for more high-end products.

4.1.5 Yield Improvement

A good classifier for memory test results before packaging will help to increase yield and profit for the following reasons.

- One can stack predicted good dies together for high end products and package predicted lower performance ("bad") dies together for low end products. If the classifier is good enough, the number of good packages will increase.
- Dies predicted as "bad" can go through the low end product process directly and the cost of downgrading is saved.

For each die, all of its early test results and available fabrication parameters can be used as the input of the classifier; in our case, we have over 1000 features per die. The output of the classifier is a prediction of its memory test result. Figures 4-2 and 4-3 illustrate the packaging and testing process without and with a classifier, respectively. In these two simple examples, 4 out of 16 incoming dies are bad or low-performance (though labeled by the earlier electrical tests as good or high-performance). With a fairly good classifier, high end yield can be improved from 25% to 50%. A similar issue also exists in 3D integrated circuit packaging technology: one chip in a stack fails, the whole package fails. In 3D packaging cases, where different types of chips or devices are stacked in a package, we would need classifiers for each kind of chip.

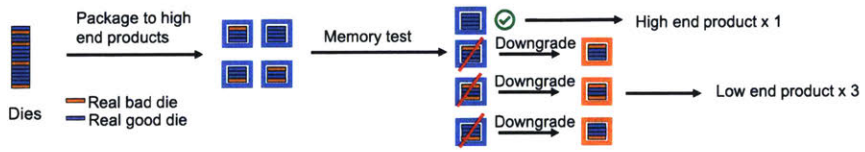


Figure 4-2: The packaging and memory testing process without a classifier.

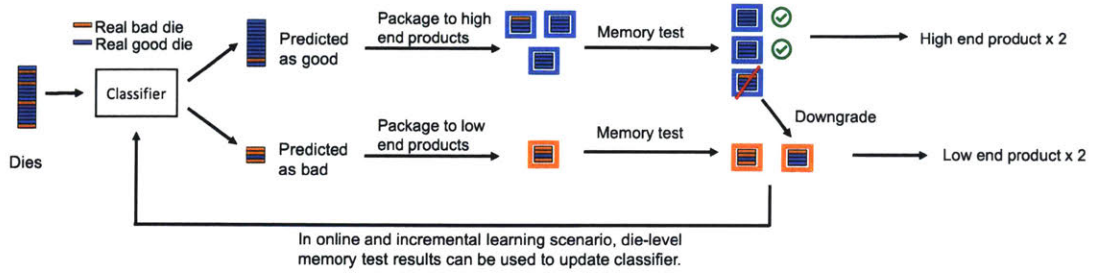


Figure 4-3: The packaging and memory testing process with a classifier, where prediction of final packaged state of the incoming die is improved.

4.2 The Data

The data used in this chapter is the memory test data from previous MIT LGO student Naomi Arnold. The data consists of the results of 924327 die SME1, SME2, KGD and memory tests. The date of SME1 test is between July 19th, 2015 and October 6th, 2015, while the date of memory test is between August 30th, 2015 and October 19th 2015.

For input, each die has around a thousand features, the parameters and measurement results from SME1 (549 features), SME2 (339 features) and KGD (104 features) combined. For output, the result of the memory test is a single numeric variable called FH (factory high temperature) soft bins. If a die passes memory test, the soft bin number is 0, otherwise the bin number is the corresponding category it lands in. In total, 20004 dies failed the memory test, 2.16% of all the dies. For each fail die, only one soft bin is assigned. Figure 4-4 shows the top 12 compositions of the fail dies. Note that according to the memory test engineers from SanDisk, soft bin 535 is actually an outlier and should not be considered [8]. It is also worth mentioning that all the dies in this dataset passed SME1, SME2 and KGD already, or otherwise the memory test result should be null.

The data has been preprocessed and cleansed, with some of the steps mentioned in [8]. Non-numeric features and parameters where most of the dies have null values are deleted. The test date, number of lots, wafers and dies are excluded as features, but some of these parameters are included in certain experiments related to temporal or spatial variations. After cleansing, each die has 997 features, including its position on the original wafer $(x, y$ coordinates).

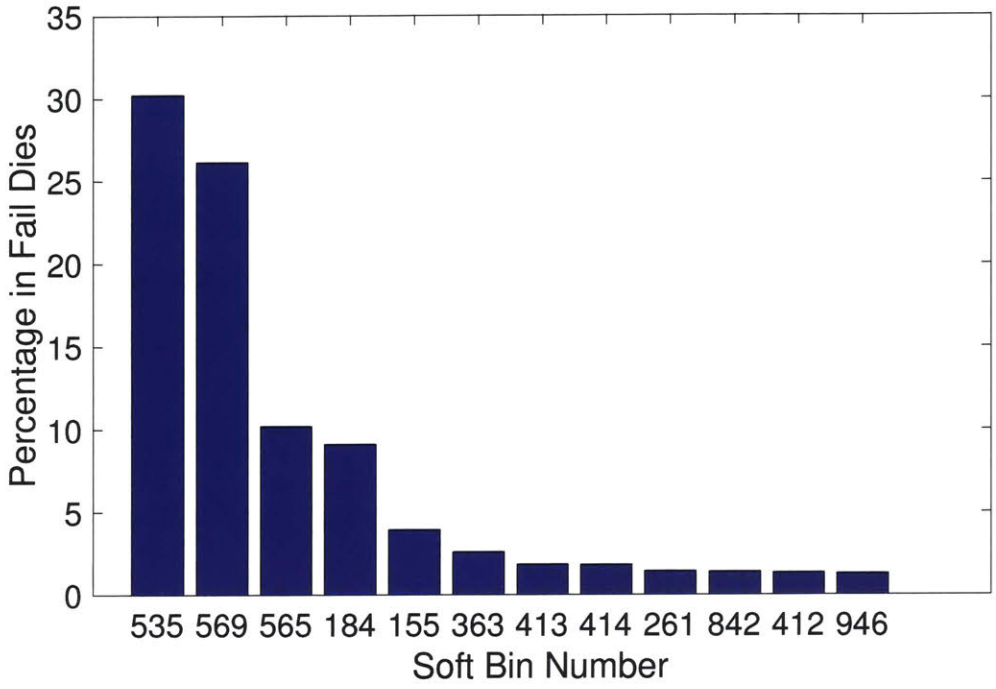


Figure 4-4: The percentage of top 12 soft bins among fail dies. Bin 535 is an outlier.

Since there are over 40 soft bins, each with different failure pattern and some of the soft bins only have less than 10 instances, it is important to select the response variable properly. Based on engineers' advice, in this project, we detect the combination of soft bin 569 and soft bin 565 (569 or 565), which consists of 0.79% of all the dies. Building classifiers on this highly imbalanced dataset will be discussed in Section 4.5.

Another challenge in this real world industrial data is that the performances of the classifier deteriorates over time. Due to subtle changes in the manufacturing process, a classifier trained on old data may not be able to give accurate prediction on new data. The topic of online learning and continuous classifier updating is discussed in Section 4.6.

Finally, it is also important to detect the most important parameters in SME1, SME2 and KGD which influence or serve to predict the outcome of the memory test. Feature selection enables engineers to accurately find the most important parameters in fabrication or early tests and to make appropriate changes. Feature selection, and its interaction with temporal variation, are discussed in Section 4.8.

4.3 Assessment Metrics

With the growing number of different models and algorithms developed in the machine learning and statistics community, it is important to standardize the methods of assessment to compare the performances of various algorithms. This section is a brief summary of metrics typically used in classification problems. Additionally, the metrics and terminology used in this project will be emphasized.

4.3.1 Confusion Matrix

Confusion matrices are perhaps the easiest and most widely used performance metric in classification problems. Suppose we have a classification problem with N possible classes, the confusion matrix is an $N \times N$ matrix visualizing the performance of a supervised learning method. Each column of a confusion matrix represents a possible predicted condition (output of the classifier), while each row of the matrix represents a possible true condition (ground truth). In this project, there are only two possible classes. The confusion matrix and corresponding terms with respect to a binary classification problem are shown in Figure 4-5.

		Predicted condition	
		Predicted positive (fail)	Predicted negative (pass)
True condition	Condition positive (fail)	True positive (TP)	False negative (FN)
	Condition negative (pass)	False positive (FP)	True negative (TN)

Figure 4-5: Confusion matrix and terminology for a binary classification problem.

In our problem, fail dies are labelled as positive in the test and pass dies are labelled as negative. The confusion matrix is constructed based on the results of a classifier on the test set. Each die in the test set must be placed into one cell in the matrix. Fail dies that are correctly predicted as fail are placed into the true positive (TP) cell, and those that are incorrectly predicted as pass are placed into the false negative (FN) cell. Good dies that are correctly predicted as pass are placed into the

true negative (TN) cell, while those that are incorrectly predicted as fail are placed into the false positive (FP) cell. Then we sum up the number of dies in each cell and obtain the confusion matrix of the classifier. FN is also called Type II error and FP is also called type I error.

Based on the confusion matrix, some terminology and derivations are developed. The accuracy (ACC) of the classification is defined as

$$ACC = \frac{TP + TN}{TP + FN + FP + TN}. \tag{4.2}$$

The true positive rate (TPR) is defined as

$$TPR = \frac{TP}{TP + FN}, \tag{4.3}$$

which shows the fraction of positive instances that are detected. The engineers in SanDisk also use the term "Underkill" which is defined as

$$\text{Underkill} = 1 - TPR = \frac{FN}{TP + FN} \tag{4.4}$$

The false positive rate (FPR) or fall-out is defined as

$$FPR = \frac{FP}{FP + TN}, \tag{4.5}$$

which shows the fraction of negative instances that are sacrificed. The engineers in SanDisk also use the term "Overkill" for FPR, or

$$\text{Overkill} = FPR. \tag{4.6}$$

In general, a good classifier is expected to have low value in both Overkill and Underkill.

In the literature on machine learning, some other metrics are often used. For example, the precision of a classifier is defined as

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{4.7}$$

and recall is defined as

$$\text{Recall} = \frac{TP}{TP + FN} = TPR. \tag{4.8}$$

Precision shows the fraction of true positive instances among those labelled as positive by the classifier. We want to have high precision and recall for good classifier. The F-measure is defined as the harmonic mean of the precision and recall:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall},} \tag{4.9}$$

or more generally

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}. \tag{4.10}$$

4.3.2 Receiver Operating Characteristic (ROC) Curve

The Receiver Operating Characteristic (ROC) curve is a graphic illustration of the performance of a binary classifier based on TPR and FPR. Here the binary classifier $h(\cdot, \cdot)$ is assumed to output a non-negative real value, the score of an instance to be classified in a class.

$$h(\cdot, \cdot) : X \times Y \to R. \tag{4.11}$$

More intuitively, we can assume

$$h(x, 1) + h(x, 0) = 1, h(x, y) \in [0, 1]. \tag{4.12}$$

Then $h(x, y)$ can be interpreted as the probability of x to be classified into class y. To make a prediction, a threshold η is set to map the scores to $\{1, 0\}$. For example, if $\eta = 0.6$, x is predicted to be 1 if $h(x, 1) > 0.6$ and to be 0 otherwise. Under each threshold, we have different TPR and FPR. The Receiver Operating Characteristic (ROC) curve is obtained by plotting TPR against FPR under different thresholds. Note that when $\eta = 1$, all instances will be classified as 0, $TPR = FPR = 0$ and when $\eta = 0$, $TPR = FPR = 1$. So an ROC curve must go through $(0, 0)$ and $(1, 1)$.

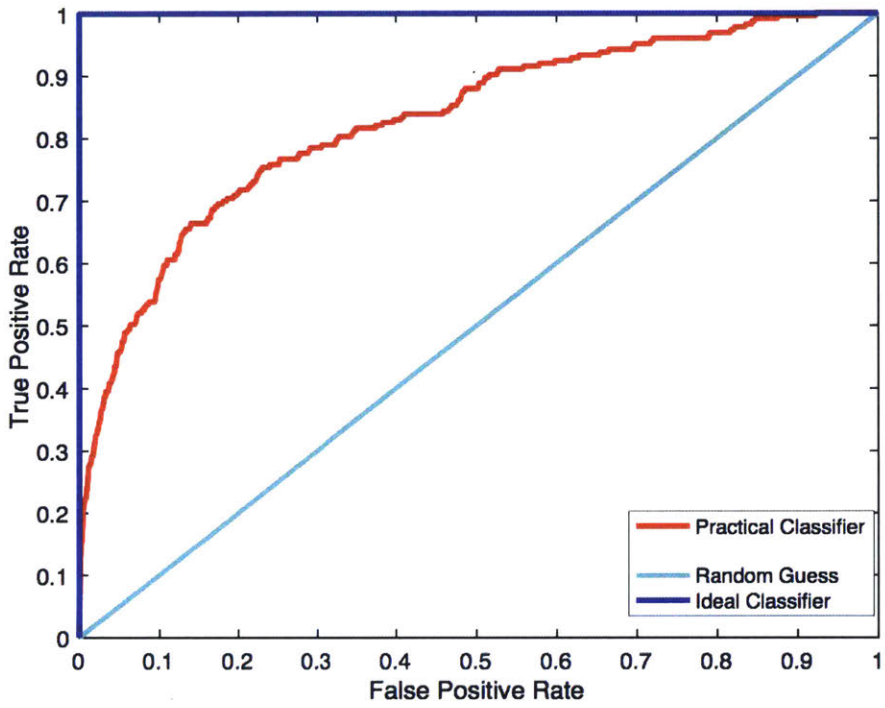


Figure 4-6: The Receiver Operating Characteristic (ROC) curves. The ROC curve of an ideal classifier goes through (0, 1). The ROC curve of a random guess is a straight line from (0, 0) to (1, 1). The ROC curve of a practical classifier is some where in between.

Figure 4-6 shows a number of different possible ROC curves. All ROC curves go through $(0, 0)$ and $(1, 1)$. An ideal ROC curve goes through $TPR = 1, FPR = 0$, which means that under some threshold, we can achieve 100% accuracy classification. A classifier based purely on a random guess is a straight line. A practical classifier is somewhere in between.

The area under an ROC curve (AUC) is a useful metric for the performances of a classifier, as shown in Figure 4-7. For an ideal classifier, $AUC = 1$; for a random guess, $AUC = 0.5$; and for a practical classifier, $0.5 < AUC < 1$.

4.4 Mathematical Formulation

With a large number of testing dies, we can estimate the underlying probability by the relative frequency. We denote "positive" or fail by 1, and "negative" or pass by

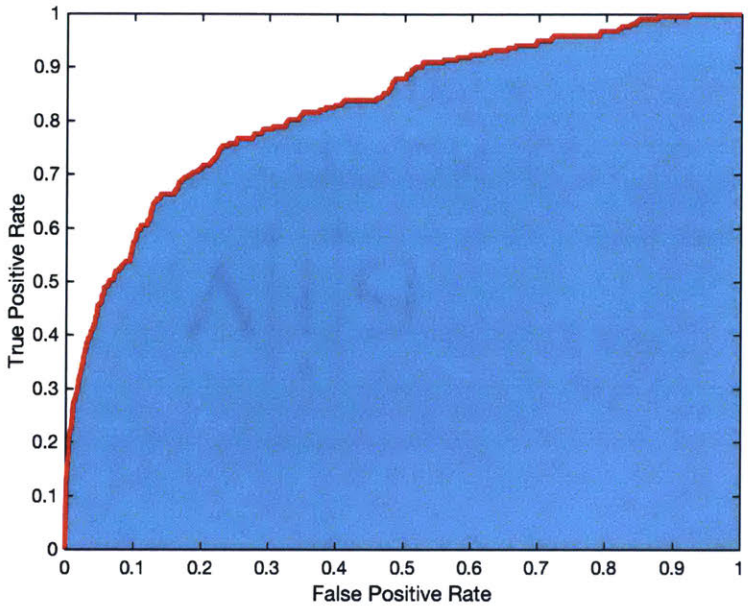


Figure 4-7: The area under ROC curve, or AUC.

0. Let H be the true condition and y be the predicted condition.

$$\begin{aligned} p(H=1,y=1) &= \frac{TP}{N},\ p(H=1,y=0) = \frac{FN}{N} \\ p(H=0,y=1) &= \frac{FP}{N},\ p(H=0,y=0) = \frac{TN}{N} \end{aligned} \tag{4.13}$$

Without any classifiers, if we randomly package the dies into packages or stacks with s die in the stack, the failure rate of the packages is

$$p_{\text{package fail}} = 1 - p(H=0)^s. \tag{4.14}$$

Then if n dies are produced, the expected number of good packages is

$$\mathbb{E}[m_1(s)] = \frac{n}{s} p(H=0)^s, \tag{4.15}$$

where round-off error is neglected (leftover dies $< s$ not packaged). With a classifier to better predict the memory test result before packaging, we can stack the dies predicted as fail together and the dies predicted as pass together. Using the law of total expectation, the expected number of good packages in this case is given by

$$\mathbb{E}[m_2(s)] = \mathbb{E}[\mathbb{E}[m_2(s)|k]] = \mathbb{E}[kp(H=0|y=0)^s] = \frac{n}{s}p(H=0|y=0)^s p(y=0), \tag{4.16}$$

where k is the number of stacks packaged as high end products and k is subject to a binomial distribution $B(\frac{n}{s},\ p(y=0))$. Again round-off error is neglected. If the classifier is ideal, all good dies are packaged into high-end products and the maximum number of high-end products is achieved,

$$\mathbb{E}[m_2^*(s)] = \frac{n}{s}p(y=0) = \frac{n}{s}p(H=0). \tag{4.17}$$

It is easy to prove that $\mathbb{E}m_2^* \geq \mathbb{E}m_1$ and $\mathbb{E}m_2^* \geq \mathbb{E}m_2$. The expected yield improvement is $\mathbb{E}[m_2(s)] - \mathbb{E}[m_1(s)]$ which can be either positive or negative. Note that, TP, TN, FP and FN are uniquely determined by N_{pass}, N_{fail}, FPR and TPR. Subtracting (4.15) from (4.16), plugging in (4.13) and with a little math, we have the expected yield improvement

$$\frac{\mathbb{E}[m_2(s) - m_1(s)]}{n/s} = \frac{[N_{\text{pass}}(1-FPR)]^s}{N[N_{\text{pass}}(1-FPR) + N_{\text{fail}}(1-TPR)]^{s-1}} - \frac{N_{\text{pass}}^s}{N^s}, \tag{4.18}$$

which is a function of TPR and FPR. If $N_{\text{fail}}(1-TPR) \ll N_{\text{pass}}(1-FPR)$ and $N_{\text{fail}} \ll N_{\text{pass}}$, which is the case in this work's classification, (4.18) can be linearized as

$$\begin{aligned}
\frac{\mathbb{E}[m_2(s) - m_1(s)]}{n/s} &= \frac{[N_{\text{pass}}(1-FPR)]^s}{N[N_{\text{pass}}(1-FPR) + N_{\text{fail}}(1-TPR)]^{s-1}} - \frac{N_{\text{pass}}^s}{N^s} \\
&\approx \frac{N_{\text{pass}}}{N}(1-FPR)[1-(s-1)\frac{N_{\text{fail}}(1-TPR)}{N_{\text{pass}}(1-FPR)}] - (1-\frac{sN_{\text{fail}}}{N}) \\
&= (s-1)\frac{N_{\text{fail}}}{N}TPR - \frac{N_{\text{pass}}}{N}FPR.
\end{aligned} \tag{4.19}$$

Here we implicitly assume that the underlying probability distribution $p(H,\ y)$ does not change over time. If it does, as in Section 4.7, Equation (4.18) from old data is not valid for future prediction.

4.5 Imbalanced Classification

An imbalanced dataset can be characterized as having many more instances of some classes, the majority, than other classes, the minority [46]. In our example, the majority consists of passing dies and the minority is the set of failing dies. In most cases, the minority is the class of interest. The main issue of an extreme imbalanced dataset is that it can compromise the performances of the regular learning algorithms [47][48]. In this section we first provide an overview of this challenge of class imbalance. We then present our adaptation of RUSBoost to address this challenge. Finally we present and summarize experimental results applying our classifier to our industrial dataset.

4.5.1 Introduction

Most statistical learning methods are designed to maximize the expected accuracy by empirical risk minimization. To avoid overfitting, a regularization term is often used as a penalty to impose sparsity and simplicity of the model, as in (4.20):

$$\min_f \sum_i L(f(x_i), y_i) + \lambda R(f), \lambda \geq 0 \tag{4.20}$$

where L is the loss, x is the input, $y \in \{1, 0\}$ is the output and $R(\cdot)$ is a regularizer whose minimum is always reached by a constant function. The regularizer is designed to "denoise" the model, leading to a simpler model. However, if the data is imbalanced, say 99% of y_i are 0, the minimum value of (4.20) may be achieved by setting f to be as simple as $f(x) \equiv 0$. That is to say, a "high accuracy" classifier may be achieved simply by declaring or classifying all samples as being in the majority class. Another problem is that, when the data is highly skewed, we may not have enough minority class instances to represent the distribution of the minor class, which leads to a noisy training set.

Imbalanced classification is pervasive in many real world problems, especially when connected with anomaly detection [46], such as in Email fraud detection, medical diagnosis or computer intrusion detections. It is a classical problem with some known state-of-the-art solutions [47], as summarized below.

Without any changes in the learning algorithm, we need to modify the dataset

itself to provide a balanced training set for the classifier. In a balanced dataset, the number of instances in the two classes are not necessarily exactly the same, but contain sufficient numbers of both classes to enable effective training. The easiest and most intuitive ways to alleviate class imbalance are oversampling the minority class, or undersampling the majority class. Though these seem to be equivalent in function, oversampling and undersampling each have problems that may be harmful to the performance of the learning. For undersampling, it is obvious that reducing the number of the training instances means that some important patterns or information may be lost and the data may become noisy. Oversampling, on the other hand, can lead to increase in computation time and to overfitting due to a large number of identical minority training instances. To tackle the problems of simple oversampling and undersampling, more sophisticated techniques have been developed. For example, instead of oversampling the minority class, SMOTE (the Synthetic Minority Oversampling Technique) generates synthetic data by k-nearest neighbors approach [49]. Based on SMOTE, various adaptive sampling techniques have been proposed [50][51].

Sampling techniques can be integrated with ensemble methods to interact with the models. Ensemble methods combine the results from multiple classifiers (base classifiers) to achieve better performance. One of the most famous ensemble methods to combine classifiers is AdaBoost [52][53]. EasyEnsemble and BalanceCascade [54]are proposed to train each classifier on independent sampling sets and combines them in an AdaBoost fashion. Shortly after SMOTE, SMOTEBoost [55] was proposed to embed SMOTE in an AdaBoost framework. Prior to the training step of each base classifier (that is, at the beginning of each iteration), SMOTE is applied to generate a new balanced dataset.

Instead of creating a balanced data set for the model to train, an alternate approach is to create various loss function that impose a larger penalty on the error in the minority class. For example, we could make $L(0,1) > L(1,0)$ as in (4.20). With the asymmetric loss metric, the classification can be transformed into a Bayesian hypothesis testing problem. Other approaches also exist; cost-sensitive decision trees or cost-sensitive neural networks [56] are also used in the community. We note that, customized loss functions can be easily embedded into the AdaBoost framework to create new ensemble methods.

4.5.2 RUSBoost

In this project, RUSBoost is used to tackle the problem of imbalanced dataset in classification. Here RUS stands for Random Under Sampling [57]. RUSBoost is a widely used hybrid approach embedding undersampling in an AdaBoost framework. The main advantage of random under sampling over SMOTE is its simplicity. In SMOTEBoost, SMOTE is used to generate new data using K-nearest neighbor before each iteration. Additionally, the training set for each base classifier is much larger than the original training set due to oversampling. These two problems are exaggerated in SMOTEBoost because multiple base classifiers are trained. Similar to SMOTEBoost, RUSBoost generates a new dataset for each base classifier by random undersampling the majority class. However, random undersampling is much faster than K-nearest neighbors, and undersampling results in a smaller data set for training.

Algorithm 2 AdaBoost

1: **input**: S instances: $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m), y_i \in \{1, 0\}$
2: base classifiers $h_1, ..., h_T$
3: **initialize** $D_1(i) = \frac{1}{m}$ for any i
4: **for** $t = 1, 2, ..., T$ **do**
5: Train base classifier h_t with S'_t and D'_t
6: Test h_t on S, $h_t(x_i, y_i) \in [0, 1]$
7: Calculate pseudo-loss on S and D_t: $\epsilon_t = \sum_{i:y_i \neq y} D_t(1 - h_t(x_i, y_i) + h_t(x_t, y))$.
8: $\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$
9: Update $D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_t,y))}$
10: Normalize $D_{t+1}(i) : D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_j D_{t+1}(j)}$
11: **end for**
12: **final output**: $H(x) = \arg\max_y \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}$

The RUSBoost method is shown as Algorithm 3 [57]. In this algorithm, base classifiers are supposed to generate a "score" or "probability": $h(\cdot, \cdot) : X \times Y \rightarrow [0, 1]$. In practice, there are different implementations for undersampling that can be used in Algorithm 3 [58]. Suppose in our input sample set S that we have $N+$ instances in the majority class ($y_i = 1$), N_- instances in the minority class ($y_i = 0$), $N_+ \gg N_-$ and we want to have CN_- majority instances after undersampling, say $C = 1$ or 2. In one approach we can keep randomly removing samples in the majority class until we have CN_- majority instances left and renormalize the weight. Alternatively, we can

Algorithm 3 RUSBoost

input: S instances: $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m), y_i \in \{1, 0\}$ with majority class $y_i = 0$ and minority class $y_i = 1$;
base classifiers $h_1, ..., h_T$
initialize $D_1(i) = \frac{1}{m}$ for any i
for $t = 1, 2, ..., T$ **do**
    Create new training set S'_t by undersampling the majority class, the weight distribution of the remaining instances is D'_t
    Train base classifier h_t with S'_t and D'_t
    Test h_t on S, $h_t(x_i, y_i) \in [0, 1]$
    Calculate pseudo-loss on S and D_t: $\epsilon_t = \sum_{i:y_i \neq y} D_t(1 - h_t(x_i, y_i) + h_t(x_t, y))$.
    $\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$
    Update $D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_t,y))}$
    Normalize $D_{t+1}(i) : D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_j D_{t+1}(j)}$
end for
final output: $H(x) = \arg\max_y \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}$

sample CN_- majority instances and N_- minority instances based on the probability distribution $D_t(i)$. As a third approach, we could even generate a new data set by sampling the original one based on $D_t(i)$, and then sample the majority class from the new dataset. In our experiments, the first approach is implemented due to its simplicity since the other two approaches require sampling based on a non-uniform distribution $D_t(i)$.

4.5.3 Experimental Results

The inputs of the model are the die-level parameters including test results from SME1, SME2 and KGD. The output of the model is the prediction of the memory test result as discussed in Section 4.1.4. For each die, we have the date of memory test and the date of the SME1 test. The dataset is sorted by SME1 test date, which reflects the date of the fabrication more accurately. The model is implemented in MATLAB and `Python`. We use the data of the dies produced in a short period of time, eight days, to test the algorithms. We assume that no major change in the manufacturing process takes place in this period and thus the concept drift in the data stream over this interval is negligible.

RUSBoost and regular AdaBoost are used to train and to test on the same dataset.

Figure 4-8 displays the ROC curves obtained by AdaBoost and RUSBoost. To detect 15% of the real bad dies using AdaBoost, 80% of the real good dies are labelled as bad. The large amount of false alarms by AdaBoost will decrease the company's profit significantly.

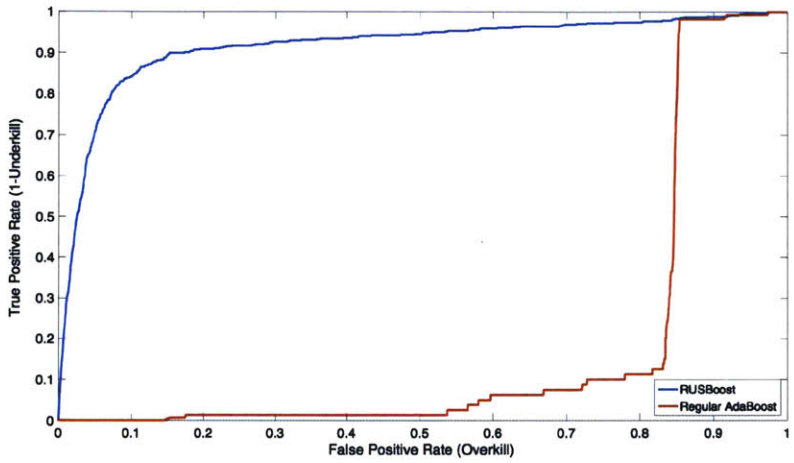


Figure 4-8: The ROC curves obtained by RUSBoost and regular AdaBoost.

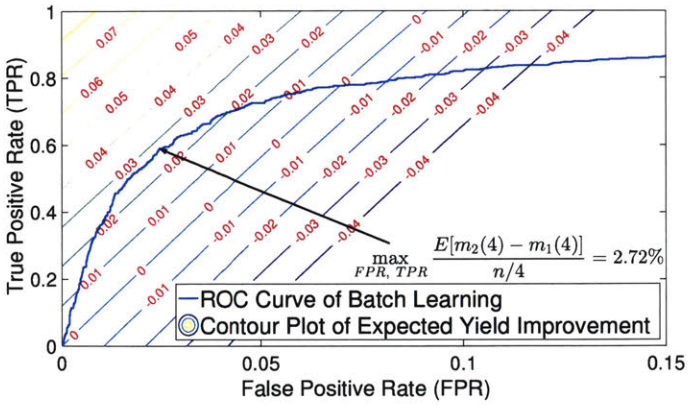
Typical confusion matrices by setting threshold to 0.5 are shown in Tables 4.1 and 4.2. As expected, AdaBoost tends to label all dies as pass to achieve both simplicity and high accuracy. Then only 0.2% of the real bad dies are detected by the classifier and the economic benefit generated is negligible. In contrast, RUSBoost identifies 523 of 634 bad dies, with significant potential economic saving as a result.

	Predicted as fail	Predicted as pass	
True fail	523	111	Underkill=17.5%
True pass	2518	26848	Overkill=8.6%

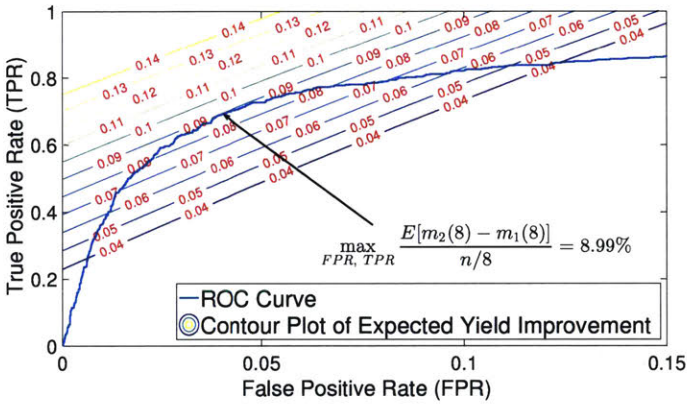
Table 4.1: Confusion matrix using RUSBoost, threshold 0.5.

	Predicted as fail	Predicted as pass	
True fail	1	633	Underkill=99.8%
True pass	6	29360	Overkill=0.0%

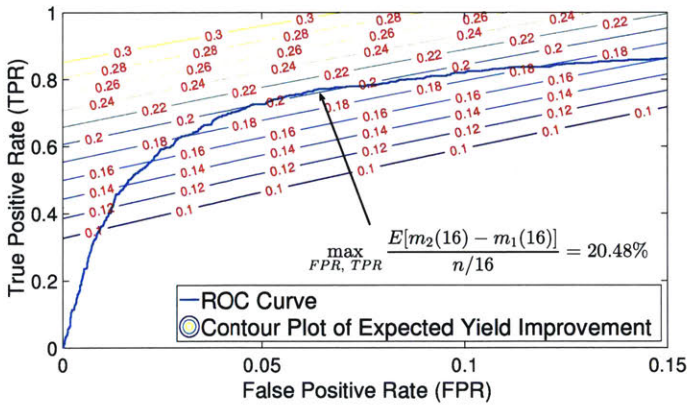
Table 4.2: Confusion matrix using AdaBoost, threshold 0.5.



(a) $s = 4$, yield without a classifier= 88.06%, yield with a classifier= 90.78%, $\gamma^* = 0.75$.



(b) $s = 8$, yield without a classifier= 77.54%, yield with a classifier= 86.53%, $\gamma^* = 0.68$.



(c) $s = 16$, yield without a classifier= 60.12%, yield with a classifier= 80.60%, $\gamma^* = 0.60$.

Figure 4-9: Contour plots of yield improvement $\frac{\mathbb{E}[m_2(s)-m_1(s)]}{n/s}$ with number of dies in a package $s = 4$, $s = 8$ and $s = 16$. The yield improvement is optimized under the constraint of ROC curve that is determined by the classifier. With more dies in a package, we can potentially achieve larger yield improvement.

Recalling Equation (4.18), the expected yield improvement is a function of TPR and FPR. However, FPR and TPR are constrained by the ROC curve of the classifier. Then the optimal point (FPR^*, TPR^*) is where the contour plot is tangent to the ROC curve and the corresponding optimal threshold γ^* is determined for future prediction. Figure 4-9 gives the ROC curve and contour plots of expected yield improvement $\frac{\mathbb{E}[m_2(s)-m_1(s)]}{n/s}$ with different s (number of dies in a package). From the contour plot we can see that our linearization in Equation (4.19) is a good approximation even at $s = 16$, which can be used as a fast estimation of the expected yield improvement.

As expected, with larger s, we can achieve larger yield improvement with more dies in a package. When $s = 16$, the maximum yield improvement is as large as 20.48% for this case of no concept drift, over an eight day period. With smaller s, the yield without a classifier itself is very high, so even 1% of improvement is difficult, though still economically valuable. Also, the optimal threshold γ^* decreases as s increases, which means more dies will be classified as fail. The results are summarized in Table 4.3.

s	Yield without a classifier	Yield with a classifier	Yield Improvement	Optimal threshold
4	88.06%	90.78%	2.72%	0.75
8	77.54%	86.53%	8.99%	0.68
16	60.12%	80.60%	20.48%	0.6

Table 4.3: Yield improvement and optimal classification threshold with number of dies in a package $s = 4$, $s = 8$ and $s = 16$. The yield improvement is optimized under the constraint of ROC curve that is determined by the classifier.

4.5.4 Summary

The batch RUSBoost learning method in this section demonstrates good performance on real data from industry. To determine the optimal classification threshold, we plot the ROC curve with a contour plot of expected yield improvement given in (4.18). The contour plots justify our linearization of expected yield improvement in (4.19). With an optimal classification threshold, at least 2.72% expected yield improvement can be achieved for four or more stack packages.

4.6 Online Learning

Conventional batch machine learning models are trained on a fixed training set. In our problem, however, all incoming new dies that are identified by the electrical test program as functional (no matter if they are predicted as pass or fail die in the memory test) go through the memory test, either as part of a high performance package or as part of a low performance package. Thus we have access to the true labels for all test instances on a continuous though delayed basis. This additional information enables us to further improve our classifier with new test data, as shown in Figure 4-3. Due to the large number of dies produced each day, it is expensive to store all the old training data and retrain a new classifier with both old and new data when more data is available. Therefore we use an online learning model to update our existing model when the final memory test information about new dies arrives. A similar online machine learning model updating technique has also been studied in the electronics CAD community, such as for hotspot detection in [59].

4.6.1 Introduction

Online machine learning is a vibrant sub area in machine learning research. Online learning methods have been proven to be very powerful in applications where training data becomes available with a sequential order. In contrast with traditional batch learning which needs to see the entire training data set before giving an output, online learning requires instant output after seeing each datum. The scenario online learning is illustrated in Figure 4-10 [9]. The online classifier is built to output prediction p_t for input x_t at each time step t, $t = 1, 2, ..., T$. At time t, classifier receives x_t from the world, shown in the figure as coming from "nature." In our project, nature refers to the early test data from SME1, SME2 or KGD. The classifier is asked for a prediction p_t for this x_t. Then the classifier receives the true label y_t. Comparing p_t and y_t, the classifier then updates its internal model.

In batch learning, training and testing phases are separated with no overlap between the training set and testing set. In the online scenario, however, the training phase and testing phase are interleaved and some data points may be used for both training and test [9]. In some situations where the training data set is too large to be

used, or timely response is desired, this interleaved training and testing set up is important. Note that a simple online classifier could be a batch classifier with memory that stores all the data it previously received $(x_1, y_1, x_2, y_2, ..., x_{t-1}, y_{t-1})$ and trains a new model each time a new datum is received. Obviously, such a simple online classifier requires a growing memory space and computation time in each iteration. The art of online learning, in contrast, is to update the model each time a new datum is received, with a limited number of previous data points or model parameters being stored. Hence the workload and memory space are expected to be constant for an efficient online classifier.

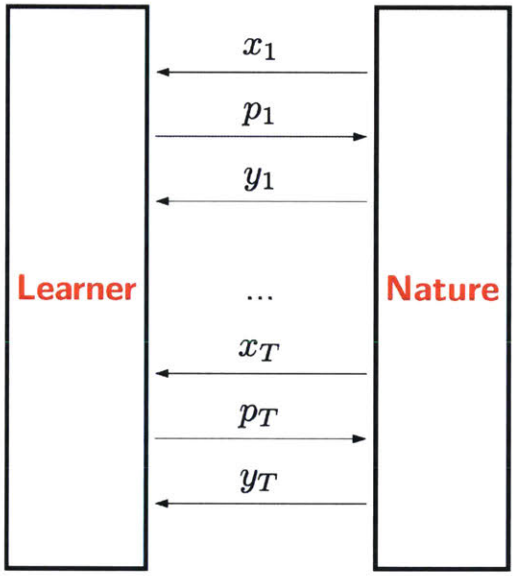


Figure 4-10: The scenario of online learning [9]. Here x_t are inputs, p_t are predictions and y_t are true labels.

4.6.2 Online RUSBoost

In this section we first introduce the idea of online boosting and its imbalanced class variant. Then we present the online version of RUSBoost, which is used to update our classifier when new die level memory test results are available.

Online Bagging and Boosting

The online versions of bagging and boosting are discussed in [60]. The key idea of online bagging and boosting is to approximate batch bagging and boosting with

the training dataset observed as a data stream. In the batch version of bagging

Algorithm 4 Bagging

1: **input**: S instances: $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N), y_i \in \{1, 0\}$
2: Base classifiers $h_1, ..., h_K$
3: **for** $k = 1, 2, ..., K$ **do**
4: Bootstrap: Sample $|S| = N$ instances from S to get D_k
5: Train base classifier h_k with D_k
6: **end for**
7: **final output**: $H(x) = \arg\max_y \sum_{k=1}^{K} \mathbf{1}_{\{h_k(x)=y\}}$

(Algorithm 4), for each base classifier, the number of copies of (x_i, y_i), a_i, are subject to a binomial distribution $B(N, \frac{1}{N})$. Indeed, we draw N independent samples with $\frac{1}{N}$ chance of drawing each copy, so that the probability of having n copies of a_i within that sample set is:

$$P(a_i = n) = \binom{N}{n} (\frac{1}{N})^n (1 - \frac{1}{N})^{N-n} \approx \frac{e^{-1}}{n!}. \tag{4.21}$$

The reason for the approximation is that in online learning, we may not know the length of the stream before observing all the data. The approximation is valid when $N \to \infty$. Then we can use a Poisson(1) probability distribution to estimate the probability distribution of a_i, and generate random values a_i accordingly. The online version of bagging is in Algorithm 5. With the idea of transforming batch bagging

Algorithm 5 Online Bagging [60]

1: **input**: Data stream: $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m), y_i \in \{1, 0\}$
2: Base classifiers $h_1, ..., h_K$
3: **When** (x_i, y_i) **arrives**:
4: **for** $k = 1, 2, ..., K$ **do**
5: Generate random variable a_i according to $Poisson(1)$
6: Do a_i times: train base classifier h_k with (x_i, y_i)
7: **end for**
8: **final output**: $H(x) = \arg\max_y \sum_{k=1}^{K} \mathbf{1}_{\{h_k(x)=y\}}$

to online bagging, the online version of AdaBoost in Algorithm 2 is also straight forward. In Algorithm 6, λ can be interpreted as $N \times D(i)$ in batch AdaBoost and ϵ_k is equivalent to the weighted pseudo-loss in batch AdaBoost. The total weights of correctly classified is λ_m^{SC} and that of the misclassified instances for each base is λ_m^{SW}.

Algorithm 6 Online AdaBoost [60]

input: Data stream: $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m), y_i \in \{1, 0\}$
Base classifiers $h_1, ..., h_K$
Initialize $\lambda_k^{SC} = \lambda_k^{SW} = 0$ for all k.
When (x_i, y_i) **arrives**:
Reset $\lambda = 1$
for $k = 1, 2, ..., K$ **do**
    Generate random variable a_i according to $Poisson(\lambda)$
    Do a_i times: train base classifier h_k with (x_i, y_i)
    if $h_k(x_i) = y_i$ **then**
        $\lambda_k^{SC} \leftarrow \lambda_k^{SC} + \lambda$
        $\epsilon_k \leftarrow \frac{\lambda_k^{SW}}{\lambda_k^{SW} + \lambda_k^{SC}}$
        $\lambda \leftarrow \frac{\lambda}{2(1-\epsilon_k)}$
    else
        $\lambda_k^{SW} \leftarrow \lambda_k^{SW} + \lambda$
        $\epsilon_k \leftarrow \frac{\lambda_k^{SW}}{\lambda_k^{SW} + \lambda_k^{SC}}$
        $\lambda \leftarrow \frac{\lambda}{2\epsilon_k}$
    end if
end for
final output: $H(x) = \arg\max_y \sum_{k=1}^{K} \mathbf{1}_{\{h_k(x)=y\}} \log \frac{1-\epsilon_k}{\epsilon_k}$

Online RUSBoost

To handle class imbalance, Algorithm 6 is further modified in [58] to approximate RUSBoost in Algorithm 3. The online RUSBoost is shown in Algorithm 7 [58]. Similar as in online AdaBoost, λ^{RUS} is an approximation of $N \times D(i)$ and it is tracked by the total weights of correctly classified and misclassified examples for each base classifier λ_m^{SC} and λ_m^{SW}. Here n^+ and n^- are the total number of positive (minority) instances and negative (majority) instances, respectively, and λ_m^{POS} and λ_m^{NEG} store the total weights of positive (minority) instances and negative (majority) instances.

Even though this family of online learning methods is not specifically designed for concept drift, experiments in [61] suggest that they may be used in a concept drift environment. More specifically, [61] studied the impact of diversity among base classifiers on the test error before and after drift. The results showed that the diversity among base classifiers in the ensemble can help reduce test error shortly after drift begins. In RUSBoost, random under sampling is expected to increase the diversity of base classifiers when there is a small number of training instances. With this

motivation, online RUSBoost is used in our approach and our empirical tests to follow.

Algorithm 7 Online RUSBoost [58]

input: Data stream: $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m), y_i \in \{1, 0\}$ with majority class $y_i = 0$ and minority class $y_i = 1$; sampling rate C
Base classifiers $h_1, ..., h_K$, $\lambda_k^{SC} = \lambda_k^{SW} = 0$ for all k.
Initialize $\lambda_k^{SC} = \lambda_k^{SW} = \lambda_k^{POS} = \lambda_k^{NEG} = 0$ for all k, $n^+ = n^- = 0$
When (x_i, y_i) **arrives**:
Reset $\lambda = 1$
for $k = 1, 2, ..., K$ **do**
    if $y_i = 1$ **then**
        $\lambda_k^{POS} \leftarrow \lambda_k^{POS} + \lambda, n^+ = n^+ + 1$
        $\lambda_k^{RUS} \leftarrow \lambda \frac{n^+}{n^+ + n^-} / \frac{\lambda_k^{POS}}{\lambda_k^{POS} + \lambda_k^{NEG}}$
    else
        $\lambda_k^{NEG} \leftarrow \lambda_k^{NEG} + \lambda, n^- = n^- + 1$
        $\lambda_k^{RUS} \leftarrow \lambda \frac{Cn^+}{n^+ + n^-} / \frac{\lambda_k^{NEG}}{\lambda_k^{POS} + \lambda_k^{NEG}}$
    end if
    Generate random variable a_i according to $Poisson(\lambda^{RUS})$
    Do a_i times: train base classifier h_k with (x_i, y_i)
    if $h_k(x_i) = y_i$ **then**
        $\lambda_k^{SC} \leftarrow \lambda_k^{SC} + \lambda$
        $\epsilon_k \leftarrow \frac{\lambda_k^{SW}}{\lambda_k^{SW} + \lambda_k^{SC}}$
        $\lambda \leftarrow \frac{\lambda}{2(1-\epsilon_k)}$
    else
        $\lambda_k^{SW} \leftarrow \lambda_k^{SW} + \lambda$
        $\epsilon_k \leftarrow \frac{\lambda_k^{SW}}{\lambda_k^{SW} + \lambda_k^{SC}}$
        $\lambda \leftarrow \frac{\lambda}{2\epsilon_k}$
    end if
end for
final output: $H(x) = \arg\max_y \sum_{t=1}^{T} \mathbf{1}_{\{h_k(x)=y\}} \log \frac{1-\epsilon_k}{\epsilon_t}$

4.6.3 Experimental Results

To show the effectiveness of online learning and compare it with the batch learning, we use the same dataset as in the batch learning in Section 4.5.3. As shown in Figure4-11, all base classifiers of both batch and online models are trained on the training set. In the batch learning scenario, these base classifiers are combined to test the dies in the test set without any further training. In the online learning scenario,

however, the dies in the test set are received sequentially. The test results are then used to update the online classifiers to achieve higher prediction accuracy.

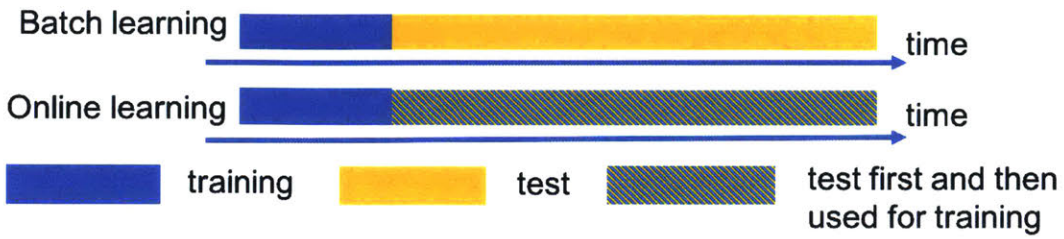


Figure 4-11: The batch and online learning scenarios. All base classifiers of both batch and online models are trained on the training set. In the batch learning scenario, these base classifiers are combined to test the dies in the test set without any further training. In the online learning scenario, however, the test results are then used to update the model.

This experiment simulates the situation displayed in Figure 4-3, where once we know the memory test result of a die in the test set, it becomes our new training instance for updating the online learning classifier. As online boosting methods are designed to approximate the batch learning asymptotically, each die in the test set is predicted by a classifier trained on the training set and all the test instances before it. As indicated in Figure 4-12, online learning could enable as much as an extra 4.5% of expected yield improvement when $s = 4$, primarily by a much improved classifier and corresponding ROC curve, when there is no concept drift present.

4.6.4 Summary

In this section the online bagging and boosting algorithms are presented. The main idea of online learning in this problem is that as we have the die level memory test result for each die, online learning can be performed to update our learning model when more data is available. Since we are facing an imbalanced classification problem, we choose an online version of RUBoost. Figure 4-12 shows that an extra 4.5% of yield improvement compared to batch learning is achieved with online learning.

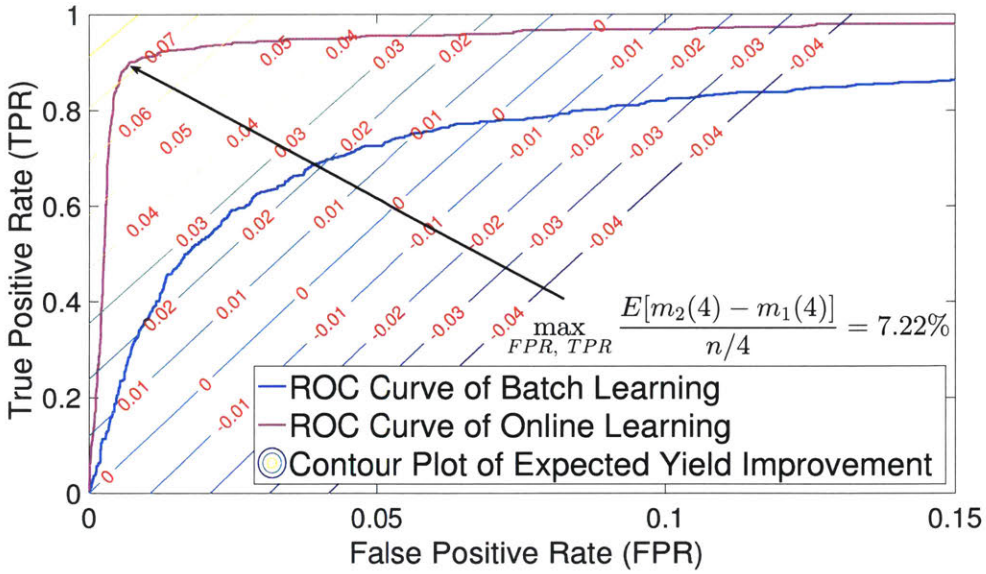


Figure 4-12: Contour plots of yield improvement $\frac{\mathbb{E}[m_2(s)-m_1(s)]}{n/s}$ with number of dies in a package $s = 4$. The yield improvement is optimized under the constraint of ROC curve that is determined by the classifier. Yield with a batch classifier= 90.78% while yield with an online classifier= 95.28%.

4.7 Concept Drift (Temporal Variation) and Incremental Learning

The classical set up of batch machine learning problems is based on learning from data drawn from an unknown but fixed distribution. Then the idea of empirical risk minimization can be applied according to the Law of Large Numbers (LLN). However, problems in the real world are more complex, especially with inclusion of a particular parameter in the learning scenario, i.e., time. The data we receive from nature may not be generated at the same time and a critical issue is that the underling distribution of the training data may have changed during the information acquisition. In this section, we introduce incremental learning to overcome realistic concept drift of semiconductor manufacturing data in a long period of time. Section 4.7.1 is an introduction of concept drift. Section 4.7.2 discusses the speed of concept drift. The experiments to detect realistic concept drift in our data stream are presented in Section 4.7.3. Some other concept drift detection techniques are also introduced in this section. Section 4.7.4 describes the challenges of concept drift with class

imbalance. Then our incremental learning framework is introduced in Section 4.7.5 to tackle concept drift and class imbalance simultaneously. Finally, the experimental results and discussion are provided in Section 4.7.6.

4.7.1 Introduction

The issue of model change or underlying time trends in the dataset is called concept drift in the statistics community, and refers to the unexpected changes in the statistical property of the dataset used to train the model. The most obvious problem caused by concept drift is that the model built on previous training data may lose its accuracy over time. In data stream from semiconductor manufacturing processes, the issue of temporal variation, or concept drift, also exists. One obvious reason is that changes in the manufacturing process are often occurring. This includes improvements or other changes in the fabrication, assembly or testing processes.

The mechanism of concept drift, however, manifests in multiple forms. Theoretically, a learning model gives the posterior distribution $p(y|x)$ of the output y with input x. By Bayes rule, the posterior distribution can be written as

$$p(y|x) = \frac{p(y)p(y|x)}{p(x)}, \tag{4.22}$$

where $p(y)$ is the distribution of the output, $p(x)$ is the distribution of the input and $p(y|x)$ is the likelihood imposed by the model. The distribution of the input, $p(x)$, is assumed to be fixed. So the three possible drifts in the model are $p(y)$, $p(x|y)$ and $p(y|x)$ [10]. The three possible concept drifts are illustrated in Figure 4-13. In scenario (a), the distribution of the class changes over time, which means that the circle class occurs more after the drift, but the boundaries of two classes remain the same. In scenario (b), the posterior changes and thus the outer boundary of the circle class changes. In scenario (c), the likelihood drifts such that the dashed boundary between the two classes changes. Though different drift models show different behaviors in the input and output over time, most algorithms and discussions in the literature do not distinguish the form of drift.

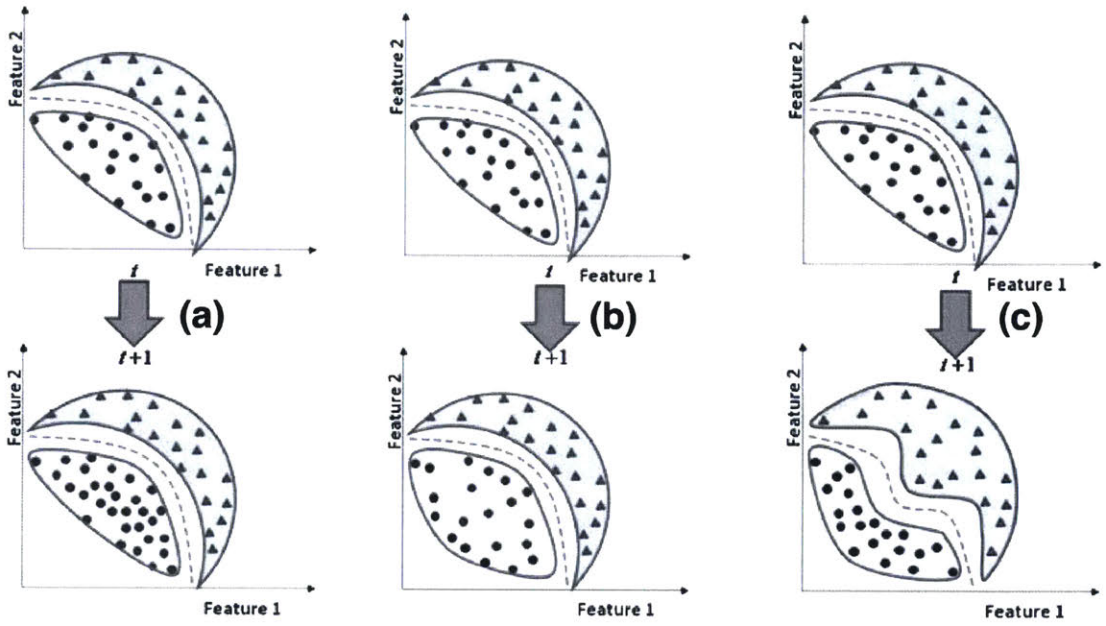


Figure 4-13: Three different concept drift types [10]: **(a)** $p_t(y) \neq p_{t+1}(y)$; **(b)** $p_t(y|x) \neq p_{t+1}(y|x)$; **(c)** $p_t(x|y) \neq p_{t+1}(x|y)$.

4.7.2 Speed of the Concept Drift

When the existence and the form of drift is confirmed, another topic related to concept drift is the speed of the drift, which describes the behavior over time. Some possible ways that a 1D variable might drift over time are shown in Figure 4-14 [11]. In general, concept drifts can be divided into two classes: sudden drift (a), and incremental/gradual drift, (b) and (c). In sudden drift cases, the distribution of the instances changes abruptly, and the model trained on the old distribution may be substantially degraded or even useless when used to predict the data drawn from the new distribution. When sudden drift occurs, we may see a sharp drop in the performances of the classifier (for example, accuracy or AUC). On the other hand, sudden concept drift such as (a) are comparably easy to detect. Gradual or incremental drift, however, shows a smooth transition in the distribution. In this case, the performance of the old classifier will deteriorate gradually. This deterioration is hard to detect and sometimes even unnoticeable at an early stage due to statistical fluctuation. A drift may also be recurrent as in (c). The existence of reoccurring concept drift implies that the old data may still be useful to improve the classifier in future predictions.

An extreme case of concept drift is an outlier in (e), which is not considered to be concept drift in a general sense.

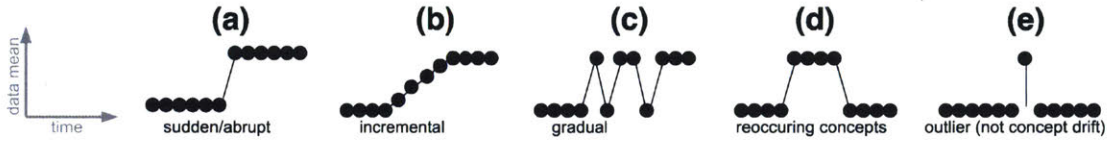


Figure 4-14: Concept drifts of a 1D variable over time [11].

4.7.3 Detection

Detection is another important topic in consideration of concept drift. As mentioned in the previous section, detecting a gradual drift in the distribution is believed to be hard. Various methods have been proposed to detect gradual concept drift in the literature. Intuitively, if the distribution of the data is stationary, the error rate of a learning algorithm should decrease as the number of training instances increases. In [62], the author proposed a drift detection method (DDM) to detect concept drift based on significant increase in the error rate of a learning algorithm. If the error rate at time t is P_t, the standard deviation is

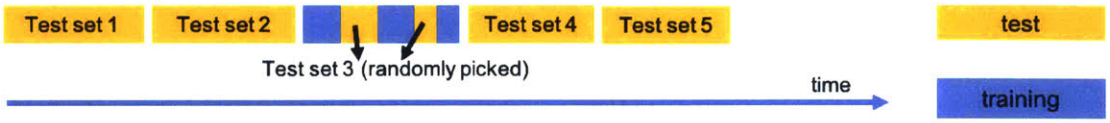
$$s_t = \sqrt{P_t(1 - P_t)/t}. \tag{4.23}$$

The detector stores the minimum value of $P_t + s_t$ as $P_{min} + s_{min}$. The thresholds of warning and drift detection is set as $P_{min}+2s_{min}$ and $P_{min}+3s_{min}$. The thresholds are determined by the significance level of 95% and 99% based on the fact of central limit theorem. A variant of DDM, the early drift detection method (EDDM), is developed in [63]. This method tracks the distance between two consecutive classification errors. This method denotes P_t as the average distance between two consecutive errors at time t, and s_t as the corresponding standard deviation. The detector stores the maximum of P_t and s_t as P_{max} and s_{max}. Warning and detection are raised if $(P_t + 2s_t)/(P_{max} + 2s_{max})$ is smaller than predefined thresholds. Other more sophisticated detection methods are discussed in more recent literature. For example, [64] proposed three adaptive concept drift detection methods based on a rank statistic on density estimates, average margin of SVM and error rate of SVM. Reference [65] splits the

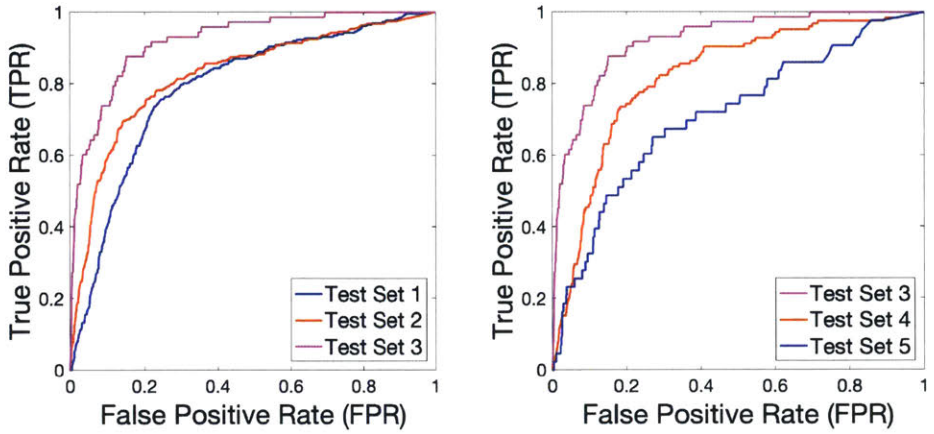
data stream into an old sub-stream with n_o instances and a new sub-stream with n_r. A more complex statistic is calculated based on the classification result on these two sub-streams and is compared with a normal distribution to detect significant deviations or drift.

In this project, we detect the existence of concept drift (temporal variation) in oursemiconductor manufacturing data stream based on the result of batch and online learning. Recalling that the dataset is the die level memory test data sorted by SME1 test date. A chunk of the data stream in July, August and September 2015 is used. The goal of this experiment is to show the deterioration in the performance of the classifier over time, as a function of distance in time between the training and test sets, considering both subsequent or "forward" test sets and earlier or "backword" test sets. The partition of test and training sets is shown in Figure 4-15a. In this experiment, our data stream is divided into 5 chunks. The training set is randomly selected from the 3rd chunk. The rest of the dies in chunk 3 are used as test set 3. Then the classifier is tested on these 5 test sets. Because the training set and test set 3 are randomly selected from the same time period, test set 3 can serve as a "control group" in which little or no concept drift exists. The batch RUSBoost is trained on the training set and the ROC curves obtained by testing the classifier on the test sets are displayed in Figure 4-15b. It is easy to see that the performance of the classifier on test sets 1, 2, 4 and 5 are much worse than on test set 3. Note that as the pass dies make up over 99% of the total dies, we only care about the very left part of the ROC curve for the purpose of bad die detection. The shifts observed in the ROC curve so means that a large number of false alarms will occur with the same true positive rate.

A similar conclusion can be drawn from an online learning experiment. In this experiment, the forward test takes each die in the data stream as both test and training instances; after receiving all data, a backward test is conducted by testing the ensemble classifier on all the data as a batch. Then the entire data stream is swapped and the online classifier is trained and tested on the data stream in a reverse or "backward" in time direction. Figure 4-16 shows the ROC curves obtained in these three tests. It is obvious that both forward and backward stream tests achieve high accuracy in prediction. The backward batch test, however, is basically a random



(a) The partition of training and test sets



(b) The ROC curves obtained in different test sets.

Figure 4-15: Existence of concept drift. The data stream is divided into five chunks. The training set is randomly selected from chunk 3. The remaining dies in chunk 3 are used as test set 3. The classifier is tested on all 5 test sets. The performance of the classifier decays over time (both forward and backward) as the ROC curves show, confirming substantial concept drift.

guess, which suggests that the online classifier obtained at the end of the forward streaming test can no longer accurately predict early dies. This test shows that the performance improvement of the online classifier over batch classifier is not because of it receives more training instances, but rather is due to its tracking concept drift (the temporal trend or changes over time).

4.7.4 Concept Drift with Class Imbalance

The problem of imbalanced classification, as encountered in Section 4.5, is exacerbated in the concept drift scenario. As the distance in time between two consecutive minority instances can be very large, the classifier may not receive any minority instances for a long time. During this time period, no significant improvement of the classifier is expected but underlying concept drift may still gradually impair the classifier's ability. One the other hand, the detection of concept drift also requires a

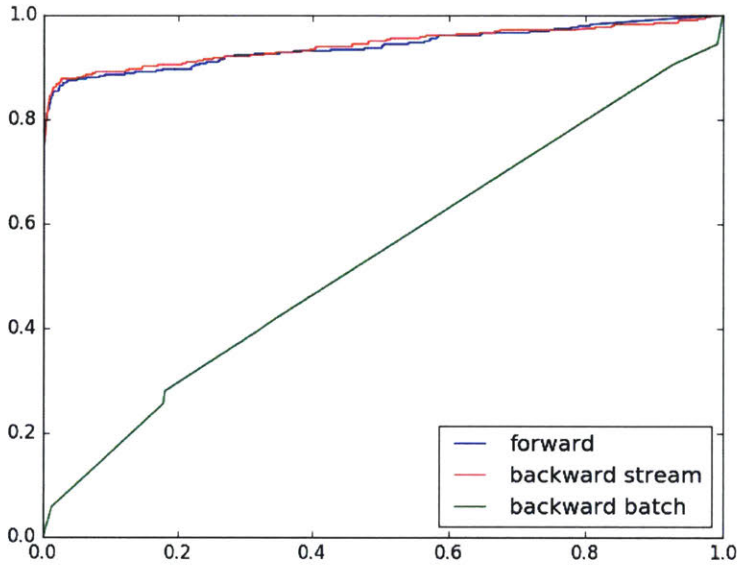


Figure 4-16: The ROC curves obtained by testing the online classifier in three different ways.

frequent check of the classifier's performance. In the time period without any minority class instances, concept drift detection is also difficult. In Section 4.8, we will discuss and visualize some typical drifting features.

Learning concept a drift stream with class imbalance is believed to be a very challenging problem, with a lack of previous research available. Virtually all existing algorithms require a "data chunk stream" which is slightly different from the "data point stream" scenario in Figure 4-10. A data chunk structure enables new classifiers to be trained from scratch for each data chunk to adapt to concept drift. In a data stream case, however, classifiers can only be updated but not rebuilt. Also, the algorithm using a data chunk stream approach implicitly assume no concept drift within the chunk.

The framework in [66] is an early method to handle concept drift and class imbalance simultaneously, based on bootstrap aggregating (bagging). The classifier stores all minority instances in its memory. When a new chunk of data arrives, the algorithm subsamples the majority class instances to balance the distribution ratio, and then combines the subsample with all minority instances in memory. Then base classifiers

are trained in a bagging fashion. The drawback of this method is that by keeping all minority instances, it assumes that the minority class does not drift [10]. SERA in [67], however, does not use all minority instances in the memory. Instead, it picks the old minority instances with small Mahalanobis distance [68] to the minority instances in the new data chunk in order to forget the old information. Reference [69] proposes another method using Hellinger distance, but it is not an online learning method since previous data chunks are required.

Learn++.SMOTE in [70] is uses a SMOTEBoost framework. The algorithm we use in our work, Learn++.NIE [71], is a similar approach. In these two algorithms, a classifier is generated and stored when a chunk of data is received. Old classifiers are weighted with an exponential decay. When a new chunk of data arrives, the algorithm invokes all old classifiers by their weighted average.

4.7.5 Incremental Learning

As mentioned in Section 4.7.4, dealing with concept drift and class imbalance simultaneously is an inherently difficult problem. With the same problem formulation, Online RUSBoost in Algorithm 2 is not the best choice to handle a nonstationary environment since it is designed to approximate the batch learning algorithm.

Intuitively, in an evolving environment, learning models should only be trained on the most "recent" data and should only be used to predict the near future. However, as the concept drift can be very fast, the qualified recent data may be inadequate to train a good classifier due to overfitting. This problem is exaggerated in a class imbalance scenario where the qualified recent minority instances are even fewer. To learn in a nonstationary environment, the idea of incremental learning has been proposed. In contrast to the online learning scenario, where the information from old instances is discarded after training, incremental learning emphasizes extending the model's knowledge by leveraging new data while preserving information from old data. To implement this idea, we store the classifiers trained on old data (since storing old data is expensive) and invoke those classifiers to help us to make predictions in the future. While a new classifier trained only on recent data may be incapable of making a good prediction, old classifiers trained on a similar pattern may participate in decision making and improve performance.

An existing incremental learning algorithm is Learn++.NIE, which assumes that there is a data stream of chunks (mini batches) available over time. At each time step t, a new chunk $\mathcal{D}_t = \{(x_1,\ y_1),\ (x_2,\ y_2),\ ...,(x_{m^{(t)}},\ y_{m^{(t)}})\}$ arrives; and a new base classifier h_t is trained on $\mathcal{D}_t$ in a batch manner. The final ensemble combines the base classifiers by calculating a weighted average of the output from each base classifier. The weight of the base classifier trained at time step $k \leq t$ is determined by that base classifier's overall performance on $\mathcal{D}_k$, $\mathcal{D}_{k+1}, ...,\ \mathcal{D}_t$ with an emphasis on recent data chunks. The detailed introduction of Learn++.NIE is available in [71].

In order to make it more suitable to our problem, we introduce a modified version of Learn++.NIE in this thesis, as Algorithm 8. First, instead of under sampling bagging as in [71], we use RUSBoost as our base classifier. Second, since the output of the base classifier is a probability, we use $\text{auc}_k^{(t)}$, the area under the ROC curve, to evaluate the performance of the base classifier. The pseudo-loss is obtained by $\epsilon_k^{(t)} = 1 - \text{auc}_k^{(t)} \in [0,\ 1]$. Finally, the final output of the ensemble is a weighted average of the probability from each base classifier.

4.7.6 Experimental Results

The real manufacturing data of dies produced over a large time span is used. As shown in Section 4.7.3, concept drift is detected in a large time interval. This is due to minor changes in fabrication, assembly and testing processes. The modified Learn++.NIE is used to handle concept drift and class imbalance simultaneously. As mentioned in Section 4.4, in a drifting environment, expected yield improvement $\frac{\mathbb{E}(m_2 - m_1)}{n/s}$ in the future can no longer be estimated by old data. The choice of the optimal threshold in classification then requires expert advice or evaluation of cost tradeoff [8]. Here we assess the overall performance of the classifiers by their AUC values over time. We compare the AUC value of our modified Learn++.NIE ensemble classifier (incremental learning) against a single RUSBoost classifier trained only on the most recent data chunk. Results are shown in Figure 4-17. With old base classifiers stored, incremental learning outperforms single RUSBoost classifier on the whole. The average AUC value with 95% confidence interval of the single classifier is 0.79 ± 0.002 while that of our incremental learning approach is 0.82 ± 0.001 (a 4% improvement). If we do have expert advice for the optimal threshold, average ex-

Algorithm 8 Modified Learn++.NIE

input: Data chunks: $\mathcal{D}_t = \{(x_i,\ y_i)\},\ i = 1,\ 2, ...,\ m^{(t)}$ with majority class $y_i = 0$ and minority class $y_i = 1$; Positive sigmoid parameters a and b.
for $t = 1, 2, ...$ **do**
    when $\mathcal{D}_t$ **arrives:**
    Call RUSBoost on $\mathcal{D}_t$ to generate a new base classifier $h_t(\cdot,\ \cdot) : X \times Y \rightarrow [0, 1]$
    For $k \in \{1,\ 2,\ ...,\ t\}$, evaluate $\text{auc}_k^{(t)}$, area under ROC curve obtained by h_k on $\mathcal{D}_t$.
    For $k \in \{1,\ 2,\ ...,\ t\}$, calculate pseudo-loss $\epsilon_k^{(t)} = 1 - \text{auc}_k^{(t)}$.
    if $\epsilon_t^{(t)} > 0.5$ **then**
        Discard h_t and train a new one.
    end if
    if $\epsilon_k^{(t)} > 0.5$ **and** $k < t$ **then**
        Set $\epsilon_k^{(t)} = 0.5$
    end if
    $\beta_k^{(t)} = \epsilon_k^{(t)} / (1 - \epsilon_k^{(t)})$
    For $k \in \{1,\ 2,\ ...,\ t\}$, calculate weights:
    Define: $\sigma_k^{(t)} = [1 + \exp(-a(t - k - b))]^{-1}$
    Normalize: $\hat{\sigma}_k^{(t)} = \sigma_k^{(t)} / \sum_{j=0}^{t-k} \sigma_k^{(t-j)}$
    $\hat{\beta}_k^{(t)} = \sum_{j=0}^{t-k} \hat{\sigma}_k^{(t-j)} \beta_k^{(t-j)}$
    $W_k^{(t)} = \log \frac{1}{\hat{\beta}_k^{(t)}}$
end for
output: $H^{(t)}(x, y) = \sum_{k=1}^{t} W_k^{(t)} h_k(x, y)$ with normalization

pected yield improvement with 95% confidence interval by our incremental learning and a single classifier is given in Table 4.4, where our incremental learning increases the average expected yield improvement by 1.5× to 4.4× when $s = 8$ or 16, compared to the single classifier.

4.8 Feature Selection

4.8.1 Introduction

In statistics, feature selection is an important process to identify relevant variables for model construction. The purpose of feature selection is generally three-fold. First, feature selection reduces model complexity with shorter training time and smaller

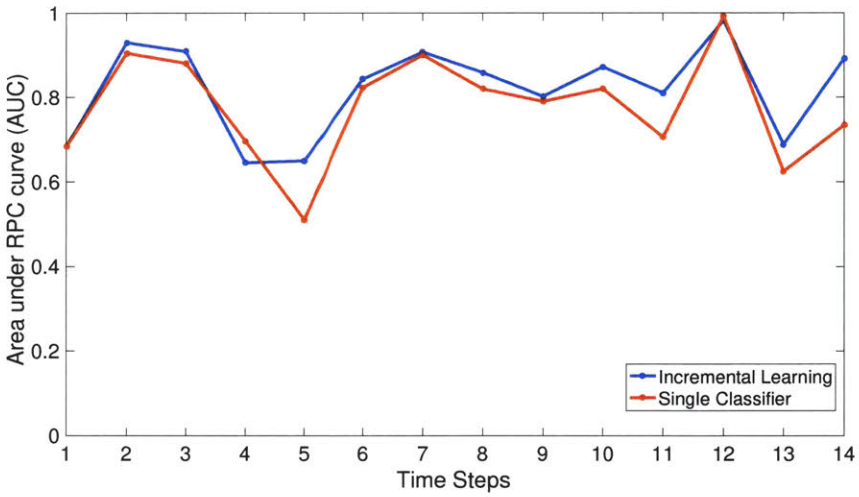


Figure 4-17: Area under ROC curves obtained on data chunks by incremental learning, and by a single classifier trained only on the most recent data chunk.

s	Incremental Learning	Single Classifier	Improvement Factor
4	$\mathbf{0.36\% \pm 0.03\%}$	$0.08\% \pm 0.02\%$	$4.2\times$
8	$\mathbf{1.4\% \pm 0.03\%}$	$0.6\% \pm 0.1\%$	$2.2\times$
16	$\mathbf{3.4\% \pm 0.05\%}$	$2.4\% \pm 0.1\%$	$1.4\times$

Table 4.4: Average expected yield improvement by incremental learning and by a single classifier when optimal thresholds are given.

space consumption. Second, it simplifies the model for easier interpretation for the users. Third, it can reduce the number of parameters to avoid overfitting. From a theoretical perspective, feature selection can be considered as a special case of regularization, which imposes Occam's razor on the model. Using fewer features improves robustness of the model and achieves better performances. In our semiconductor manufacturing scenario, feature selection also helps process engineers to identify the most relevant early stage parameters with respect to the yield.

Feature selection methods can be seen as searching and ranking methods on the subsets of features. The easiest way of feature selection is to test each possible combination of features. But this brutal-force method has an intractable complexity of $O(2^N)$. To more feasibly search on the space of subsets, various feature selection methods are proposed and can by roughly classified into three groups [72]: filter methods, wrapper methods and embedded methods.

Filter Methods

Filter methods are perhaps the easiest methods for feature selection. Filter methods measure the relevance between the feature and the output label with some state-of-the-art metrics (instead of using the model). For example, the measure could be the correlation coefficient or the mutual information between the feature and the output. Note that in filter methods, no information about the model is required and thus it is a good preprocessing technique before training any learning models.

mRMR, developed in [73], is one of the most cited filter method. Here mRMR is a general criterion used in the feature selection. Intuitively, the goal of feature selection is to select the subset with maximum relevance with the output (MR), and minimum redundancy within its features (mR). mRMR in [73] uses the average mutual information between the feature and output to measure the relevance, and the average mutual information between features to measure the redundancy. The score of mRMR is defined as

$$\max_S \frac{1}{|S|}\sum_{k\in S} I(x_k, y) - \frac{1}{|S|^2}\sum_{i,j\in S} I(x_i, x_j), \tag{4.24}$$

where S is the subset, x_i are features and y is the output. The mutual information is defined as

$$I(x, y) = \iint p(x, y)\log\frac{p(x, y)}{p(x)p(y)}dxdy. \tag{4.25}$$

Then the optimization is solved in a greedy way: each step, a feature maximizing the score on the current S is added into S. Some recent works based on this method can be found in [74] and [75].

Wrapper Methods

Wrapper methods, in contrast to filter methods, use the model in feature selection. The wrapper methods use the performance of the model on a subset as its measure. The model is invoked as a blackbox and no internal information is required. Then the optimality of a subset depends on the model and is customized for the model. To check the relevance of an individual feature, a simple wrapper method is to train the model on that single feature and calculate the AUC in the ROC curve in validation

set.

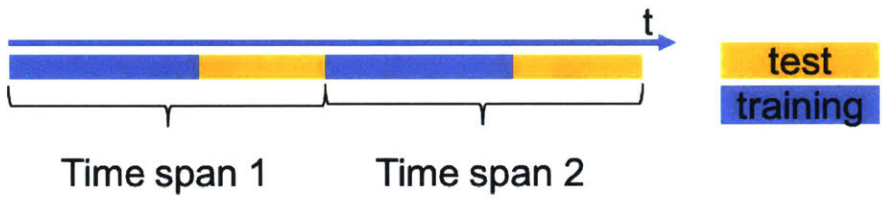
Embedded Methods

Some machine learning algorithms impose sparsity within the learning algorithm. Rather than use a previous feature selection step, learning and selecting are conducted simultaneously in many learning models. A good example of this kind of learning algorithm is LASSO, which achieves feature selection by L_1 norm penalty. In decision trees, pruning can also be considered as feature selection. Other well-known embedded feature selection methods include SVM-RFE [76], first developed for gene selection.

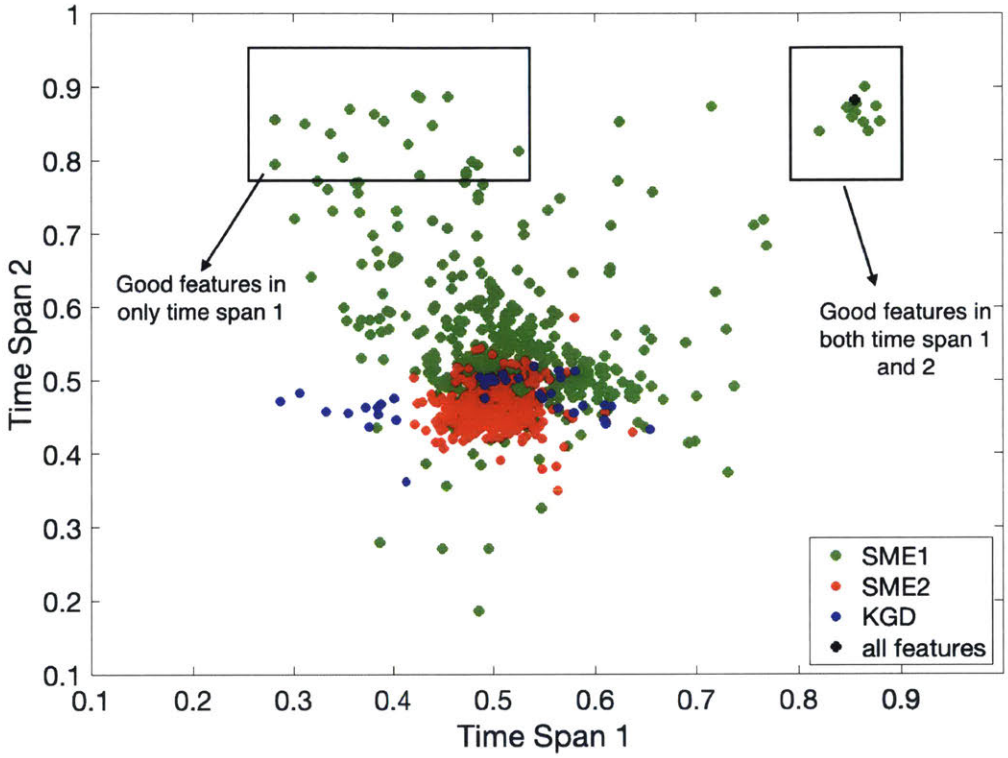
4.8.2 Test Results on the Data

In this work, we use the area under ROC curves (AUC) to measure the overall performance of models trained on a feature set. Recall that AUC values close to 1 indicate a high quality classifier. As mentioned in Section 4.7, concept drift is observed in the data stream; thus the optimal feature set for training is also expected to change over time. Figure 4-18a shows the partition of training and test set in this experiment. Two consecutive time spans are studied. In each time span, the first 70000 dies are used for training, which is followed by 30000 dies for testing. RUSBoost classifiers are trained with individual features (only one feature for each classifier). Figure 4-18b shows each features' AUC values obtained in the two test sets. From Figure 4-18b, features can be divided into three groups: features that are useful in both time spans, such as feature 333; features that are useful in only time span 1 or 2, such as feature 356; and features that are useful in neither time spans, such as feature 400. Figures 4-19 and 4-20 show the normalized histograms of feature 333 and feature 356 in two time spans. Both features are in SFBC part of SME1. Feature 4-19 shows high separability in both time spans, while feature 356 is only separable in time span 2. Figure 4-21 is a typical feature in the third group, which is not a good feature in either time span.

Obviously, features in the first group are the most critical ones, while those in the second group are drifting over time. Almost all features in the first and second group are from SME1, indicating that SME1 parameters are actually the most important



(a) The partition of training set and test set.



(b) The area under curve (AUC) obtained in different time spans.

with respect to memory test. Also note that some features even have larger AUC than the whole feature set. This proves the necessity of feature selection to avoid adding noisy features. More studies are needed to understand the real performance of a feature, particularly in cases where we care about classification with low false alarm rate.

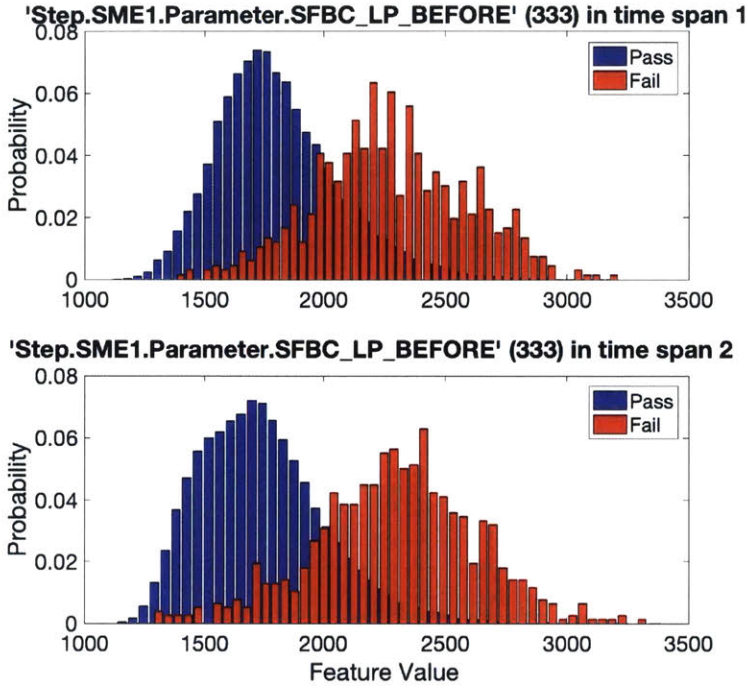


Figure 4-19: The normalized histogram of feature 333 in two time spans.

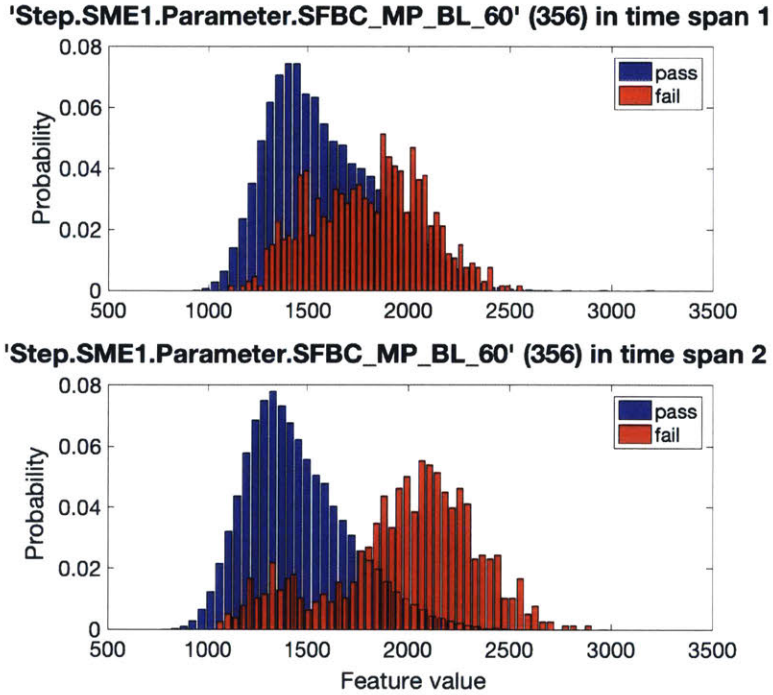


Figure 4-20: The normalized histogram of feature 356 in two time spans.

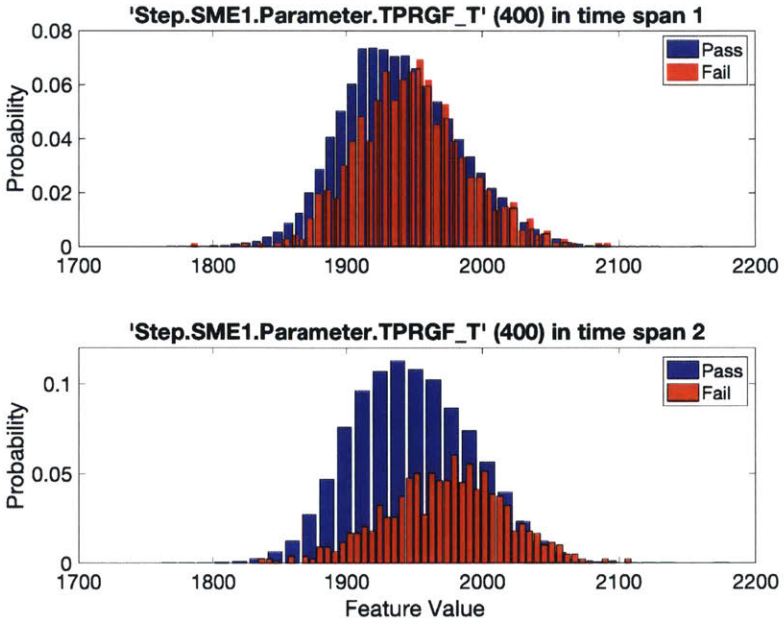


Figure 4-21: The normalized histogram of feature 400 in two time spans.

4.9 Summary

This chapter discusses the machine learning techniques to predict the die level results of the final memory test from fabrication and early stage test data. The challenges encountered in this work are class imbalance and concept drift (temporal variation). Learning models are proposed to solve these challenges. As an application, we study the packaging process of chip stack flash memory manufacturing at SanDisk. We prove the potential of yield improvement in an environment without concept drift by adding a classifier before the packaging process to predict the outcome of the final memory test.

The class imbalance is due to high yield in this manufacturing process. Most dies pass the high performance memory test, which limits the number of die failure instances. To tackle class imbalance, sampling methods are used. RUSBoost is an ensemble method that randomly under-samples the pass instances before training base classifiers. RUSBoost is tested on a data stream from a short period of time and the concept drift is neglected. Since we have access to the die level memory test result of each die in both training and test sets, an online learning version of RUSBoost is

also introduced to enhance the model's classification power.

Concept drift (temporal variation) is another problem in the prediction. The memory test dies can be sorted as a data stream by test date. Due to minor changes in fabrication, assembly and the testing processes, there exists concept drift in the data. The performance of the batch RUSBoost classifier decays over a longer time period. An incremental learning technique is used to store the old knowledge from the data stream and invoke it to update the model with new data received. The experiments compare the incremental learning framework with a single classifier only trained on the most recent data chunk. The potential of significant yield improvement is demonstrated.

Finally, this chapter also mentions the role of feature selection in learning models for prediction. Feature selection reveals the parameters with largest impact on the memory test results. The most important feature group identified is SME1. Also, feature drifting is detected in the parameters, which confirms the claims in the concept drift section.

Chapter 5

Conclusion and Future Work

In this chapter we first summarize results and contributions of this thesis. We then offer suggestions for future research related to machine learning approaches for spatial variation modeling, online and incremental learning and combination of the two.

5.1 Conclusion

The learning models studied in this work present a promising new area of research for the semiconductor industry and the machine learning community. As mentioned in Chapter 2, various machine learning and data mining techniques have been proposed to improve different steps of the semiconductor manufacturing process. Two main topics are discussed in this thesis: virtual metrology approaches for spatial variation modeling and recovery based on low-rank matrix completion in Chapter 3; and online and incremental learning techniques for semiconductor manufacturing data streams under class imbalance and concept drift in Chapter 4.

In Chapter 3, a novel spatial variation virtual metrology algorithm is proposed to recover the die and wafer level variation from a limited number of samples. Compared with a previous algorithm based on discrete cosine transform, this matrix-based algorithm can handle large scale spatial variation problems more efficiently by exploiting recent developments in compressed sensing. The spatial variation across wafers or dies is modeled as a matrix and missing entries recovered by rank minimization. Then the optimization problem is relaxed and solved by a gradient-descent-like algorithm with

singular value thresholding. The optimization can be further improved with device type information.

Chapter 4 discusses machine learning approaches to predict final test results in semiconductor manufacturing from early stage measurement data. Two main challenges, imbalanced classification and concept drift are identified in this chapter. As an application, we study the packaging process in chip stack flash memory manufacturing. The possibility of yield enhancement with a classifier predicting the final test result for each die before packaging is demonstrated in Section 4.4. To address class imbalance, we introduce RUSBoost, an ensemble method under an AdaBoost learning framework, that uses random undersampling before each iteration. The potential for significant expected yield improvement with RUSBoost classifier is shown on real data from industry in Section 4.5. As we have the memory test result of each die after packaging, the learning model can be further updated in an online learning manner. Thus the online version of RUSBoost is introduced in Section 4.6, and an extra yield improvement is achieved in a stationary scenario. In a longer time period, concept drift is detected in our realistic data stream, and the idea of incremental learning is proposed in Section 4.7 to tackle the problem of scarcity of qualified recent data. The state-of-the-art Learn++.NIE framework is modified and implemented to address concept drift and class imbalance simultaneously for incremental learning. The performance of this incremental learning approach is then compared with a single classifier only trained on the most recent data chunk, and an average 1.4× to 4.2× expected yield improvement increase is achieved.

5.2 Limitations and Next Steps

Two key areas for future research are suggested here. First, improvement opportunities for low rank methods in spatial variation modeling are described. Second, online and incremental learning extensions to address temporal variation modeling are proposed.

5.2.1 Low Rank Methods for Spatial Variation Recovery

Other Spatial Variation Patterns

In this thesis, the low rank matrix completion technique is applied to a die-level contact resistance variation problem. A similar methodology can also be applied to other spatial variations such as etch depth or width variation, flush-delay values or fabrication process spatial variation in silicon photonics. Also, the low rank method should be tested on more complex spatial variation patterns, especially those with high frequency components, to compare its performance with the virtual probe approach from [3].

Low Rank Models

In Chapter 3, the low rank matrix is recovered by solving the relaxed optimization problem

$$\min_X ||X||_* + \frac{\lambda}{2} \sum_{(i,j)\in\Omega} |X_{i,j} - M_{i,j}|^2.$$

In fact, there are many other relaxations and corresponding solvers of the rank minimization problem (3.3) [77]. Here some possible problem formulations for future research are listed. Cai, Candès and Shen [78] proposed a algorithm also based on singular value thresholding to solve a tighter relaxation as in (3.4):

$$\begin{aligned} &\min_X ||X||_* \\ &\text{subject to } X_{i,j} = M_{i,j}, (i,j) \in \Omega. \end{aligned}$$

In [79], the authors proposed an algorithm to minimize the cost function $F(U,V)$ defined by

$$\begin{aligned} F(U,V) &= \min_{S\in\mathbb{R}^{r\times r}} \mathcal{F}(U,V,S) \\ \mathcal{F}(U,V,S) &= \frac{1}{2} \sum_{(i,j)\in\Omega} (M_{i,j} - (USV^T)_{i,j})^2, \end{aligned} \tag{5.1}$$

where U and V are orthogonal matrices. The spatial variation recovery problem can also be formulated as a robust Principle Component Analysis problem. In [80], the original matrix M is assumed to be low rank and corrupted by a small noise matrix S.

The original matrix is recovered from partial observations by solving the constraint optimization problem

$$\begin{aligned} &\min_{X,\ S} ||X||_* + \lambda||S||_1 \\ &\text{subject to } X_{i,j} + S_{i,j} = M_{i,j}, (i,j) \in \Omega. \end{aligned} \tag{5.2}$$

Studying the spatial variation recovery problem under different problem formulations could be an immediate next step of research.

5.2.2 Temporal Variations, Online and Incremental Learning

Anomaly Detection

As mentioned in Section 4.5, the problem of test failure prediction can also be regarded as an anomaly detection problem [81] that identifies unexpected instances or events in datasets. Generally, anomaly detection problems can be divided into three main types: supervised anomaly detection, semi-supervised anomaly detection and unsupervised anomaly detection [81] as illustrated in Figure 5-1 [12].

- *Supervised anomaly detection* assumes that the training data comprises fully labeled instances from both normal and anomaly classes. Then the anomaly detection problem is basically equivalent to a classification problem.

- *Semi-supervised anomaly detection* also needs training data for the model. However, the training data only consists of instances from the normal class with no anomaly instances available. The model is then built to represent the distribution of the normal data and any instances that deviate from that distribution are predicted or identified to be anomalies.

- *Unsupervised anomaly detection* requires no training data. It assumes normal instances are far more numerous than anomalies. The prediction is solely based on the features' distribution.

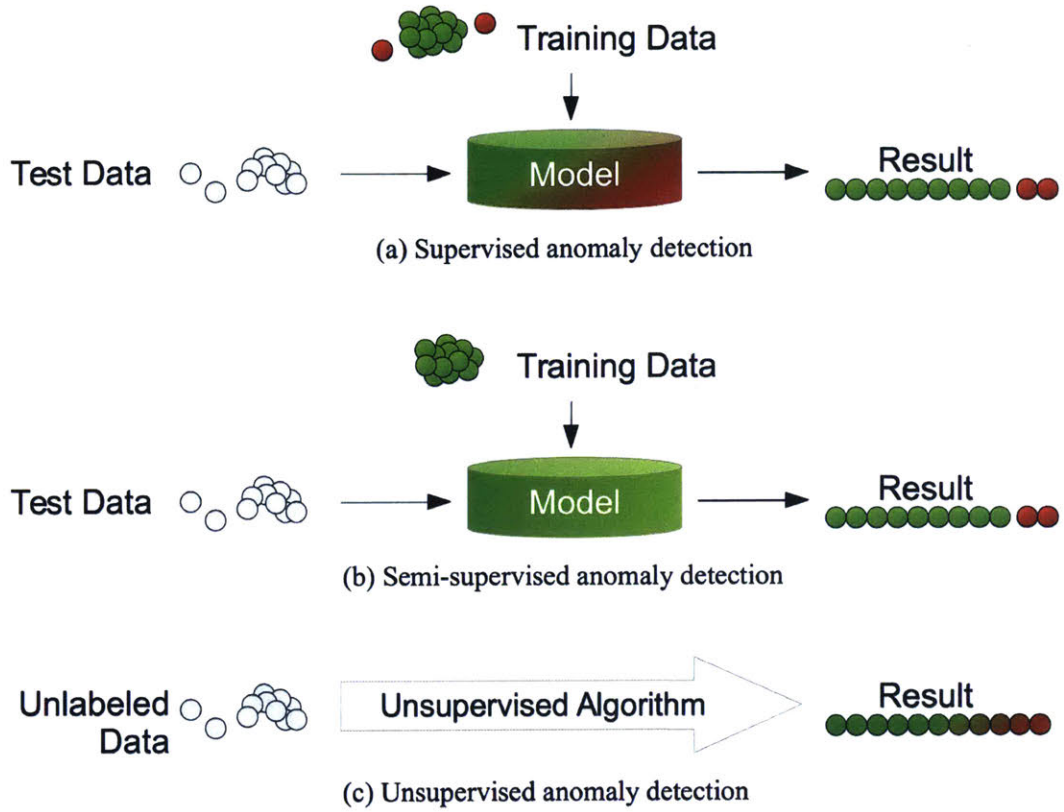


Figure 5-1: Illustration of supervised, semi-supervised and unsupervised anomaly detection [12].

In our memory test failure detection problem, since we have labeled data from both teh normal and the anomaly classes, a supervised anomaly detection setup is straight forward. As discussed in Section 4.7, concept drift exists in the data stream, and thus we need to constantly update our supervised model over time to learn different failure patter.

A possible next step is to build models in a semi-supervised framework. Intuitively, instead of tracking new patterns in fail dies, we only learn the pattern of good dies and predict all those dies that do not show this pattern as fail. Ideally, if the distribution of good dies is stationary, it is not even necessary to update the semi-supervised model. Popular techniques for this problem include classification-based one-class SVM [82], Replicator Neural Network [83], cluster-based method [84] and information-theoretic based method [85].

Wafer Level Spatial Pattern

The work in this thesis is based on die level memory test results. In fact, in the dataset we also have wafer level information. For example, we have the x and y coordinates of each die on its original wafer. We can thus recover a map of fail dies on a wafer, as in Figure 5-2. Clearly, bad dies (red dots) tend to cluster on the wafer, which is valuable spatial information that can be studied in future work. Another interesting

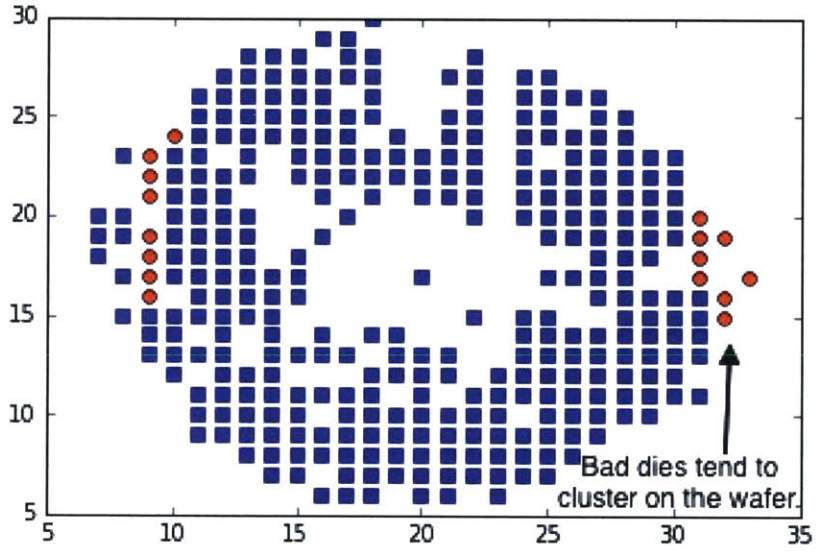


Figure 5-2: A map of fail dies. Good dies are marked as blue squares. Bad dies that are believed to be good by the incoming electrical test criteria but ultimately fail the final high performance memory test after packaging are marked as red dots. A blank space means the die at that position fails previous tests and thus does not go through pacackging process and memory test.

wafer level spatial pattern is the number of dies in the memory test dataset at each position of the wafer as in Figure 5-3, where we can see that dies at the center and the edge of a wafer have higher probability of failure in the tests before the memory test. Wafer level spatial patterns can be studied with the algorithms in Chapter 3 or can be combined with temporal variations to create a combined spatio-temporal learning problem.

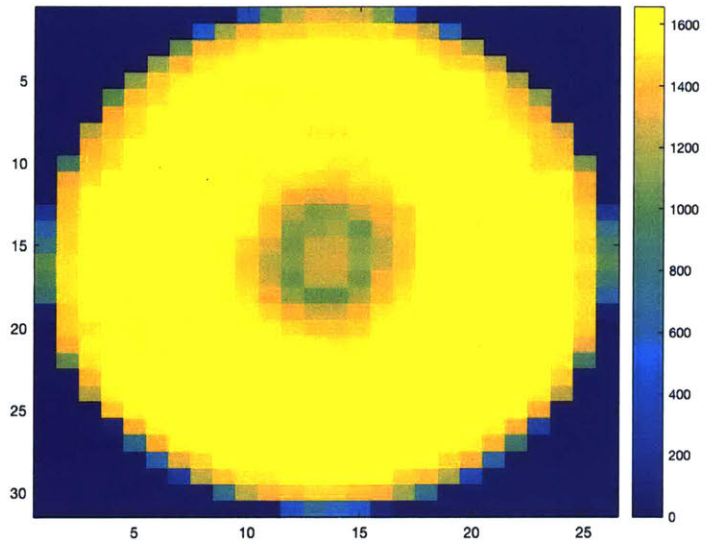


Figure 5-3: The number of dies in memory test dataset at each position of the wafer.

Predicting Failure Categories

As presented in Figure 4-4, FH soft bins indicate the types of failure at final memory test. Predicting the FH soft bin (failure type) is also an important issue for process control and yield improvement. However, predicting the soft bin is a much harder problem. For example, as in Figure 4-4, if there are 12 possible failure categories (12 soft bins), the output of our classifier should be a 12 dimensional vector with each element indicating the probability of failure in the corresponding category. Clearly, this model is 12 times more complicated than our current binary classification model. Also, instances in some soft bins, like bin 412 or bin 946 in Figure 4-4, are so scarce that we do not have enough instances to build a robust classifier.

Data Set Improvement

Another important next step is to enhance the data sources. Currently the output of the model is a binary variable of a given soft bin. The model would be significantly improved if we have access to the criteria and decision variables of binning in the memory test.

Also, our current model requires almost no information of the features. Many of

the features are parameters and test results from the fabrication process. Packaging engineers downstream of the fabrication process do not know the meaning of most features. It is necessary to consult engineers in the fab about the accurate meaning of the features, especially those identified in Section 4.8. Such prior knowledge can help us to filter out irrelevant and redundant features.

Bibliography

[1] Clive Maxfield. Overcoming 32/28-nm IC implementation challenges, *EE Times*, December, 2010.

[2] Brian E. Stine, Duane S. Boning, and James E. Chung. Analysis and decomposition of spatial variation in integrated circuit processes and devices. *IEEE Transactions on Semiconductor Manufacturing*, 10(1):24–41, 1997.

[3] Wangyang Zhang, Xin Li, Frank Liu, Emrah Acar, Rob A. Rutenbar, and Ronald D. Blanton. Virtual probe: a statistical framework for low-cost silicon characterization of nanoscale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(12):1814–1827, 2011.

[4] Xin Li, Rob R. Rutenbar, and Ronald D. Blanton. Virtual probe: a statistically optimal framework for minimum-cost silicon characterization of nanoscale integrated circuits. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 433–440. ACM, 2009.

[5] Dana Cheree Krueger. *Semiconductor yield modeling using generalized linear models.* PhD thesis, Arizona State University, 2011.

[6] Seokho Kang, Sungzoon Cho, Daewoong An, and Jaeyoung Rim. Using wafer map features to better predict die-level failures in final test. *IEEE Transactions on Semiconductor Manufacturing*, 28(3):431–437, 2015.

[7] Karthik Balakrishnan and Duane Boning. Measurement and analysis of contact plug resistance variability. In *2009 IEEE Custom Integrated Circuits Conference*, pages 415–422. IEEE, 2009.

[8] Naomi Aiko Arnold. Wafer defect prediction with statistical machine learning. Master's thesis, Massachusetts Institute of Technology, 2016.

[9] Percy Liang. CS229T/STAT231: Statistical learning theory (winter 2014). Stanford University, 2014.

[10] T. Ryan Hoens, Robi Polikar, and Nitesh V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.

[11] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014.

[12] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[13] Duane Boning, James Chung, Dennis Ouma, and Rajesh Divecha. Spatial variation in semiconductor processes: Modeling for control. In *Proc. of 2nd Int. Symp. on Process Control, Diagnostics, and Modeling in Semiconductor Manufacturing*, pages 72–83, 1997.

[14] Yaw-Jen Chang, Yuan Kang, Chih-Liang Hsu, Chi-Tim Chang, and Tat Yan Chan. Virtual metrology technique for semiconductor manufacturing. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5289–5293. IEEE, 2006.

[15] Jonathan Chang Yung-Cheng and Fan-Tien Cheng. Application development of virtual metrology in semiconductor industry. In *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.*, pages 6–pp. IEEE, 2005.

[16] Aftab A. Khan, James R. Moyne, and Dawn M. Tilbury. Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares. *Journal of Process Control*, 18(10):961–974, 2008.

[17] Gian Antonio Susto, Simone Pampuri, Andrea Schirru, Guiseppe De Nicolao, Seán McLoone, and Alessandro Beghi. Automatic control and machine learning for semiconductor manufacturing: Review and challenges. In *Proceedings of the 10th European Workshop on Advanced Control and Diagnosis (ACD 2012)*, 2012.

[18] Min-Hsiung Hung, Tung-Ho Lin, Fan-Tien Cheng, and Rung-Chuan Lin. A novel virtual metrology scheme for predicting cvd thickness in semiconductor manufacturing. *IEEE/ASME Transactions on Mechatronics*, 12(3):308–316, 2007.

[19] Duane S. Boning and James E. Chung. Statistical metrology: Understanding spatial variation in semiconductor manufacturing. In *Proc. SPIE 2874, Microelectronic Manufacturing Yield, Reliability, and Failure Analysis II,*, pages 16–26, 1996.

[20] Dirk J. Bartelink. Statistical metrology: At the root of manufacturing control. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 12(4):2785–2794, 1994.

[21] Liang-Teck Pang, Kun Qian, Costas J. Spanos, and Borivoje Nikolic. Measurement and analysis of variability in 45 nm strained-si cmos technology. *IEEE Journal of Solid-State Circuits*, 44(8):2233–2243, 2009.

[22] Shigeru Sakai, Masaaki Ogino, Ryosuke Shimizu, and Yukihiro Shimogaki. Deposition uniformity control in a commercial scale HTO-CVD reactor. In *MRS Proceedings*, volume 989, pages 0989–A08. Cambridge University Press, 2007.

[23] E. Chang, B. Stine, T. Maung, R. Divecha, D. Boning, J. Chung, K. Chang, G. Ray, D. Bradbury, O.S. Nakagawa, S. Oh, and D. Bartelink. Using a statistical metrology framework to identify systematic and random sources of die-and wafer-level ILD thickness variation in CMP processes. In *Electron Devices Meeting, 1995. IEDM'95., International*, pages 499–502. IEEE, 1995.

[24] Dawn Dougherty Fitzgerald. Analysis of polysilicon critical dimension variation for submicron cmos processes. Master's thesis, Massachusetts Institute of Technology, 1994.

[25] C. Yu and C. Spanos. Manufacturability evaluation of exposure tools using statistical metrology. In *Statistical Metrology Workshop, Austin, TX*, 1995.

[26] Ariane Ferreira, Agnès Roussy, and Lamine Condé. Virtual metrology models for predicting physical measurement in semiconductor manufacturing. In *Advanced Semiconductor Manufacturing Conference, 2009. ASMC'09. IEEE/SEMI*, pages 149–154. IEEE, 2009.

[27] Yi-Ting Huang, Hsien-Cheng Huang, Fan-Tien Cheng, Tai-Siang Liao, and Fu-Chien Chang. Automatic virtual metrology system design and implementation. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 223–229. IEEE, 2008.

[28] Pilsung Kang, Hyoung-joo Lee, Sungzoon Cho, Dongil Kim, Jinwoo Park, Chan-Kyoo Park, and Seungyong Doh. A virtual metrology system for semiconductor manufacturing. *Expert Systems with Applications*, 36(10):12554–12561, 2009.

[29] Shane Lynn, John Ringwood, Emanuele Ragnoli, Sean McLoone, and Niall MacGearailty. Virtual metrology for plasma etch using tool variables. In *Advanced Semiconductor Manufacturing Conference, 2009. ASMC'09. IEEE/SEMI*, pages 143–148. IEEE, 2009.

[30] Tung-Ho Lin, Fan-Tien Cheng, Wei-Ming Wu, Chi-An Kao, Aeo-Juo Ye, and Fu-Chien Chang. NN-based key-variable selection method for enhancing virtual metrology accuracy. *IEEE Transactions on Semiconductor Manufacturing*, 22(1):204–211, 2009.

[31] Yi-Ting Huang, Fan-Tien Cheng, and Min-Hsiung Hung. Developing a product quality fault detection scheme. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 927–932. IEEE, 2009.

[32] Sholom M. Weiss, Amit Dhurandhar, and Robert J. Baseman. Improving quality control by early prediction of manufacturing outcomes. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1258–1266. ACM, 2013.

[33] Lihui Wu, Jie Zhang, and Gong Zhang. A fuzzy neural network approach for die yield prediction of wafer fabrication line. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 3, pages 198–202. IEEE, 2009.

[34] Bingjun Xiao, Jinjun Xiong, and Yiyu Shi. Novel applications of deep learning hidden features for adaptive testing. In *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*, pages 743–748. IEEE, 2016.

[35] Q. Peter He and Jin Wang. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 20(4):345–354, 2007.

[36] Q. Peter He and Jin Wang. Large-scale semiconductor process fault detection using a fast pattern recognition-based method. *IEEE Transactions on Semiconductor Manufacturing*, 23(2):194–200, 2010.

[37] Tomás Sarmiento, Sang J. Hong, and Gary S. May. Fault detection in reactive ion etching systems using one-class support vector machines. In *Advanced Semiconductor Manufacturing Conference and Workshop, 2005 IEEE/SEMI*, pages 139–142. IEEE, 2005.

[38] N. Kumar, K. Kennedy, K. Gildersleeve, R. Abelson, C.M. Mastrangelo, and D.C. Montgomery. A review of yield modelling techniques for semiconductor manufacturing. *International Journal of Production Research*, 44(23):5019–5036, 2006.

[39] Katina R. Skinner, Douglas C. Montgomery, George C. Runger, John W. Fowler, Daniel R. McCarville, T. Reed Rhoads, and James D. Stanley. Multivariate statistical methods for modeling and analysis of wafer probe test data. *IEEE Transactions on Semiconductor Manufacturing*, 15(4):523–530, 2002.

[40] Dongil Kim, Pilsung Kang, Sungzoon Cho, Hyoung-joo Lee, and Seungyong Doh. Machine learning-based novelty detection for faulty wafer detection in semiconductor manufacturing. *Expert Systems with Applications*, 39(4):4075–4083, 2012.

[41] Lee-Ing Tong and Li-Chang Chao. Novel yield model for integrated circuits with clustered defects. *Expert Systems with Applications*, 34(4):2334–2341, 2008.

[42] Daewoong An, Hyo-Heon Ko, Turghun Gulambar, Jihyun Kim, Jun-Geol Baek, and Sung-Shick Kim. A semiconductor yields prediction using stepwise support vector machine. In *Assembly and Manufacturing, 2009. ISAM 2009. IEEE International Symposium on*, pages 130–136. IEEE, 2009.

[43] Seung Hwan Park, Cheong-Sool Park, Jun Seok Kim, Sung-Shick Kim, Jun-Geol Baek, and Daewoong An. Data mining approaches for packaging yield prediction in the post-fabrication process. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 363–368. IEEE, 2013.

[44] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[45] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[46] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[47] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowledge and Data Eng,*, 21(9):1263–1284, 2009.

[48] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[49] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[50] B.X. Wang and N. Japkowicz. Imbalanced data set learning with synthetic samples. In *Proc. IRIS Machine Learning Workshop*, page 19, 2004.

[51] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, IEEE International Joint Conference on*, pages 1322–1328. IEEE, 2008.

[52] Yoav Freund, Robert Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[53] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[54] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[55] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 107–119. Springer, 2003.

[56] Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[57] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1):185–197, 2010.

[58] Boyu Wang and Joelle Pineau. Online bagging and boosting for imbalanced data streams. *IEEE Trans. on Knowledge and Data Eng.*, 28(12):3353–3366, 2016.

[59] Hang Zhang, Bei Yu, and Evangeline F.Y. Young. Enabling online learning in lithography hotspot detection with information-theoretic feature optimization. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 47. ACM, 2016.

[60] Nikunj C. Oza. Online bagging and boosting. In *Systems, Man and Cybernetics, 2005 IEEE Cnternational Conference on*, volume 3, pages 2340–2345. IEEE, 2005.

[61] Leandro L. Minku, Allan P. White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.

[62] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer, 2004.

[63] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, volume 6, pages 77–86, 2006.

[64] Anton Dries and Ulrich Rückert. Adaptive concept drift detection. *Statistical Analysis and Data Mining*, 2(5-6):311–327, 2009.

[65] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In *International Conference on Discovery Science*, pages 264–269. Springer, 2007.

[66] Jing Gao, Wei Fan, Jiawei Han, and Philip S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 3–14. SIAM, 2007.

[67] Sheng Chen and Haibo He. SERA: selectively recursive approach towards non-stationary imbalanced stream data mining. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 522–529. IEEE, 2009.

[68] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[69] Ryan N. Lichtenwalter and Nitesh V. Chawla. Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53–75. Springer, 2009.

[70] Gregory Ditzler, Robi Polikar, and Nitesh Chawla. An incremental learning algorithm for non-stationary environments and class imbalance. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2997–3000. IEEE, 2010.

[71] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Trans. on Knowledge and Data Eng.*, 25(10):2283–2301, 2013.

[72] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[73] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[74] Hai Thanh Nguyen, Katrin Franke, and Slobodan Petrovic. Towards a generic feature-selection measure for intrusion detection. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1529–1532. IEEE, 2010.

[75] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of Machine Learning Research*, 13(Jan):27–66, 2012.

[76] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[77] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016.

[78] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[79] Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[80] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[81] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[82] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[83] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[84] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[85] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pages 130–143. IEEE, 2001.