Classification of Wafer Process Quality and Equipment Fault Diagnosis

Takuro Tsutsui
takurot@stanford.edu

Hironori Moki
hironori@stanford.edu

Abstract

Semiconductor device manufacturing equipment is very complex. The wafer manufacturing process also is a complex mix of chemical and physical phenomena, monitored with many sensors. This complexity makes it hard to fully understand the nature of the process. The yield[1] of the process is most important for the device-maker. To ensure the quality of wafers, device-makers measure sample wafers. However wafer quality measurement is expensive, takes a long time, and reduces throughput. This project aims to classify process sensor readings into those producing good or bad wafers. A good classification algorithm can reduce the number of measurements and increase throughput. This project also tries to modify the classification algorithm to determine the sensor most likely causing poor quality. From this, an engineer can determine the nature of the process failure.

1. Introduction

This project focuses on the classification of the wafer process. To motivate this, the semiconductor device manufacturing equipment vendor is required to identify the cause when the process result gets out of the range the device-maker expects. Normally quality measurement values are continuous, but they are given as discrete values of Good/Bad to hide the details of the process[2]. In addition, the equipment vendor wants to know the *cause sensor* which is the sensor indicating the root cause of the problem. Analysis conditions are very tough as typically device-makers provide very small samples to investigate, the number of bad wafers is small because trouble is very rare, and the equipment has many sensors.

There are two applications that this project focuses on

1) Classification of Good/Bad wafers
   Train with many good, many bad wafers then predict test wafers.
2) *Cause Sensor* Diagnosis
   Train with many good, few bad wafers then check the causing sensor.

This project uses the SVM (Support Vector Machine) method to solve these problems.

2. Datasets

This project used the following datasets.

Datasets	Samples(Good/Bad)	Sensors	Sensor Read	Process Steps[3]
1) Classification Good/Bad wafer	154 wafers Good : 111/Bad : 43	84 sensors	3463 times/wafer (=346.3sec) (0.1 sec interval sampling)	21 steps
2) *Cause Sensor* Diagnosis	100 wafers Good : 91/ Bad : 9	47 sensors	325 times/wafer (=325 sec) (1 sec interval sampling)	24 steps

Here is an example of one wafer data.

WaferID	Time	Step	GasFlow A	GasFlow B	⋯	Temperature A	Temperature B	⋯	Electric A	Electric B	⋯	Electric Z
1	0.1	1	0	0.4		61.8	20.5		19.9	20		0
1	0.2	1	200	0.4		61.9	20.5		19.9	20		0
1	0.3	1	200	0.4		61.9	20.5		19.9	20		0
1	0.4	1	200	0.3		61.9	20.5		19.9	20		0
1	0.5	1	200	0.3		61.9	20.5		19.9	20		0
1	346.2	21	75	75		60.8	20.9		20	20		15
1	346.3	21	75	75		60.8	20.9		20	20		15

The first dataset has 3463 * 84 = 290892 features per wafer and second dataset has 325 * 47 = 15275. In these datasets, the number of features n is much larger than the number of samples m.

In addition, the classification problem is made difficult by the following

[1] Yield is the proportion of good chips on the wafer.

[2] Usually real quality measurements are confidential. They are hard to get.

[3] A wafer process is divided into process steps. Sensor setting values are usually different in each step to control the process condition.

a. Sensor values include noise.
b. Some sensors have strong correlation with others giving redundant features

We need to find a solution given these hard conditions.

3. Preprocessing

Datasets have many features. Feature selection is a useful way to avoid overfitting for machine learning. So we evaluated applying the following preprocessing steps to reduce the number of features before training.

1) Binning
   Grouping data by some duration.
   Compute average, minimum, maximum, standard deviation for each process step.
   (Usually each process step has different condition for processing, e.g. sensor control value is different)
2) Compression
   PCA (less computation time, noise reduction)
3) Transformation
   Uniform Whitening: Append random noise within sensor resolution amount to avoid sensor noise effect

4. Classification

This project uses algorithms in libsvm-3.17[4] for classification

C-SVM

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i, \quad i = 1,\cdots,m$$
$$\xi_i \geq 0, \quad i = 1,\cdots,m$$

ν-SVM

$$\min_{w,b,\xi,\rho} \frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } y^{(i)}\left(w^T x^{(i)} + b\right) \geq \rho - \xi_i, \quad i = 1,\cdots,m$$
$$\xi_i \geq 0, \quad i = 1,\cdots,m$$
$$\rho \geq 0$$

For the classification application, we use dataset 1) described before. We reduced features from 290892(=3463 times * 84 sensors) to 7056 (=21 steps * 4 statistics * 84 sensors) by using binning. We used hold-out cross validation: trained 70% samples (107 wafers) of dataset selected by random sampling from each of good/bad labels and predicted 30% samples (47 wafers). We used linear kernel with tuning parameter C of C-SVM (1000, 100, 10, 1, 0.1). We ran C-SVM 10 times for each parameter setting. Each trial had prediction errors but no training errors **(Fig 1, Fig 2)**. We realized that there were big differences for each trial even if using the same parameter C.

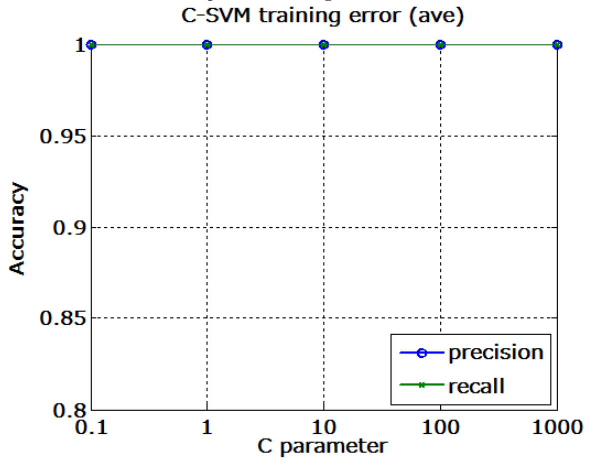


Fig 1. C-SVM Training Accuracy

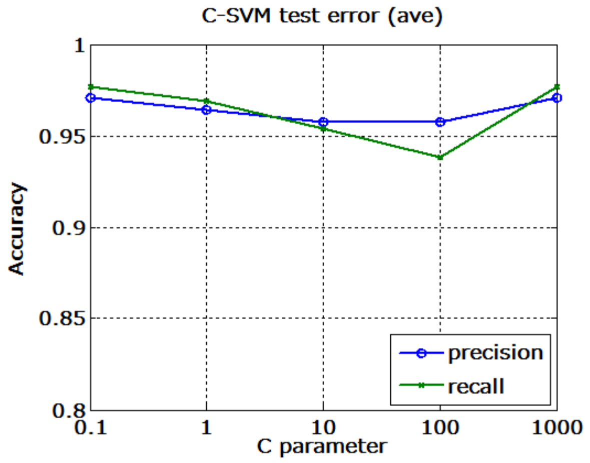


Fig 2. C-SVM Prediction Accuracy

After considering the reason, we found that the dataset includes two inseparable samples **(Fig 5. Dataset samples distribution)**. It seems that these inseparable samples do not have enough information (e.g. the key feature is not measured by sensor or incoming wafer already had some problems caused by previous process). When a training dataset includes an inseparable sample, no training errors occur but prediction errors increase. These results are depending on inseparable sample. Inseparable samples dramatically change the hyperplane angle if a training dataset includes them. This makes overfitting problems easily occur in the hard margin case. We realized that the C parameter of C-SVM is the key factor to avoid the effect of inseparable

[4] libsvm-3.17 is A Library for Support Vector Machines provided by National Taiwan University

samples. It is difficult to choose a good value because it depends on the training dataset, especially whether the training dataset contains inseparable samples or not.

Thus we evaluated ν-SVM. The ν-SVM accepts error samples at a given error rate ν. It is preferable for our situation because our dataset might have some inseparable samples at a certain rate. It is easy to define in advance because if we expect 5% training error samples, we just set ν parameter to 0.05. We tried ν-SVM with various ν parameters (0.5, 0.25, 0.1, 0.05, and 0.01) to find the best classifier. Using ν-SVM with a sufficient value of ν, we get training or prediction errors only on inseparable samples. But too small a ν parameter has a high penalty for errors so that SVM is eager to find the hyperplane. As a result it creates overfitting models. This overfitting model has many prediction errors. Thus it is important to choose a correct ν parameter.

Here is the result of ν-SVM (ν=0.05). The training automatically ignores inseparable samples **(Fig 3. ν-SVM Training)**, and prediction can detect other inseparable sample **(Fig 4. ν-SVM Prediction)**. In addition, the *cause sensor* given by this model makes sense in this dataset (*Cause sensor* is described in next section).

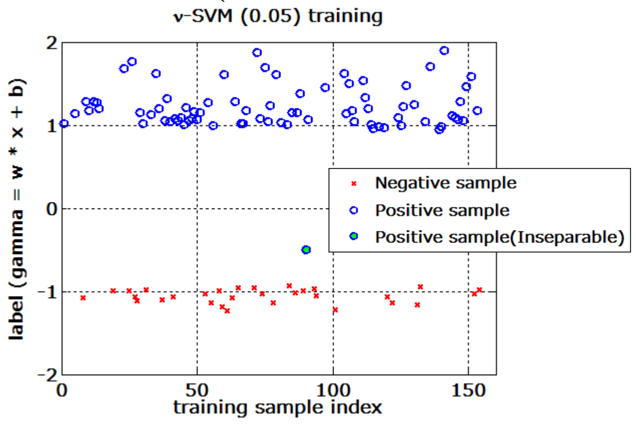


Fig 3. ν-SVM Training

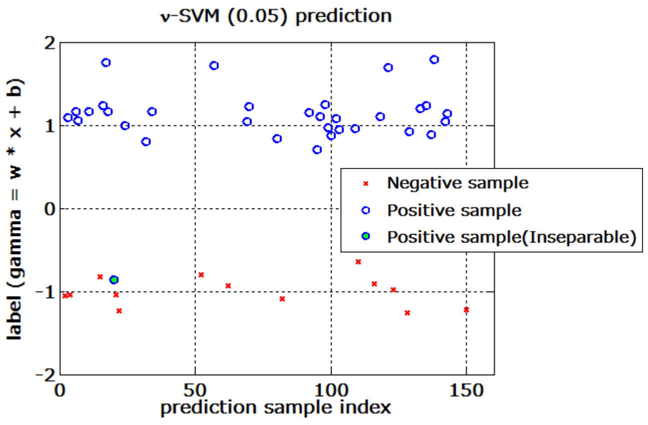


Fig 4. ν-SVM Prediction

We also tried using PCA to reduce features. Define x_{train} as an original training feature vector, V as a projection matrix of PCA which is made by x_{train} and z_{train} as a projection of point x_{train} onto the direction of matrix V. Then

$$z_{train} = x_{train}V^{-1}$$

Input first k components in z_{train} as features into SVM. In this project, compute the accumulated proportion of components and take k components such that the accumulated proportions exceed 0.98. PCA can reduce the feature number to 4 or 5 components. For prediction, the test dataset feature vector x_{test} is also projected by same matrix $V(z_{test} = x_{test}V^{-1})$ and then predicted using the first k components in z_{test}. In this way, ν-SVM with PCA still provides good predictions. The results both of ν-SVM and ν-SVM with PCA are not significantly different for this dataset. However, PCA is helpful to understand visually what the sample distribution will be. We expect that PCA will not improve SVM much, but it tells us that there are only few (4/5) dimensions (independent features) in the training data.

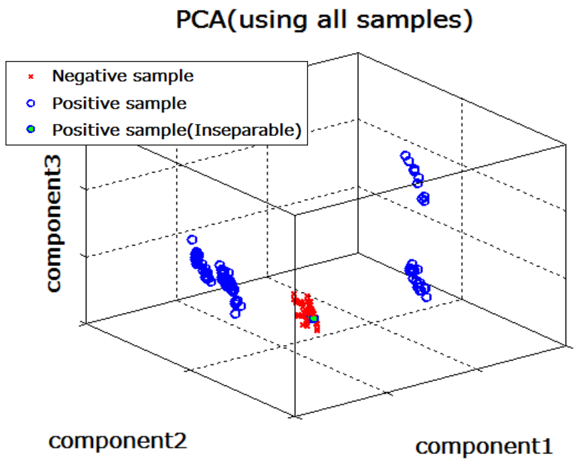


Fig 5. Dataset samples distribution
(Plot using top 3 principle components)

4.1.1. Classification Summary

The model that is made with a hard margin is too sensitive to the training dataset. There are many complex conditions in the wafer process therefore the model generated with hard margin makes misclassifications easily. It is easier to choose parameters for ν-SVM than C-SVM and we can get a better classifier which is independent of the effect of inseparable samples by giving a sufficient parameter.

4.1.2. Classification Next Action

Currently we don't have enough datasets to evaluate classification application. We need to evaluate a variety of datasets to establish a robust algorithm for classification. We wish to use predicted values as a quality index to reduce real measurement. In order to achieve the purpose, we need to collect real quality values from device-makers to clarify if the predicted value is reasonable or not. For example, this wafer is good but it is almost a bad wafer. If this application works well, we will be one step closer to the goal of replacement of real measurement.

We also wish to create *Inseparable Sample* diagnosis by excluding mislabeled or unexplainable samples in advance using a one-class SVM algorithm.

5. *Cause Sensor* Diagnosis

For *Cause Sensor* diagnosis we aimed to analyze semiconductor equipment trouble and find the *cause sensor*, which is the sensor indicating the root cause of the problem. In order to identify the *cause sensor*, this project uses *w* value of the SVM result as a weight vector of cause features. The weight vector indicates which axes, thus which features, are most significant for classification. We define the unsigned feature magnitude vector *F* as follows

$$\boldsymbol{F_j} = \frac{|\boldsymbol{w_j}|}{\sum_{i=1}^{n}|\boldsymbol{w_i}|}$$

The summary of this diagnosis procedure is as follows
-Binning by step
Bin the data by calculating statistical information for each process step.
-Learning the good/bad labels
Execute SVM learning with good/bad labels and all statistics as explanatory variables.
-Ranking the features based on the feature magnitude
Calculate *F* from the SVM model and ranked the features by feature magnitude value.
-Evaluation of result
Evaluate the quality of the learning by measuring how well the ranking predicts the (known) cause sensor.

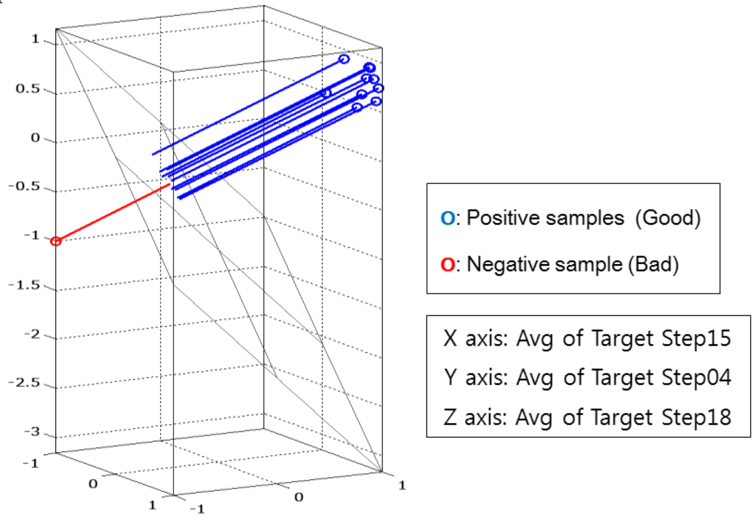
For this application, we use dataset 2 described before.
Field engineers told us which sensor is the highest impact on quality for this data. This sensor, which we will call “Target sensor”, has a different value between normal wafers and error wafers. Based on Classification results, we use ν-SVM with ν parameter value 0.05 to evaluate the feasibility of this algorithm.

In actual practice, we can expect small unbalanced datasets. Thus we evaluated a dataset containing the sample with the strongest error and the 10 normal samples preceding it. At first we used 4 kinds of statistical data (Mean, Max, Min, Stddev). So the training data for SVM learning was an 11-by-4512 matrix (11 samples and 47 sensors * 24 steps * 4 kinds statistical values). The target sensor was not within the top 10 features of *F*. We assumed this was caused by an unbalanced dataset with too many features relative to the number of samples. We tried again using only Mean with an 11-by-1124 training data matrix. Then results were better than previously, but the target sensor didn’t come up consistently. So, we investigated the correlation between label information and each feature and found many features had too high of a correlation caused by incidental sensor noise. So we used uniform whitening as follows:

$$\boldsymbol{Whiten\,SensorValue = SensorValue + SensorResolution(2 * UniformRandomValue - 1)}$$

After whitening, the high correlations caused by sensor noise disappeared and we could get the expected result consistently. One example of a good result is shown in the below table and graph. The left table shows the top 10 features of *F* and the right graph shows the top 3 features with their separating hyperplane.

Ranking	Feature	Magnitude
1	**Avg of Target Step15**	0.00693
2	**Avg of Target Step04**	0.00691
3	**Avg of Target Step18**	0.00641
4	**Avg of Target Step05**	0.00637
5	**Avg of Target Step10**	0.00628
6	**Avg of Target Step02**	0.00604
7	Avg of Other B_Step15	0.00596
8	Avg of Other A_Step05	0.00594
9	**Avg of Target Step11**	0.0059
10	Avg of Other C_Step15	0.0059



Using the preprocessing strategy established above, we evaluated how many samples would be needed to obtain satisfactory

Takuro Tsutsui takurot@stanford.edu
Hironori Moki hironori@stanford.edu

results. We randomly sampled bad/good wafers from the 100 wafer dataset (5/50, 3/30, 2/20, etc.). We performed 30 trials on each sample size and calculated the average ranking of the target sensor in the results, i.e. TOP *k* means the target sensor appears in the top *k* highest features in *F*.

Sample Num [Bad / Good]	55 samples [5 / 50]	33 samples [3 / 30]	22 samples [2 / 20]	16 samples [1 / 15]	11 samples [1 / 10]	6 samples [1 / 5]
TOP 1	1.000	0.933	0.900	0.567	0.467	0.400
TOP 3	1.000	1.000	1.000	0.833	0.767	0.544
TOP 10	1.000	1.000	1.000	0.967	0.933	0.780

We also tried PCA for feature number reduction on *Cause Sensor* diagnosis. We can reduce the number of features to about 30 components (over 0.98 cumulative contribution ratio) using PCA. Using these components as features in SVM gave almost the same results as non-PCA. In order to calculate *F* with PCA, we need to project component weights to sensor weights by matrix *V* as follows:

$$F^{PCA}{}_j = \frac{|Vw_j|}{\sum_{i=1}^{k} |Vw_i|}$$

5.1.1. *Cause Sensor* Diagnosis Summary

Using the sample with the strongest error, we got consistently good results with 1/10 bad/good samples. However, to diagnose a random error more samples are needed, around 20 or so.
Given our results,
- When we must use few samples of data with many features, we should use whitening to remove incidental sensor noise.
- We need around 20 samples to get satisfactory results. In practice, most analysis situations have around 10-25 samples, so this diagnosis can be effective.
- The visualization of *Cause Sensor* diagnosis is very effective.

5.1.2. *Cause Sensor* Diagnosis Next Action

Evaluate how the algorithm works when some sensors are highly correlated with each other.
Create Multiple SVMs for one classifier
a). Execute SVM with all good wafers and one bad wafer
b). Repeat a) for each bad wafer
c). Combine the results of each SVM and summate of all feature magnitude

6. Project Summary

We found SVM very useful for our problems, for the following reasons.
- It works well with coarse quality information like good/bad, and we often encounter this situation.
- It also works well if the dataset has few samples.
- Furthermore, there is an effective way to handle inseparable samples.

We would like to keep evaluating and try new approaches to apply machine learning into our product.

7. Acknowledgement

We would like to express our gratitude to Prof. Andrew who taught us a lot of machine learning knowledge that was immeasurably valuable throughout the course of our study. Special thanks also to TAs who gave us invaluable comments and support.