Automatic Excursion Detection in Manufacturing: Preliminary Results

Branislav Kveton
Intelligent Systems Program
University of Pittsburgh
bkveton@cs.pitt.edu

Denver Dash
Intel Research
Santa Clara
denver.h.dash@intel.com

Abstract

In the past decades, high-volume manufacturing processes have grown increasingly complex. If a failure in these systems is not detected in a timely manner, it often results in tremendous costs. Therefore, the demand for methods that automatically detect these failures is high. In this work, we address the problem of automatic excursion detection based on parametric tests. Overlooking the complexity of wafer fabrication processes, we propose two structurally simple excursion detection models: Naïve Bayes classifier and boosted decision stumps. We apply these models in the domain of semiconductor manufacturing, compare them to off-the-shelf classification techniques, and show significant gains in the precision and recall of detected excursions. These results encourage our future work that should primarily focus on increasing the recall of our models.

Introduction

From supercomputers to appliances, semiconductors have slowly pervaded every aspect of our life. In the last half of the 20th century, they forever changed the character of our society and enhanced its potential beyond our dreams. Although we have gone a long way since the transistor was invented, semiconductors are still manufactured on the same silicon basis – purified sand. The past five decades of their manufacturing can be viewed as a successful story of making the technology smaller, faster, cheaper, and more reliable.

Semiconductor devices are usually manufactured on a lot level, a collection of *wafers*[1] which are at the end of the line cut into hundreds of dice representing individual chips. The fabrication of wafers is a complicated process that involves hundreds of operational steps, most of which remove or deposit thin layers of material on the wafers. Even if these steps are highly automatized, there are many factors that can cause an *excursion*. An excursion is, roughly speaking, an error in the manufacturing process that causes systematic damage to the product as it passes through some step in the process. Possible causes are equipment malfunction, operator error, mishandling, or material issues such as contamination. To prevent excursions, physical and electrical properties of the wafers are measured by *in-fab parametric tests* at every stage of the fabrication.

Wafer fabrication usually takes from several weeks up to several months, depending on many factors. After the fabrication is completed, wafers undergo extensive die-level *parametric* and *functional testing*. Consequently, the wafers are cut into individual dice, and the ones that passed the tests are packaged. Packaging consists of attaching the dice to support structures, bonding, and encapsulation to protect the semiconductors from the outer world. Once the chips are assembled, they are tested under extreme conditions to assure that no defective product is shipped to customers.

If an excursion occurs, we assume that it can be reliably detected by the functional tests because they directly measure the functional properties of the product. Unfortunately, these tests reveal excursions with a long delay and do not help to prevent them in early stages. Comparing to this end-of-line procedure, excursion detection based on the in-fab parametric tests offers several benefits. First, the in-fab tests are closely related to the stages of the fabrication, and therefore can be used to localize the source of an excursion. Second, in-fab excursion detection gives us an opportunity to repeat failed manufacturing steps, or to discontinue the fabrication of seriously damaged lots. Third, if we know that a lot was affected by an excursion, we can adjust its die-level testing procedure at the end of the line. Finally, all discussed benefits have a positive impact on reducing the cost of the production.

However, the in-fab tests are performed on a wafer level, and thus are less informative than the functional tests. After all, the die-level testing would not be necessary if the in-fab tests perfectly detected every defect. In this work, we try to build a statistical model based on the in-fab parametric tests for the automatic detection of defective lots. We refer to the model as an *excursion detection model*, assuming that the large scale defects are caused by excursions.

The paper is organized as follows. First, we discuss

[1] Wafer is a small thin circular slice of a semiconducting material, such as pure silicon, on which an integrated circuit can be formed.

related work in the manufacturing domain. Second, we point to the challenging character of our problem and propose two excursion detection models. Third, we present relevant theoretical background on used machine learning techniques. Fourth, we compare the two excursion detection models to off-the-shelf classification techniques and discuss their benefit. Finally, we outline future directions of our work.

Related work

Wafer fabrication is a highly automatized process that involves numerous monitoring devices and sensors. For analytical purposes, measurements recorded by these devices are often stored for a long period of time. Even if the fabrication process can be quite complex, its flow is clearly defined. As a result, machine learning techniques can be applied to ease a variety of existing decision problems in this domain.

Currently, we are not aware of any academic work that primarily deals with the automatic excursion detection. The closest to our objective is the work of Fountain *et al.* 2000; 2003 on the optimization of die-level functional testing. The authors constructed a probabilistic model that captures spatial dependencies between failing dice on a wafer. This model was extended into an influence diagram for the purpose of selective die-level testing. The policy corresponding to the diagram was applied on manufacturing data and outperformed other competing strategies. The policy was also tested on defective wafers and shown to be responsive.

Comparing to the work of Fountain *et al.* 2000; 2003, we apply machine learning techniques in an earlier stage of wafer fabrication on a higher, wafer level. Our study primarily focuses on the learning of an excursion detector rather than formalizing a full decision-theoretical framework.

Excursion detection model

In this work, we assume that the lots manifest their state, good or defective, in the results of their in-fab tests. Therefore, to predict that a lot is affected by an excursion, it is sufficient to learn a statistical model $P(Y \mid \mathbf{X})$, where Y is a boolean excursion variable and $\mathbf{X} = \{X_1, \dots, X_k\}$ represents the results of k in-fab tests corresponding to the lot. When predicting the state of a lot, we assume that the results of all its in-fab tests are available. Therefore, our classifier can be viewed as operating by the end of the fabrication line, prior to the more informative die-level testing. Under this assumption, performance of the classifier serves as an upper bound on the real-world scenario when the test results are available gradually as wafer fabrication progresses.

Challenges of the domain

To demonstrate the challenges of learning an excursion classifier, we discuss an example of the in-fab tests dataset[2]. The dataset contains results of 1186 in-fab tests for 583 lots, which corresponds to approximately four months of manufacturing data for a single product. The dataset can be characterized as having: (1) high dimensionality of the input vector comparing to the number of training samples, (2) low number of excursions, unlabeled with unknown sources, (3) large portion of missing values, (4) mostly continuous test results, and (5) no single test that is a strong predictor of defective lots. In the following paragraphs, we return to each of these issues in detail.

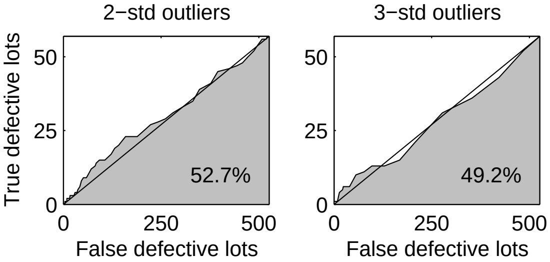

Figure 1: ROC curves corresponding to the number of 2-std and 3-std outliers in all in-fab tests for a specific lot. A test result is considered to be a k-std outlier if its value is more than k standard deviations from the expected result of the test.

The high number of in-fab tests and the low number of lots pose serious threat for many machine learning techniques. In this setting, overfitting is likely to happen, and a close fit to the training set may not generalize well to the test set. Moreover, a sufficient amount of training data may never be available, and thus it is crucial to learn efficiently from a small number of training samples. Therefore, good candidates for the excursion detection model should be structurally simple and yet powerful enough to distinguish between the concepts of good and defective lots.

In general, information on whether a specific lot was affected by an excursion is hard to obtain, which is primarily caused by the diversity of the phenomenon and the human factor involved in the investigation of excursions. Therefore, we automatically label lots as *defective* if they falls into the lower 10 percent tail of the distribution of the number of good dice yielded by the lot[3]. Following this recipe, we get 526 and 57 good and defective lots, respectively. Surprisingly, the defective lots are weakly correlated with the lots that deviate in a large number of their in-fab tests (Figure 1). In fact, simultaneous deviations in several in-fab tests are likely to happen due to the high number of tests performed. Therefore, it is important that our excursion detection model distinguishes between these random and real deviations.

[2] The dataset is a private property of Intel Corporation and cannot be disclosed publicly. All numerical values presented in the paper are normalized.

[3] The 10 percent threshold was arbitrarily chosen prior to the experiments.

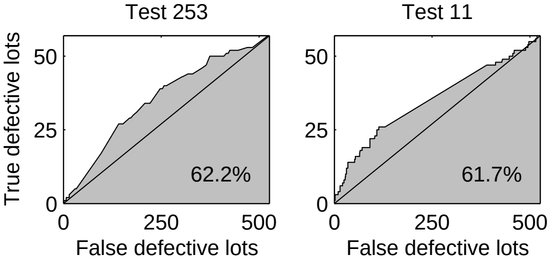

Figure 2: ROC curves corresponding to the two most discriminative in-fab tests. To allow a fair comparison of the tests, missing values are replaced by their expectations.

Finally, half of the tests in the dataset are not measured for more than a half of the lots, which poses a serious challenge for the proper handling of missing values. In such a setting, Bayesian networks (Pearl 1988) allow us to model relations among partially observed attributes. However, the representation of complex dependencies between continuous and discrete variables is in general hard, and so is inference (Lerner & Parr 2001). On the other hand, the discriminative power of individual in-fab tests is very low. Measured by the area under the ROC curve (Figure 2), the best discriminator achieves only 62 percent. Therefore, a model with no capability to learn a multivariate signal is likely to perform poorly.

Model selection

Following upon our discussion, we choose two structurally simple yet powerful candidates for the excursion detection model: Naïve Bayes classifier and boosted decision stumps. The Naïve Bayes model is a popular choice for classification tasks due to its simplicity and ability to discriminate well even if the attributes are strongly dependent (Domingos & Pazzani 1997). A decision stump is a one-level decision tree that allows discrimination along a single attribute. On a variety of classification tasks, Freund and Schapire 1996 showed that the boosting of decision stumps results in a classifier as good as C4.5.

Both Naïve Bayes classifier and boosted decision stumps can be viewed as defining a similar discriminative boundary along individual attributes. In the Naïve Bayes model, the log-likelihood ratio of class probabilities can be expressed as a linear combination of the log-likelihood ratios corresponding to individual attributes:

$$\log \frac{P(Y=1 \mid \mathbf{X})}{P(Y=0 \mid \mathbf{X})} = \sum_{j=1}^{k} \log \frac{P(X_j \mid Y=1)}{P(X_j \mid Y=0)} + \log \frac{P(Y=1)}{P(Y=0)}.$$

On the other hand, boosting produces an ensemble $\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ which is linear in the votes of the experts h_t. If the experts are represented by decision stumps, each of them is restricted to a single attribute, and thus the ensemble becomes linear in the discriminators over individual features.

Machine learning methods

Naïve Bayes classifier

Assume that we have a classification problem, where Y is a class variables and $\mathbf{X} = \{X_1, \dots, X_k\}$ denotes a vector of k attributes. By Bayes theorem, the conditional probability $P(Y \mid \mathbf{X})$ can be rewritten as:

$$P(Y \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid Y) P(Y)}{\sum_y P(\mathbf{X} \mid y) P(y)}. \quad (1)$$

By making a “naive” assumption that the attributes $\mathbf{X}$ are independent given the class label Y, Equation 1 simplifies to:

$$P(Y \mid \mathbf{X}) = \frac{\left[\prod_{j=1}^{k} P(X_j \mid Y)\right] P(Y)}{\sum_y \left[\prod_{j=1}^{k} P(X_j \mid y)\right] P(y)}, \quad (2)$$

where the denominator is only a normalizing constant. The classifier corresponding to Equation 2 is well known as the *Naïve Bayes classifier* or *model* (Duda & Hart 1973). In real-world domains, this classifier performs surprisingly well, even if the “naive” assumption is often unrealistic. Domingos 1997 explored this issue and showed that the classifier has a much larger region of optimality than previously believed, including conjunctive and disjunctive concepts. In fact, the classifier can perform optimally even if the estimates of $P(Y \mid \mathbf{X})$ are incorrect by a wide margin.

The Naïve Bayes classifier and the logistic regression model form a generative-discriminative pair (Rubinstein & Hastie 1997). More concretely, the classifier assigns a label 1 to the vector $\mathbf{X}$ if and only if $\log \frac{P(Y=1|\mathbf{X})}{P(Y=0|\mathbf{X})} = \sum_{j=1}^{k} \log \frac{P(X_j|Y=1)}{P(X_j|Y=0)} + \log \frac{P(Y=1)}{P(Y=0)} > 0$. If the conditionals $P(X_j \mid Y)$ follow normal densities, this discriminative boundary becomes identical to the one of the logistic regression model.

The conditional probabilities of continuous variables are often assumed to follow normal densities. Unfortunately, if the training sample is small and contaminated by outliers, estimates of the mean and variance become very sensitive and are no longer representative. John and Langley 1995 proposed an extension to the Naïve Bayes model that can deal with these issues. Instead of fitting a parametric conditional $P(X_j \mid Y)$, the *Flexible Naïve Bayes model* uses the following nonparametric density estimator:

$$P(X_j \mid Y) = \frac{1}{N} \sum_i g(X_j, x_j^{(i)}, \sigma), \quad (3)$$

where i ranges over the training points of the attribute X_j and class Y, $g(X_j, x_j^{(i)}, \sigma)$ is a Gaussian kernel centered in the data point $x_j^{(i)}$, and σ is a kernel width.

Decision stumps

A *decision stump* is a one-level decision tree that allows discrimination along a single attribute. The optimal decision boundary is found by minimizing a chosen *impurity measure*. Two popular choices for the measure are

```
Inputs:
    dataset of pairs $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})$,
        where $\mathbf{x}^{(i)} \in \mathbf{X}$, $y^{(i)} \in Y = \{-1, 1\}$

Algorithm:
    initialize the distribution $D_1(i) = \frac{1}{N}$
    for $t = 1$ to $T$
        train a weak learner $h_t$ on the samples from $D_t$
        $\epsilon_t = \frac{1}{N} \sum_{i=1}^{N} \{h_t(\mathbf{x}^{(i)}) \neq y^{(i)}\}$
        $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
        create a new distribution over the training samples
            $D_{t+1}(i) = D_t(i) \exp[-\alpha_t y_i h_t(\mathbf{x}^{(i)})]$
        normalize the distribution $D_{t+1}$

Outputs:
    binary classifier
        $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$
```

Figure 3: A pseudo-code implementation of AdaBoost.

the *entropy impurity*, which is used in C4.5 (Quinlan 1993), and the *Gini impurity*, which is used in CART (Breiman *et al.* 1984).

This learner can be very effective in discriminating between two classes, whereas its simplicity makes it hard to overfit. Its recent popularity is due to achieving high accuracy when boosted. On a variety of classification tasks, Freund and Schapire 1996 showed that the boosting of decision stumps results in a classifier as good as C4.5. At the same time, decision stumps are easy to learn and interpret.

Boosting

Building of highly accurate classifiers in real-world domains is a challenging task. Fortunately, we can often come up with the rules of thumb that have low accuracy, but are significantly better than random guessing. *Boosting* is a technique that allows us to combine these *weak learners* into an ensemble that represents a highly accurate classifier.

AdaBoost (Freund & Schapire 1997) is an adaptive boosting algorithm that belongs among the most popular machine learning techniques (Figure 3). Initially, the algorithm sets the distribution of training samples D_1 to uniform[4]. In every boosting round t, a weak learner h_t is trained on the samples from the distribution D_t, evaluated by its misclassification error ϵ_t, and assigned a weight a_t. Consequently, the distribution D_t is reweighted such that the samples drawn from the new distribution D_{t+1} are half correctly classified and misclassified. This procedure is repeated for T rounds and outputs an ensemble of T weak learners.

	Training set			Test set		
	Prec	Rec	Acc	Prec	Rec	Acc
DS	0.91	0.07	0.91	0.18	0.01	0.90
LR	0.53	0.88	0.91	0.10	0.48	0.51
NB	0.16	1.00	0.50	0.10	0.67	0.38
FNB	1.00	0.49	0.95	1.00	0.03	0.90

Table 1: Comparison of four baseline classifiers: decision stump (DS), logistic regression (LR), Naïve Bayes model (NB), and Flexible Naïve Bayes (FNB).

Recently, boosting has attracted a lot of attention. The main reason is its superior performance in many real-world domains that seems to avoid overfitting (Freund & Schapire 1996). Schapire *et al.* 1997 tried to explain this phenomenon and bounded the generalization error of the algorithm in terms of its margin on the training data, complexity of the classifier, and size of the training sample. Even if this result is more qualitative than quantitative, it suggests that the improvement obtained by boosting can be low if: the learner performs poorly on the training set, the learner is overly complex, or there is not enough training data. Friedman *et al.* 1998 offered an alternative explanation of boosting in terms of additive modeling and the maximum likelihood principle.

Experimental results

Experimental section is divided into three parts. First, we establish baselines for the excursion detection problem. Second, we test the proposed excursion detection models and discuss their benefits. Third, we apply one of the models online and show how its performance varies with the amount of observed test results. Comparison of the models is performed on the in-fab tests dataset discussed earlier in the paper, and the models are evaluated by their precision, recall, and accuracy[5]. Reported statistics are obtained by 8-fold cross-validation and averaged over 4 randomly chosen 8-fold partitionings.

Baselines

The procedure that classifies every lot as non-defective achieves a high accuracy of 0.9. However, it is completely useless for the purpose of excursion detection. To establish more informative baselines, we test four off-the-shelf classification techniques: a single decision stump, logistic regression, Naïve Bayes model (the conditionals of continuous variables follow normal densities), and Flexible Naïve Bayes. Splits in the decision stumps are scored by the entropy impurity measure. To allow a fair comparison of all decision stumps, missing values are replaced by their expectations. Kernel width in the Flexible Naïve Bayes model is set to $\sigma = 1/\sqrt{N}$

[4] Uniform distribution of the training samples corresponds to the zero-one loss function. Preference for other utility models can be incorporated by reweighting of the distribution (Schapire, Singer, & Singhal 1998).

[5] Precision is defined as the number of correctly classified defective lots over the total number of lots classified as defects. Recall is defined as the number of correctly classified defective lots over the total number of defects. Accuracy is computed as the number of correctly classified lots over the total number of lots.

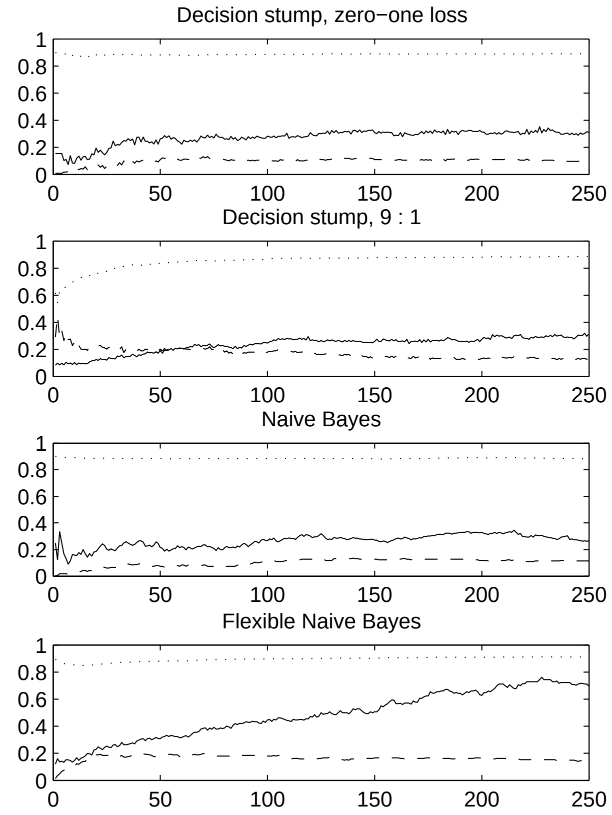

Figure 4: Comparison of boosted decision stumps and discriminatively learned Naïve Bayes models for varying model complexity. Solid, dashed, and dotted lines denote the precision, recall, and accuracy of the classifiers on the test set.

(John & Langley 1995). Performance of the classifiers is reported in Table 1.

Surprisingly, none of the baseline methods performs extraordinary well. The decision stump learner slightly overfits, which is apparent in the decline of its precision and recall on the test set. This behavior is caused by a large number of similar weak discriminators on the training set, the best of which generalizes poorly to the unseen test data[6]. The logistic regression model needs to estimate 1187 parameters, which is at least twice as many as can be fit by 510 (= 7/8 × 583) training samples, and thus overfits. In terms of accuracy, the Flexible Naïve Bayes model surpasses the Naïve Bayes classifier both on the training and test sets. The weak performance of the Naïve Bayes classifier is caused by the lack of flexibility in modeling the conditionals of continuous variables.

Excursion detection models

To improve the decision stump learner, we boost it by AdaBoost (Figure 3) up to 250 rounds. The ensemble is expected to perform better due to (1) the combination of multiple features, and (2) the maximization of the margin on the training set, which translates into a superior upper bound on the generalization error (Schapire *et al.* 1997). Boosting experiments are performed with two utility models: zero-one loss and 9 times higher reward for the correct classification of a defective lot. The latter of the utility models evens the disproportion between the good and defective lots in the dataset. To improve the performance of the Naïve Bayes models, we train them for the purpose of classification – to maximize their conditional likelihood. Both models are initialized empty and new nodes are greedily added if the conditional likelihood increases (Grossman & Domingos 2004). Performance of the improved models is shown in Figure 4.

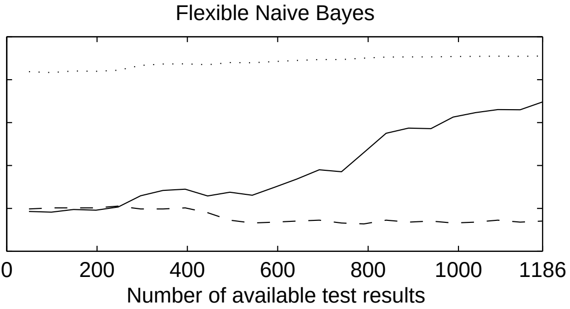

Figure 5: Performance of the discriminatively learned Flexible Naïve Bayes model (250 nodes) in online excursion detection. Solid, dashed, and dotted lines denote its precision, recall, and accuracy on the test set.

Boosted decision stumps improve significantly over their baseline in both utility scenarios. After 250 boosting rounds, the classifiers recall between 0.1 and 0.15 of defective lots with the precision exceeding 0.3. Different utility models clearly affect the trade-off between the precision and recall, but their effect diminishes with a higher number of boosting rounds. The discriminative learning of the Naïve Bayes models outperforms their generative counterparts. After adding 250 nodes, the Flexible Naïve Bayes model recalls more than 0.15 of defective lots with a precision of 0.7. Moreover, the model yields an accuracy of 0.91 and tops the other compared techniques. Finally, the model outperforms the discriminatively learned Naïve Bayes model in all measured quantities, which is attributed to its more flexible nonparametric density estimate. Its superior performance against boosted decision stumps stems from (1) the proper handling of missing values and (2) the ability to define a non-linear discriminative boundary along a single feature. Interestingly, the model needs only few features to recall about 0.2 of excursions. The rest of the features serves for the purpose of eliminating false positives – non-defective lots classified as defects. This result is consistent with our earlier observation that simultaneous deviations in several in-fab tests are likely to happen just by chance.

[6] A better selection of the decision stump can be achieved by using the wrapper method (Kohavi & John 1997). However, the effect of this method on later compared machine learning techniques was minor.

Online excursion detection

So far, we assumed that prior to the classification of a lot, the results of all its in-fab tests are at our disposal. However, excursion detection is an online task where the complete information is rarely available. To show that our excursion detection models can operate in this online setting, we choose one model and test it on a set of partially observed test results. We know the partial ordering of the in-fab tests in the wafer fabrication flow, so we can simulate how their results are gradually available as the fabrication advances (Figure 5).

The trends in Figure 5 closely resemble Figure 4. As the number of available test results grows, the accuracy and precision of the model improve at a relatively small drop in the number of detected excursions. Our results suggest that even if the model was trained at the end of the line, it can be used to detect excursions in earlier stages of wafer fabrication. Unfortunately, earlier excursion detection is paid by a lower precision in the detection of defective lots.

Conclusions

High-volume manufacturing is a highly automatized process that involves numerous monitoring devices and sensors. In this work, we explored this data-rich environment and used in-fab parametric tests to construct an automatic excursion detection system. Preliminary results show that we can recall about 0.15 of defective lots with a precision of 0.7. It should be emphasized that although we only discover 0.15 of excursions, we operate on a training set that contains only excursions that were able to get past current in-fab excursion detection systems. Detecting even a small fraction of these troublesome lots can translate into large monetary savings. Nevertheless, increasing the recall is an important direction of our future research.

To further improve the performance of our excursion detection systems, we plan to incorporate engineering knowledge on common excursion causes. Moreover, the ability to merge data for similar manufacturing processes can be the key in obtaining sufficiently large and rich training sets. Finally, this work can viewed as a step towards building a full statistical model for the discovery of excursion sources.

Acknowledgment

The first author did an internship at Intel Corporation, Santa Clara, California, during the summer 2004, which gave him an opportunity to participate in a joint effort to improve manufacturing processes by machine learning techniques. The first author was also supported by an Andrew Mellon Predoctoral Fellowship for the academic year 2004-05. We thank to John Mark Agosta, Nuriel Amir, Gary Bradski, Bob Davies, Richard Pelikan, Alex Rohas, Jeff Snodgrass, and anonymous reviewers for providing insightful comments that led to the improvement of the paper.

References

Breiman, L.; Friedman, J.; Olshen, R.; and Stone, C. 1984. *Classification and Regression Trees.* Wadsworth.

Domingos, P., and Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29:103–130.

Duda, R., and Hart, P. 1973. *Pattern Classification and Scene Analysis.* Wiley.

Fountain, T.; Dietterich, T.; and Sudyka, B. 2000. Mining IC test data to optimize VLSI testing. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 18–25.

Fountain, T.; Dietterich, T.; and Sudyka, B. 2003. *Exploring Artificial Intelligence in the New Millennium.* Morgan Kaufmann. chapter Data Mining for Manufacturing Control: An Application in Optimizing IC Tests, 381–400.

Freund, Y., and Schapire, R. 1996. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, 148–156.

Freund, Y., and Schapire, R. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Friedman, J.; Hastie, T.; and Tibshirani, R. 1998. Additive logistic regression: A statistical view of boosting. Technical report, Stanford University.

Grossman, D., and Domingos, P. 2004. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the 21st International Conference on Machine Learning*, 361–368.

John, G., and Langley, P. 1995. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, 338–345.

Kohavi, R., and John, G. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2):273–324.

Lerner, U., and Parr, R. 2001. Inference in hybrid networks: Theoretical limits and practical algorithms. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 310–318.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Rubinstein, D., and Hastie, T. 1997. Discriminative vs informative learning. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 49–53.

Schapire, R.; Freund, Y.; Bartlett, P.; and Lee, W. S. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning*, 322–330.

Schapire, R.; Singer, Y.; and Singhal, A. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*, 215–223.