Wafer Spatial Signature Analysis

Abhishek Singh and Wojtek Poppe

Abstract

Semiconductor manufacturing is a complex multi-step process that can be prone to processing issues that lead to nonfunctional chips. A significant proportion of systematic defects can manifest as spatial patterns (signatures) of failing chips on the silicon wafers. These systematic spatial patterns are indicative of process defects that need to be identified and tracked. The primary challenges of classifying failure signatures across wafers are: (a) presence of more than one systematic failure patterns (or mechanisms) on a single wafer, (b) presence of random defects that make chips fail randomly adding noise to the patterns, and (c) small amounts of labeled data. In this paper, we apply machine learning algorithms to automatically track these spatial systematic failure patterns in midst of abovementioned challenges and assess their performance.

Introduction

During the manufacturing process, semiconductor wafers go through numerous chemical and mechanical processing steps. A finished wafer is cut up into chips before they are packaged (Figure 1). Each processing step is susceptible to process variations as well as tool recipe issues that may render affected chips useless. In a majority of cases, especially during the early days of a process generation, process defects are systematic in nature and result in a systematic wafer signature of failing dies. Figure 2 shows an example of wafer with systematic pattern where more chips fail at the outer right edge of the wafer. Since each processed wafer consists of over a hundred process steps, it is common for multiple signatures to exist on one wafer. It is important to quickly identify which signature affects the most wafers, so that resources can be focused on addressing the biggest yield detractors. In this paper we compare various learning algorithms in tracking these spatial signatures across thousands of wafers. We also present a signature quality metric to identify the presence of a systematic which serves as a diagnostic tool in assessing the performance of any algorithm.

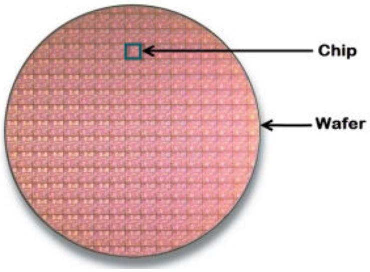


Figure 1: Manufactured Semiconductor Wafer consisting of several chips

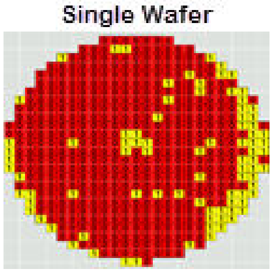


Figure 2: Yellow squares indicate nonfunctional chips. Note the high failure right on the right edge

Experimental Data and Dimensionality Reduction

Data Set

For this project we used 11 months (Jan-Nov) of data consisting 52,010 wafers. This data-set is known to capture several process issues that emerged at various points and were eventually addressed and fixed. Therefore, the systematic signatures that we aimed to track using various learning algorithms were prevalent in the early portion of the data-set and faded away as the process matured. We focused our efforts on three known signatures (Figure 3) and in particular the *Checker Rin*g pattern, which was the most common and well understood. Note the single wafer shown in Figure 2 constitutes one training example. The checker ring pattern is actually present, but it is mixed up with other less frequently observed issues such as high failure rate on the right-edge.

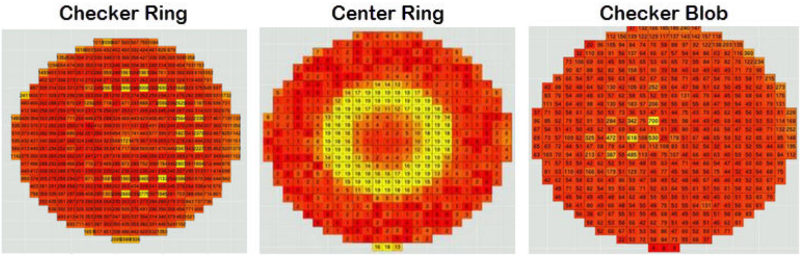


Figure 3: Real Signatures. Yellow indicates region of higher failure rates.

Feature Encoding

We chose the (x, y) location of each chip on the wafer as a feature and the PASS(0) or FAIL(1) status of the corresponding chip as the feature value. Each wafer or training example was represented by a 516 bit long feature vector, (1,1,0,1…etc). This feature encoding is referred to as "*full-wafer*" training set. To reduce dimensionality in an effort to decrease minimum training size requirements we tried three different feature encodings. These include:

- *Adaptive features*: We crop chips that form the spatial signature (also referred to as *hot-spots*) on the wafer along with their immediate neighbors. The neighboring chips are included in order to retain the contrast information between the frequently failing chips compared to other chips. Similar to the full-wafer case, the pass/fail status of each chip is used as the feature value. The length of adaptive features would change between signatures, but we train separate SVMs for each signature anyways.
- *36 Reticle-zones*: Instead of having a feature for each die, we calculate the yield or "average intensity" in multiple chips that are within one of 36 zones. The zone definitions are pretty commonly used in industry and in our case are very well suited to the checker ring pattern
- *14 Radial-zones*: Similar to 36-Reticle-zone except with 14 radial zones (also common zonal map).

Signature Quality Metric

One of the main challenges we face is a dearth of accurately labeled training examples, so we implemented a *Signature Quality Metric (SQM)* which turned out to be a very useful diagnostic for evaluating classification results. As seen in Figure 2, in a lot of cases a single wafer may not exhibit a clear signature, thus we resort to analyzing a stack of wafers. The SQM for a stack of wafers is defined as:

$$\boxed{Signature\ Quality\ Metric = \frac{\sum_{i=1}^{N} Number\ of\ failures\ at\ hotspot\ locations}{\sum_{i=1}^{N} Expected\ Number\ of\ failures\ at\ hotspot\ locations}}$$

In order to compute the SQM we need to first identify the hot-spots locations on the wafer. This can be accomplished by maintaining a baseline set of wafers that represent the majority of the wafer population unaffected by the signature being analyzed. Next, the target wafer stack is normalized with respect to the baseline wafer stack in order to fix the global process bias. The difference of normalized wafer stack and the baseline is computed for each chip location. The chip locations high delta values (>1-sigma) correspond to the hotspot locations. The baseline population is used to compute the expected failure rates at the hotspot locations. The ratio of total failure rates at the hotspot locations in the target wafer stack with respect to the total expected failure rates at the corresponding locations is used as SQM score. This metric can then be used to quantify how pure a particular group of wafers is relative to some target signature.

Results

Comparison of various Algorithms

In order to gain better insight into what types of learning algorithms will work best for spatial wafer signatures we decided to look at *Logistic Regression (LR), SVM,* and *Nearest Neighbor (kNN)*. Logistic Regression is simple to understand, SVM has shown very positive results for character recognition[1], and Nearest Neighbor could give us insights into distance metrics, which would be useful for other algorithms, such as unsupervised clustering. Figure 4 demonstrates performance for the various algorithms.

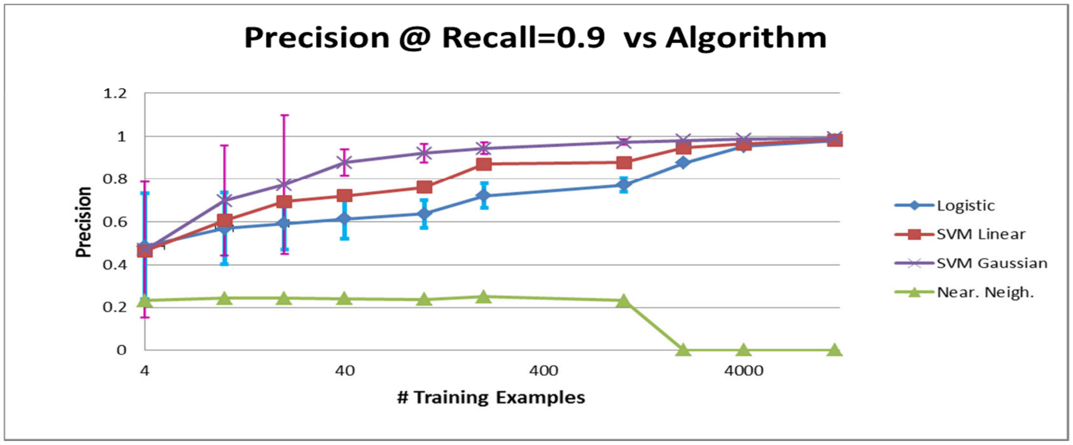


Figure 4: Comparison of various learning algorithms. Plotting precision given Recall=0.9. If recall is never 0.9, then we say precision=0. Error bars indicate a 2-sigma bound based on 100 random "bagging" runs.

We looked at a variety of metrics ranging from PRC Area to average accuracy for quantifying algorithm performance and found precision given a fixed recall to serve our purposes best. We had the following takeaways from each algorithm:

Logistic Regression (LR): It worked, but only for large training sets. Training error was always 0, which indicated Logistic regression suffers from high variance. We looked at the theta vector as seen in Figure 5a, and noticed that it took Logistic regression a lot of samples to accurately identify the "hotspot" locations, which eventually garnered a high corresponding theta value.

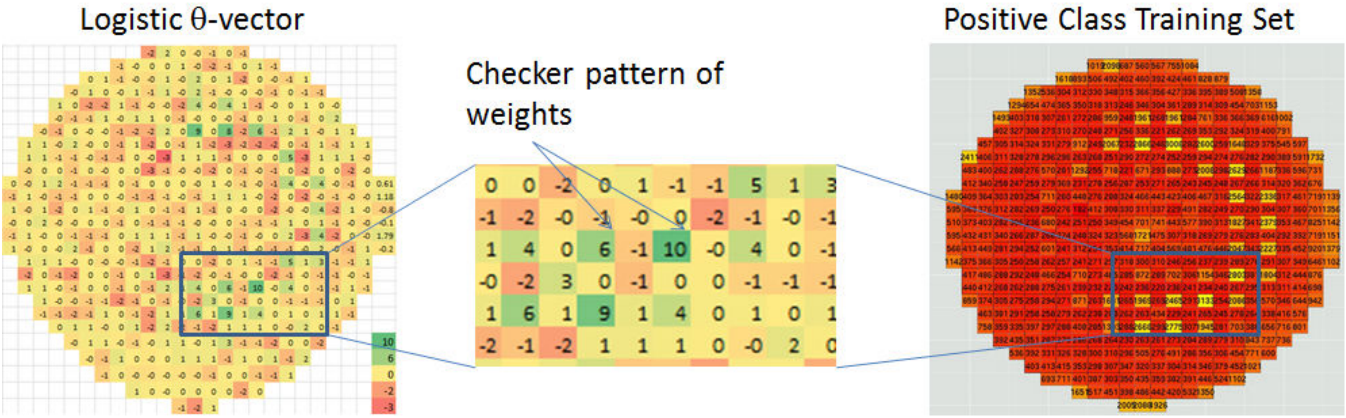


Figure 5: Example of how Logistic regression converges on a θ vector that weighs "hotspot" chip locations much more than surrounding die. Also note immediately surrounding chips are negative to indicate the need for only the hotspot being a fail.

SVM: SVM had excellent performance and still classified with some precision down to 4 training examples. This is likely due to the fact that SVM classifier not only iterates until it correctly classifies the two classes but also maximizes the margin between them. It's worth noting that the margin for error is small for smaller training sets and minimal changes in threshold could lead to drastic changes in precision. This is reflected in the significantly higher variance with training set sizes < 40 (we ran 100 random training sets for each training set size).

kNN – Nearest neighbor showed the poorest results, which is likely due to the distance metric we used. As seen in in Figure 5, LR (and most likely SVM) weigh specific chips more than others. However, kNN tends to weigh all the features equally which may wash out the signal (higher failure rate at specific locations). We do think that this can be addressed, but would require a custom distance metric for each signature.

K-Means and Mixture of Gaussians – None of the clustering algorithms we tried effectively clustered large sample sets. Small sample sets with well-chosen examples split into predictable clusters, but running a full month lead to uninteresting clusters. We feel this is largely due to the "unweighted" distance metric that also affected kNN. We observed slightly improved performance using radial and 36-zone encoding, but not enough to justify a deeper analysis in the time provided.

Comparison of Feature Encoding Schemes

Using SVM with a Gaussian kernel on the four different feature encodings lead to performance seen in Figure 6. Surprisingly the Adaptive encoding with 195 features always outperformed the 36 zone encoding, even with the smallest training set. Adaptive encoding with the reduced binary feature set worked best for all signatures in all cases. 80% precision was achievable at 90% recall, but only for a highly "tuned" threshold setting. In other words, the algorithm would pick the correct class, but with very low confidence, and classification results would vary between training sets of the same size. Fortunately the SQM score could be used to help alleviate this issue. We found that we could sub-sample even a small training set, predict groups with high SQM, and then re-train on previously unlabeled data to get more stable and predictable SVMs. As another vote of confidence for binary encoding, the full wafer SVM, with 516 features, outperformed the 36 zone SVM when we had at least 20 training examples.

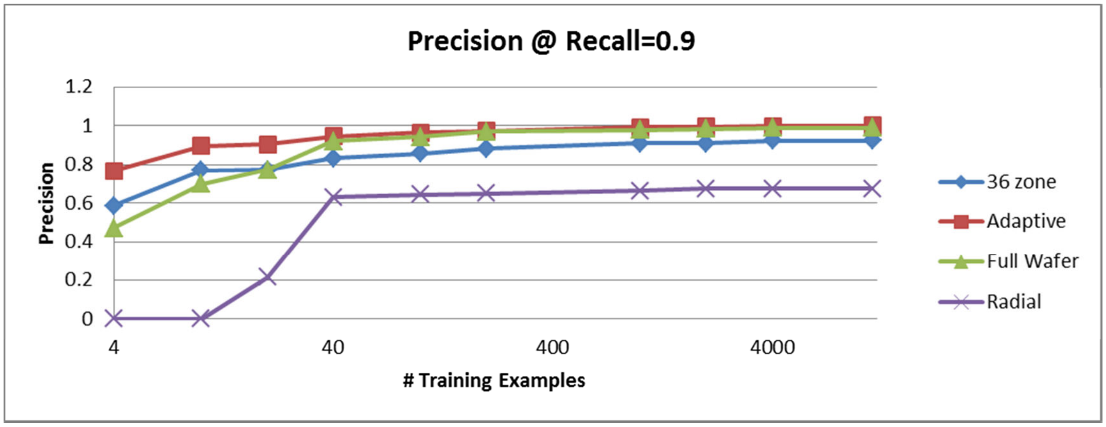
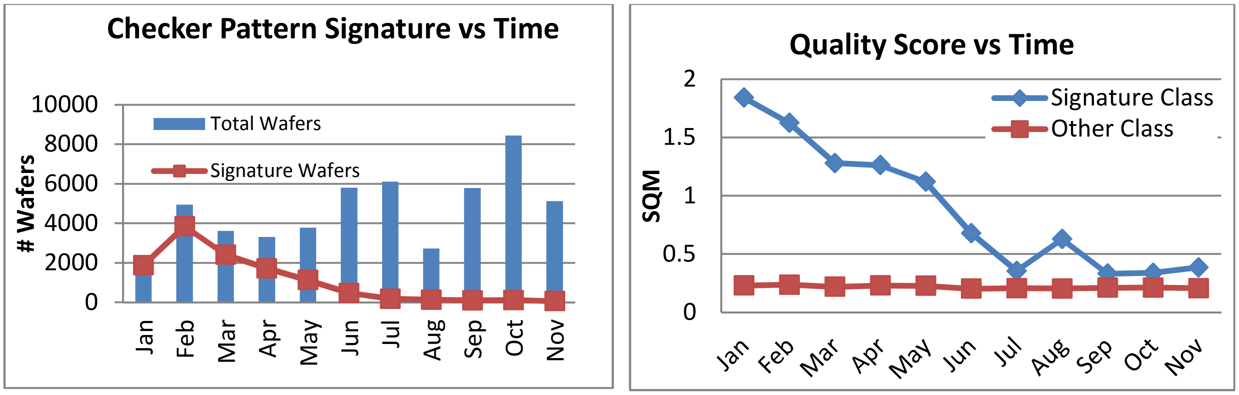


Figure 6: Performance of the four different feature encoding schemes. Error bars were not drawn to keep the graph clean, but are similar to the SVM Gaussian error bars seen in Figure 4

To demonstrate the robustness of our algorithm we ran the Adaptive SVM one month at a time on the full training set and were able to produce a chart of impacted wafers over time. We were able to accurately detect 80% of the wafers in January when this issue was prevalent and also in August when <5% of wafers were affected. Note the quality score in Figure 7b is flat for the "*Other*" class, which is a diagnostic we use when looking at unlabeled data. The signature quality or signal from this particular process defect also decreased over time as the issue was addressed and fixed. There was an interesting bump in August when a couple lots of wafers were released from quarantine that were actually processed earlier in the year (hence had a stronger signature). The chart in Figure 7a is the essence of what we wanted to achieve with this project.



(a) (b)

Figure 7 (a) Signature wafers identified vs time. 79% wafers identified in Jan and only 1% in November. (b) Quality score for Signature and Other class. Note how flat the Other class signature quality is. This is a strong indicator of a good classifier.

Discussion

Support Vector Machines have demonstrated high performance in character recognition[1] and we feel have performed even better in spatial wafer pattern recognition. One potential reason why is that spatial wafer patterns will generally have the same exact chips (or pixels) failing. Image recognition techniques tend to do well with good centering of target objects, so ours is somewhat the extreme case of that. This may be related to why binary features perform so much better as specific chip (or pixel) location is an important factor. Adaptive features also show a dramatic improvement as a lot of the noise from other parts of the wafer that may have other systematics is filtered out. In order to push training set requirements even further, we are developing a “semi-supervised” type strategy where we can iteratively “evolve” a training set using unlabeled data and SQM scores from multiple runs with slightly different training sets[4]. This is an area of active development we plan to pursue more rigorously in the future.

Conclusion

We have demonstrated in this paper that Support Vector Machines can be very effectively trained to identify spatial wafer patterns of failing chips. The most effective feature encoding is a binary one, for pass=0 and fail=1, with an “adaptive” feature vector that filters out “uninteresting” chips on the wafer. We are hence able to take a relatively small training set and calculate yield impact with very high precision. We still need to automate most of the flow in a robust manner before users can just throw in a handful of wafer and get good results, but the steps necessary are much clearer after completing this project.

Acknowledgements

We would like to thank Andrew Maas for all his guidance and advice during all his project office hours and beyond. We also want to thank Prof. Andrew Ng for such an insightful course on Machine Learning.

References

[1] **Fast and Accurate Digit Classification** (*eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-159.pdf*)

[2] **LIBSVM: A library for Support Vector Machines** (*http://www.csie.ntu.edu.tw/~cjlin/libsvm*)

[3] **Weka 3: Data mining software in Java** (*http://www.cs.waikato.ac.nz/ml/weka*)

[4] **A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data** (jmlr.csail.mit.edu/papers/volume6/ando05a/ando05a.pdf)