Modeling and Selection for Real-time Wafer-to-Wafer Fault Detection Applications

Jae Yeon (Claire) Baek

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-215
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-215.html

December 1, 2015

Modeling and Selection for Real-time Wafer-to-Wafer Fault Detection Applications

by

Jae Yeon Baek

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Costas J. Spanos, Chair
Professor Laura Waller
Professor Robert Leachman

Spring 2015

Modeling and Selection for Real-time Wafer-to-Wafer Fault Detection Applications

Abstract

Modeling and Selection for Real-time Wafer-to-Wafer Fault Detection Applications

by

Jae Yeon Baek

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Costas J. Spanos, Chair

The semiconductor manufacturing industry currently faces many challenges in terms of metrology and process control. With the delay of EUV and the advent of high aspect-ratio 3D structures, there is an increase both the number of complex processing steps and systematic/random errors, and optical resolution for metrology has now reached its limit for sub-14nm devices. The industry now requires real-time wafer-to-wafer control and in-line metrology, such as scatterometry or virtual metrology, for effective process monitoring.

Data models provide a quick and flexible way for integrating different forms of information. For example, in metrology, often times it is useful to combine sensor data, previous measurements, and other types of signals to extract the best possible measurement. Moreover, as the number of process steps continues to increase, explicit physical modelling of each step becomes extremely time-consuming and empirical data models will quickly become an effective alternative. In this dissertation, we discuss the application and usefulness of empirical data inference models in the context of W2W advanced process control, specifically focusing on wafer fault detection.

We first use virtual metrology, a type of in-line metrology technique, to determine whether the introduction of such data inference models is actually useful for the fab. Moreover, results show that the effective cost is determined by not only the model type and accuracy, but also the resulting false and missed alarm patterns. In the next chapter, we demonstrate an application of data models to fault detection by constructing a support vector machine classifier (SVM) and using only the diffraction signatures from scatterometry measurements to detect alarms. In the last chapter, we develop an algorithm for the SVM that allows one to choose the optimal false and missed alarm combination based on an asymmetric cost function. Moreover, our algorithm can be generalized for optimal hyperparameter selection for any SVM problem.

To my family and CC

Contents

List of Figures

List of Tables

Acknowledgments

Foremost, I would like to thank my adviser and mentor, Professor Costas J. Spanos, for guiding me through my graduate studies at Berkeley. He has given me both freedom and guidance to pursue my research interests in novel and interesting directions. I cannot thank him enough.

I would like to thank Robert Socha of ASML Santa Clara for providing me with an internship opportunity to test my ideas with YieldStar. The internship also laid the groundwork for the hyperparameter selection work discussed in Chapter 4 of this dissertation. He was an excellent mentor to me and I gained substantial knowledge of the industry through him.

I would like to thank Professors Kameshwar Poolla, Laura Waller, and Robert Leachman for being on my Qualifying Exam committee and dissertation committee. They have given me very helpful feedback for my graduate work.

I would like to thank Yuxun Zhou of our group. He was my co-researcher for the latter part of my graduate studies and even more, a good friend.

I would like to thank all our group members: Zhaoyi Kang, Ming Jin, Ruoxi Jia, Ioannis Konstantakopoulos, and Yovana Gomez for all their support and help.

I would like to thank all members of IMPACT+. I had a great exchange of ideas through presenting my work and meeting new colleagues at its workshops.

Finally, I would like to thank Samsung Scholarship for funding both my undergraduate and graduate studies. I could not have completed my degrees without their help.

Chapter 1

Introduction

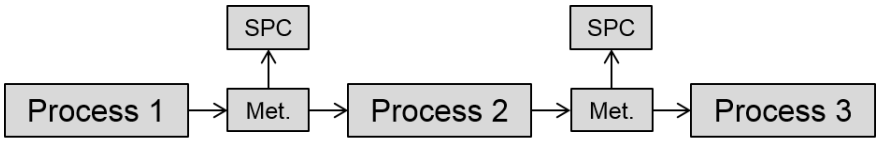
1.1 Wafer-to-Wafer Advanced Process Control

Increasingly complex manufacturing processes and reduced device dimensions demand ever tighter control in semiconductor manufacturing. Traditionally, improvement efforts were focused on the tools and process themselves. In the mid-1980's, as the number of process steps and amount of process data increased, the industry began looking into *in-situ* metrology as well as off-line metrology, in an attempt to increase process visibility. Most of the efforts were focused on utilizing sensor and metrology data. [1]

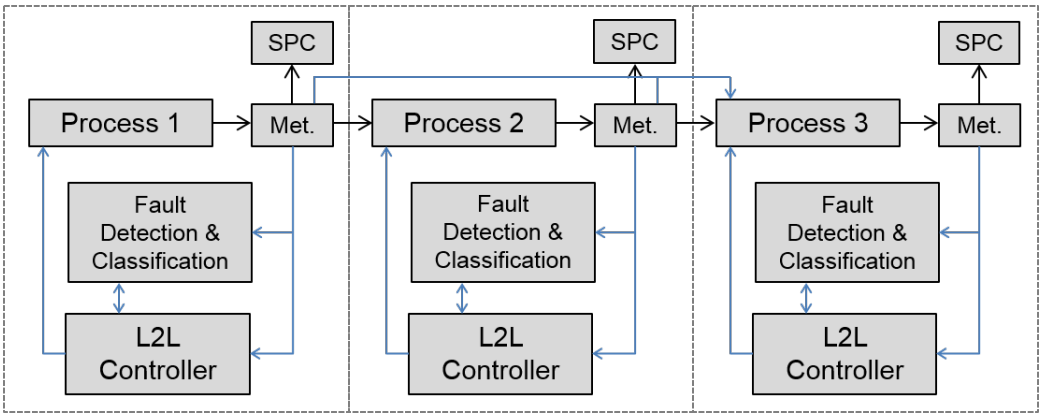
Statistical process control (SPC) was the first mechanism that arose to address this need. Introduced in the early 1930's by W. Shewhart, statistical process control (SPC) has been used in the microelectronics fabrication industry to monitor deviations from statistical control [1]. Using the control chart as its main tool and assuming a Gaussian distribution on the sensor data, SPC aimed to detect occurrence of shifts in process performance so that investigation and corrective action may be undertaken to bring incorrectly behaving manufacturing processes back under control [2, 3]. A process with SPC correction is shown in Figure 1.1a. After being processed, wafers are randomly selected from a lot at each step and measured at an off-line metrology station. These measurements are then recorded on the control chart for outlier monitoring.

Although effective in monitoring process variability, SPC does not prescribe automatic control actions [4]. Due to this need for continuous process tuning, advanced process control (APC) evloved rapidly in the semiconductor industry in the 1980's and 1990's with Run-to-Run (R2R) control emerging as the first technologically viable product of that evolution [1]. Since collecting *in-situ* metrology data was difficult, R2R control mainly considered modification of a product recipe with respect to a particular machine process *ex-situ*, or between machine runs, to minimize process drift, shift, and variability. [1] In general, APC now referred to a control framework that allowed R2R control, fault detection and prediction, predictive maintenance, and SPC.

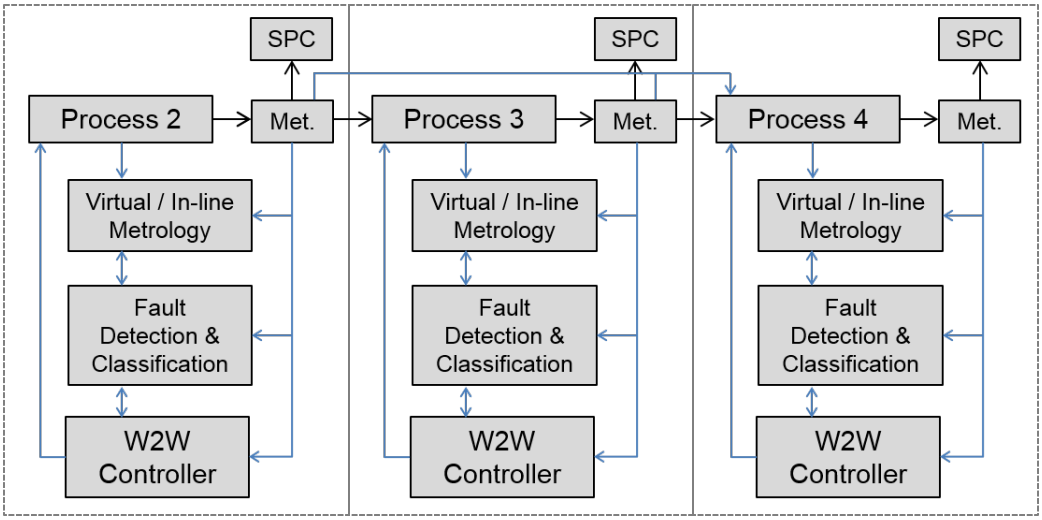
The first R2R control systems employed were lot-to-lot, and uses lot-level metrology



(a) Statistical process control. A few wafers are randomly sampled from a lot. Measurements are then recorded on a control chart and statistical parameters are used as indicators of out of control processes.



(b) Lot-to-lot process control. In contrast to statistical process control, lot-to-lot control is a form of advanced process control in which automated correction steps are taken at a lot-level. That is, the recipes are held constant for each lot.



(c) Wafer-to-wafer process control. Advanced process control occurs at a wafer level and due to off-line metrology delay, in-line metrology tools such as scatterometry or virtual metrology are necessary.

Figure 1.1: Evolution of advanced process control in semiconductor manufacturing.

data [5, 6]. This means that like SPC, a few wafers within a lot are measured at the metrology station and the control values are kept the same for all of the wafers in a lot [6, 7]. Figure 1.1b shows a process with lot-to-lot control. SPC is still being used but measurements are fed into fault detection and classification engines and a lot-to-lot controller. Occasionally, measurements from previous steps are feed-forward into the controller. Depending on the measurement received, the controller modifies the recipe for the process so as to minimize variation. However, with shrinking device dimensions and increasing wafer diameters, even L2L control is insufficient and wafer-to-wafer (W2W) control becomes crucial to maintaining high quality and productivity.[6, 7, 8, 9, 10]

The largest obstacle in deploying W2W control is the delay of off-line metrology equipment. Although conventional, off-line metrology produces relatively accurate measurements, it is also costly and time-consuming because of the time gap from sending the test wafer to the equipment to getting the results [9]. For example, even if there is a sudden drift or shift in the process, this cannot be detected real-time, causing defectiveness to all wafers processed in the delay period [8]. The fab incurs a significant cost due to these various disadvantages of conventional metrology. [6] Since monitoring is now needed at a wafer-to-wafer level, off-line metrology alone is insufficient and this drove the need for *in-situ* metrology, where the wafer is measured during or right after it is processed. A W2W control schematic is shown in Figure 1.1c.

Although all process parameters are important for effective product manufacturing, some so-called critical dimensions (CDs) are determined to be the most important. CD's typically encompass those that define the electrical performance of transistors or interconnect. Currently, the main metrology workhorse for measuring the length of transistor gates has been the scanning electron microscope (SEM), where measurements are collected off-line using dedicated, throughput-limited equipment. Due to the need for W2W control, the two candidates for in-line metrology that arose in the 2000's were scatterometry, in which gratings are measured through matching diffraction signatures, and virtual metrology, in which tool sensor data is used to infer an estimate of the CD through statistical modeling techniques. Figure 1.1c shows real-time tool sensor data and other signals entering the in-line metrology, FDC, and W2W control systems in addition to the off-line metrology data. Control actions are taken on a wafer-to-wafer level.

In the next section, we briefly go over the background of some in-line metrology tools that are currently being deployed or tested.

1.2 Background on In-line Metrology

In this section, we introduce three important in-line metrology techniques. The first is virtual metrology, in which statistical modeling or control techniques are used to estimate a CD value based on tool sensor data or basically any other features that may provide additional information on the measurement. The second is scatterometry, or optical CD (OCD), in which a separate test grating with periodic structures is measured through shining

light and collecting the intensity and/or phase information of the diffracted orders. The last but not least is hybrid metrology, which is a framework for combining measurements from different tools to get an enhanced measurement on the primary tool, usually scatterometry.

1.2.1 Virtual Metrology

The metrology delay from physical metrology stations makes wafer-to-wafer monitoring and control difficult for today's manufacturing lines, as thousands of wafers are processed in a fab per day. A potential solution to this is virtual metrology (VM), which takes the processing data produced by the processing tool in real time, (e.g. plasma etching data during isolation trench formation, for example) and predicts an outcome of the wafer (e.g. critical dimension of the trench) utilizing an inference model. Tool data utilized in fault detection is usually used as the input to the VM model. These may be statistics of gas flows, pressure, temperature, plasma parameters, etc. Implementation of a good virtual metrology model enables W2W real-time quality control and reduces the cycle time, in addition to decreasing number of test wafers [11].

VM models are constructed by collecting highly accurate off-line measurements and fitting a predictive model. We assume we have a vector of off-line measurements $\boldsymbol{y}_{SEM}$, and a matrix of tool data $\boldsymbol{X}_{SEM} = [\boldsymbol{x}_1^T \dots \boldsymbol{x}_n^T]^T$, where there are a series of n samples and each $\boldsymbol{x}_i^T \in \mathbb{R}^p$ is a row vector of p different tool data statistics. The most simple but widely used methods are regression-based algorithms, in which the CD measurement is assumed to be modeled as a linear additive model

$$\boldsymbol{y}_{SEM} = \boldsymbol{X}_{SEM}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1.1}$$

where $\boldsymbol{\beta}$ is a column vector of coefficients, and $\boldsymbol{\epsilon}$ is usually assumed to be a vector of independent (although often times they are not) Gaussian noise variables with each element sampled from $N(0, \sigma^2)$. The best estimate for $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}_{SEM}^T \boldsymbol{X}_{SEM})^{-1} \boldsymbol{X}_{SEM}^T \boldsymbol{y}_{SEM}.$$

Thus, the CD estimate for an incoming tool signal with feature vector $\boldsymbol{x}_{new}^T$ would be

$$\hat{\boldsymbol{y}}_{new} = \boldsymbol{x}_{new}^T \hat{\boldsymbol{\beta}}.$$

The equations described above are for ordinary least squares (OLS). Although this model is the core of many regression-based algorithms, in practice more complex models such as principal component analysis (PCA) and partial least squres (PLS) are used to further reduce dimensionality of the data matrix and capture linear combinations of features that are more effective in explaining the correlation between the tool data and off-line CD measurements.

After the CD has been estimated for an incoming sample, R2R control algorithms are used to automate recipe corrections. One well-known algorithm is the exponentially weighted moving average (EWMA) filter. Although the process model is very similar to (1.1), automatic control actions are taken one sample at a time. We assume again, the true process at time t is modeled by

$$y_t = \beta_0 + \boldsymbol{x}_t^T \boldsymbol{\beta} + \epsilon_t,$$

where we have included an offset term, β_0, omitted from equation (1.1). The EWMA filter estimates the noise as

$$\hat{\epsilon}_{t+1} = \lambda(y_t - \hat{\beta}_0 - \boldsymbol{x}_t^T \hat{\boldsymbol{\beta}}) + (1-\lambda)\hat{\epsilon}_t \tag{1.2}$$

where $\lambda \in [0, 1]$ is the EWMA coefficient. If the target value is τ, the control action taken for the next step of tool settings is

$$\boldsymbol{x}_{t+1} = \hat{\boldsymbol{\beta}}^{-1}(\tau - \hat{\beta}_0 - \hat{\epsilon}_{t+1}). \tag{1.3}$$

If successfully employed, VM can potentially introduce many benefits. Since VM monitors each wafer, this leads to higher yield and less scrapped wafers. Moreover, tighter process control provides a way to overcome metrology delay in L2L control, and less time can be spent on time-consuming *ex-situ* metrology, increasing the throughput of a process. [12] However, one of the biggest drawbacks of VM is the limited accuracy of the model. In contrast to direct metrology, such as the SEM or AFM, VM utilizes a data model to infer the CD indirectly, which often results in a compromise in the accuracy of the measurement. A question we will be exploring in this work is whether the introduction of such VM inference models is beneficial to the fab.

1.2.2 Scatterometry

Scatterometry, or optical metrology, is a non-destructive metrology technique that infers the parameter values of a 3D geometric profile by analyzing the changes in the intensity of light reflected from a periodic grating. [13, 14, 15] In contrast to local measurement tools such as the CD-SEM or TEM, scatterometry reports an average measurement across the grating. With in-line integration, scatterometry is a very powerful tool for manufacturing, especially with high-aspect-ratio finFET and 3D structures. [16] Although there is room for improvement, optical CD (OCD) metrology now significantly matches the CD-SEM and has shown high accuracy and precision. Often times, the high accuracy of scatterometry is paired with significant computation time, as we will see later in this section.

Scatterometry is the characterization of light diffracted from periodic structures. Figure 1.2 shows a general setup for scatterometry, with a simple periodic grating consisting of bottom anti-reflective coating (BARC) and photoresist. The scattered or diffracted light pattern, often referred to as a signature, can be used to characterize the details of the grating shape itself. For periodic gratings, the scattered light consists of distinct diffraction orders at angular locations specified by the simple grating equation, which is given below. Often times, to increase sensitivity, the light is polarized into transverse electric (TE) and transverse magnetic (TM) polarizations.

In Figure 1.2, we show the 0^{th} and 1^{th} orders only. Due to complex interactions between the incident light and the materials that constitute the grating, the intensity or phase of light diffracted into any order is sensitive to the shape and parameters of the diffracting structure. The intensity or phase information is then used to characterize that structure itself. [17] In Figure 1.2, we show an example of parameters such as underlying BARC height, photoresist

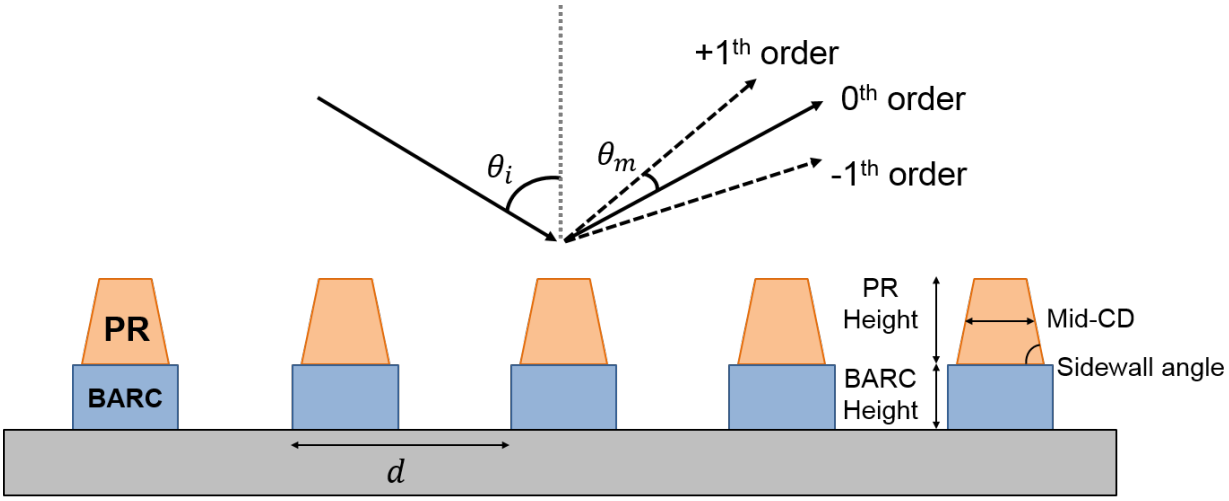


Figure 1.2: Overview of scatterometry.

height, mid-CD, and sidewall angle, where the photoresist is modeled as an infinitely long trapezoid.

The simple grating equation is specifically given by

$$n_i \sin(\theta_m) = n_i \sin(\theta_i) + m\frac{\lambda}{d}, \quad m = 0, \pm 1, \pm 2, \ldots \tag{1.4}$$

where θ_i is the angle between the incident light and the normal to the grating surface, θ_m is the angle between the m^{th} diffracted order and the normal to the grating surface, n_i is the incident medium's index of refraction, and d is the grating period. [13] In practice, usually information pertaining to only the zeroth order is collected and proves sufficient for profile reconstruction.

1.2.2.1 Normal Incidence Spectroscopic Reflectometry

In normal-incidence spectroscopic reflectometry (NISR), a broadband light source is split into a reference beam and a beam directed towards the sample. The reflectivity at various wavelengths is calculated by comparing the intensity of these two beams. [18] Defining the angle between the polarizer and grating lines to be ϕ, the complex reflectivities of TE and TM waves to be $\tilde{r}_s = r_s e^{i\delta_1}$ and $\tilde{r}_p = r_p e^{i\delta_2}$, respectively, and the phase difference to be $\Delta = \delta_1 - \delta_2$, the total reflectivity is given by [18]

$$R(\phi) = r_s^2 \cos^4\phi + r_p^2 \sin^4\phi + 2 r_s r_p \cos\Delta \cos^2\phi \sin^4\phi. \tag{1.5}$$

By varying the polarization angles, r_s, r_p, and $\cos\Delta$ can be calculated.

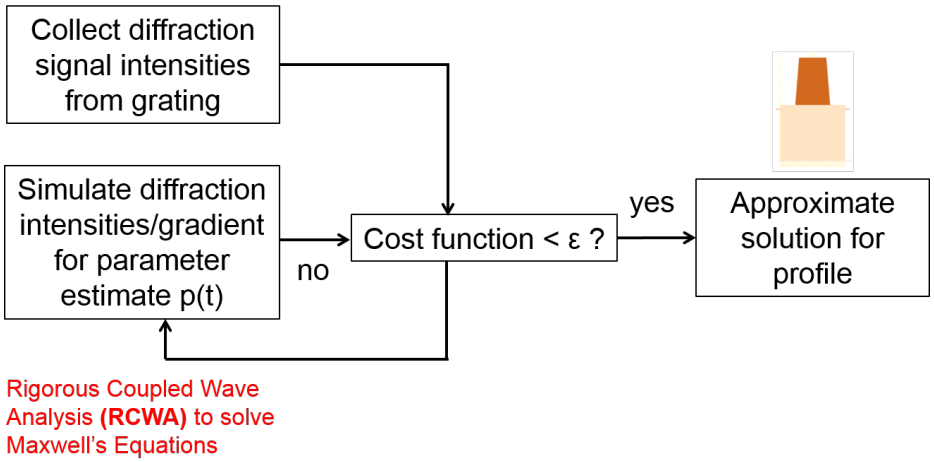


Figure 1.3: The forward and inverse problem of scatterometry.

1.2.2.2 Spectroscopic Ellipsometry

Spectroscopic ellipsometry measures the ratio of the zeroth-order complex TE and TM reflection coefficients $\rho = \tilde{r}_p/\tilde{r}_s = \tan\Psi e^{i\Delta}$, where $\tan\Psi = |\tilde{r}_p|/|\tilde{r}_s|$, and again, δ is the phase difference. [13] Since ellipsometry measures the ratio of the two coefficients, no reference beam is necessary. [19] The phase information contained in Δ provides high sensitivity to film properties, especially thickness of ultra-thin films. [20]

1.2.2.3 Variable-Angle Scatterometry

In variable-angle scatterometry, the incident beam is scanned through a series of discrete angles, and the detector of the scatterometer is able to follow and measure the orders as the incident angle is varied. The light can also be a laser at different wavelengths. [17] Since the two angles in equation (1.4) are varied, this system is also referred to as a "2-Θ" scatterometer. [17] One of the key advantages of this system is the ability to "visualize" the grating profile at various incident angles. Although in general only intensity measurements are used, various forms of variable-angle scatterometry can incorporate polarization and also measure both magnitude and phase information like ellipsometry.

1.2.2.4 The Forward and Inverse Problem

Figure 1.3 shows the forward and inverse problems in reconstructing the grating profile. The forward problem in scatterometry concerns itself with accurately simulating a diffraction signature for a given set of profile parameters. Simple gratings are usually modeled as infinite trapezoids but more complex shapes can also be analyzed. In the case of a trapezoid, the grating profile can be explained by three parameters: CD (or linewidth), height, and sidewall angle (Figure 1.2). The inverse problem then tries to "match" as closely as possible the

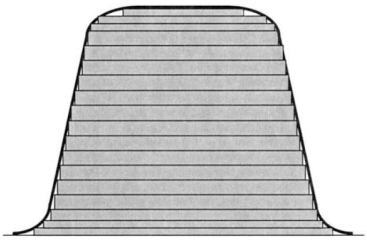


Figure 1.4: Slicing technique used to accommodate arbitrary profile shapes.

diffraction signal collected from the sample to a simulated signal. The parameters that result in the closest match are used as measurements.

The most popular method for simulating grating diffraction signatures is rigorous coupled-wave analysis (RCWA). It is a mathematical mechanism that allows for the direct solution of the electromagnetic fields diffracted by a grating. The profile is sliced into many thin rectangular layers (Figure 1.4). By utilizing Maxwell's equations and applying boundary conditions at all interfaces of the rectangles, RCWA calculates the field strengths in all regions of the grating. The flexibility of this method makes modeling more complex grating shapes feasible. [17]

The inverse problem can be solved in two ways. One is a library search, where hundreds of thousands or more diffraction signatures are simulated for a very fine grid of parameters. The library signature that is closest to that of the sample is then identified as the measurement. The library construction often takes hours or even days, and may range from hundreds of thousands of simulations or more. The signatures are compared through a cost function, which usually is a squared sum. The profile parameter set that results in the minimum cost is used as the measurement.

Another approach is an iterative optimization algorithm to search for a non-linear least squares minima. One example is the well-known Levenberg-Marquardt (LM) algorithm. Assuming there are n optical response (usually intensity or phase) points and k parameters for the profile, we denote the following.

- y_i: optical response of the sample diffraction signal for the i^{th} point.
- x_i: measurement condition, e.g. values of wavelength or angle, for the i^{th} point.
- $\boldsymbol{p} = [p_1, \ldots, p_k]^T$: profile parameter vector containing variables such as linewidth (CD), height, sidewall angle, etc.
- $f(x_i; \boldsymbol{p})$: simulated RCWA optical response for condition x_i and parameter values $\boldsymbol{p}$ for the i^{th} point.

The collected signature is modeled as the simulated signature plus noise

$$y_i = f(x_i; \boldsymbol{p}) + \epsilon_i, \;\; \epsilon_i \sim N(0, \sigma^2).$$

The function $f(\cdot)$ is highly non-linear and we do not know the explicit form of it. Thus, the goal of LM is to find the set of optimal profile parameters $\boldsymbol{p}^*$ that results in the minimum of the cost function

$$C(\boldsymbol{p}) = ||\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})||^2. \tag{1.6}$$

Since the analytical form of $f(\cdot)$ is not known, the function at an updated point $\boldsymbol{p}^{t+1} = \boldsymbol{p}^t + \boldsymbol{s}$ is approximated by the Taylor expansion. For one point i, this results in

$$f(x_i; \boldsymbol{p}^t + \boldsymbol{s}) \approx f(x_i; \boldsymbol{p}^t) + \boldsymbol{J}_i \boldsymbol{s} \tag{1.7}$$

where

$$\boldsymbol{J}_i = \left[\frac{\partial f(x_i; \boldsymbol{p}^t)}{\partial p_1}, \cdots, \frac{\partial f(x_i; \boldsymbol{p}^t)}{\partial p_k} \right].$$

Now the cost function at the updated location becomes

$$C(\boldsymbol{p}^t + \boldsymbol{s}) \approx ||\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p}^t) - \boldsymbol{J}\boldsymbol{s}||^2. \tag{1.8}$$

Notice that finding the optimal $\boldsymbol{s}^*$ that minimizes the approximated $C(\boldsymbol{p}^t + \boldsymbol{s})$ is a linear least squares problem. Thus, the best estimate for $\boldsymbol{s}^*$ is

$$\boldsymbol{s}^* = (\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{J}^T (\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p}^t)) \tag{1.9}$$

where $\boldsymbol{J} = [\boldsymbol{J}_1 \cdots \boldsymbol{J}_n]^T$. Due to some additional issues,

$$\boldsymbol{s}^* = (\boldsymbol{J}^T \boldsymbol{J} + \lambda \text{diag}(\boldsymbol{J}^T \boldsymbol{J}))^{-1} \boldsymbol{J}^T (\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p}^t))$$

is used in practice. The LM algorithm finally updates $\boldsymbol{p}^{t+1} = \boldsymbol{p}^t + \boldsymbol{s}^*$ for each iteration until convergence is reached or the algorithm goes out of bounds.

Both methods have their pros and cons. While constructing a library may take an exhaustive amount of time beforehand, iterative methods may take a significant amount of effort to converge during the measurement. In addition, as seen in by the LM algorithm, most methods require precise calculations of the gradient, and in some cases, even the second-order Hessian matrix, making them computationally intensive as well. However, iterative methods do not choose an optimal set from a predetermined grid of values. Lastly, while many solver-based algorithms achieve local minima, library based methods may achieve values close to global minima since the search is exhaustive over all possible parameter sets.

1.2.3 Holistic or Hybrid Metrology

Recently, there has been significant interest in so-called hybrid or holistic metrology, the practice of combining measurements from multiple sources in order to improve the measurement of one or more critical parameters. [21, 22] Hybrid metrology allows us to combine the strong points of each metrology tool, increasing both the accuracy and precision of CD measurements. Vaid *et al.* [21] define hybrid metrology to be the combined use of two or

more metrology tools to measure aspects of the same dataset. The central idea is that data obtained from these tools is used in a complementary or synergistic way to enhance the resolving power of either or both tools. Similarly, Zhang *et al.* [22] have used a Bayesian statistical approach to combine measurement information from different metrology sources directly in the parametric fitting process, improving measurement uncertainty.

Although there are multiple ways to mix measurements, we briefly go over an example of combining scatterometry measurements, specifically from [22]. From Section 1.2.2.4, the optimal parameter set $\boldsymbol{s}^*$ was found by (1.9). For demonstration purposes, we only deal with one profile parameter, say the CD. In Bayesian linear regression, we go one step further from (1.9) and put a *prior* distribution over s. That is, we propagate some belief we have about s through (1.9) from previous measurements.

In hybrid metrology, prior information is known about a measurement site of interest, usually through past off-line metrology measurements. For example, this could be a set of SEM measurements across a previously processed wafer. The *prior* distribution of p (the CD) can be modeled as a Gaussian variable from the distribution $N(\mu_0, \sigma_0^2)$, where μ_0 is the mean and σ_0^2 is the variance of the off-line measurements. Since at each iteration, $s = p^{t+1} - p^t$, s now has a prior of $N(\mu_0 - p^t, \sigma_0^2)$ for a fixed p^t value at the start of iteration $t+1$. Mixing this information with the OCD diffraction signatures, the *posterior* distribution of s, $P(s|\boldsymbol{J}, \boldsymbol{y})$, is Gaussian distributed with

$$\begin{aligned} E(s|\boldsymbol{J}, \boldsymbol{y}) &= \left(\frac{\mu_0 - p^t}{\sigma_0^2} + \frac{(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}; p^t))}{\sigma^2(\boldsymbol{J}^T\boldsymbol{J})^{-1}} \right) \left(\frac{1}{\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2(\boldsymbol{J}^T\boldsymbol{J})^{-1}}} \right) \\ Var(s|\boldsymbol{J}, \boldsymbol{y}) &= \left(\frac{1}{\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2(\boldsymbol{J}^T\boldsymbol{J})^{-1}}} \right). \end{aligned} \tag{1.10}$$

After convergence, the mean is usually taken as the estimate for s. The posterior distribution is a mix of the prior and the OLS estimate, as seen by the terms in the mean and variance. Notice that the posterior estimate of s has a smaller variance than the variance of the prior distribution or the OLS estimate, improving the precision of the measurement.

1.3 Motivation and Organization of Thesis

Historically, metrology started with measuring a characteristic in question against a reference, such as a ruler. In semiconductor processing, hardware-intensive and highly intricate off-line microscope tools, such as the SEM, have been considered the workhorse for metrology. As technology nodes continue to shrink and 3D structures emerge, it will be a challenge for microscope tools to resolve and "see" next generation devices. Thus, a significant amount of current resources are focused on developing computational metrology, such as scatterometry, that provide full profile information and W2W process monitoring through physical modeling and optimization techniques.

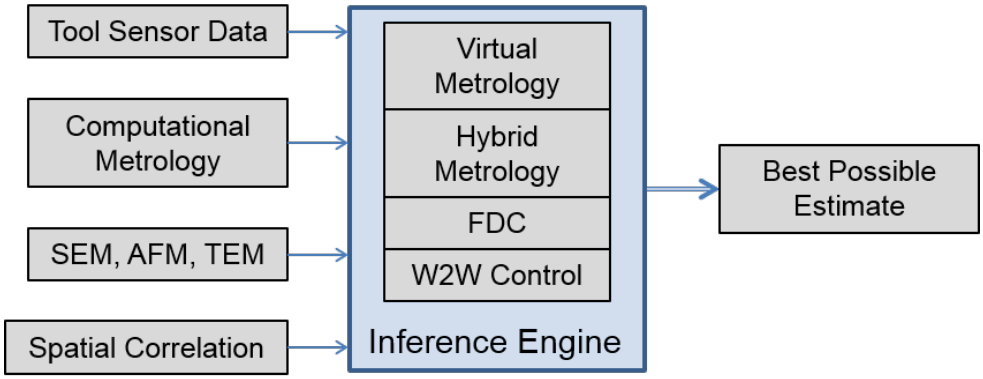


Figure 1.5: Inference engine showing various forms of information, such as tool sensor data, optical-CD measurements, off-line microscope measurements, and spatial/temporal correlation, being integrated for the best possible estimate of a wafer. The inference model can be used for functions such as in-line metrology, fault detection, or wafer-to-wafer control.

Currently, many process control models are based on explicit modeling of the process or electromagnetic phenomena. However, this level of modeling will now be difficult as products go through hundreds of complex manufacturing steps and the time for model development is limited. Although physical modeling will always be a priority, the industry must also focus on complementing this with empirical inference models that have faster development time and more flexibility in terms of integrating different forms of data. Moreover, most facilities already have access to a huge pool of information from tool and optical sensors that may help significantly in building such data models.

Figure 1.5 shows an inference engine. Very much like hybrid metrology, we would like to combine or hybridize various forms of information, such as sensor data, OCD measurements, various off-line measurements, and spatial/temporal correlations to extract the best estimate of a wafer state. This information would be utilized for in-line metrology, hybrid metrology, fault detection and classification, and W2W control. In this dissertation, we explore the application and flexibility of inference engines in the semiconductor processing arena in the context of W2W fault detection, since in many situations a method of quickly determining whether the wafer is in-spec or out-of-spec is essential.

We start by exploring virtual metrology, which is in essence an inference model with tool signals and off-line measurements as inputs. In reality, VM models are constructed, or recalibrated, based on more accurate off-line metrology data. In a fault detection context, we try to answer whether the introduction of such inference models actually improves the overall performance of the fabrication sequence, and if so, analyzing the conditions under which such a blended metrology scheme would be advantageous and optimized.

Specifically, the first part of the thesis, Chapter 2, illustrates possible cost scenarios when using virtual metrology for fault detection, taking into account recalibration period length and the type of VM model. We start by estimating cost parameters and delve into the

effect of using VM for fault detection and its impact on total profit. For demonstration purposes, we utilize three types of virtual metrology models, and two possible metrology sampling scenarios. Simulation results allow us to explore when virtual metrology may or may not be useful depending on the tradeoff between false and missed alarm rates. This demonstration is not only for traditional virtual metrology models, but can also be extended to other inference models that utilize any kind spatial or temporal information.

Since ultimately, the total cost is a function of false and missed alarm rates, we would like to utilize a fault detection and classification model that provides flexibility in tuning these two types of errors. In the next part of the thesis, the support vector machine (SVM) classifier, which is another type of inference engine, is presented as the main FDC tool. We not only demonstrate the speed and accuracy of SVMs with scatterometry grating classification, but also demonstrate an algorithm that quickly chooses the appropriate parameters for tuning false and missed alarm rates as needed.

Specifically, in Chapter 3, we construct a real-time inspection tool using SVM classifiers that will allow us to detect erroneous or out-of-spec grating CD's much faster and potentially more accurate than scatterometry measurements. In contrast to Chapter 2, estimation of the CD value of the grating is skipped entirely, and only the diffraction signatures of the scatterometry tool are used for the full purpose of fault detection. Finally, in Chapter 4, we develop an algorithm that will allow us to quickly determine the optimal parameter set for a classifier depending on which false and missed alarm rate combination minimizes the loss function. Moreover, this framework can also be used to choose appropriate scaling factors that further improve the accuracy of the model.

Chapter 2

Effect of Periodic Recalibration on Virtual Metrology

2.1 Introduction

Past research on virtual metrology has been heavily focused on constructing empirical models for process modeling. Various authors have developed VM models for wafer etch rate using high dimensional sensor data. [23, 24, 25] Su *et al.* and Hung *et al.* developed VM models for CVD film thicknesses using neural networks. [26, 27] Moreover, some have implemented an APC model using VM measurements as inputs [6, 7]. The performance of the VM model is usually measured against conventional metrology data by criteria such as the error variance or the mean squared error (MSE).

Although VM has its advantages, VM model prediction quality is not as good as that of conventional metrology, and the models need frequent recalibration in order to maintain acceptable predictive capability. In real life, we envision that practical metrology schemes will involve VM in combination with conventional or actual metrology, the latter being used for the needed periodic recalibration of the VM empirical model. Such a scenario is shown in Fig. 2.1. The initial training step is used to construct the VM model; the model is then applied to incoming wafers and later recalibrated to incorporate process drift and other faults. We also envision an algorithm that would respond to a fault being predicted by the VM model by possibly requiring additional actual measurements.

In this chapter, we explore whether introducing in-line metrology models for fault detection is advantageous for the manufacturing sequence and propose a general framework that can be used to quickly lead to the optimal design of such schemes given the characteristics of the process in question. This is done by exploring the effects of variables such as the frequency of samples that go through actual metrology, the prediction quality of the VM model, the cost of missed or false alarms, processing and actual metrology costs etc.

In our earlier work [28], we demonstrated that it is more beneficial to see VM in context of a faulty process than a well-controlled process. In fact, a well-controlled process would need

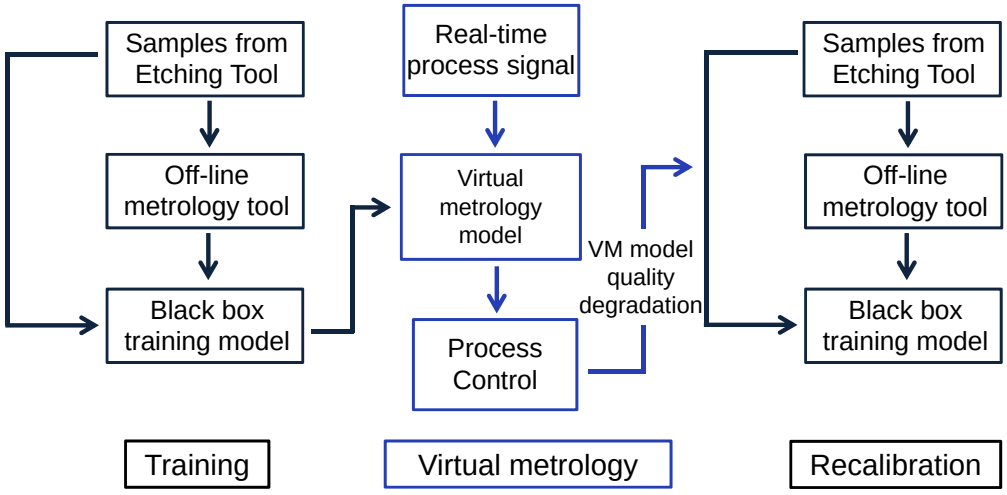


Figure 2.1: A blended metrology scenario: the virtual metrology model is constructed through the initial training step, then applied to the process, and recalibrated later to incorporate process drift.

minimal metrology. Thus, we simulate a drifting process with low C_{pk} and calculate the total profit for three different but realistic VM models: Linear Regression, Exponentially-weighted Linear Regression (EWLR), and the Kalman Filter. This is done for two possible blended metrology sampling scenarios, one where flagged wafers are automatically discarded, and another where they are re-inspected and the process is re-tuned. From here on, our analysis assumes conventional metrology to be the SEM and the unknown quantities to be the critical dimension (CD).

2.2 Costs Associated with Imperfect Detection

In any metrology operation, whether conventional, virtual or blended, the value obtained is an estimate of the true value that is being measured. For either conventional or virtual metrology, let us assume that the true value of the quantity in question (unknown to the process engineer), is y, and the value estimated by the metrology operation is $\hat{y}$. We assume there are three types of cost associated with each sample wafer going through a metrology scheme:

- Revenue per wafer: If the true value of a wafer sample is in-spec and the wafer goes through the processing line, it ultimately becomes an integrated circuit product (e.g. microprocessor). The fab then generates revenue from selling this product. This is given by

$$\frac{\text{revenue}}{\text{wafer}} = \frac{\text{revenue}}{\text{die}} \times \frac{\text{number of dies}}{\text{wafer}} \times \text{yield}. \tag{2.1}$$

- Processing cost per wafer: Cost required to further process one wafer until it becomes a final product. This is given by

$$\frac{\text{process cost}}{\text{wafer}} = \frac{\text{process cost}}{\text{die}} \times \frac{\text{number of die}}{\text{wafer}}. \tag{2.2}$$

- SEM cost for w wafers: Cost of using the SEM for processing w wafers. This includes depreciation cost and employment of the operation engineer. This is given by

$$\text{SEM cost for } w \text{ wafers} = \frac{\text{SEM tool cost}}{\text{years used}} \times \frac{w \text{ wafers}}{\text{wafers started per month} \times 12}. \tag{2.3}$$

Here we assume with no loss of generality that the entire wafer is accepted or rejected by the metrology operation. In addition, we assume for simplicity that there is no cost of operating a virtual metrology software tool. Thus, we can define a total profit for each wafer as

$$\text{total profit} = \text{revenue} - \text{process cost} - \text{SEM cost}. \tag{2.4}$$

The financial parameters mentioned above vary widely depending on the type of chip (e.g. microprocessor or memory). As we are proposing a methodology for blended metrology optimization, we chose to demonstrate it using approximate but realistic values for current microprocessor manufacturing processes. We list the numbers used in equations (2.1)-(2.3) in Table 2.1.

Table 2.1: Values for parameters in (2.1)-(2.3). We demonstrate using approximate but realistic values.

Parameter	**Value**
Wafers started per month	40,000
Die yield / wafer	85%
# dies per wafer	280
Years SEM used	4
SEM tool cost	$1.6 million
Revenue / die	$120
Process cost / die	$16

Due to imperfect estimation, fault detection models may produce false alarms (Type I error) or missed alarms (Type II error). Those two types of errors are associated with operational costs that depend on the metrology scheme and the algorithm used in response to an alarm. We determine that a sample is "faulty" or "bad" if a metrology value is under or over the respective specification limits. Thus, if the estimated metrology value is bad, the

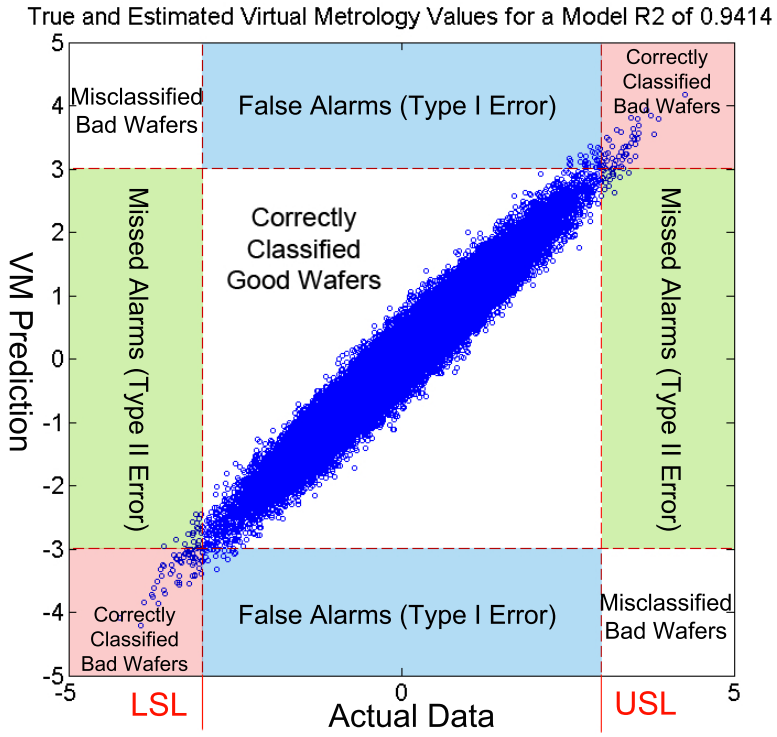


Figure 2.2: Plot of true and estimated metrology values for an R^2 of 0.94 for a virtual metrology model. The dotted lines are the $\pm 3\sigma$ limits. The false and missed alarm error regions are highlighted and labeled.

process engineer will have to throw away (or re-work) the sample, and if the estimated value is good, the wafer continues onto subsequent manufacturing processes. Overall, there are four cases that can happen when a process engineer uses either of the metrology schemes. Depending on which category a wafer is in, it incurs a different set of costs.

- Correctly classified as good: The metrology tool (either conventional or virtual) classifies the sample as good when the true value is good. The wafer goes through subsequent processes and is turned into final product, generating revenue for the company. Given some upper specification limit (USL) and lower specification limit (LSL), this directly translates to

$$P(\hat{y}_{vm} = \text{good} \mid y = \text{good}) \tag{2.5}$$
$$= P(\text{USL} \geq \hat{y}_{vm} \geq \text{LSL} \mid \text{USL} \geq y \geq \text{LSL}). \tag{2.6}$$

- Correctly classified as bad: The metrology tool classifies the sample as bad when the true metrology value is bad, and the wafer is discarded. The correct metrology

estimate saves the company cost that could have incurred if the wafer was processed subsequently, as the final product would have been defective due to the faulty true value.

$$P(\hat{y}_{vm} = \text{bad} \mid y = \text{bad}) \tag{2.7}$$
$$= P(\hat{y}_{vm} > \text{USL} \parallel \hat{y}_{vm} < \text{LSL} \mid y > \text{USL} \parallel y < \text{LSL}). \tag{2.8}$$

- Type I error: Also known as a "false alarm", the metrology tool classifies the sample as bad when actually the true value is good (in-spec). The proportion of samples with Type I error is given by:

$$P(\hat{y}_{vm} = \text{bad} \mid y = \text{good}) \tag{2.9}$$
$$= P(\hat{y}_{vm} > \text{USL} \parallel \hat{y}_{vm} < \text{LSL} \mid \text{USL} \geq y \geq \text{LSL}). \tag{2.10}$$

  In this case, the wafer is discarded even though it could have been processed to become a final product.

- Type II error: Also known as a "missed alarm", the metrology tool classifies the sample as good when actually the true value is bad. Similarly, the proportion of samples with Type II error is given by:

$$P(\hat{y}_{vm} = \text{good} \mid y = \text{bad}) \tag{2.11}$$
$$= P(\text{USL} \geq \hat{y}_{vm} \geq \text{LSL} \mid y > \text{USL} \parallel y < \text{LSL}). \tag{2.12}$$

As a visual example, we plot simulated pairs of true and estimated VM values with correlation coefficient ρ of 0.96, as seen in Fig. 2.2. Notice that this is a very high correlation for a statistical model. The corresponding false and missed alarm error regions discussed above are highlighted and labeled.

Table 2.2: Different kinds of costs that occur for a metrology scheme.

	Revenue	**Process Cost**	**Metrology Cost**
Both Good	Yes	Yes	Yes
Both Bad	No	No	Yes
False Alarm	No	No	Yes
Missed Alarm	No	Yes	Yes

A missed alarm generally incurs the greatest cost because even though the wafer goes through the whole manufacturing process, it is not made into a final product at the end due to its defect. Also note that for W2W process control, all wafers generate metrology cost. The different costs that occur for each classification category are summarized in Table 2.2.

2.3 Sampling and Process Scenarios for Blended Metrology

One question that remains is how to devise a metrology sampling scenario. Traditionally, 2-3 wafers are sampled out of a lot and are observed through the SEM. We devise two sampling scenarios, both applied to a process with low C_{pk}. The first is a short, non-interrupted process of 3,000 wafers (or 120 lots) in which flagged wafers by VM or SEM are automatically discarded. The second is a long-term process of 10,000 wafers (or 400 lots) that allows for a SEM re-inspection whenever VM flags a wafer as faulty. Moreover, this scenario allows for process re-tuning. Since the objective of this work is to determine when virtual metrology is beneficial, we compare the total profit for the blended case and conventional case (no VM used) for both scenarios. We define some terms before we delve into details.

1. Initial Training Length (r_{it}): Number of wafers used to initially train the VM model.
2. Length of VM Run (r_{vm}): Number of wafers "measured" only through VM before additional SEM measurements are collected for recalibration.
3. Recalibration Length (r_{rc}): Number of wafers that need to be measured by the SEM for each recalibration step.
4. Metrology Delay Length (r_{md}): Number of wafers processed while waiting for SEM measurements to become available. These wafers are considered to be at-risk of being processed under faulty process conditions.
5. Retune Delay Length (r_{rd}): Number of wafers that could have been processed but missed because the process is being re-tuned back to target.

2.3.1 Scenario without Re-inspection and Re-tuning

The non-interrupted process of 120 lots consists of initially training the VM model with real-time processing data and SEM metrology data (given by r_{it}). We assume that whenever a wafer is sent to the SEM metrology station, there is a metrology delay of 2 lots (given by r_{md}), or 50 wafers, that are unconditionally processed while waiting for the metrology results. When the new metrology data is available, the VM model is updated using this new data and applied to the next number of wafers. When the SEM or VM model flags a wafer as being faulty, it is discarded. Finally, a number of wafers equal to r_{rc} are sent to the SEM station to update the model again. This whole process is repeated throughout the simulation, as seen in Fig. 2.3. As usual, $\boldsymbol{X_i}$ represents a $n \times p$ matrix of process conditions, where n is the number of wafers and p is the dimension of sensor data, and $\boldsymbol{y_i}$ is a $n \times 1$ vector that represents actual metrology values or VM predictions depending on the step. The conventional metrology scheme for this scenario follows the same SEM metrology

sampling frequency as that of blended metrology but no VM is applied and all other wafers unconditionally go through subsequent processing.

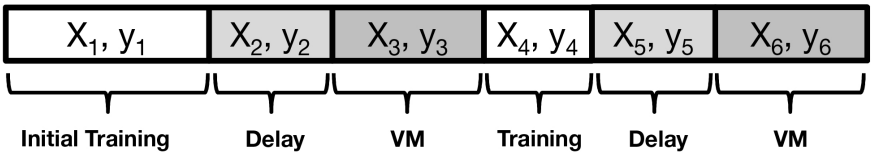


Figure 2.3: Sampling scenario without re-inspection. SEM metrology is initially used to construct the virtual metrology model. Following a metrology delay period, the virtual metrology model is used for in-line fault detection before being recalibrated again. The figure above shows the process condition matrices ($\boldsymbol{X_i}$) and CD values/estimates ($\boldsymbol{y_i}$) for each phase.

2.3.2 Scenario with Re-inspection and Re-tuning

The extended process of 400 lots similarly trains the first r_{it} number of samples to construct the first VM model and applies it to the next r_{vm} number of wafers. However, we account for the limited prediction quality of VM by inspecting all flagged wafers. That is, when VM estimates a wafer to be faulty, the sample is sent to the SEM and inspected one more time with the risk of delaying r_{md} wafers. If the wafer is determined to be in-spec after the SEM inspection (i.e. VM has generated a false alarm), the process goes back to being monitored by VM; if the SEM confirms the wafer is bad, the process is halted and re-tuned back to target, delaying Re-tune Delay Length (r_{rd}) amount of wafers to be processed. After either finishing one VM run or re-tuning the process, r_{rc} number of wafers is collected from the SEM to update the VM model. If a wafer is flagged during this phase, the process is re-tuned again (there is no need to double-check as we assume SEM is 100% accurate). An overview of this scenario is shown in Fig. 2.4. For the conventional case, process re-tuning only occurs when SEM flags a wafer and all other wafers are processed unconditionally.

Fig. 2.5 shows a cartoon of what would happen to the total fab profit vs. r_{vm} in a blended metrology scheme. With short VM runs, there would be frequent updates of the VM model but also more frequent SEM metrology delays, leading to a lot of at-risk wafers. At the upper r_{vm}, the delay would not impact as much, but since the VM model is not updated enough, an increase in false and missed alarm rates would bring down the profit. In the middle would be a maximum profit optimized to the fab's characteristics.

2.3.3 Drifting Regression Model

As mentioned in Section 2.1, we propose that a well-controlled process setting is not a compelling application for virtual metrology. In addition, we want a dynamic yet realistic process that will allow us to explore the dependence between VM model accuracy and r_{vm}.

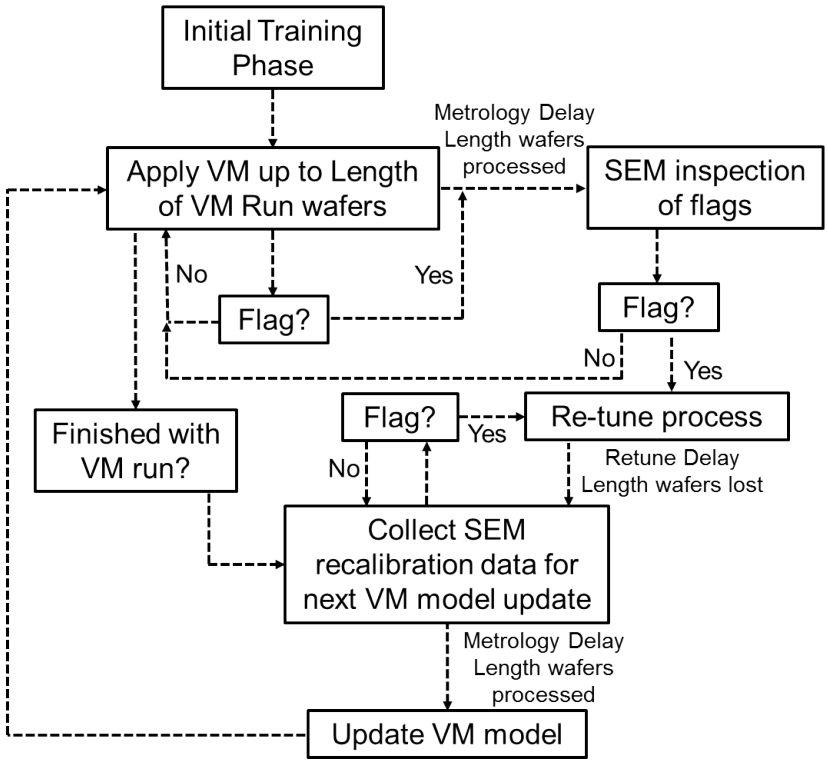


Figure 2.4: Extended sampling scenario with interrupted flag inspection. After initial training and deployment of the virtual metrology model, the samples that are flagged by virtual metrology as bad are sent to the SEM for confirmation. If so, the process is re-tuned to control with some delay.

Such a process is a regression model with drifting coefficients. Consider a general linear process

$$y_t = \beta_t^0 + \beta_t^1 x_t^1 + \beta_t^2 x_t^2 + \ldots + \beta_t^{hid} x_t^{hid} + \epsilon_t \tag{2.13}$$

where subscript t corresponds to time, the numerical superscripts correspond to the dimension of the sensor data, and ϵ_t is the observation error at time t. It is possible that the VM model misses estimating an independent variable x_t^{hid} and its coefficient β_t^{hid}, and suppose this hidden variable drifts over time by the amount Δx^{hid}. At the next time step, if all other variables are constant, the process is now explained by

$$\begin{aligned} y_{t+1} &= \beta_t^0 + \beta_t^1 x_t^1 + \beta_t^2 x_t^2 + \ldots + \beta_t^{hid}(x_t^{hid} + \Delta x^{hid}) + \epsilon_t \\ &= \beta_t^0 + (\beta_t^1 + \Delta\beta) x_t^1 + \beta_t^2 x_t^2 + \ldots + \beta_t^{hid} x_t^{hid} + \epsilon_t. \end{aligned} \tag{2.14}$$

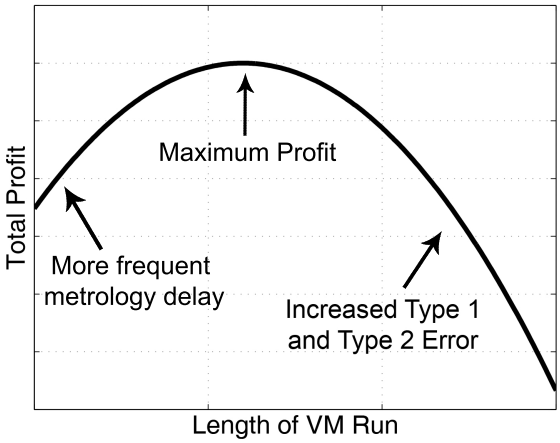


Figure 2.5: Optimization of blended metrology schemes as a function of r_{vm}. When r_{vm} is small, there is frequent recalibration and more metrology delay. When r_{vm} is large, there is a risk for increased false and missed alarm rates due to process drifts and less recalibration of the model.

Now the drift Δx^{hid} is captured in β_t^1 with $\Delta\beta x_t^1 = \beta_t^{hid}\Delta x^{hid}$ as a linearly drifting coefficient. Thus, the final model of the true process would be

$$\begin{aligned} y_t &= \boldsymbol{x}_t^T\boldsymbol{\beta}_t + \epsilon_t, \qquad \epsilon_t \sim N(0,\sigma_\epsilon^2) \\ \boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t + \boldsymbol{u} + \boldsymbol{\eta}_t, \qquad \boldsymbol{\eta}_t \sim N(0,\boldsymbol{Q}_t) \end{aligned} \tag{2.15}$$

where y_t is the true CD, $\boldsymbol{x}_t$ are sensor values for one sample at time t, and ϵ_t, $\boldsymbol{\eta}_t$ are white noise variables. Finally, $\boldsymbol{u}$ controls how much the coefficient drifts over time.

2.4 Virtual Metrology Models

In this section, we provide a brief theoretical background on the three VM models that are used in the analysis: Linear Regression, Exponentially-weighted Least Squares (EWLS), and the Kalman Filter (KF). To clarify how they are used in our work, we explain the background in context of the sampling scenario shown in Fig. 2.3.

2.4.1 Linear Regression

Assuming there is a collection of n_{SEM} sensor and CD data from the SEM, linear regression assumes the true process is of the form (note there is no incorporation of the drift in the coefficient vector $\boldsymbol{\beta}$ that is present in the true process model)

$$\boldsymbol{y}_{SEM} = \boldsymbol{X}_{SEM}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad \epsilon_i \sim N(0,\sigma_\epsilon^2) \quad i = 1,2,\ldots,n_{SEM} \tag{2.16}$$

and the well-known estimate of the coefficients is

$$\hat{\boldsymbol{\beta}}_{SEM} = (\boldsymbol{X}_{SEM}^T \boldsymbol{X}_{SEM})^{-1} \boldsymbol{X}_{SEM}^T \boldsymbol{y}_{SEM}. \tag{2.17}$$

In the context of this work,

$$\boldsymbol{X}_{SEM} = \begin{pmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_4} \\ \vdots \end{pmatrix}, \quad \boldsymbol{y}_{SEM} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_4 \\ \vdots \end{pmatrix}$$

where $\boldsymbol{X_i}$ and $\boldsymbol{y_i}$, $i = 1, 4, 7...$ refer to Fig. 2.3 and denote the training data from the SEM. That is, at a certain point in time, we accumulate all data from the SEM that is available for modeling not knowing the true process is faulty. Given our VM model, the CD prediction for subsequent VM samples is

$$\hat{\boldsymbol{y}}_{VM} = \boldsymbol{x}_{VM}^T \hat{\boldsymbol{\beta}}_{SEM} \tag{2.18}$$

$$= \boldsymbol{x}_{VM}^T (\boldsymbol{X}_{SEM}^T \boldsymbol{X}_{SEM})^{-1} \boldsymbol{X}_{SEM}^T \boldsymbol{y}_{SEM} \tag{2.19}$$

where $\boldsymbol{x}_{VM}$ is an incoming wafer's sensor data vector.

2.4.2 Exponentially-weighted Linear Regression

For this model, we assume that the engineer knows there is some drift in the process and wants to weigh the recent observations more heavily than the earlier ones. The weights are given by

$$w_t = s(1-s)^{(n_{SEM}-t)} \tag{2.20}$$

where s is a tunable parameter. The best value for s was found by training the model on different s values and finding the one that minimized the MSE. With $\boldsymbol{W} = \text{diag}(\boldsymbol{w})$, the EWLS estimates of $\boldsymbol{\beta}$ are given by

$$\hat{\boldsymbol{\beta}}_{SEM} = (\boldsymbol{X}_{SEM}^T \boldsymbol{W} \boldsymbol{X}_{SEM})^{-1} \boldsymbol{X}_{SEM}^T \boldsymbol{W} \boldsymbol{y}_{SEM} \tag{2.21}$$

and similarly, the estimated CD predictions are given by

$$\begin{aligned} \hat{\boldsymbol{y}}_{VM} &= \boldsymbol{x}_{VM}^T \hat{\boldsymbol{\beta}}_{SEM} \\ &= \boldsymbol{x}_{VM}^T (\boldsymbol{X}_{SEM}^T \boldsymbol{W} \boldsymbol{X}_{SEM})^{-1} \boldsymbol{X}_{SEM}^T \boldsymbol{W} \boldsymbol{y}_{SEM}. \end{aligned} \tag{2.22}$$

2.4.3 Kalman Filter

This section largely references and follows the notation from [29]. More details about the Kalman Filter are in the text. In our case, $y_t, \sigma_\epsilon^2, \epsilon_t$ are scalars, $\boldsymbol{\alpha}_t, \boldsymbol{z}_t, \boldsymbol{\eta}_t$ are vectors, and

$\boldsymbol{T}_t, \boldsymbol{Q}_t, \boldsymbol{R}_t$ are matrices. The Consider the general linear Gaussian state space model

$$\begin{aligned} y_t &= \boldsymbol{z}_t^T \boldsymbol{\alpha}_t + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma_\epsilon^2) \\ \boldsymbol{\alpha}_{t+1} &= \boldsymbol{T}_t \boldsymbol{\alpha}_t + \boldsymbol{R}_t \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim N(0, \boldsymbol{Q}_t) \\ \boldsymbol{\alpha}_1 &\sim N(\boldsymbol{a}_1, \boldsymbol{P}_1), \quad t = 1, \ldots, n \end{aligned} \tag{2.23}$$

where y_t are the observations (e.g. CDs), $\boldsymbol{\alpha}_t$ are hidden state variables controlling the process (e.g. process conditions and drift of the process), and ϵ_t, $\boldsymbol{\eta}_t$ are independent Gaussian noise sequences with corresponding variance and covariance matrices σ_ϵ^2 and $\boldsymbol{Q}_t$.

The objective of the filter is to obtain the conditional distribution of $\boldsymbol{\alpha}_{t+1}$ given $\boldsymbol{Y}_t$ for $t = 1, \ldots, n$ where $\boldsymbol{Y}_t = \{y_1, \ldots, y_t\}$. That is, given SEM measurements up to time t, we want to estimate the process conditions $\boldsymbol{\alpha}_{t+1}$. Since all distributions are normal, conditional distributions of subsets of variables given other subsets of variables are also normal. Now all we need is $\boldsymbol{\alpha}_{t+1}|\boldsymbol{Y}_t \sim N(\boldsymbol{a}_{t+1}, \boldsymbol{P}_{t+1})$ and then

$$\begin{aligned} \boldsymbol{a}_{t+1} &= E(\boldsymbol{\alpha}_{t+1}|\boldsymbol{Y}_t) \\ \boldsymbol{P}_{t+1} &= Cov(\boldsymbol{\alpha}_{t+1}|\boldsymbol{Y}_t). \end{aligned} \tag{2.24}$$

Since $\boldsymbol{\alpha}_{t+1} = \boldsymbol{T}_t \boldsymbol{\alpha}_t + \boldsymbol{R}_t \boldsymbol{\eta}_t$, we have

$$\begin{aligned} \boldsymbol{a}_{t+1} &= E(\boldsymbol{T}_t \boldsymbol{\alpha}_t + \boldsymbol{R}_t \boldsymbol{\eta}_t | \boldsymbol{Y}_t) \\ &= \boldsymbol{T}_t \, E(\boldsymbol{\alpha}_t | \boldsymbol{Y}_t), \\ \boldsymbol{P}_{t+1} &= Cov(\boldsymbol{T}_t \boldsymbol{\alpha}_t + \boldsymbol{R}_t \boldsymbol{\eta}_t | \boldsymbol{Y}_t) \\ &= \boldsymbol{T}_t \, Cov(\boldsymbol{\alpha}_t | \boldsymbol{Y}_t) \boldsymbol{T}_t^T + \boldsymbol{R}_t \boldsymbol{Q}_t \boldsymbol{R}_t^T \end{aligned} \tag{2.25}$$

for $t = 1, \ldots, n$. We now define v_t, the one-step forecast error of y_t given $\boldsymbol{Y}_{t-1}$.

$$\begin{aligned} v_t &= y_t - E(y_t | \boldsymbol{Y}_{t-1}) \\ &= y_t - E(\boldsymbol{z}_t^T \boldsymbol{\alpha}_t + \epsilon_t | \boldsymbol{Y}_{t-1}) \\ &= y_t - \boldsymbol{z}_t^T \boldsymbol{a}_t. \end{aligned} \tag{2.26}$$

Observing $\boldsymbol{Y}_t$ is the same as observing $\{\boldsymbol{Y}_{t-1}, v_t\}$, we see that $E(\boldsymbol{\alpha}_t|\boldsymbol{Y}_t) = E(\boldsymbol{\alpha}_t|\boldsymbol{Y}_{t-1}) + E(\boldsymbol{\alpha}_t|v_t)$. It is easy to see that $E(v_t) = 0$, and $Cov(v_t, \boldsymbol{Y}_{t-1}) = 0$. Through the lemma in multivariate normal regression

$$\begin{aligned} E(\boldsymbol{\alpha}_t|\boldsymbol{Y}_t) &= E(\boldsymbol{\alpha}_t|\boldsymbol{Y}_{t-1}, v_t) \\ &= E(\boldsymbol{\alpha}_t|\boldsymbol{Y}_{t-1}) + E(\boldsymbol{\alpha}_t|v_t) \\ &= E(\boldsymbol{\alpha}_t|\boldsymbol{Y}_{t-1}) + Cov(\boldsymbol{\alpha}_t, v_t)[Var(v_t)]^{-1} v_t \\ &= \boldsymbol{a}_t + \boldsymbol{M}_t \boldsymbol{F}_t^{-1} v_t, \end{aligned} \tag{2.27}$$

where $\boldsymbol{M}_t = Cov(\boldsymbol{\alpha}_t, v_t) = \boldsymbol{P}_t \boldsymbol{z}_t$, and $g_t = Var(v_t) = \boldsymbol{z}_t^T \boldsymbol{P}_t \boldsymbol{z}_t + \sigma_\epsilon^2$, when $\boldsymbol{Y}_{t-1}$ and v_t are uncorrelated. Substituting everything into expression for $\boldsymbol{a}_{t+1}$ and $\boldsymbol{P}_{t+1}$, we get

$$\begin{aligned} \boldsymbol{a}_{t+1} &= \boldsymbol{T}_t \boldsymbol{a}_t + \boldsymbol{T}_t \boldsymbol{M}_t g_t^{-1} v_t \\ &= \boldsymbol{T}_t \boldsymbol{a}_t + \boldsymbol{K}_t v_t \\ \text{where } \boldsymbol{K}_t &= \boldsymbol{T}_t \boldsymbol{M}_t g_t^{-1} = \boldsymbol{T}_t \boldsymbol{P}_t \boldsymbol{z}_t g_t^{-1}. \end{aligned} \tag{2.28}$$

Through similar analysis,

$$\begin{aligned} \boldsymbol{P}_{t+1} &= \boldsymbol{T}_t \boldsymbol{P}_t \boldsymbol{L}_t^T + \boldsymbol{R}_t \boldsymbol{Q}_t \boldsymbol{R}_t^T \\ \text{where } \boldsymbol{L}_t &= \boldsymbol{T}_t - \boldsymbol{K}_t \boldsymbol{z}_t^T. \end{aligned} \tag{2.29}$$

All the filtering recursion equations are given by

$$\begin{aligned} v_t = y_t - \boldsymbol{z}_t^T \boldsymbol{a}_t, \quad & \boldsymbol{g}_t = \boldsymbol{z}_t^T \boldsymbol{P}_t \boldsymbol{z}_t + \sigma_\epsilon^2, \\ \boldsymbol{K}_t = \boldsymbol{T}_t \boldsymbol{P}_t \boldsymbol{z}_t g_t^{-1}, \quad & \boldsymbol{L}_t = \boldsymbol{T}_t - \boldsymbol{K}_t \boldsymbol{z}_t^T, \\ \boldsymbol{a}_{t+1} = \boldsymbol{T}_t \boldsymbol{a}_t + \boldsymbol{K}_t v_t, \quad & \boldsymbol{P}_{t+1} = \boldsymbol{T}_t \boldsymbol{P}_t \boldsymbol{L}_t^T + \boldsymbol{R}_t \boldsymbol{Q}_t \boldsymbol{R}_t^T. \end{aligned} \tag{2.30}$$

For missing observations and observations to be forecasted, v_t and $\boldsymbol{K}_t$ of the filter are set to zero [29], and the updates just become

$$\boldsymbol{a}_{t+1} = \boldsymbol{T}_t \boldsymbol{a}_t, \qquad \boldsymbol{P}_{t+1} = \boldsymbol{T}_t \boldsymbol{P}_t \boldsymbol{T}_t^T + \boldsymbol{R}_t \boldsymbol{Q}_t \boldsymbol{R}_t^T.$$

Given these update equations, the following $\boldsymbol{z}_t$, $\boldsymbol{T}_t$, and $\boldsymbol{\alpha}_t$ matrices

$$\boldsymbol{\alpha}_t = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\beta}_t \end{bmatrix}, \quad \boldsymbol{z}_t^T = \begin{bmatrix} 0 & \cdots & 0 & \boldsymbol{x}_t^T \end{bmatrix}, \quad \boldsymbol{T}_t = \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{I} & \boldsymbol{I} \end{bmatrix}$$

are used for the Kalman Filter to model the true process in (2.15), where $\boldsymbol{I}$ is the identity matrix with appropriate dimensions.

2.5 Results and Discussion

In this section, we simulate data for the true process (2.15) and report results for the total profit, VM model MSE, false and missed alarm rates as a function of r_{vm} for both the scenario with and without re-inspection and re-tuning.

2.5.1 Scenario without Re-inspection and Re-tuning

We generate data based on (2.15) with the following parameters

$$\boldsymbol{\beta}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \ \boldsymbol{u} = \begin{bmatrix} 0.0004 \\ 0 \\ 0.0004 \end{bmatrix}, \ \sigma_\epsilon^2 = 0.2, \ \boldsymbol{Q}_t = \mathbf{0}_{2p \times 2p}.$$

The r_{it} is set to 200 wafers, the r_{rc} to 20 wafers, and the r_{md} to 50 wafers (2 lots). 3,000 wafers are simulated and the corresponding total profit, false and missed alarm rates, and the MSE are calculated for each VM model (Fig. 2.6, 2.7, and 2.8). For repeatability, 100 simulations are repeated and the average is plotted in bold to distinguish them from the first first 20 simulations. For the Kalman Filter, we start with a randomized hidden state vector as our initial guess. Estimates of the beta coefficients for each model are also shown in Fig. 2.9 (a), (b), and (c) along with the true coefficient values in the background.

2.5.1.1 Linear Regression

One noticeable feature about Fig. 2.6 is the non-existent false alarms and the very large amount of missed alarms. This is due to the under-estimation of the regression coefficients, as seen in Fig. 2.9(a). Since all past and present observations are weighed equally in the recalibration phase, the coefficient estimates have significantly lower values than the true ones. This constraint in the gain leads to VM estimates that are mostly inside the LSL/USL limits and as a result, the model fails to catch most of the alarms that happen, leading to the significant missed alarm probabilities. Relatively constant false and missed alarm patterns result in a flat total profit curve. The maximum mean profit occurs at $\$5.874 \times 10^7$ when the r_{vm} is 145.

2.5.1.2 Exponentially-weighted Linear Regression

EWLR shows a significantly reduced VM MSE than Linear Regression (Fig. 2.7), as only the recent observations are used in the recalibration stage. This is also seen in Fig. 2.9(b), where the EWLR estimates accurately track the drift in the true coefficients. However, in contrast to Fig. 2.9(a), the estimates have more variance. This translates to an increased probability of false alarms, and as seen in the third column of Fig. 2.7, the false alarm probability is not zero anymore. We see an increase in missed alarms as the r_{vm} increases because the coefficient estimates for a certain VM stage stay constant while the process keeps on drifting. The tradeoff between false and missed alarms result in an optimal point on the total profit curve. The maximum mean profit occurs at $\$5.921 \times 10^7$ when the r_{vm} is 665.

2.5.1.3 Kalman Filter

The Kalman Filter gives us the least MSE out of all three VM models. Note that this is partially because the filter is constructed on such a state-space model like the true process in (2.15). Moreover, we have assumed that the exact value of some of the parameters are

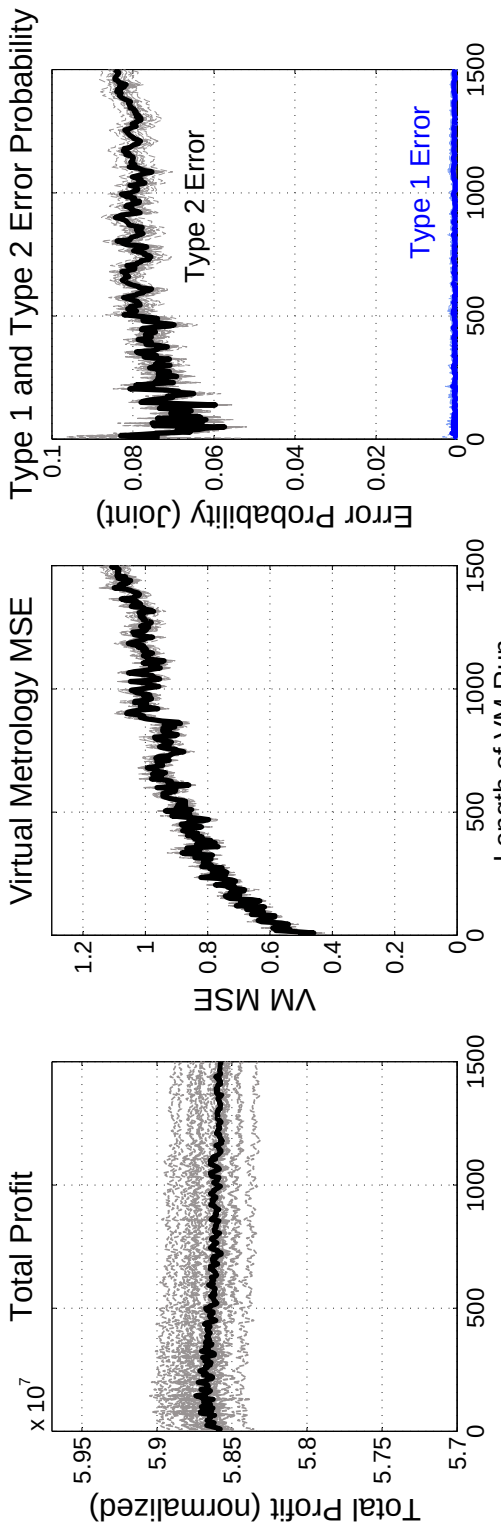


Figure 2.6: Total profit, mean-squared error, false and missed alarm probabilities for the Linear Regression virtual metrology model. The increase in estimation error is significant as the R_{vm} increases. Relatively constant false and missed alarm patterns result in a flat total profit curve.

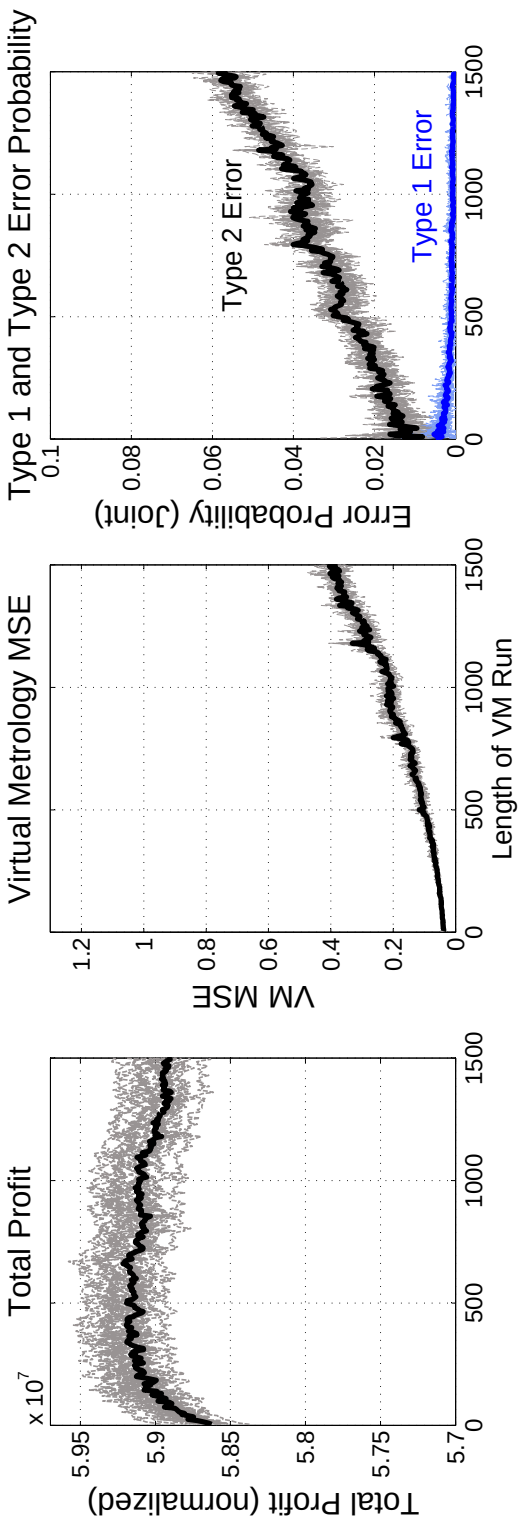


Figure 2.7: Total profit, mean-squared error, false and missed alarm probabilities for the Exponentially-weighted Linear Regression virtual metrology model. The mean-squared error and alarm probabilities have significantly improved compared to Linear Regression. Tradeoff between false and missed alarm rates result in an optimal point on the total profit curve.

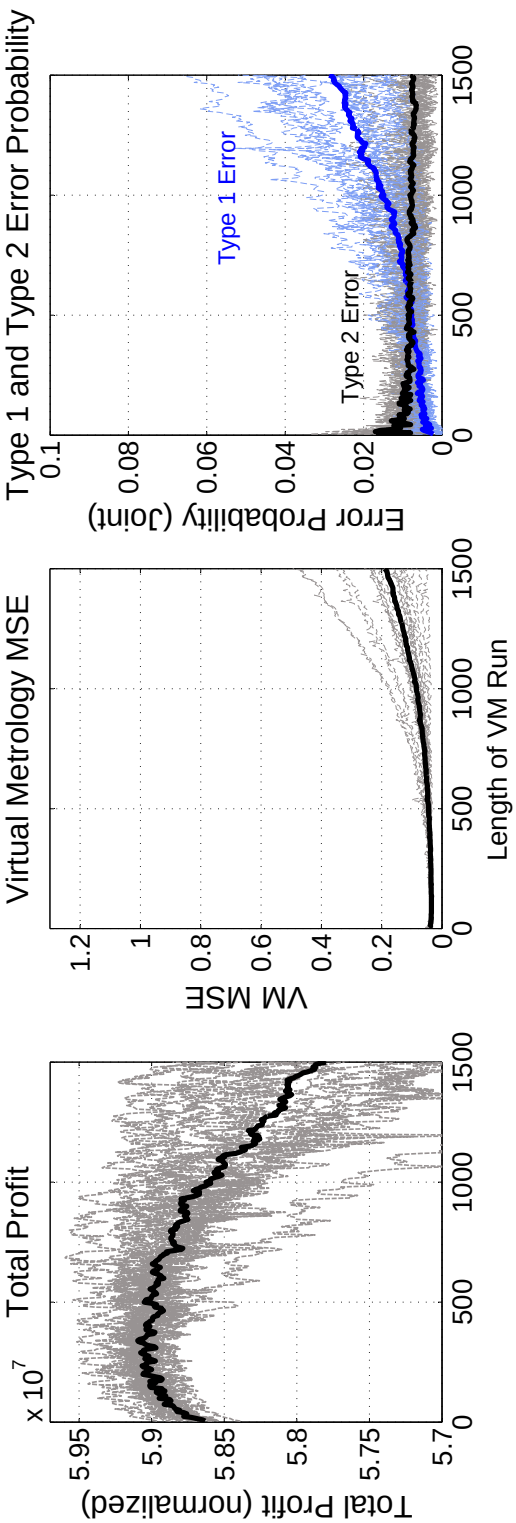


Figure 2.8: Total profit, mean-squared error, false and missed alarm probabilities for the Kalman Filter virtual metrology model. This model shows the highest accuracy and lowest error probabilities as it is a filter constructed for a state-space model. Again, the tradeoff between false and missed alarms result in an optimal point on the total profit curve.

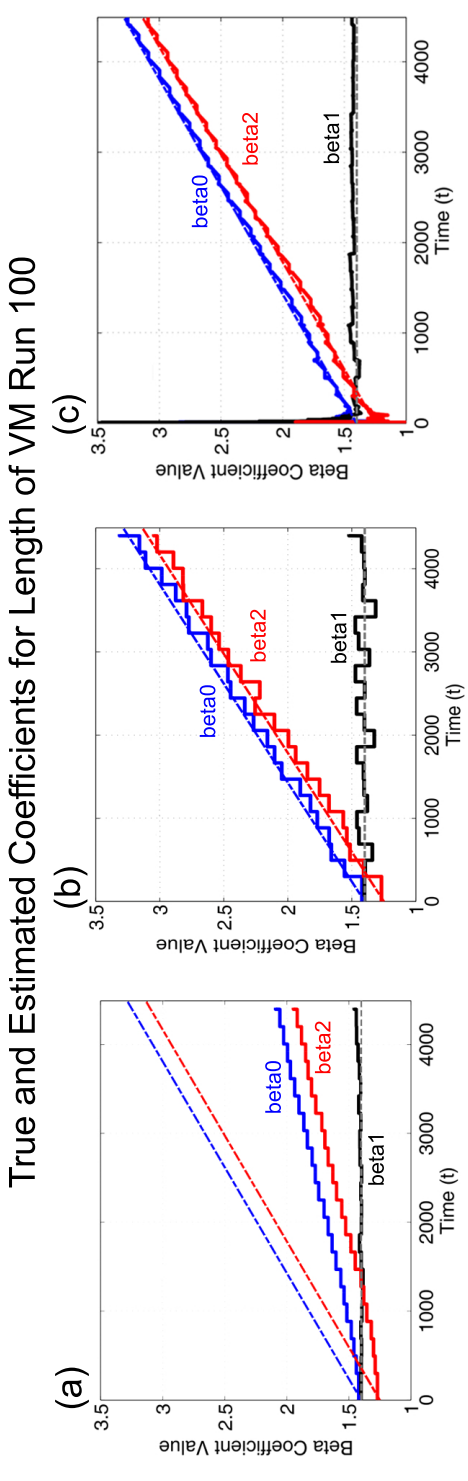


Figure 2.9: Regression coefficient estimates for each VM model for a R_{vm} of 100. (a) Estimates for Linear Regression. Notice that the coefficients are underestimated due to retained memory from start of the process, leading to an increase in missed alarm probabilities. (b) Estimates for Exponentially-weighted Linear Regression. Variance of estimations have increased due to smaller training size, but tracks the drifting coefficients. (c) Estimates for the Kalman Filter. Most accurate performance out of the three models.

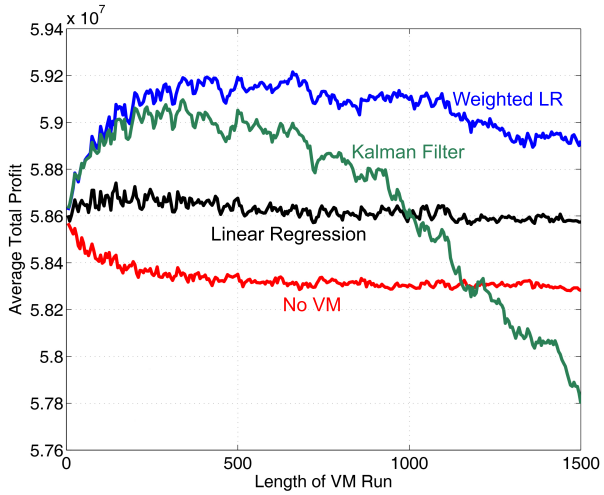


Figure 2.10: Averaged total profit for the three VM models and no VM case. This is for the scenario without re-inspection/re-tuning. "Length of VM Run" would actually be the number of unmonitored wafers between SEM recalibration measurements for the no virtual metrology case.

known, including the noise covariance matrices. As seen in Fig. 2.9(c), the Kalman Filter estimates accurately track the true values. Although the false and missed alarm probabilities are quite low, we see that the variances of these quantities increase as r_{vm} increases. Note the increase in total profit variance as the r_{vm} increases. This is due to the increased variance of the hidden state variable estimate. Again, the tradeoff between false and missed alarms result in an optimal point on the total profit curve. The maximum mean profit occurs at $\$5.909 \times 10^7$ when the r_{vm} is 340. Although the KF has the highest accuracy, this quantity is lower than the weighted linear regression model due to the time the KF needs to converge from the initial guess.

Table 2.3: Maximum mean profit and corresponding r_{vm} for each VM model for case without re-inspection.

	Max. Profit ($)	r_{vm} (# wafers)
Linear Regression	5.874×10^7	145
Exponentially-weighted LR	5.921×10^7	665
Kalman Filter	5.909×10^7	340

The average total profit for all VM models was plotted along with that of the conventional metrology scenario in Fig. 2.10. For this scenario, blended metrology schemes profit more than the conventional scheme in most regions. Although Fig. 2.10 shows that the weighted

linear regression VM model has the highest average total profit out of all three VM models, we want to emphasize that the example given in this paper is one of many cases and the results depend on different cost and revenue values. Consider an extreme example of a process with a very high false alarm cost; in this case, a linear regression-like model would be the most beneficial to the fab, even though this model has the worst MSE out of the three models. Even more, for different processes, using VM may not result in higher profit for the fab.

Given a LSL/USL, each model generates a different false and missed alarm pattern that directly translates to total profit. It is the combination of model accuracy and missed / false alarm patterns that determine the optimal total metrology profit.

2.5.2 Scenario with Re-inspection and Re-tuning

We generate data based on (2.15) with the following parameters

$$\beta_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad u = \begin{bmatrix} 0.0002 \\ 0 \\ 0.0002 \end{bmatrix}, \quad \boldsymbol{H}_t = 0.2, \quad \boldsymbol{Q}_t = \mathbf{0}_{2p \times 2p}.$$

The r_{it} is set to 200 wafers, the r_{rc} to 20 wafers, and the r_{md} to 50 wafers (2 lots) as well as the r_{rd}. 10,000 wafers are simulated and the corresponding total profit is plotted in Fig. 2.11. We do not plot the MSE, false and missed alarm rates of the VM models because the VM phase is no longer continuous as in the first scenario.

As discussed in section 2.3.2, a wafer is re-inspected every time it is flagged by VM. If there were no virtual metrology, the total profit would again look like Fig. 2.5. Too short of a VM run will decrease profit due to more frequent SEM recalibration and more delays. Too long of a VM run will decrease profit because it takes longer for the SEM to catch faults and the process will have drifted farther from target. This is what we see in the fourth column of Fig. 2.11.

The LR VM model underestimates the CD and hardly flags a wafer. Thus, the graph is similar to that of the no VM case, as seen in the first column of Fig. 2.11. For Exponentially-weighted Linear Regression, the total profit steadily decreases and plateaus as the r_{vm} increases because the VM model is not updated and again, the CD's are underestimated. Total profit for the Kalman Filter steadily increases and plateaus because the filter continuously tracks the process drift accurately. The total profit curve for both EWLR and the KF converges as r_{vm} increases because even though the VM model is not recalibrated enough, subsequent SEM measurements detect the drift and the process is retuned. We again emphasize that even though the Kalman Filter seems to generate the most profit, it does not mean it is the best VM model in general. The maximum total profit and corresponding r_{vm} is shown in Table 2.4.

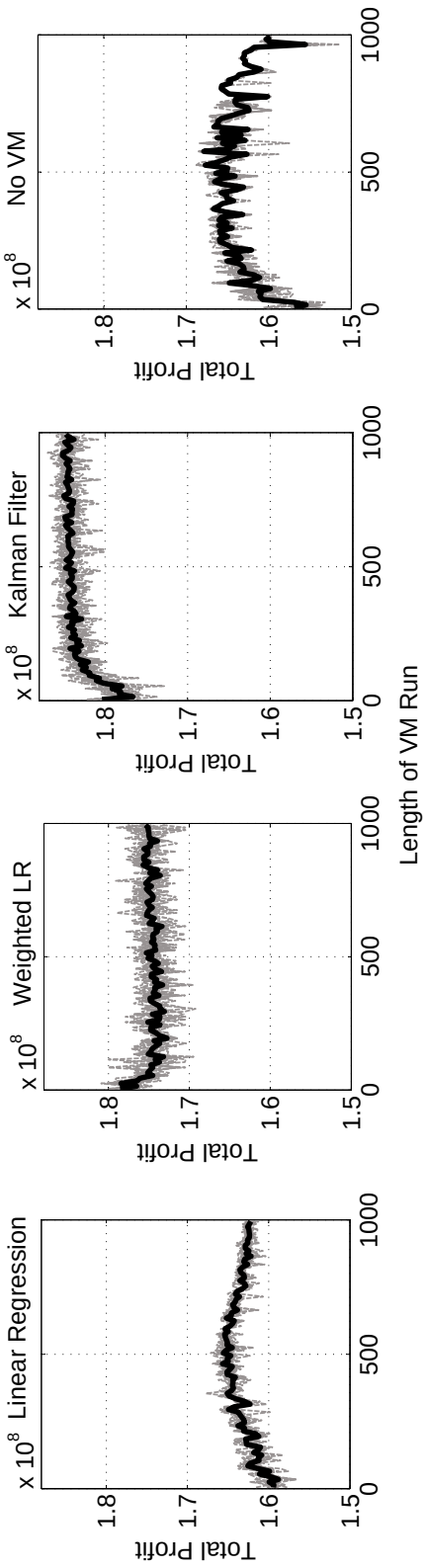


Figure 2.11: Total profit for each virtual metrology model and the conventional metrology scheme. This is for the scenario with re-inspection/re-tuning.

Table 2.4: Maximum mean profit and corresponding r_{vm} for each VM model for case with re-inspection.

	Max. Profit ($)	r_{vm} (# wafers)
No VM	1.677×10^8	525
Linear Regression	1.657×10^8	465
Exponentially-weighted LR	1.786×10^8	0
Kalman Filter	1.848×10^8	∞

2.6 Summary

This work takes a step back from empirical VM model construction and analyzes when VM is actually useful for the fab and how it can be optimized as a function of VM recalibration frequency. This is done by identifying 4 categories of VM estimate classification and defining critical parameters such as the revenue, processing cost, and actual metrology cost. These parameters were combined into a metric we called the total profit. We concentrated our analysis on a largely deviating process, the regression model with drifting coefficients, to link the relationship between VM recalibration frequency, VM model accuracy, and total profit.

Two blended metrology scenarios were carried out on a simulated drifting process for 3 VM models, Linear Regression, Exponentially-weighted Linear Regression, and the Kalman Filter. The first scenario was for a relatively short process with less than optimal C_{pk}. The second scenario was a long-term process with better but still low C_{pk}, which also allowed re-inspection to compensate for limited VM prediction quality and process re-tuning. Although the second scenario is more realistic, the continuity of the first scenario provided us with some valuable insights about blended metrology. Results indicated that each VM model had different false (Type I) and missed (Type II) alarm patterns that translated to different total profit patterns as a function of r_{vm}. Moreover, an optimal value of r_{vm} that gave the maximal total profit was identified for each model. The analysis indicated the need for missed and false alarm pattern analysis, rather than solely focusing on increasing a certain accuracy metric.

In practical reality this means that a high VM proportion may be beneficial early into the process lifecycle, while this proportion may be gradually reduced as the process becomes more mature, and therefore more stable. This finding is similar to the intuition that the overall role of metrology changes with process maturity, starting with detecting and diagnosing significant deviations, and continuing to using metrology to drive more subtle run-to-run control adjustments.

In this work, we use plasma etching as the example because etching has historically been a step more suitable for VM deployment. One could easily extend our analysis to physical metrology tools such as scatterometry, or even to "virtual" tools that consist of blending

various physical tools (i.e. CD-SEM and scatterometry).

Although this work gives us a view into how the cost, error probabilities, and recalibration sizes are related, a lot of assumptions were incorporated into the process model. For instance, we assumed the observation errors were independent from one time to the next, whereas in reality some autocorrelation would need to be accounted for. In addition, we assumed exact parameter values for the Kalman Filter. It would be interesting to see how the results change for a real processing dataset where the parameters of the Kalman Filter would have to be estimated via some parameter estimation algorithm. Moreover, we believe the next step is to provide a theoretical framework past Monte Carlo simulations that would provide robust and accurate decision rules for the metrology industry.

Chapter 3

Real-time Inspection System using Scatterometry Pupil Data

3.1 Introduction

Following the background from Appendix A, in this chapter, we utilize support vector machine (SVM) classifiers as a tool for fault detection with an application to scatterometry. In a general sense, this example will demonstrate not only the actual use of empirical inference models in the semiconductor processing arena, but also their fast development time and flexibility. Moreover, the work in this chapter does not require any alterations to existing hardware configurations.

Although there have been many proposals on optimal metrology sampling for semiconductor processing, current sites for measurements are usually picked at random or around locations on the wafer that are thought to have high process variability (e.g. wafer edge spots). As mentioned in Chapter 2, most measurements show well-controlled behavior if chosen at random, especially in high-volume matured processes. Ideally, we would like to concentrate efforts to sites that express more interesting behavior than well-controlled ones, especially for computational metrology techniques such as scatterometry. In addition, both library-based and solver-based OCD metrology outputs a number based on the assumption that the grating has a certain input profile shape, which generates missed alarms when the grating is faulty or does not conform to the specified shape. For example, the gratings studied in this paper were measured by ASML's S(T)-200 "YieldStar" in-line metrology tool with the assumption that they were infinitely long trapezoids. These missed alarms in scatterometry are mainly caused by practical limitations of the optical system (only the 0^{th} order is collected in most systems) and computational limitations, which sometimes leads to valid profile reconstruction for erroneous gratings.

There were many classification models to choose from. Some of the most prominent ones are artificial neural networks (ANN), decision trees, linear discriminant analysis (LDA) / quadratic discriminant analysis (QDA), and ADABOOST. The SVM is currently one of the

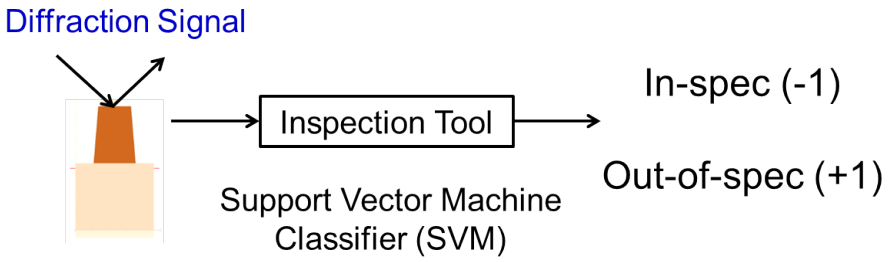


Figure 3.1: Basic schematic of proposed inspection tool. For each grating under test, only the diffracted 0^{th}-order intensity signals are used to determine whether the wafer will go onto subsequent manufacturing steps.

most sophisticated classifiers available for binary decision-making. It has a very fast, yet simple training process; moreover, as seen in Appendix A, it is very flexible across data with high feature dimension spaces, in addition to the ability to transform the original space into larger Hilbert spaces. As we will see, this aspect was suitable for scatterometry optical diffraction data, since the number of features (or varying angles and wavelengths) can get quite large. In contrast, ANN's have a longer training time for the same performance, and many other models, such as ADABOOST, can be prone to overfitting. However, by Wolpert and Macready's "No Free Lunch Theorem", it has its drawbacks. Since the complexity of the SVM is $\mathcal{O}(n^2) - \mathcal{O}(n^3)$, the training time can quickly get larger as the number of instances is increased. Moreover, it is empirically known that the SVM underperforms compared to other classifiers for data with more than 2 classes. [97, 96] Regardless, for our fault detection purposes, which classify an incoming die into 2 classes (in-spec or out-of-spec), binary SVMs performed well.

In this chapter, we construct an inspection tool with the SVM classifier that requires significantly less time and computation power and is potentially capable of detecting alarms missed by scatterometry. We use ASML's YieldStar S(T)-200 in-line metrology tool, which uses high aperture angle-resolved scatterometry with a non-linear optimization solver. As seen in Section 1.2.2.4, solver-based scatterometry requires repeatedly comparing RCWA signatures to that of the collected one until the cost function is minimized. In contrast to this process, we utilize only the signal of the diffracted 0^{th}-order to determine whether a sample CD is in-spec or out-of-spec. In essence, our objective is to construct a fast diagnostic tool that determines whether a sample should or should not proceed to subsequent manufacturing steps. This allows measurement time to be focused on samples that generate alarms. We shift our focus from the optimization algorithm used to minimize the cost function to the signal itself, which means no more iterative RCWA calculations during inspection, and thus almost instant computation time. The objective of this tool is not replacing scatterometry, but to provide more information for effective resource allocation and missed outlier detection.

As seen in Figure 3.1, the proposed system is constructed by using each pixel of the intensity signature as a feature, resulting in a high dimensional binary (or more) classification

problem. Various SVM classifiers are trained and applied to predict whether an incoming sample should be labeled as in-spec or out-of-spec depending on the middle critical dimension (MCD) 10% process window. To demonstrate that no hardware alterations are needed to use inference models, we train the classifiers on simulated RCWA diffraction signatures and apply them to collected signatures from focus-exposure matrix gratings with nominal CD dimensions ranging from 45-300nm.

3.2 Experimental Design

As mentioned in Sec. 3.1, ASML's YieldStar S(T)-200 is a high-numerical aperture (NA) angle-resolved scatterometer. The OCD signals are the intensities of the diffracted zero-order, and are observed through the pupil of the metrology objective. [30] Every point in the pupil corresponds to one incident or azimuth angle for a fixed wavelength. [31] A wavelength of 425nm is used for this work. Due to confidentiality issues, we could not publish the metrology signal used in this work. However, for an example image, please refer to Fig. 3 in Benschop *et al.* [31]. From recent sources, the measurement acquire time (MA) for YieldStar is approximately 0.40 seconds. [32]

The FEM testing set is composed of 11 1-D gratings, all with photoresist on top of bottom anti-reflective coating (BARC). Thus, the relevant profile parameters for scatterometry are photoresist height, SWA, and mid-CD (our parameter of interest). 6 matrices have a nominal pitch of 600nm, and 5 have a nominal pitch of 100nm (Table 3.1). All were exposed at IMEC, Belgium, with each wafer having a different nominal CD value from 45nm to 300nm. The data acquired from this experiment includes 1 SEM image of the grating per field, 1 YieldStar measurement per field, and the corresponding diffraction intensity signatures. Due to the objective of this work, this includes erroneous, under and over-exposed samples. We use 7 FE matrices for analysis, all printed in Appendix B (Figures B.1-B.7). The depth of focus is varied horizontally and the exposure is varied vertically. The numbers under each image represent the local CD-SEM measurement.

We first construct Bossung curves (Figures B.8-B.14) from the YieldStar measurements for each FEM wafer and determine the 10% process window. Each sample is labeled as in-spec or out-of-spec according to the specification limits given in the last column of Table 3.1. Moreover, SEM images are visually inspected for erroneous gratings (this includes necking, bridging, unexposure, etc. of the gratings) and the label is adjusted accordingly if YieldStar missed to flag such a sample. For example, in Figure B.1, faulty gratings can be seen at positions (4,0), (3,2), and (3,3), where the indices correspond to (*row*,*col*). This data is set aside for the final testing stage.

For the training and validation phase, classifiers are constructed and selected by using 1500 diffraction signatures simulated around the process window for each FE matrix. All 3 parameters, including the mid-CD, are varied to generate this data. Finally, this process is repeated for multi-categorical SVM classifiers (Section A.2.1), to see whether we can also differentiate between below LSL and above USL samples in addition to classifying them

Table 3.1: Nominal grating dimensions for focus exposure matrix.

Name	Pitch (nm)	CD (nm)	Num. Samples	LSL/USL (nm)
P90CD45	90	45	80	36/44
P100CD45	100	45	80	22.5/27.5
P100CD50	100	50	80	36/44
P600CD100	600	100	83	63/77
P600CD150	600	150	83	108/132
P600CD200	600	200	83	162/198
P600CD300	600	300	83	256.5/313.5

as out-of-spec. Although for maximum accuracy, training of the classifiers can be done on exposed wafers, we would optimally like to save resources and demonstrate that the inspection tool can be constructed without major alterations to current YieldStar or other scatterometry hardware.

3.3 Results

In this section, we demonstrate the inspection tool on physical gratings and analyze the performance with respect to different kernel functions. Linear, ellipsoid, and radial basis functions are used as the kernels.

Figure 3.2 shows the misclassification (black square), false alarm (blue diamond), and missed alarm (red triangle) rates for the FEM gratings. A more detailed breakdown can be found in Tables 3.3-3.5 for each kind of classifier. For all gratings, the RBF SVM shows the most promising results, with low false alarm and missed alarm rates. The classification rates range from 87% to over 98%, as shown in Table 3.5. In addition, the RBF SVM classifier flagged 42 out of the 43 erroneous grating samples. We also implement the multi-categorical classification given in (A.34). Results showed that all samples correctly estimated as out-of-spec by the RBF SVM classifier were also correctly classified as either above the upper limit or below lower limit.

Table 3.2 shows the SEM images for best focus and exposure (first column) and erroneous samples we found interesting for each grating (second and third columns). The third column shows 4 images that YieldStar missed to flag. As expected, gratings P90CD45, P100CD45, and P100CD50 have a significantly higher number of erroneous fields than those with 600nm pitches. Disregarding under and over-exposed samples, there were 43 faulty gratings. For 39 samples, YieldStar "measured" the mid-CD of these gratings as a very small number (e.g. 5nm instead of 30nm) and correctly flagged the fault. It missed only 4 samples, shown in the third column of Table 3.2, and instead gave a mid-CD measurement that was in-spec.

One key thing is the flexibility of the RBF kernel to capture various non-linear boundaries, hence the high classification performance across the varying gratings. Another interesting behavior is the extreme 100% false alarm and 0% missed alarm points, usually occurring in linear and ellipsoid classifiers. If a classifier is not flexible enough to handle the separation, we can see that it weighs the label with more training data to minimize the objective function in (A.13). As mentioned in Chapter 1 we would ideally like to construct a classifier that allows flexible tuning of these rates for maximum operating profit. In Chapter 4 we develop

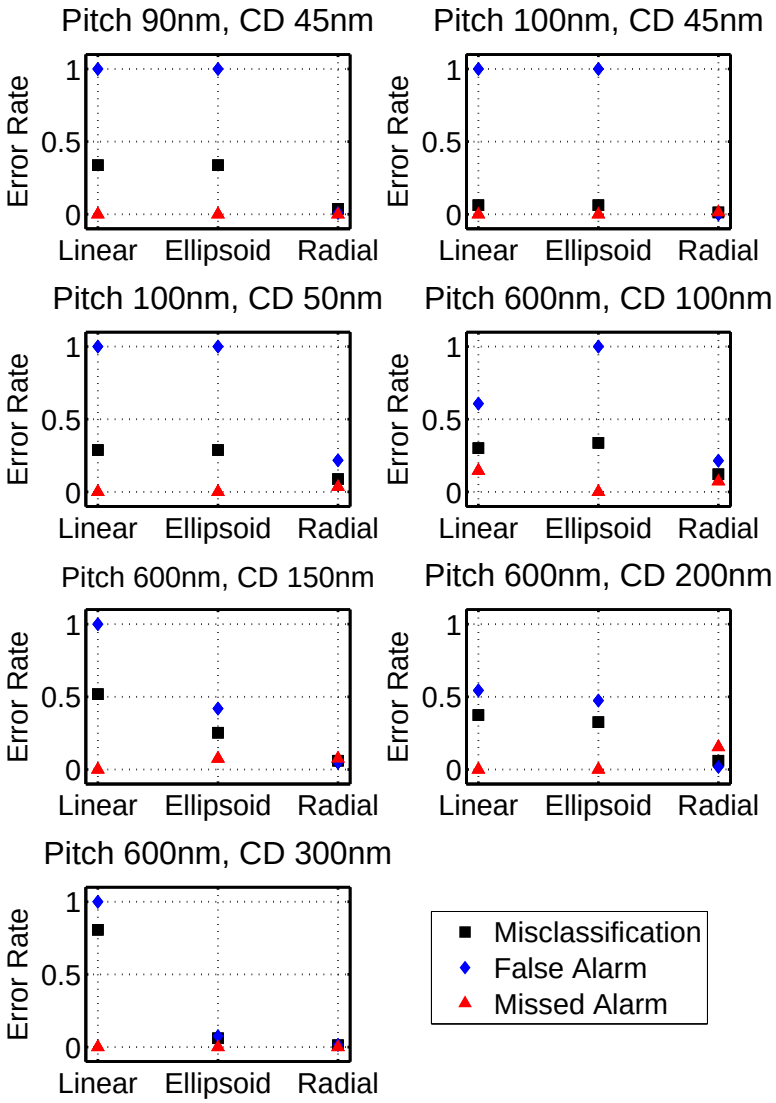
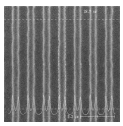
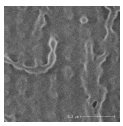
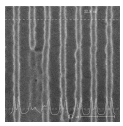
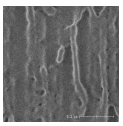
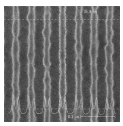
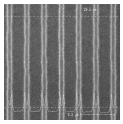
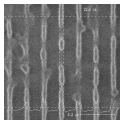
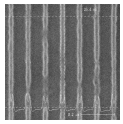
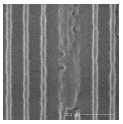
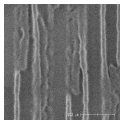
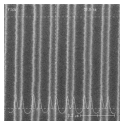
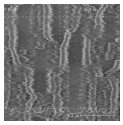
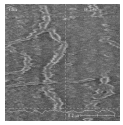
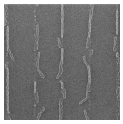
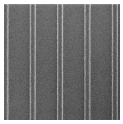
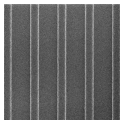
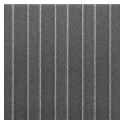


Figure 3.2: Test set misclassification, false alarm, and missed alarm rates for focus-exposure matrix gratings. The classifiers are trained only through RCWA simulations. Each plot graphs the total misclassification (black), false alarm (blue), and missed alarm (red) rates.

Table 3.2: Sample images from FEM gratings. The first column shows the best focus and exposure for each grating. The second column shows faulty gratings that YieldStar flagged out-of-spec by outputting a small numeric measurement. The third column shows faulty gratings that YieldStar missed to flag. For context, there were 43 erroneous gratings judged from SEM images; YieldStar correctly flagged 39 samples.

	Best	Faulty Gratings	
		Detected by Scatterometry	Not Detected by Scatterometry
90/45		 	 
100/45		 	 
100/50		 	
600/100			
600/150			
600/200			
600/300			

a method of quickly selecting such a classifier through cost-sensitive SVMs.

Table 3.3: FEM classification results for a linear SVM.

Pitch/CD (nm)	Actual		Estimated		Linear SVM Performance (%)		
	in-spec	out-spec	in-spec	out-spec	correct	false	missed
90/45	27	53	0	53	66.25	100	0
100/45	5	75	0	75	93.75	100	0
100/50	23	57	0	57	71.25	100	0
600/100	28	55	11	47	69.88	60.71	14.55
600/150	43	40	0	40	48.19	100	0
600/200	57	26	26	26	62.65	54.39	0
600/300	67	16	0	16	19.28	100	0

Table 3.4: FEM classification razesults for an ellipsoid SVM.

Pitch/CD (nm)	Actual		Estimated		Ellipsoid SVM Performance (%)		
	in-spec	out-spec	in-spec	out-spec	correct	false	missed
90/45	27	53	0	53	66.25	100	0
100/45	5	75	0	75	93.75	100	0
100/50	23	57	0	57	71.25	100	0
600/100	28	55	0	55	66.27	100	0
600/150	43	40	25	37	74.70	41.86	7.50
600/200	57	26	30	26	67.47	47.37	0
600/300	67	16	62	16	93.98	7.46	0

Table 3.5: FEM classification results for a radial basis kernel SVM.

Pitch/CD (nm)	Actual		Estimated		RBF SVM Performance (%)		
	in-spec	out-spec	in-spec	out-spec	correct	false	missed
90/45	27	53	24	53	96.25	11.11	0
100/45	5	75	5	7	98.75	0	1.33
100/50	23	57	18	55	91.25	21.74	3.51
600/100	28	55	22	51	87.95	21.42	7.27
600/150	43	40	41	37	93.97	4.65	7.50
600/200	57	27	56	22	92.85	1.75	18.52
600/300	67	16	66	16	98.79	1.49	0

3.4 Summary

In this chapter, we presented a pre-inspection tool that was capable of real-time CD monitoring through construction of a classification model. We first used simulated RCWA diffraction signals for parameter ranges of interest to train a SVM classifier. This was done by tagging all signals with MCD's outside the specification limits as out-of-spec, and all other samples as in-spec. The gratings were assumed to be infinite trapezoids. We conducted classifications on gratings exposed from a focus-exposure matrix at IMEC.

The RBF kernel SVM had the highest classification rates, ranging from 87-98%, and flagged all but one erroneous grating. We also successfully constructed multi-categorical classifiers to differentiate samples above the upper specification limit (USL) and below the lower specification limit (LSL). It should be noted that our tool was applied to one type of scatterometry implementation, the YieldStar, and the performance of the classifier on different scatterometers is to be further explored and may result in different statistics then those reported here. However, the overall design framework of the tool provides an easy extension over to other scatterometry methods, and each will have its limitations and advantages. For example, although the off-axis beams in angle-resolved scatterometry cause increased sensitivity to the profile, it is hard to differentiate between changes in profile parameters and changes in underlying thin-film thickness due to the available wavelengths. These thin-film effects may decrease in spectroscopic reflectometry (SR) by using a shorter wavelength.

The inspection tool presented in this paper cannot be utilized as a standalone metrology system. However, it provides a few additional functions that scatterometry does not have. One, it preserves more information about the grating by using all feature dimensions instead of reducing the intensities to one value through a cost function. This gives the classifier the flexibility and sensitivity to flag erroneous gratings missed by scatterometry, since it does not have to bias the grating into a fixed shape. Second, the computation time of the classifier for an incoming signal is one vector dot product. This allows the tool to be used real-time and focus the computation power of scatterometry on samples that are deemed more interesting. Both of these advantages could be particularly helpful in reducing measurement setup time, as it would allow fast recipe setup for OCD measurements.

The final significant but non-obvious benefit is a further increase in processing flexibility due to the use of classification techniques. For example, Tables 3.3-3.5 show resulting error rates for equal false alarm and missed alarm costs. That is, the two types of errors contribute equally in constructing the classifier. However, additional statistical techniques applied to the classifier can allow tuning of the two error rates depending on individual situations, an aspect that is not available in cost minimization algorithms. For example, missed alarms cause significantly more loss in the fab then do false alarms, and the flag rate can be adjusted accordingly. Such a method will be discussed in Chapter 4.

As device structures get more complicated and the dimensions even smaller, a holistic approach to metrology is the only way to meet tight measurement requirements. [22, 21, 33] In a more general sense, this means that all evidence from current tools or any combination of tools should be utilized and combined to provide an optimal manufacturing environment.

Although we utilized only OCD signals in this paper, signals from other sources, such as processing conditions from previous steps, can easily be included in the analysis by increasing the dimensionality of the data. In essence, we have demonstrated a framework that provides real-time monitoring capabilities and incorporates the idea of holistic metrology by providing a flexible and easy framework in which signals from other sources can easily be incorporated into the computational inference process.

Chapter 4

Fast Model Selection for Grating Classification

4.1 Introduction

In Chapter 3, SVM classifiers were used to quickly determine a fault status of a grating without iterative optimization steps. As seen in Chapter 2, operational costs depend on the cost of false and missed alarms, and we would like to have the flexibility of choosing the classifier that results in an optimal combination. Note that we are discussing the error rates on unseen data, such as the validation set or testing set, and not the training data. As seen in Figure 4.1, a tradeoff between the two types of errors occur for every classification model. The tradeoff curve is a straight line for random guessing (e.g. coin flip), and gets steeper for high-performing classifiers. Notice that for all performance levels, all curves are bounded by the extreme cases of 100% false alarms or 100% missed alarms. As seen in Tables 3.3-3.5, conventional SVM models result in only one point on the curve. However, depending on relative costs for these two errors, one may benefit from tuning the classifier such that it gives a different error combination. For example, in Figure 4.1, we would like to shift the current 50-50 error combination to one with a higher false alarm rate, if the missed alarm costs are significantly higher.

As we will see, this can be done by using cost-sensitive 2ν-SVMs (Section A.2), with an additional hyperparameter γ that weights one category over the other for different treatments between the two error rates. Since the tradeoff curve for a classifier is unknown, the question becomes how to choose the optimal hyperparameter set in (A.27) or (A.28), such that it minimizes some asymmetric generalization cost in the form of (A.32). For cost-sensitive problems, choosing the right value of γ becomes especially important. However, a significant obstacle of this approach is the substantial amount of computational power needed for the cross-validation process as the number of hyperparameters is increased. Although our motivation behind this chapter is cost-sensitive classification, our problem is just one example of how to *quickly* choose an optimal hyperparameter set for any classification problem

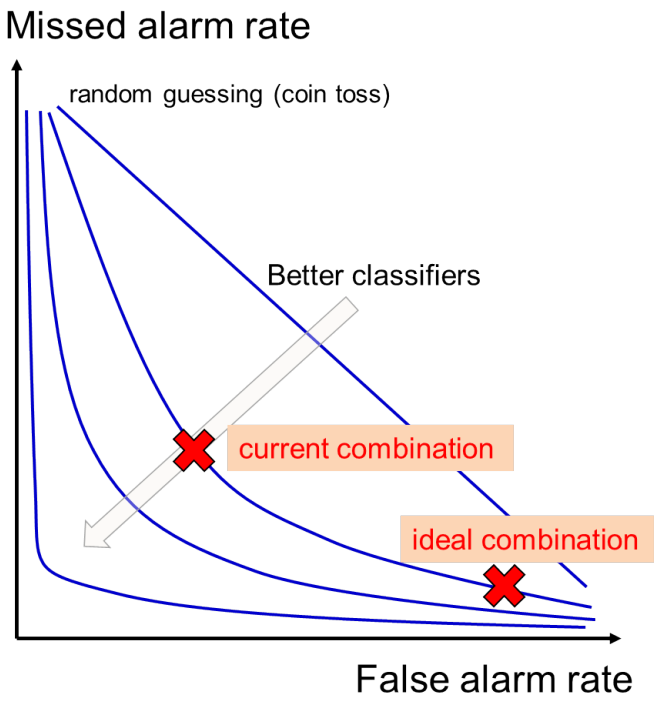


Figure 4.1: Tradeoff between false and missed alarm rates for classification models. As seen in this figure, the curve is a straight line for random guessing, and gets steeper for high-performing classifiers.

that introduces additional hyperparameters to incorporate more flexibility and performance improvement in specific conditions.

Conventional application of the soft margin C-SVM or ν-SVM with a radial basis kernel function (RBF) involves the selection of 2 hyperparameters and can be solved by aforementioned exhaustive search methods (Section A.0.4). [34] Nonetheless, alternate forms of the SVM introduce additional parameters that result in an improved performance of the resulting classifier. Two examples are the aforementioned cost-sensitive 2ν-SVM and the ARD-Gaussian kernel SVM. In the ARD-Gaussian kernel SVM, each feature dimension is assigned a separate kernel parameter to compensate for differences in overfitting and underfitting between different feature spaces. [35] In addition, several works [36, 37, 38], have proposed to use the ARD-Gaussian kernel for feature selection.

Since the exhaustive approach quickly becomes intractable for SVMs with 3 or more parameters, the design of scalable hyperparameter selection is not only favorable but also necessary. In the context of C-SVM, Chapelle *et al.* (2002) [39] proposed to implicitly calculate gradients of various error bounds and loss functions and proceeded to use this in a gradient-descent algorithm. Based on this method, Keerthi *et al.* [40] proposed an improved algorithm that does not require large matrix inversion. This method enables fast selection

of hyperparameters and is widely applicable to various situations.

In this chapter, we tackle the problem of constructing a classifier which enables flexible tuning of error rates as needed. In a broader perspective, we deal with shortening the cross-validation process for any type of SVM with additional hyperparameters for increased degree of freedom. As an example, we consider the problem of optimal hyperparameter selection for the cost-sensitive 2ν-SVM with the ARD-Gaussian kernel. We propose a new approach by exploiting the inherent properties of the dual SVM problem, and adopting a parametric programming framework. [41] We show that the challenging problem of finding the explicit form of the dual SVM solution as a function of its hyperparameters can be solved under mild conditions. Our theorem indicates that in the feasible set, the solution is a piecewise differentiable function of the hyperparameters, with an explicit form for a given critical region. This allows us to write the generalization cost as a function of the hyperparameters, so that explicit computation of the gradient is made possible with smooth approximations. Finally, a gradient descent algorithm is constructed for optimal hyperparameter selection. Our solution has nice properties in terms of continuity and piecewise differentiability. While we developed our method for the 2ν-SVM with the ARD Gaussian Kernel, it can be readily adopted for $\nu - SVM$'s and $C - SVM$'s with other Mercer kernels.

This chapter is organized as follows. Section 4.2 reviews the 2ν-SVM with the ARD-Gaussian kernel and introduces notation used throughout the chapter. Section 4.3 gives the derivation for the explicit form of the dual solution and the classifier as a function of the hyperparameters in a parametric optimization framework. Section 4.4 defines the empirical generalization cost and calculates its gradient with smooth approximations. Based on these results, Section 4.5 proposes a gradient descent algorithm for optimal parameter selection. Finally, Section 4.6 demonstrates the performance of our method on two datasets, and show that it is significantly faster than exhaustive search without compensating accuracy. The work developed in this chapter was co-authored by Yuxun Zhou.

4.2 Background on 2ν-SVM with ARD-Gaussian Kernel

The dual problem for 2ν-SVM with an ARD-Gaussian kernel was

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\delta} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_i^T \boldsymbol{x}_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le \frac{\gamma}{n} \quad \text{for} \quad i \in I_+ \\ & 0 \le \alpha_i \le \frac{1-\gamma}{n} \quad \text{for} \quad i \in I_- \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \sum_{i=1}^{n} \alpha_i \ge \nu. \end{aligned} \tag{4.1}$$

with

$$\kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{ -\sum_{k=1}^{p} \beta_k \left(x_i^k - x_j^k \right)^2 \right\} \tag{4.2}$$

where we denote β_k aks the kernel parameter for k^{th} feature, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^T \in (0, \infty)^p$, and x_i^k and x_j^k denote the k^{th}-dimension of samples i and j. Note that the kernel function is written as $\kappa_{\boldsymbol{\beta}}(\cdot)$ to emphasize the dependence on hyperparameters $\boldsymbol{\beta}$. Since the ARD-Gaussian kernel is also a Mercer kernel, its positive definite (PD) property guarantees the strict convexity of (4.1). Therefore, the solution $\boldsymbol{\alpha}^*$ is unique for a fixed hyperparameter configuration. With fixed training data, this implies that $\boldsymbol{\alpha}^*$ can be determined as a function of the hyperparameters.

Throughout this chapter, we denote $\boldsymbol{\theta} = [\gamma, \nu]^T \in (0,1) \times (0, 1/2]$, and $\boldsymbol{\phi} = [\boldsymbol{\theta}^T, \boldsymbol{\beta}^T]^T$. Note that according to (A.31), only a subset of the parameter space $(0,1) \times (0, 1/2] \times (0, \infty)^p$ can yield feasible prime or dual problems. With slight abuse of terminology, we call this set the "feasible set" of hyperparameters. Moreover, after solving (4.1), we define the following set of partitions:

Definition 1. ***(Partition of Training Data)*** *After solving the dual problem of 2ν-SVM, we obtain a partition of training samples as $\{\mathcal{S}_0, \mathcal{S}_b, \mathcal{S}_{ub}\}$ based on the value of corresponding $\boldsymbol{\alpha}^*$. We define the index set of non-support vectors as $\mathcal{S}_0 \triangleq \{i \mid \alpha_i^* = 0\}$, bounded support vectors as $\mathcal{S}_b \triangleq \mathcal{S}_b^+ \cup \mathcal{S}_b^- \triangleq \{i \mid i \in \mathcal{T}_+, \alpha_i^* = \frac{\gamma}{n}\} \cup \{i \mid i \in \mathcal{T}_-, \alpha_i^* = \frac{1-\gamma}{n}\}\}$ and unbounded support vectors as $\mathcal{S}_{ub} \triangleq \mathcal{S}_{ub}^+ \cup \mathcal{S}_{ub}^- \triangleq \{i \mid i \in \mathcal{T}_+, 0 < \alpha_i^* < \frac{\gamma}{n}\} \cup \{i \mid i \in \mathcal{T}_-, 0 < \alpha_i^* < \frac{1-\gamma}{n}\}$.*

4.3 Parametric Optimization Method

In the terminology of operational research and optimization, a problem that depends on multiple parameters is referred to as a parametric program (PP). The goal is to analyze the dependence of the optimal solution $\boldsymbol{\alpha}^*$ on the hyperparameter vector $\boldsymbol{\phi}$. However, a non-linear PP is not a trivial problem, and existing theories are usually restricted to special cases or require additional assumptions. In this section, we derive the explicit form of $\boldsymbol{\alpha}^*$ as a function of $\boldsymbol{\phi}$, by exploring the partition of support vectors. We start with reformulating (4.1) into a PP framework.

4.3.1 Dual Problem Reformulation

Since the order of training samples does not matter, we re-write (4.1) with a systematic form by rearranging training data as $[\mathcal{T}_+ \ \mathcal{T}_-]^T$. Thus the corresponding label vector $\boldsymbol{y}$ is simply $[+1, \ldots, +1, -1, \ldots, -1]^T$ with dimension $n_+ + n_- = n$. We define the following matrices,

$$\boldsymbol{Q}_{\boldsymbol{\beta}} = \begin{bmatrix} y_1 y_1 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_1, \boldsymbol{x}_1) & y_1 y_2 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & y_1 y_n \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ y_2 y_1 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_2, \boldsymbol{x}_1) & y_2 y_2 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & y_2 y_n \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ y_n y_1 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_n, \boldsymbol{x}_1) & y_n y_2 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & y_n y_n \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

$$\boldsymbol{C}^{\alpha} = \begin{bmatrix} -1 & \cdots & 0 \\ \vdots & & \vdots \\ \vdots & & \vdots \\ 0 & \cdots & -1 \\ \hline 1 & \cdots & 0 \\ \vdots & & \vdots \\ \hline \vdots & & \vdots \\ 0 & \cdots & 1 \\ \hline -1 & \cdots & -1 \end{bmatrix} \qquad \boldsymbol{C}^{\theta} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ \vdots & \vdots \\ 0 & 0 \\ \hline 1/n & 0 \\ \vdots & \vdots \\ \hline -1/n & 0 \\ \vdots & \vdots \\ \hline 0 & -1 \end{bmatrix} \qquad \boldsymbol{C}^{0} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ \hline 0 \\ \vdots \\ \hline \frac{1}{n} \\ \vdots \\ \hline 0 \end{bmatrix}$$

and (4.1) becomes

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Q}_{\boldsymbol{\beta}} \boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{C}^{\boldsymbol{\alpha}} \boldsymbol{\alpha} \leq \boldsymbol{C}^{\theta} \boldsymbol{\theta} + \boldsymbol{C}^{0} \\ & \boldsymbol{\alpha}^T \boldsymbol{y} = 0. \end{aligned} \tag{4.3}$$

where the inequality constraints are encapsulated into a matrix form. Notice that $\boldsymbol{C}^{\boldsymbol{\alpha}}$ has dimensions $(2n+1) \times n$, with each block having n, n_+, n_- and 1 rows. $\boldsymbol{C}^{\theta}$ has dimensions

$(2n+1) \times 2$, and $\boldsymbol{C}^0$ has dimensions $(2n+1) \times 1$, with blocks defined similarly.

Note that (4.3) is in fact a quadratic PP with $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ as hyperparameters. The study and application of quadratic PP's have been previously addressed by operational research and control communities [41, 42]. However, most results do not allow variations of the matrix $\boldsymbol{Q}_{\boldsymbol{\beta}}$. More importantly, they rely on the Linear Independence Constraint Qualification (LICQ) condition [43], which in our case cannot be satisfied due to the existence of the equality constraint, $\boldsymbol{\alpha}^T \boldsymbol{y} = 0$. In fact, problem (4.3) corresponds to a degenerate case, and usually requires special treatment. In subsequent parts, we show that the explicit form of $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ can be obtained under mild conditions. For convenience, we largely borrow notation from [42] and introduce some definitions before we proceed.

Definition 2. ***(Active Constraint)*** *Assume that a solution of (4.3) has been obtained as $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$. Then the i^{th} row of the constraint is said to be active at $\boldsymbol{\phi}$, if $\boldsymbol{C}_i^{\alpha}\boldsymbol{\alpha}^*(\phi) = \boldsymbol{C}_i^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_i^0$, and inactive if $\boldsymbol{C}_i^{\alpha}\boldsymbol{\alpha}^*(\phi) < \boldsymbol{C}_i^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_i^0$. The index set of all active inequality constraints i is denoted by $\mathcal{A}$, and all inactive inequality constraints by $\mathcal{A}^C$. We denote $\boldsymbol{C}_{\mathcal{A}}^{\alpha}$ as the $\boldsymbol{C}^{\alpha}$ matrix with only rows that correspond to the active constraints, and $\boldsymbol{C}_{\mathcal{A}^C}^{\alpha}$ as the matrix with only rows that correspond to inactive constraints.*

Definition 3. ***(Non-degenerate SVM)*** *We say that a solution of a soft-margin SVM is Non-degenerate if the set of unbounded support vectors $\mathcal{S}_{ub}$ contains at least one $i \in \mathcal{T}_+$ and at least one $i' \in \mathcal{T}_-$, i.e. both $\mathcal{S}_{ub}^+$ and $\mathcal{S}_{ub}^-$ are non-empty.*

4.3.2 Lemmas for Matrix Invertibility

In this section, we discuss the invertibility of a matrix that is crucial in solving (4.3). With some constraints active, we define a matrix

$$\boldsymbol{P}_{\boldsymbol{\beta}} \triangleq \boldsymbol{C}_{\mathcal{A}}^{\alpha} \left(\frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{M}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1} \right) \boldsymbol{C}_{\mathcal{A}}^{\alpha T}. \tag{4.4}$$

Note that if all $(\boldsymbol{x}_i, y_i) \in \mathcal{T}$ are unique, $\boldsymbol{Q}_{\boldsymbol{\beta}}$ is positive definite (PD), and thus invertible. However, $\boldsymbol{P}_{\boldsymbol{\beta}}$ is symmetric, but in general not always invertible. A simple example is the case of training data with samples from only one class. The invertibility issue is one of the major difficulties in solving the parametric optimization problem. With a series of lemmas given in Appendix C, we prove the following result. For the proof, please refer to the end of the thesis.

Theorem 1. *If the solution $\boldsymbol{\alpha}^*$ of (4.3) is non-degenerate, the matrix $\boldsymbol{P}_{\boldsymbol{\beta}}$ is symmetric negative definite, hence invertible.*

Theorem 1 guarantees the uniqueness of the Lagrangian multipliers, as we will see later in this section. Another issue is the non-degeneracy requirement in the theorem. Since except for deliberately designed cases, meaningful classifiers have at least one unbounded support vector in both classes, the vast majority of classifiers will satisfy non-degeneracy.

4.3.3 Explicit Form of $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$

With Theorem 1, we are now able to derive a piecewise explicit form for the optimal solution $\boldsymbol{\alpha}^*$ of (4.3) as a function of the hyperparameters. We present one of the main results of this chapter.

Theorem 2. *Assume that the solution of a 2ν-SVM is non-degenerate and induces a set of active and inactive constraints $\mathcal{A}$ and $\mathcal{A}^C$, respectively. Then in the critical region defined by*

$$\begin{cases} \boldsymbol{P}_{\boldsymbol{\beta}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^{0}) \geq 0 \\ \boldsymbol{C}_{\mathcal{A}^C}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}^C}^{0} - \boldsymbol{C}_{\mathcal{A}^C}^{\alpha}\boldsymbol{T}_{\boldsymbol{\beta}}\boldsymbol{P}_{\boldsymbol{\beta}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^{0}) \geq 0 \end{cases} \tag{4.5}$$

the optimal solution $\boldsymbol{\alpha}^*$ of (4.3) admits a closed form

$$\boldsymbol{\alpha}^*(\boldsymbol{\phi}) = \boldsymbol{T}_{\boldsymbol{\beta}}\boldsymbol{P}_{\boldsymbol{\beta}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^{0}) \tag{4.6}$$

where

$$\boldsymbol{T}_{\boldsymbol{\beta}} \triangleq \left(\frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1} \right) \boldsymbol{C}_{\mathcal{A}}^{\alpha T}. \tag{4.7}$$

Proof. Problem (4.3) is equivalent to

$$\begin{aligned} \max_{\lambda,\boldsymbol{\mu}} \min_{\boldsymbol{\alpha}} \quad & \mathcal{L}(\boldsymbol{\alpha}, \lambda, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Q}_{\boldsymbol{\beta}}\boldsymbol{\alpha} \\ & \quad + \lambda(\boldsymbol{\alpha}^T\boldsymbol{y}) + (\boldsymbol{C}^{\alpha}\boldsymbol{\alpha} - \boldsymbol{C}^{\theta}\boldsymbol{\theta} - \boldsymbol{C}^{0})^T\boldsymbol{\mu} \\ \text{subject to} \quad & \boldsymbol{\mu} \geq 0. \end{aligned} \tag{4.8}$$

where $\lambda, \boldsymbol{\mu}$ are the Lagrangian multipliers for the equality and inequality constraints, respectively.

The KKT conditions specify for optimal $(\boldsymbol{\alpha}^*, \lambda^*, \boldsymbol{\mu}^*)$,

$$\nabla_{\alpha}\mathcal{L} = \boldsymbol{Q}_{\boldsymbol{\beta}}\boldsymbol{\alpha}^* + \lambda^*\boldsymbol{y} + \boldsymbol{C}^{\alpha T}\boldsymbol{\mu}^* = 0 \tag{4.9a}$$

$$\boldsymbol{y}^T\boldsymbol{\alpha}^* = 0 \tag{4.9b}$$

$$\mu_i^* \geq 0 \qquad \text{for } i = 1, \ldots, n \tag{4.9c}$$

$$\mu_i^*(\boldsymbol{C}^{\alpha}\boldsymbol{\alpha}^* - \boldsymbol{C}^{\theta}\boldsymbol{\theta} - \boldsymbol{C}^{0})_i = 0 \qquad \text{for } i = 1, \ldots, n \tag{4.9d}$$

$$(\boldsymbol{C}^{\alpha}\boldsymbol{\alpha}^* - \boldsymbol{C}^{\theta}\boldsymbol{\theta} - \boldsymbol{C}^{0})_i \leq 0 \qquad \text{for } i = 1, \ldots, n. \tag{4.9e}$$

From (4.9a),

$$\boldsymbol{\alpha}^* = -\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}(\lambda^*\boldsymbol{y} + \boldsymbol{C}^{\alpha T}\boldsymbol{\mu}^*). \tag{4.10}$$

where $\boldsymbol{Q}_{\boldsymbol{\beta}}$ is invertible for any $\boldsymbol{\beta} \in (0, \infty)^p$ due to its positive definiteness. Now plugging in the above expression for $\boldsymbol{\alpha}^*$ into (4.9b),

$$\begin{aligned} &\boldsymbol{y}^T\boldsymbol{\alpha}^* = -\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}(\lambda^*\boldsymbol{y} + \boldsymbol{C}^{\alpha T}\boldsymbol{\mu}^*) = 0 \\ \Rightarrow &-\lambda^*\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{C}^{\alpha T}\boldsymbol{\mu}^* = 0 \\ \Rightarrow &\lambda^* = -\frac{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{C}^{\alpha T}\boldsymbol{\mu}^*}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}}. \end{aligned} \tag{4.11}$$

There are two cases for the inequality constraints.

$$\begin{cases} \mu_i^* > 0, \;\; \boldsymbol{C}_i^{\alpha}\boldsymbol{\alpha}^* - \boldsymbol{C}_i^{\theta}\boldsymbol{\theta} - \boldsymbol{C}_i^0 = 0 & \text{for } i \in \mathcal{A} \\ \mu_i^* = 0, \;\; \boldsymbol{C}_i^{\alpha}\alpha^* - \boldsymbol{C}_i^{\theta}\theta - \boldsymbol{C}_i^0 < 0 & \text{for } i \in \mathcal{A}^C \end{cases} \tag{4.12}$$

We define vectors $\boldsymbol{\mu}_{\mathcal{A}}$ and $\boldsymbol{\mu}_{\mathcal{A}^C}$, where each one is composed of elements from $\boldsymbol{\mu}$ with $i \in \mathcal{A}$ and $i \in \mathcal{A}^C$, respectively. Since $\boldsymbol{C}^{\alpha T}\boldsymbol{\mu}^* = \boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\mu}_{\mathcal{A}}^*$, the formula for $\boldsymbol{\alpha}^*, \lambda^*$ in equations (4.10) and (4.11) reduces to

$$\boldsymbol{\alpha}^* = -\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\left(\lambda^*\boldsymbol{y} + \boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\mu}_{\mathcal{A}}^*\right) \tag{4.13}$$

$$\lambda^* = -\frac{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\mu}_{\mathcal{A}}^*}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}}. \tag{4.14}$$

Substituting (4.14) into (4.13), we get

$$\begin{aligned} \boldsymbol{\alpha}^* &= \left(\frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\right)\boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\mu}_{\mathcal{A}}^* \\ &= \boldsymbol{T}_{\boldsymbol{\beta}}\boldsymbol{\mu}_{\mathcal{A}}^*. \end{aligned} \tag{4.15}$$

where we have defined

$$\boldsymbol{T}_{\boldsymbol{\beta}} \triangleq \left(\frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\right)\boldsymbol{C}_{\mathcal{A}}^{\alpha T}.$$

Another equality we get from the active constraints is

$$\boldsymbol{C}_{\mathcal{A}}^{\alpha}\boldsymbol{\alpha}^* = \boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^0. \tag{4.16}$$

Combining (4.15) and (4.16), we get an expression for $\boldsymbol{\mu}_{\mathcal{A}}^*$ since

$$\begin{aligned} &\boldsymbol{C}_{\mathcal{A}}^{\alpha}\left(\frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\right)\boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\mu}_{\mathcal{A}}^* \\ &= \boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^0 \\ \Rightarrow &\boldsymbol{\mu}_{\mathcal{A}}^* = \boldsymbol{P}_{\boldsymbol{\beta}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^0) \end{aligned} \tag{4.17}$$

with $\boldsymbol{P}_{\mathcal{B}}$ defined as before, whose invertibility has already been proved. Finally, we get

$$\boldsymbol{\alpha}^*(\boldsymbol{\phi}) = \boldsymbol{T}_{\mathcal{B}}\boldsymbol{P}_{\mathcal{B}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^{0}) \tag{4.18}$$

as an explicit function of $\boldsymbol{\phi}$.

We now derive the boundaries of the critical region in which (4.18) holds. For active constraints, (4.9c) and (4.9e) require that

$$\boldsymbol{\mu}_{\mathcal{A}}^* \geq 0 \tag{4.19}$$

$$\boldsymbol{C}_{\mathcal{A}^C}^{\alpha}\boldsymbol{\alpha}^* \leq \boldsymbol{C}_{\mathcal{A}^C}^{\theta} + \boldsymbol{C}_{\mathcal{A}^C}^{0}. \tag{4.20}$$

These two inequalities yield

$$\begin{aligned} &\boldsymbol{P}_{\mathcal{B}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^{0}) \geq 0 \\ &\boldsymbol{C}_{\mathcal{A}^C}^{\alpha}\boldsymbol{T}_{\mathcal{B}}\boldsymbol{P}_{\mathcal{B}}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^{0}) \leq \boldsymbol{C}_{\mathcal{A}^C}^{\theta}\boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}^C}^{0}. \end{aligned} \tag{4.21}$$

Hence in the region defined above, the optimum $\boldsymbol{\alpha}^*$ can be computed explicitly as a function defined by (4.18). □

Note that in each critical region, the partition $\{\mathcal{S}_0, \mathcal{S}_b, \mathcal{S}_{ub}\}$ does not change, because they are uniquely determined by the set of active and inactive constraints, $\mathcal{A}$ and $\mathcal{A}^C$. Since the dual of the classic soft margin ν-SVM is simply the formulation (4.3) with $\gamma = 1/2$, the parametric solution for the ν-SVM is just a special case. In addition, equation (4.10) only relies on the PD property and is readily extended to other Mercer kernels used in SVMs. Therefore, the solution given in Theorem 2 is quite general for soft-margin SVMs.

In short, Theorem 2 concludes that the optimal solution $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ is a piecewise defined function, and for a given critical region, the SVM has to be solved only once since other $\boldsymbol{\alpha}^*$ for any $\boldsymbol{\phi}$ inside the region can be constructed through (4.18). From the explicit form, we see that $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ is differentiable in each critical region. The following subsection provides some interesting properties of the critical regions and its boundaries.

4.3.4 Some Properties of $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$

Proposition 1. *Assuming a non-degenerate SVM, $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ is continuous over the feasible set. The number of critical regions N_r is finite and upper bounded by $|\mathcal{I}_-| \cdot |\mathcal{I}_+| \cdot 2^{|\mathcal{I}|-2}$. The set of all critical regions $R_1, R_2, \ldots, R_{N_r}$ defined by Theorem 2 constitute a partition of the feasible set, i.e. any feasible $\boldsymbol{\phi}$ belongs to one and only on region.*

For proof, refer to Appendix C. At first glance, the upper bound of N_r is quite large since it is exponential to the sample size. However, in practice, this number is limited due to conflicts and inclusion of constraints. We will also show in Section 4.6.1 that the number and area of the critical regions vary according to the choice of parameters. The proposition implies that since there are only finite number of boundaries, their overall measure is zero

when an appropriate measure is defined for regions. Consequently, with the differentiability in each critical region, we conclude that $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ is differentiable almost everywhere. Moreover, if we consider a fixed kernel parameter and look at the $2D$ space of γ and ν, we have the following proposition.

Proposition 2. *In the 2D plane of $\boldsymbol{\theta}$ with fixed $\boldsymbol{\beta}$, $\boldsymbol{\alpha}^*_{\boldsymbol{\beta}}(\boldsymbol{\theta})$ is a piecewise affine function of $\boldsymbol{\theta}$ on a set of polyhedrons, and the optimal objective function is piecewise quadratic.*

Proof is obvious by observing (4.6) with constant $\boldsymbol{\beta}$. Illustrations of this proposition will be also given in Section 4.6.1.

4.3.5 Explicit Form of the Classifier

Since the explicit form of $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ is known, the classifier can also be constructed through (A.12). For simplicity of notation, we define

$$\boldsymbol{d_j} = [y_1 \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_1, \boldsymbol{x}_j), \ldots, y_n \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_n, \boldsymbol{x}_j)]^T \tag{4.22}$$

for any $j \in \mathcal{Z}$. In addition, if we denote the vector $\boldsymbol{d}_{\mathcal{U}} = \frac{1}{2|\mathcal{U}|} \sum_{u \in \mathcal{U}} \boldsymbol{d}_u$, the calculation of the intercept b^* in (A.12) can be reformulated as

$$b^* = \boldsymbol{d}_{\mathcal{U}}^T \boldsymbol{\alpha}^* = \frac{1}{2|\mathcal{U}|} \sum_{u \in \mathcal{U}} \boldsymbol{d}_u^T \boldsymbol{\alpha}^*. \tag{4.23}$$

Note that $\boldsymbol{d_j}$ also depends on $\boldsymbol{\beta}$. By this notation, we define the hyperplane for a validation feature vector $h(\boldsymbol{x}_j)$

$$h(\boldsymbol{x}_j) = \sum_{i=1}^{n} \alpha_i^* y_i \kappa_{\boldsymbol{\beta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) + b^* = (\boldsymbol{d}_j^T - \boldsymbol{d}_{\mathcal{U}}^T) \boldsymbol{\alpha}^*(\boldsymbol{\phi}) \tag{4.24}$$

$$= (\boldsymbol{d}_j^T - \boldsymbol{d}_{\mathcal{U}}^T) \boldsymbol{T}_\beta \boldsymbol{P}_\beta^{-1} (\boldsymbol{C}_{\mathcal{A}}^\theta \boldsymbol{\theta} + \boldsymbol{C}_{\mathcal{A}}^0), \tag{4.25}$$

and the classifier is merely the sign of $h(\boldsymbol{x}_j)$. Now we have the explicit form of the classifier as a function of the hyperparameters.

4.4 Generalization Cost Estimation and Gradient Calculation

The parametric solution in Theorem 2 allows us to construct the hyperplane and the classifier as a piecewise defined function of hyperparameters. In this section, we extend these results to the empirical generalization cost. Finally we calculate its gradients with a smooth approximation for the purpose of hyperparameter selection.

4.4.1 Estimation of Asymmetric Generalization Cost

As seen in Section A.2, an example of asymmetric empirical generalization cost on the validation set was given by

$$\begin{aligned}\hat{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}) =& \frac{c_1}{2m_-}\sum_{j\in\mathcal{J}_-}[1-y_j\,\mathrm{sign}(h_j)]\frac{m_-}{m} \\ &+\frac{c_2}{2m_+}\sum_{j\in\mathcal{J}_+}[1-y_j\,\mathrm{sign}(h_j)]\frac{m_+}{m}\end{aligned} \tag{4.26}$$

where $\frac{1}{2m_-}\sum_{j\in\mathcal{Z}_-}[1-y_j\,\mathrm{sign}(h_j)]$ and $\frac{1}{2m_+}\sum_{j\in\mathcal{Z}_+}[1-y_j\,\mathrm{sign}(h_j)]$ are unbiased estimates of false and missed alarms, c_1 and c_2 are their corresponding costs, and $\frac{m_-}{m}$ and $\frac{m_+}{m}$ are the estimated probability of occurrences for the two classes. Although other forms of generalization cost can be considered, we use (A.32) for simplicity and demonstration. Our problem of hyperparameter selection becomes

$$\boldsymbol{\phi}^* = \underset{\phi}{\mathrm{argmin}} \quad \hat{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}). \tag{4.27}$$

Now by plugging in the classifier (4.25), an explicit form of $\hat{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ can be obtained. Nonetheless, the discontinuity of the sign function will make the minimization a combinatorial problem. A natural solution is to consider approximating the sign function with a sigmoid function. In particular, we consider the $\tanh(\tau x)$ function with parameter τ chosen to be large enough for good approximation[1]. We denote the smoothed empirical generalization cost as $\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ where

$$\begin{aligned}\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}) =& \frac{c_1}{2m_+}\sum_{j\in\mathcal{J}_-}[1-y_j\tanh(\tau h_j)] \\ &+\frac{c_2}{2m_-}\sum_{j\in\mathcal{J}_+}[1-y_j\tanh(\tau h_j)].\end{aligned} \tag{4.28}$$

From the differentiability of h_j and the $\tanh(\tau x)$ function, $\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ is also differentiable in each critical region, and by Proposition 1 it is almost everywhere differentiable.

4.4.2 Gradient Calculation

In this section, we calculate the gradient of $\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ at specific $\boldsymbol{\phi}$, assuming that we have solved the dual problem once and have obtained sets $\mathcal{A}$ and $\mathcal{A}^C$. The partial derivative of

[1]The choice of τ is actually a trade-off. A large value of τ gives a good approximation, but will increase the sensitivity of the gradient and slow down the convergence of the gradient based algorithm.

$\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ with respect to one dimension $\boldsymbol{\phi}_i$ of $\boldsymbol{\phi}$ follows directly from the chain rule.

$$\begin{aligned}\frac{\partial\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})}{\partial\boldsymbol{\phi}_i} = &\frac{c_1}{2m}\sum_{j\in\mathcal{J}_-} -y_j(\tau-\tau\tan^2(\tau h_j))\frac{\partial h_j}{\partial\boldsymbol{\phi}_i}\\ &+\frac{c_2}{2m}\sum_{j\in\mathcal{J}_+} -y_j(\tau-\tau\tan^2(\tau h_j))\frac{\partial h_j}{\partial\boldsymbol{\phi}_i}\end{aligned} \tag{4.29}$$

where for the first two dimensions $\boldsymbol{\theta} = [\gamma,\nu]^T$ of $\boldsymbol{\phi}$,

$$\frac{\partial h_j}{\partial\boldsymbol{\theta}} = (\boldsymbol{C}_{\mathcal{A}}^{\theta})^T\boldsymbol{P}_{\beta}^{-1}\boldsymbol{T}_{\beta}{}^T(\boldsymbol{d}_j-\boldsymbol{d}_{\mathcal{U}}). \tag{4.30}$$

The partial derivative of the hyperplane with respect to β_k is more involved.

$$\begin{aligned}\frac{\partial h_j}{\partial\beta_k} = &\frac{\partial(\boldsymbol{d_j}-\boldsymbol{d}_{\mathcal{U}})^T}{\partial\beta_k}\boldsymbol{T}_{\beta}\boldsymbol{P}_{\beta}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\boldsymbol{\theta}+\boldsymbol{C}_{\mathcal{A}}^{0})\\ &+(\boldsymbol{d_j}-\boldsymbol{d}_{\mathcal{U}})^T\frac{\partial\boldsymbol{T}_{\beta}}{\partial\beta_k}\boldsymbol{P}_{\beta}^{-1}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\theta+\boldsymbol{C}_{\mathcal{A}}^{0})\\ &+(\boldsymbol{d}_j-\boldsymbol{d}_{\mathcal{U}})^T\boldsymbol{T}_{\beta}\frac{\partial\boldsymbol{P}_{\beta}^{-1}}{\partial\beta_k}(\boldsymbol{C}_{\mathcal{A}}^{\theta}\theta+\boldsymbol{C}_{\mathcal{A}}^{0}).\end{aligned} \tag{4.31}$$

The partial derivative of $\dfrac{\partial\boldsymbol{d}_j}{\partial\beta_k}$ is just

$$-\left[y_1\kappa(\boldsymbol{x_1},\boldsymbol{x_j})(x_1^k-x_j^k)^2\cdots y_n\kappa(\boldsymbol{x_n},\boldsymbol{x_j})(x_n^k-x_q^k)^2\right]^T. \tag{4.32}$$

Using the fact that $\frac{\partial\boldsymbol{A}^{-1}}{\partial\beta_k} = -\boldsymbol{A}^{-1}\frac{\partial\boldsymbol{A}}{\partial\beta_k}\boldsymbol{A}^{-1}$, the partial derivatives of $\boldsymbol{T}_{\beta}$ with respect to β_k is

$$\begin{aligned}\frac{\partial\boldsymbol{T}_{\beta}}{\partial\beta_k} = &\frac{1}{\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}}\frac{\partial(\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{C}_{\mathcal{A}}^{\alpha T})}{\partial\beta_k}\\ &-\frac{\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{C}_{\mathcal{A}}^{\alpha T}}{(\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y})^2}\frac{\partial(\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y})}{\partial\beta_k}-\frac{\partial\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{C}_{\mathcal{A}}^{\alpha T}}{\partial\beta_k}\end{aligned} \tag{4.33}$$

with

$$\begin{aligned}\frac{\partial(\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1})}{\partial\beta_k} = &-\boldsymbol{Q}_{\beta}^{-1}\frac{\partial\boldsymbol{Q}_{\beta}}{\partial\beta_k}\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\\ &-\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\frac{\partial\boldsymbol{Q}_{\beta}}{\partial\beta_k}\boldsymbol{Q}_{\beta}^{-1}\end{aligned} \tag{4.34}$$

and

$$\frac{\partial\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}}{\partial\beta_k} = -\boldsymbol{y}^T\boldsymbol{Q}_{\beta}^{-1}\frac{\partial\boldsymbol{Q}_{\beta}}{\partial\beta_k}\boldsymbol{Q}_{\beta}^{-1}\boldsymbol{y}. \tag{4.35}$$

Finally, $\frac{\partial \boldsymbol{Q}_\beta}{\partial \beta_k}$ is given in terms of the Hadamard Product

$$\frac{\partial \boldsymbol{Q}_\beta}{\partial \beta_k} = -\boldsymbol{Q}_\beta \circ \boldsymbol{\Pi}_d \tag{4.36}$$

where $(\boldsymbol{\Pi}_d)_{ij} = (x_i^d - x_j^d)^2 \cdot \mathbf{1}(i \neq j)$ or in matrix form

$$\boldsymbol{\Pi}_d = \begin{bmatrix} 0 & (x_1^d - x_2^d)^2 & \cdots & (x_1^d - x_n^d)^2 \\ (x_2^d - x_1^d)^2 & 0 & \cdots & (x_2^d - x_n^d)^2 \\ \vdots & & & \vdots \\ (x_n^d - x_1^d)^2 & (x_n^d - x_2^d)^2 & \cdots & 0 \end{bmatrix}$$

Similarly, we also have

$$\frac{\partial \boldsymbol{P}_\beta^{-1}}{\partial \beta_k} = -\boldsymbol{P}_\beta^{-1} \frac{\partial \boldsymbol{P}_\beta}{\partial \beta_k} \boldsymbol{P}_\beta^{-1} = -\boldsymbol{P}_\beta^{-1} (\boldsymbol{C}_{\mathcal{A}}^{\alpha})^T \frac{\partial \boldsymbol{T}_\beta}{\partial \beta_k} \boldsymbol{P}_\beta^{-1}. \tag{4.37}$$

By plugging in everything to (4.29), the gradient of $\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ is computed as a vector valued function of hyperparameters. The following property for its smoothness in each critical region is useful for the convergence of the gradient based algorithm.

Proposition 3. *The gradient $\nabla \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi})$ is Lipschitz continuous in each critical region.*

Proof is given in Appendix C.

4.5 Gradient-based Hyperparameter Selection Algorithm

In this section, we propose the gradient descent algorithm for optimal parameter selection using gradients calculated in Section 4.4. We also discuss the choice of step size and some other issues. As a batch stochastic gradient descent method, we update $\boldsymbol{\phi}$ at each iteration as

$$\boldsymbol{\phi}^{t+1} = \boldsymbol{\phi}^t - \eta^t \nabla_{\boldsymbol{\phi}} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t) \tag{4.38}$$

where η^t is the step size at iteration t, and $\nabla_{\boldsymbol{\phi}} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t)$ can be obtained by (4.29). Again, $\tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t)$ is calculated by first constructing the critical region and classifier at $\boldsymbol{\phi}^t$ using $\mathcal{T}$, and then applying it to $\mathcal{Z}$.

The overall gradient descent algorithm is summarized in Algorithm 1. In each iteration, the function *randSample*() randomly assigns the data to either $\mathcal{T}$ or $\mathcal{Z}$ with a predefined ratio. The function *svm2ν*() takes $\mathcal{T}$ and $\boldsymbol{\phi}^t$, and solves (4.3)[2]. The outputs of this function

[2]We used CVX package for quadratic optimization.

Algorithm 1: Gradient Descend Algorithm for Parameter Selection

$t \leftarrow 0,\ r \in [0.2, 0.8]$
$\boldsymbol{\phi}^0 \leftarrow \nu^0, \boldsymbol{s}^0, \boldsymbol{\beta}^0$
while $||\nabla_{\boldsymbol{\phi}} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t)|| \geq \epsilon$ **do**
  $\eta^t \leftarrow initStep(t)$
  $\mathcal{T}, \mathcal{Z} \leftarrow randSample(\mathcal{D})$
  $\boldsymbol{\alpha}^* \leftarrow svm2\nu(\mathcal{T}, \boldsymbol{\phi}^t)$
  $\{\nabla_{\boldsymbol{\phi}} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t), \mathcal{R}^t\} \leftarrow getGrad(\mathcal{Z}, \boldsymbol{\alpha}^*)$
  $\tilde{\boldsymbol{\phi}}^{t+1} = \boldsymbol{\phi}^t - \eta^t \nabla_{\boldsymbol{\phi}} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t)$
  while $\tilde{\Psi}(\tilde{\boldsymbol{\phi}}^{t+1}) - \tilde{\Psi}(\boldsymbol{\phi}^t) > \epsilon'$ **do**
    $\eta^t = r\eta^t$
  end while
  $\boldsymbol{\phi}^{t+1} = \boldsymbol{\phi}^t - \eta^t \nabla_{\boldsymbol{\phi}^t} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t)$
  $t \leftarrow t + 1;$
end while
return $\boldsymbol{\phi}^t;$

comprises of the solution $\boldsymbol{\alpha}^*$. Using these outputs, the $getGrad()$ function first constructs the explicit classifier and critical region containing $\boldsymbol{\phi}^t$, and then calculates $\nabla_{\boldsymbol{\phi}} \tilde{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}^t)$.

The choice of step size η^t is very important for any gradient based method. Following a similar argument as in [44][3] , it can be shown that if we choose a step size with appropriate decreasing rate, such as $\sum_t (\eta^t)^2 \leq \infty$ and $\sum_t \eta^t = \infty$, the stochastic gradient method is guaranteed to converge at least to a local minima. We also incorporate backtrack line search for better convergence. In each iteration, the function $initStep(t)$ provides a decaying initial step size. A line search is conducted along the gradient to find better update, i.e. if an improvement condition is not satisfied, we further decrease η^t with ratio $r \in [0.2, 0.8]$. [45] If $\tilde{\boldsymbol{\phi}}^{t+1}$ is in the current critical region, Theorem 2 can be exploited to compute $\tilde{\Psi}(\tilde{\boldsymbol{\phi}}^{t+1})$. Otherwise, we invoke $svm2nu()$ again and proceed.

By resampling at each iteration, we avoid local minima arisen from fixed training and validation partition. Also note that the computation of the explicit solution involves the inversion of matrix $\boldsymbol{P}_{\boldsymbol{\beta}}$. The size of this matrix equals to the number of training samples and seems to be computationally expensive. However, by Theorem 1, $\boldsymbol{P}_{\boldsymbol{\beta}}$ is symmetric negative definite and many matrix decomposition techniques can be used for fast inversion.

4.6 Experiments

Two datasets were used to test the validity of our method. The first dataset, S500, has 1500 samples and 500 features, consisting of reflected intensities from semiconductor

[3] Constructing Lyapunov process and using Prop. 3

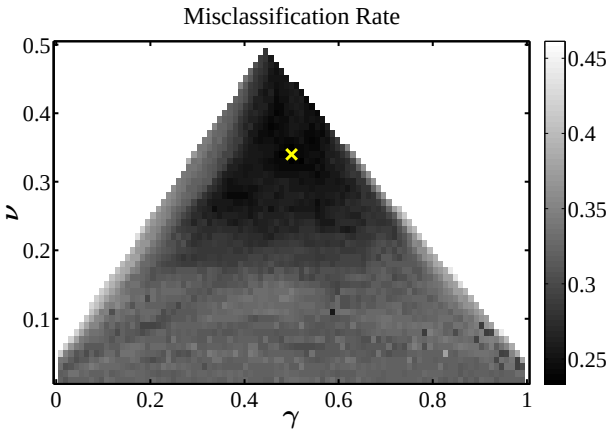
gratings. The second dataset, W11, has 1359 samples and 11 features, consisting of wine tasting quality with chemical characteristics of the wine as features[4]. In section 4.6.1, we visualize the region construction and explicit solution of $\tilde{\Psi}(\boldsymbol{\phi})$ based on Theorem 2. For visualization purposes, we use a RBF kernel and fix the value of β to a single value and plot the (γ, ν) space. In Section 4.6.2, we provide some examples of the gradient descent algorithm discussed in Section 4.5. In addition to showing the convergence results, we list the optimal choice of hyperparameters and the corresponding cost for various asymmetric cases and also demonstrate differences in performance for the RBF and ARD kernel.

4.6.1 Visualization of Critical Regions and Explicit Solutions

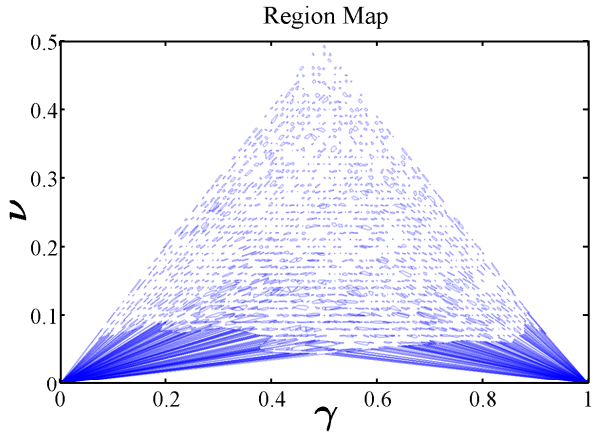
For visualization purposes, we use a RBF kernel with a single kernel parameter. Figures 4.2 and 4.3 show the critical regions and explicit solutions $\tilde{\Psi}(\boldsymbol{\phi})$ for W11 and S500, with kernel parameter $\beta = 1$ and $\beta = 10$, respectively. All results are based on a balanced cost, i.e. $c_1 = c_2 = 1$. For comparison purposes, we also add the exhaustive grid calculation[5] of the estimated generalization cost in Figures 4.2a and 4.3a.

Figures 4.2b and 4.3b illustrate the polyhedral critical regions derived in Theorem 2. We observe that the critical regions exhibit lots of interesting patterns depending on the values of γ and ν. For instance, the area of the critical region becomes larger as we decrease the value of ν, and increase the value of β. In addition, Figures 4.2c and 4.3c show the calculated $\tilde{\Psi}(\boldsymbol{\phi})$ for a fixed training and validation set using the fact that $\boldsymbol{\alpha}^*(\boldsymbol{\theta})$ is an affine function of $\boldsymbol{\theta}$ in each critical region. We observe that the explicit solution for the generalization error matches very well to that of the exhaustive one.

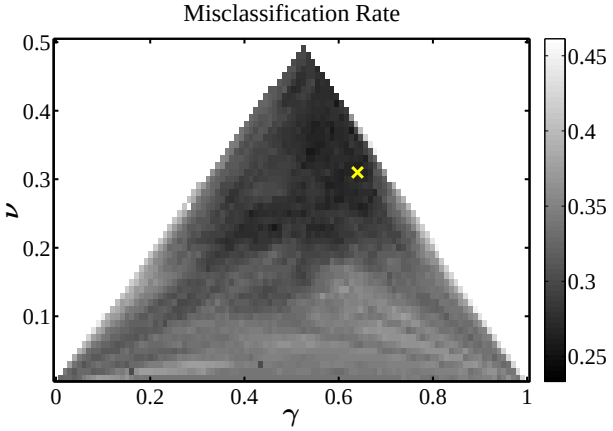
We now discuss the effect of the parameters on region size. For a given critical region, the partition $\{\mathcal{S}_0, \mathcal{S}_b, \mathcal{S}_{ub}\}$ is invariant according to Theorem 2. Because α_i^*, $i \in \mathcal{S}_b \cup \mathcal{S}_0$ is fixed (α_i^* is 0 or $\frac{\gamma}{n}$ or $\frac{1-\gamma}{n}$), only the numerical values of unbounded support vectors' α_i^* change in each critical region. Intuitively, this means that for a region with a large number of unbounded support vectors $|\mathcal{S}_{ub}|$, we have more degrees of freedom and hence a larger area that satisfies the partition. As ν decreases, the classifier tends to overfit and increase the number of unbounded support vectors $|\mathcal{S}_{ub}|$. The effect of β_k for RBF kernels can be viewed very similarly. In fact, the kernel operator can be viewed as a similarity measure. As discussed in [35], when β_k is large, the dissimilarity between samples are high, and overfitting will happen. On the other hand, when β_k is small, the similarity between samples is high, and underfitting will happen. Thus the region pattern reflects the fact that, a small (large) value of ν or a large (small) value of β lead to overfitting (underfitting) and more (less) unbounded support vectors.



(a)

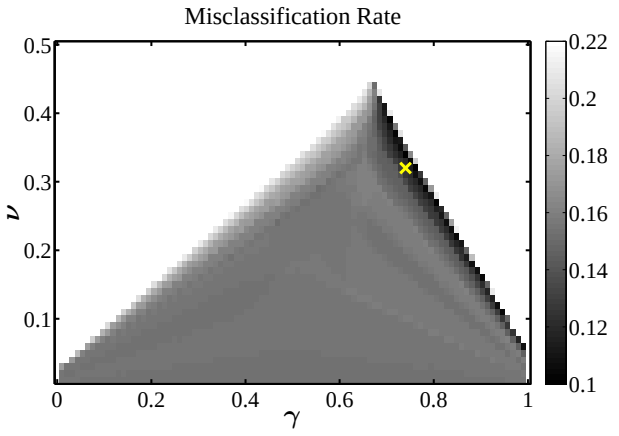


(b)

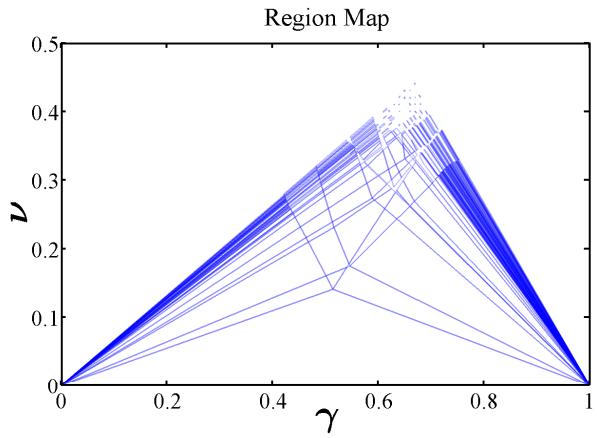


(c)

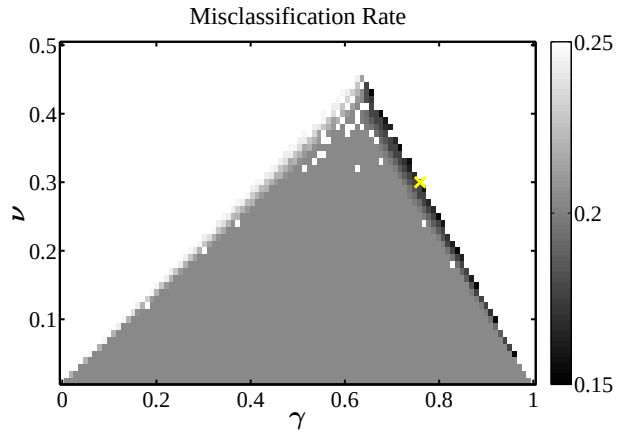
Figure 4.2: W11 $c_1 = c_2 = 1$. (a) Exhaustive 2D calculation of $\hat{\Psi}$ with optimal point marked. (b) Highly fragmented critical regions due to a smaller value of β. (c) Calculated $\tilde{\Psi}$ using Theorem 2.



(a)



(b)



(c)

Figure 4.3: S500 $c_1 = c_2 = 1$. (a) Exhaustive 2D calculation of $\hat{\Psi}$ with optimal point marked. (b) Simple critical region structure due to a larger value of β. (c) Calculated $\tilde{\Psi}$ using Theorem 2.

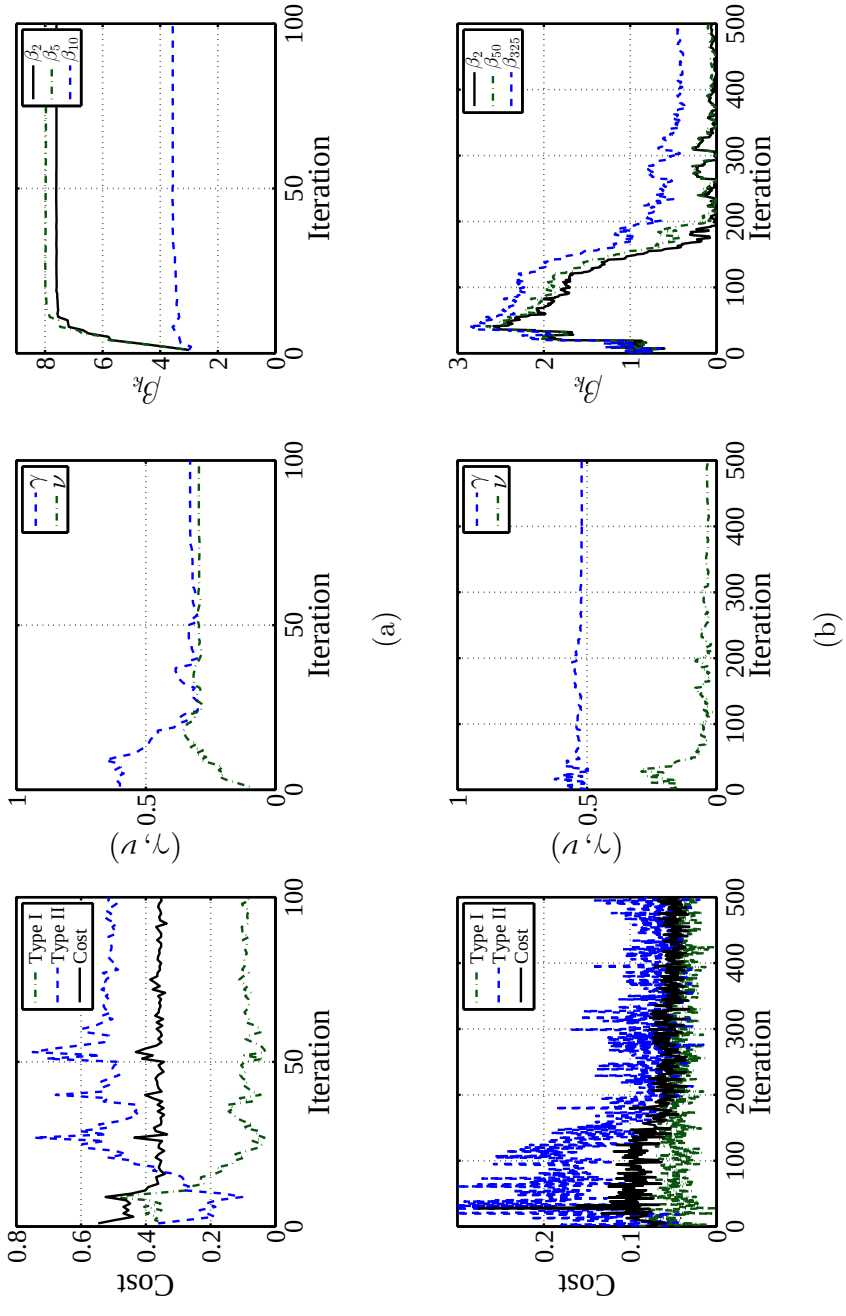


Figure 4.4: (a) Convergence result for W11, $c_1 = 2, c_2 = 1$. (b) Convergence result for S500, $c_1 = 1, c_2 = 1$.

4.6.2 Gradient Descent Algorithm Results

We now demonstrate the effectiveness of Algorithm 1 for hyperparameter selection of the 2ν-SVM with an ARD-Gaussian kernel.

Figures 4.4a and 4.4b demonstrate the convergence for two examples with the ARD kernel. Figure 4.4a shows the approximation generalization cost and the hyperparameters in each iteration for W11. The costs were $c_1 = 2, c_2 = 1$, and the initial values were $\gamma^0 = 0.6, \nu^0 = 0.1, \beta_k^0 = 3\ \forall k$. To test for robustness, we deliberately set $\gamma^0 = 0.6$, while we expect the optimal value of γ^* to be below 0.5 for $c_1 = 2, c_2 = 1$. Even with a bad initial guess for γ^0 and β_k^0, our algorithm converges to a minimum within 60 steps. Figure 4.4b shows similar results for S500. The costs were $c_1 = c_2 = 1$, and the initial values were $\gamma^0 = 0.5, \nu^0 = 0.15, \beta_k^0 = 1\ \forall k$. We set a reasonable initial value of $\gamma^0 = 0.5$, since $c_1 = c_2 = 1$. The algorithm converges within 350 steps, but note that the number of iterations required to converge is significantly small, considering we have 502 hyperparameters to choose from. The costs in both cases exhibit fluctuations because we incorporate resampling of training and validation sets in the algorithm.

In Tables 4.1-4.5, we list the optimal cost, error rates, hyperparameters, and time/steps to convergence for different combinations of c_1, c_2 using the gradient algorithm with the ARD Gaussian kernel (grad-MPK) and the RBF kernel (grad-SPK) for various datasets, including W11 and S500. For comparison purposes, we also list the optimal parameters for the RBF case found from an exhaustive search (ES-SPK)[6] in the (γ, ν, β) space. Note that optimal values for β_k^* in the grad-MPK case are omitted due to space limitation. For S500, a dimension reduced version was used to construct the table.

We discuss some interesting observations from Table 4.1. As seen in the rows labeled ES-SPK and grad-SPK, the optimal hyperparameters γ^*, ν^* found by Algorithm 1 match relatively well with those in the exhaustive search, and the final γ^* reflects the cost variations. Note that "at boundary" means the algorithm converged to the boundary of the feasible set due to the cost of one error type being too high. For β^*, the gradient algorithm converges to a local minima and is much different from the results of the exhaustive search. However, the cost is not significantly compromised, and in some cases, our algorithm even achieves a lower cost. In addition, we have observed that the optimal point is not necessarily at a single point of β. Rather, a range of β values have similar values of γ^* and ν^*, leading to multiple minima. This can be further verified by the fact that the gradient of the cost with respect to β is relatively small in this range. Finally, as expected, the grad-MPK performs best for all combinations of costs with reasonable computation time.

Similar trends are observed for Tables 4.2-4.5. In contrast to W11, S500 is relatively separable, resulting in lower error rates. Since the number of errors is significantly smaller

[4]https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/

[5]Grid size is set to 0.01. Approximately 600 training and validation samples.

[6]Parallel processing with 4 cores. Grid size was 0.025 for $\gamma, \nu \in (0, 1)$, and 7 values of β, between 10^{-2} and 10^3, were used. For both data sets, approximately 400 training and validation samples were used for exhaustive search and the gradient descent algorithm.

than W11, the variance for the estimate of γ^* was quite large. Thus, we see discrepancies for γ^* between the algorithm and the exhaustive search, which also lead to higher generalization cost for the grad-SPK, and grad-MPK results. However, we also want to point out that there is less use for the algorithm if the data set is initially well-separated. One final aspect of these results is that grad-MPK does not always outperform grad-SPK. This may be due to random error from subset sampling of the data, but may also be due to overfitting. Similar observations are discussed in [39]. Thus, one should be cautious before using kernel functions with many hyperparameters.

Finally, Figures 4.5a and 4.5b show final β_k^* values for both datasets with $c_1 = c_2 = 1$. The black dotted line is the converged β^* value for the RBF kernel, and the blue lines represent the converged β_k^* values for each dimension in the ARD-Gaussian kernel. Optimal scaling of the data is significantly different for each dimension, as seen by the different β_k^* values in the ARD-Gaussian kernel. Moreover, Figure 4.5b shows a periodic pattern in the scaling, which coincides with what we observe in the data.

In summary, we produced a rigorous alternative to the classic exhaustive search method for solving the hyperparameter selection problem for any general SVM. In one of our sample problems, a single parameter basis function formulation was used to produce essentially identical results to the exhaustive search, at approximately 1/50 of the computational cost. Also, a fully parameterized kernel formulation was used to produce consistently better results than the exhaustive search (by 20-30%), and did so at approximately 1/5 of the computational cost. This not only justifies the benefit of feature rescaling by the ARD-Gaussian kernel, but also verifies that our algorithm can be used to effectively choose the optimal configuration in a high-dimensional hyperparameter space.

4.7 Summary

We started out with the question of how to tune the hyperparameters of the SVM such that it minimizes an asymmetric generalization cost. In this work, we explored beyond this question and proposed a new method for optimizing parameter selection for general SVMs with additional hyperparameters, specifically demonstrating on the 2ν-SVM with the multiparametric ARD-Gaussian kernel. Through a parametric optimization framework, we showed that the solution of the dual problem was a piecewise explicit function of the hyperparameters. This allowed us to derive the explicit form of the generalization cost and construct a gradient descent algorithm for optimal parameter selection. Two datasets were used to verify the algorithm, and results showed that our method successfully converged to a parameter set with low cost. Although we chose to demonstrate our method on the 2ν-SVM with the ARD-Gaussian kernel, our method can be extended to any soft-margin SVMs with multiple hyperparameters.

The contributions are many-fold. Theoretically, we solved a degenerate parametric program, which is considered to be a challenging problem in the operational research community, by identifying conditions that lead to a closed-form solution. In this process, we derived the

Table 4.1: Comparison using RBF/ARD kernel, W11, starting points: 0.5,0.1,3

c_1,c_2	Method	Cost	False Alarm	Missed Alarm	γ^*	ν^*	β^*	Time (min)	Steps
1,2.5	grad-MPK	0.319	0.521	0.055	0.752	0.212	*	45.26	89
	grad-SPK	0.372	0.524	0.095	0.709	0.290	2.43	8.071	36 (at boundary)
	ES-SPK	0.365	0.680	0.0413	0.725	0.260	3.162	248.18	*
1,2	grad-MPK	0.306	0.416	0.101	0.655	0.265	*	30.59	61
	grad-SPK	0.346	0.443	0.129	0.646	0.273	1.856	6.930	56
	ES-SPK	0.333	0.501	0.0982	0.675	0.260	0.1	248.18	*
1,1.5	grad-MPK	0.284	0.232	0.225	0.523	0.385	*	45.74	91
	grad-SPK	0.303	0.361	0.165	0.592	0.291	2.13	8.049	39
	ES-SPK	0.296	0.315	0.186	0.600	0.310	0.316	248.18	*
1,1	grad-MPK	0.213	0.209	0.212	0.507	0.307	*	44.23	92
	grad-SPK	0.251	0.273	0.233	0.524	0.304	1.54	8.4091	41
	ES-SPK	0.242	0.221	0.260	0.500	0.335	0.0316	248.18	*
1.5,1	grad-MPK	0.250	0.135	0.294	0.439	0.305	*	21.22	43
	grad-SPK	0.292	0.145	0.357	0.407	0.311	2.244	12.81	62
	ES-SPK	0.289	0.178	0.313	0.425	0.285	0.0316	248.18	*
2,1	grad-MPK	0.283	0.089	0.384	0.374	0.329	*	34.38	67
	grad-SPK	0.308	0.085	0.431	0.343	0.269	1.687	8.965	46
	ES-SPK	0.320	0.138	0.359	0.375	0.260	0.1	248.18	*
2.5,1	grad-MPK	0.299	0.072	0.406	0.276	0.237	*	36.68	77
	grad-SPK	0.348	0.094	0.450	0.332	0.279	0.267	2.981	15 (at boundary)
	ES-SPK	0.350	0.102	0.432	0.400	0.335	0.316	248.18	*

grad-MPK: Gradient descent algorithm with multi-parameter ARD kernel SVM.
grad-SPK: Gradient descent algorithm with single-parameter RBF kernel SVM.
ES-SPK: Exhaustive search for single-parameter RBF kernel SVM.

Table 4.2: Comparison using RBF/ARD kernel, S500, starting points: 0.5,0.1,3

c_1,c_2	Method	Cost	False Alarm	Missed Alarm	γ^*	ν^*	β^*	Time (min)	Steps
1,2.5	grad-MPK	0.0664	0.0492	0.0345	0.6492	0.0642	*	37.18	62
	grad-SPK	0.0646	0.0579	0.0294	0.6036	0.0881	0.0813	8.37	33 (at boundary)
	ES-SPK	0.0467	0.0378	0.0251	0.800	0.035	0.01	305.86	*
1,2	grad-MPK	0.0422	0.0368	0.0265	0.6609	0.1133	*	85.46	137
	grad-SPK	0.0641	0.0513	0.0402	0.6315	0.0657	0.0899	14.16	52
	ES-SPK	0.0467	0.0378	0.0251	0.800	0.035	0.01	305.86	*
1,1.5	grad-MPK	0.0556	0.0346	0.0621	0.5218	0.1335	*	39.92	44
	grad-SPK	0.0573	0.0374	0.0558	0.5641	0.0751	0.591	12.78	54
	ES-SPK	0.0467	0.0378	0.0251	0.800	0.035	0.01	305.86	*
1,1	grad-MPK	0.0486	0.0441	0.0570	0.6133	0.0775	*	37.88	66
	grad-SPK	0.0462	0.0369	0.0589	0.5456	0.0689	0.1561	18.85	78
	ES-SPK	0.0329	0.0313	0.0362	0.525	0.06	0.0316	305.86	*
1.5,1	grad-MPK	0.0526	0.0284	0.0660	0.4558	0.0758	*	45.26	79
	grad-SPK	0.0531	0.0283	0.0665	0.4679	0.0977	0.0679	13.06	52
	ES-SPK	0.0376	0.0116	0.0771	0.400	0.06	0.0316	305.86	*
2,1	grad-MPK	0.0608	0.0256	0.0805	0.4149	0.0783	*	26.49	46
	grad-SPK	0.0591	0.0142	0.1113	0.3352	0.0726	0.1487	7.18	30
	ES-SPK	0.0431	0.0101	0.0865	0.300	0.06	0.0316	305.86	*
2.5,1	grad-MPK	0.0510	0.0099	0.1027	0.3791	0.1198	*	47.81	78
	grad-SPK	0.0679	0.0085	0.1317	0.1727	0.0403	0.1208	10.52	42 (at boundary)
	ES-SPK	0.0456	0.0048	0.1062	0.275	0.06	0.0316	305.86	*

grad-MPK: Gradient descent algorithm with multi-parameter ARD kernel SVM.
grad-SPK: Gradient descent algorithm with single-parameter RBF kernel SVM.
ES-SPK: Exhaustive search for single-parameter RBF kernel SVM.

Table 4.3: Comparison using RBF/ARD kernel, yeast, starting points: 0.5,0.1,3

c_1,c_2	Method	Cost	False Alarm	Missed Alarm	γ^*	ν^*	β^*	Time (min)	Steps
1,2	grad-MPK	0.164	0.275	0.0434	0.585	0.189	*	133.82	44
	grad-SPK	0.184	0.219	0.0805	0.492	0.196	10.0	26.83	23
	ES-SPK	0.178	0.271	0.0571	0.5	0.185	10	2182	*
1,1	grad-MPK	0.129	0.205	0.0778	0.463	0.1749	*	97.92	26
	grad-SPK	0.138	0.198	0.0990	0.435	0.188	10.11	45.35	40
	ES-SPK	0.134	0.192	0.0945	0.375	0.21	10	2182	*
2,1	grad-MPK	0.219	0.13	0.193	0.284	0.149	*	281.35	102
	grad-SPK	0.227	0.0851	0.249	0.383	0.172	9.99	108.31	97
	ES-SPK	0.213	192	0.0945	0.375	0.21	10	2182	*

grad-MPK: Gradient descent algorithm with multi-parameter ARD kernel SVM.
grad-SPK: Gradient descent algorithm with single-parameter RBF kernel SVM.
ES-SPK: Exhaustive search for single-parameter RBF kernel SVM.

Table 4.4: Comparison using RBF/ARD kernel, wdbc, starting points: 0.5,0.1,3

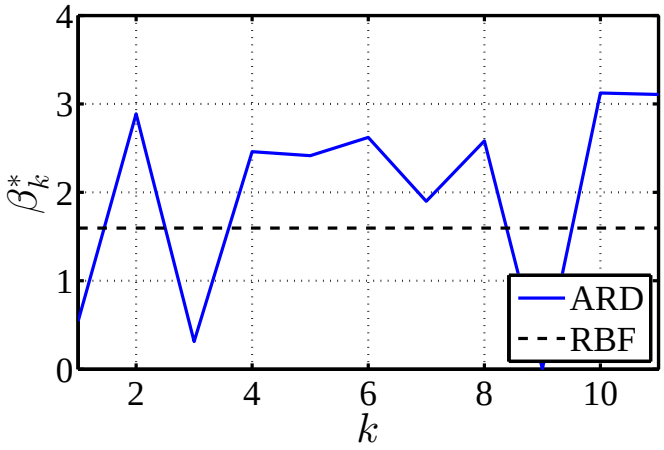
c_1,c_2	Method	Cost	False Alarm	Missed Alarm	γ^*	ν^*	β^*	Time (min)	Steps
1,2	grad-MPK	0.0176	0.0145	0.0112	0.536	0.061	*	12.95	37
	grad-SPK	0.0220	0.021	0.012	0.611	0.0861	0.0495	4.75	32
	ES-SPK	0.0162	0.0072	0.0154	0.575	0.11	0.01	276.5	*
1,1	grad-MPK	0.0132	0	0.0345	0.499	0.0833	*	14.33	43
	grad-SPK	0.0176	0.0147	0.022	0.49	0.0732	1.823	6.06	41
	ES-SPK	0.0073	0	0.0217	0.500	0.11	1.00	276.5	*
2,1	grad-MPK	0.0264	0	0.0645	0.483	0.136	*	17.61	51
	grad-SPK	0.0199	0	0.0561	0.477	0.0877	0.797	5.03	34
	ES-SPK	0.0147	0	0.0217	0.5	0.11	1	276.5	*

grad-MPK: Gradient descent algorithm with multi-parameter ARD kernel SVM.
grad-SPK: Gradient descent algorithm with single-parameter RBF kernel SVM.
ES-SPK: Exhaustive search for single-parameter RBF kernel SVM.

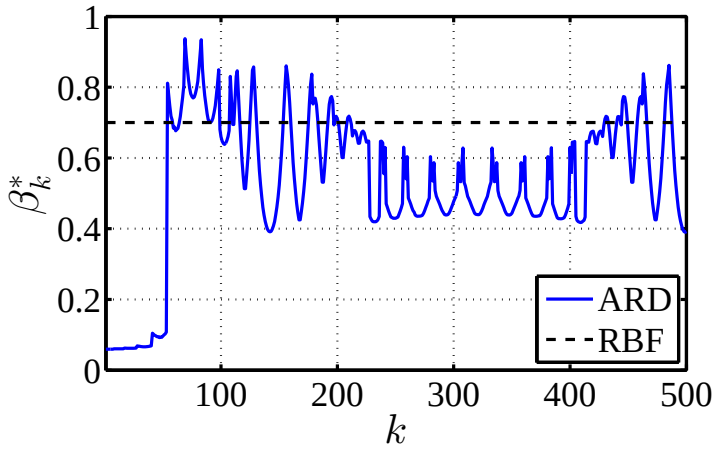
Table 4.5: Comparison using RBF/ARD kernel, waveform, starting points: 0.5,0.1,3

c_1,c_2	Method	Cost	False Alarm	Missed Alarm	γ^*	ν^*	β^*	Time (min)	Steps
1,2	grad-MPK	0.0894	0.13	0.0246	0.6892	0.1058	*	67.8	113
	grad-SPK	0.090	0.1483	0.0163	0.692	0.105	0.266	35.86	88
	ES-SPK	0.0912	0.139	0.0214	0.725	0.085	0.010	4491	*
1,1	grad-MPK	0.0753	0.112	0.0379	0.580	0.1003	*	298.54	506
	grad-SPK	0.0729	0.102	0.0432	0.555	0.110	0.209	66.15	164
	ES-SPK	0.0729	0.0899	0.0561	0.550	0.135	0.0316	4491	*
2,1	grad-MPK	0.0947	0.0296	0.130	0.348	0.174	*	76.37	128
	grad-SPK	0.105	0.0377	0.134	0.294	0.120	0.297	159.05	396
	ES-SPK	0.101	0.0511	0.1002	0.40	0.135	0.0316	4491	*

grad-MPK: Gradient descent algorithm with multi-parameter ARD kernel SVM.
grad-SPK: Gradient descent algorithm with single-parameter RBF kernel SVM.
ES-SPK: Exhaustive search for single-parameter RBF kernel SVM.



(a)



(b)

Figure 4.5: Convergence for β_k^* for datasets (a) W11 (b) S500. The costs are set to $c_1 = c_2 = 1$. The black dashed line represent the converged value of β^* for the RBF kernel, and the blue lines represent converged values of β_k^* for the ARD-Gaussian kernel.

explicit forms for the dual solution, the classifier, the generalization cost, and its gradient, which had nice continuity and differentiability properties. Our solution provides many insights into the characteristics of the soft-margin SVM. Specifically, Theorem 2 describes the overall picture of the dual solution and makes it possible to analyze the effect of the number of support vectors of the classifier on the area of the critical region.

Practically, we turned an intractable high-dimensional parameter search problem into a scalable non-linear minimization problem, which can be solved by gradient descent algorithms. Hence, this allows optimal usage of the cost-sensitive SVM and/or the flexible ARD-Gaussian kernel, which is important in many applications.

A lot of work remains to be done in the future. To further improve the efficiency of the algorithm, matrix completion techniques can be applied to problems with a large number of training data. In addition, high-order gradient-based methods can be explored for enhanced convergence. Finally, since an explicit form of the generalization cost has been obtained, we can regularize the usage of features by adding appropriate regularization penalties for $\boldsymbol{\beta}$ in the cost minimization.

Chapter 5

Conclusions and Outlook

5.1 Conclusions

In this dissertation we discussed the importance of inference models in the context of advanced process control fault detection. Such models are beneficial for various reasons. As mentioned in Chapter 1, Bayesian inference frameworks are used to overcome the physical limitations of current tools and processes by integrating information coming from various sources (previous measurements, tool sensors, spatial profiles, etc.). Moreover, as the number of complex processing steps gets significantly large, data inference models provide a way for engineers to quickly model processes using real-time sensor data, especially since developing explicit physical models for each step is time-consuming.

This work explores the application and possible issues of inference models on in-line metrology and fault detection. In Chapter 2, we looked at virtual metrology, which uses inference models to estimate measurements from tool sensor signals, for fault detection purposes. Specifically, we were interested in the question of whether the introduction of such inference models actually benefit the fab. Two different metrology sampling scenarios were explored, one with and one without process correction. We also simulated three types of VM model candidates for fault detection that resulted in different alarm patterns, and thus different cost consequences for the fab.

In Chapters 3 and 4, we explored support vector machines as an inference model for fault detection and classification. In Chapter 3, SVMs were used to construct a real-time inspection tool for erroneous grating detection to demonstrate the use of data models. In contrast to virtual metrology, the tool directly used the diffraction signatures from the grating in test, without any profile measurement reconstruction. To demonstrate the feasibility of such models, the inspection tool was demonstrated without alterations to the current YieldStar scatterometry hardware setup. Results showed that by preserving the dimensionality of the data, SVM classifiers outperformed the corresponding scatterometer on detecting out-of-spec grating detection, with almost instantaneous time.

In Chapter 4, we rediscovered the problem of choosing optimal combinations of false and

missed alarm rates. Specifically, in the context of the 2ν-SVM, we demonstrated a gradient-descent algorithm that not only gave the optimal hyperparameter set for asymmetric cost minimization, but also for any additional parameters in the SVM that were introduced for accuracy and data scaling improvements. Not only did the algorithm significantly improve hyperparameter search time compared to exhaustive search methods (two orders of magnitude for our example), but also allowed us to turn infeasible search problems in a high-dimensional hyperparameter space into a feasible one. It successfully selected combinations which resulted in minimal generalization cost.

5.2 Outlook

Semiconductor manufacturing currently faces many challenges. Multi-patterning will continue due to the delay in EUV development, leading to resolution and alignment issues. The advent of 3D devices, such as the FinFET and 3D-NAND, make processing even more difficult as many steps require high aspect-ratio etching and full profile information. As process windows continue to shrink, tighter control of the manufacturing process is required through real-time process monitoring and control. In this context, data mining and machine learning techniques will have a larger role in the semiconductor manufacturing industry due to the increased need for fast empirical modeling.

Constantly monitoring a significant number of process steps for tighter control will require both direct measurements and indirect monitoring through the utilization of sensor signals, measurement data from previous steps, and other relevant information. "Big data" approaches to manufacturing will become essential, as a substantial amount of unorganized, heterogeneous data will have to be analyzed real-time. This can include continuous variables, such as temperature, pressure, and optical signatures, or categorical variables, such as wafer and tool labels. As a consequence, preprocessing the data into a structured, usable form will become critical for successful model deployment. Moreover, filtering out useful features from such high-dimensional data will be a challenge. For instance, the stream from many tool sensors are "noise" variables that are irrelevant to the process model in question. Feature selection and reduction methods from machine learning can be useful for tackling such a problem.

Another aspect that will result in more empirical data modeling is the consolidation of the manufacturing industry into huge fabs backed by few major companies. Currently, there exist a handful of very large facilities that handle chip manufacturing for multiple design houses. For design-manufacturing optimization, recipes and tool setting have to be tweaked for each new design. Explicit modeling for each design is extremely time-consuming and process engineers may turn to empirical models for their quick setup time and flexibility.

5.3 Broader Implications

The concepts presented in this thesis are not only applicable to the semiconductor manufacturing industry, but can also be generalized to other contexts. The model selection algorithm can easily be extended to any SVM classification problem that needs flexible tuning of the weights of samples and features. For example, SVMs are now being explored in medical diagnosis, where the cost of a missed alarm (or missed diagnosis) is severe. In biocomputation, researchers deal with genomic microarray data, which has a significantly higher number of features than samples; the feature weighing done through the ARD-Gaussian kernel can be advantageous. In addition, an overarching theme throughout this work was the tradeoff between using costly, time-consuming but accurate measurements and cheaper but less accurate sensor signals. This tradeoff happens to occur in many other areas. For instance, in smart buildings, researchers often need to estimate the number of people in a given room (occupancy detection), since this has an effect on energy-expensive heating, ventilating, and air-conditioning (HVAC) and lighting systems. Although cameras can be used for this purpose, installing the hardware in a room is costly, let alone the devices themselves. An alternative way is to estimate such quantities by using data from CO_2 sensors, WiFi signals, and passive infrared sensors (PIR). Eventually, the problem is to indirectly estimate meta properties from sensor signals and balance the use of costly equipment with inference models, which was explored throughout this thesis.

Bibliography

[1] J. Moyne, E. D. C., and A. M. Hurwitz. *Run-to-Run Control in Semiconductor Manufacturing*. first. Boca Raton, FL: CRC Press, 2001.

[2] C. J. Spanos and G. J. May. *Fundamentals of Semicondcutor Manufacturing and Process Control*. first. Hoboken, NJ: Wiley-IEEE Press, 2006.

[3] C. J. Spanos. "Statistical Process Control in Semiconductor Manufacturing". In: *Microelectronic Engineering* 10 (Feb. 1991), pp. 271–276.

[4] Emanuel Sachs, Albert Hu, and Armann Ingolfsson. "Run by Run Process Control: Combining SPC and Feedback Control". In: *IEEE Trans. Semicond. Manuf.* 8.1 (Feb. 1995), pp. 26–43.

[5] F. T. Cheng, H. C. Huang, and C. A. Kao. "Dual-Phase Virtual Metrology Scheme". In: *IEEE Trans. Semicond. Manuf.* 20.4 (Nov. 2007), pp. 566–571.

[6] A. A. Khan and J. R. Moyne. "An Approach for Factory-Wide Control Utilizing Virtual Metrology". In: *IEEE Trans. Semicond. Manuf.* 20.4 (Nov. 2007), pp. 364–375.

[7] A. A. Khan, J. R. Moyne, and D. M. Tilbury. "Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares". In: *Journal of Process Control* 18 (Dec. 2008), pp. 961–974.

[8] Y. C. Su et al. "Accuracy and Real-Time Considerations for Implementing Various Virtual Metrology Algorithms". In: *IEEE Trans. Semicond. Manuf.* 21.3 (Aug. 2008).

[9] J. C. Y. Cheng and F. T. Cheng. "Application Development of Virtual Metrology in Semiconductor Industry". In: *Industrial Electronics Society, 2005. 31st Annual Conference of IEEE*. 2005, pp. 124–129.

[10] P. H. Chen et al. "Virtual Metrology: A Solution for Wafer to Wafer Advanced Process Control". In: *Semiconductor Manufacturing Conference Proceedings, 2005. IEEE International Symposium on*. Sept. 2005, pp. 155–157.

[11] An-Jhih Su et al. "Control relevant issues in semiconductor manufacturing: Overview with some new results". In: *Control Eng. Pract.* 15 (2007), pp. 1268–1279.

[12] Y. J. Chang et al. "Virtual Metrology Technique for Semiconductor Manufacturing". In: *Neural Networks, The 2006 International Joint Conference on*. July 2006, pp. 5289–5293.

[13] Xinhui Niu et al. "Specular spectroscopic scatterometry". In: *Semiconductor Manufacturing, IEEE Transactions on* 14.2 (2001), pp. 97–111.

[14] Christopher J. Raymond et al. "Multiparameter grating metrology using optical scatterometry". In: *Journal of Vacuum Science Technology B: Microelectronics and Nanometer Structures* 15.2 (1997), pp. 361–368.

[15] Xinhui Niu et al. "Specular spectroscopic scatterometry in DUV lithography". In: *Proc. SPIE*. Vol. 3677. 1999, pp. 159–168.

[16] Hock-Chun Chin et al. "Metrology solutions using optical scatterometry for advanced CMOS: III-V and Germanium multi-gate field-effect transistors". In: *Proc. SPIE*. Vol. 8788. 2013, 87881R.

[17] C. J. Raymond. "Scatterometry for Semiconductor Metrology". In: *Handbook of Silicon Semiconductor Metrology*. Ed. by A. C. Diebold. CRC Press, 2001.

[18] Junwei Bao. "An Optical Metrology System for Lithography Process Monitoring and Control". PhD thesis. U.C. Berkeley, 2003.

[19] G. A. Al-Jumaily. In: *Optical Metrology*. Ed. by G. A. Al-Jumaily. 1st ed. Critical Reviews of Optical Science and Technology Series, SPIE Press, Jan. 1999.

[20] John A. Woollam et al. "Variable angle spectroscopic ellipsometry in the vacuum ultraviolet". In: *Optical Metrology Roadmap for the Semiconductor, Optical, and Data Storage Industries*. Vol. 4099. Proc. SPIE. Nov. 2000, pp. 197–205.

[21] A. Vaid et al. "A holistic metrology approach: hybrid metrology utilizing scatterometry, CD-AFM, and CD-SEM". In: vol. 7971. Proc. SPIE. 2011, pp. 797103–1–20.

[22] N. Zhang et al. "Improving optical measurement uncertainty with combined multitool metrology using a Bayesian approach". In: *Applied Optics* 51.25 (2012), pp. 6196–6206.

[23] D. Z. and C. J. Spanos. "Virtual Metrology Modeling for Plasma Etch Operations". In: *IEEE Trans. Semicond. Manuf.* 22.4 (Nov. 2009), pp. 419–431.

[24] P. Kang et al. "A virtual metrology system for semiconductor manufacturing". In: *Expert Syst. Appl.* 36 (2009), pp. 12554–12561.

[25] S. Lynn et al. "Virtual Metrology for Plasma Etch using Tool Variables". In: *Advanced Semiconductor Manufacturing Conference, 2009. IEEE/SEMI*. 2009, pp. 143–148.

[26] M. H. Hung et al. "A Novel Virtual Metrology Scheme for Predicting CVD Thickness in Semiconductor Manufacturing". In: *Mechatronics, IEEE/ASME Transactions on* 12.3 (2007), pp. 308–312.

[27] Y. C. Su et al. "Implementation Considerations of Various Virtual Metrology Algorithms". In: *IEEE Trans. Autom. Sci. Eng.* Sept. 2007, pp. 276–281.

[28] J. Y. Baek and C. J. Spanos. "Optimization of Blended Virtual and Actual Metrology Schemes". In: *Proc. of SPIE*. Vol. 8324. Apr. 2012, 83241K–83241K–9.

[29] J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods*. first. Oxford University, 2001.

[30] Jan Mulkens et al. "Across Scanner Platform Optimization to enable EUV Lithography at the 10-nm Logic Node". In: vol. 9048. Proc. SPIE. 2014, p. 90481L.

[31] Jos Benschop et al. In: vol. 8683. Proc. SPIE. 2013, 86830P.

[32] Marlene Strobl et al. "Integrated ADI optical metrology solution for lithography process control of CD and OV". In: vol. 9050. Proc. SPIE. 2014, 90501J.

[33] J. Foucher et al. "Hyrbrid Metrology for Critical Dimension based on Scanning Methods for IC Manufacturing". In: vol. 8378. Proc. SPIE. 2012, 83780F–1–8.

[34] Christopher J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Min. Knowl. Discov.* 2.2 (June 1998), pp. 121–167.

[35] Qun Chang, Qingcai Chen, and Xiaolong Wang. "Scaling Gaussian RBF kernel width to improve SVM classification". In: *Neural Networks and Brain, 2005. ICNN B '05. International Conference on*. Vol. 1. 2005, pp. 19–22.

[36] Yi-Wei Chen and Chih-Jen Lin. "Combining SVMs with Various Feature Selection Strategies". In: *Feature Extraction*. Ed. by Isabelle Guyon et al. Vol. 207. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2006, pp. 315–324.

[37] Shui-Sheng Zhou, Hong-Wei Liu, and Feng Ye. "Variant of Gaussian Kernel and Parameter Setting Method for Nonlinear SVM". In: *Neurocomput.* 72.13-15 (Aug. 2009), pp. 2931–2937.

[38] J. Weston et al. "Feature selection for SVMs". In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2000, pp. 668–674.

[39] Olivier Chapelle et al. "Choosing Multiple Parameters for Support Vector Machines". In: *Machine Learning* 46.1-3 (2002), pp. 131–159.

[40] S. Sathiya Keerthi, Vikas Sindhwani, and Olivier Chapelle. *An efficient method for gradient-based adaptation of hyperparameters in svm models*. Tech. rep. 2007.

[41] J. C. G. Boot. "On Sensitivity Analysis in Convex Quadratic Programming Problems". In: *Operations Research* 11.5 (1963), pp. 771–786.

[42] M. Baotic. *An Efficient Algorithm for Multiparametric Quadratic Programming*. Tech. rep. Institut für Automatik, ETH Zürich, Apr. 2002.

[43] Petter Tndel, Tor Arne Johansen, and Alberto Bemporad. "An algorithm for multiparametric quadratic programming and explicit {MPC} solutions". In: *Automatica* 39.3 (2003), pp. 489–497.

[44] Léon Bottou. "On-line Learning in Neural Networks". In: Cambridge University Press, 1998. Chap. On-line Learning and Stochastic Approximations.

[45] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[46] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297.

[47] O. L. Mangasarian. "Arbitrary-norm separating plane". In: *Operations Research Letters* 24 (1-2 Feb. 1999), pp. 15–23.

[48] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. fourth. Springer, 2001.

[49] P.S. Bradley and O. L. Mangasarian. "Feature Selection via Concave Minimization and Support Vector Machines". In: *Machine Learning Proceedings of the Fifteenth International Conference*. 1998, pp. 82–90.

[50] Bernhard Schölkopf et al. *Shrinking the Tube: A New Support Vector Regression Algorithm*. 1999.

[51] B. Schölkopf et al. "New Support Vector Algorithms". In: *Neural Comput.* 12.5 (May 2000), pp. 1207–1245.

[52] Pai hsuen Chen, Chih jen Lin, and Bernhard Schø"lkopf. "A tutorial on ν-Support Vector Machines". In: *Appl. Stochastic Models Bus. Ind.* Vol. 21. 2005, pp. 111–136.

[53] Hong-Gunn Chew, Robert E. Bogner, and Cheng-Chew Lim. "Dual nu;-support vector machine with error rate and training size biasing". In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*. Vol. 2. 2001, pp. 1269–1272.

[54] Edgar Osuna, Robert Freund, and Federico Girosi. *Support Vector Machines: Training and Applications*. Tech. rep. Cambridge, MA, USA, 1997.

[55] Mark Davenport. *The 2ν-SVM: A Cost-Sensitive Extension of the ν-SVM*. Tech. rep. Rice University, Dec. 2005.

[56] Murat M. Dundar et al. "Polyhedral Classifier for Target Detection: A Case Study: Colorectal Cancer". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. 2008, pp. 288–295.

[57] F. T. Cheng et al. "Evaluating Reliance Level of a Virtual Metrology System". In: *IEEE Trans. Semicond. Manuf.* 21.1 (Feb. 2008).

[58] W. F. Finan. *Metrology-Related Costs in the U.S. Semiconductor Industry, 1990, 1996, and 2001*. Tech. rep. National Institute of Standards and Technology, May 1998.

[59] M. Kanellos. *Intel's manufacturing cost: $40 per chip*. CNET News. Sept. 2005. URL: http://news.cnet.com/Intels-manufacturing-cost-40-per-chip/2100-1006_3-5862922.html.

[60] J. Musacchio. "Run to Run Control in Semiconductor Manufacturing". MA thesis. Berkeley: U.C. Berkeley, 1998.

[61] E. Sachs. "Tuning a Process While Performing SPC: An Approach Based on the Sequential Design of Experiments". In: *Advanced Semiconductor Manufacturing Conference and Workshop, 1990. IEEE/SEMI 1990*. 1990, pp. 126–130.

[62] M. Sarfaty et al. “Advance Process Control Solutions for Semicondcuctor Manufacturing”. In: *Advanced Semiconductor Manufacturing 2002 IEEE/SEMI Conference and Workshop*. 2002, pp. 101–106.

[63] C. Schneider, L. Pfitzer, and H. Ryssel. “Integrated metrology: An enabler for advanced process control (APC)”. In: *Proc. of SPIE*. Vol. 4406. 2001, pp. 118–123.

[64] Jesper Jung et al. *Ellipsometry*. Tech. rep. Aalborg University, Dec. 2004.

[65] Prasad Dasari et al. “Scatterometry metrology challenges of EUV”. In: vol. 8324. Proc. SPIE. 2012, p. 83240M.

[66] Shahin Zangooie et al. “Scatterometry Accuracy Improvement using 3D Shapes”. In: vol. 8681. Proc. SPIE. 2013, 86811S.

[67] Jinlong Zhu et al. “Identification and reconstruction of diffraction structures in optical scatterometry using support vector machine method”. In: *J. Micro/Nanolith. MEMS MOEMS* 12.1 (2013), p. 013004.

[68] Chih-Yu Chen et al. “Direct-scatterometry-enabled optical-proximity-correction-model calibration”. In: vol. 8681. Proc. SPIE. 2013, 86810U.

[69] Ji Zhu et al. “1-norm Support Vector Machines”. In: *Neural Information Processing Systems*. 2003, p. 16.

[70] Petre-Catalin Logofatu and John R. McNeil. “Measurement precision of optical scatterometry”. In: vol. 4344. 2001, pp. 447–453.

[71] H. Huang and Fred Terry Jr. “Spectroscopic ellipsometry and reflectometry from gratings (Scatterometry) for critical dimension measurement and in situ, real-time process monitoring”. In: *Thin Solid Films*. Vol. 455-456. 2004, pp. 828–836.

[72] Q. Zhang, K. Poolla, and C. J. Spanos. “Across Wafer Critical Dimension Uniformity Enahncement Through Lithography and Etch Process Sequence: Concept, Approach, Modeling, and Experiment”. In: *IEEE Journ. of Semicond. Manuf.* 20.4 (2007).

[73] L-T. Pang et al. “Measurement and Analysis of Variability in 45nm Strained Si CMOS Technology”. In: *IEEE Journ. of Solid-State Circuits* 44 (8 2009), pp. 2233–2243.

[74] J. Xiong et al. “Robust Extraction of Spatial Correlation”. In: *IEEE Trans. Comput.-Aided Integr. Circuits Syst.* 26.4 (2007), pp. 619–631.

[75] L. Cheng et al. “Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across Wafer Variability”. In: *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 30.3 (2011), pp. 619–631.

[76] T. L. Vincent, B. Stirton, and K. Poolla. “Metrology Sampling Strategies for Process Monitoring Applications”. In: *IEEE Trans. Semicond. Manuf.* 24.4 (2011), pp. 489–498.

[77] S. Reda and S. Nassif. “Analyzing the Impact of Process Variations on Parametric Measurements: Novel Models and Applications”. In: Design, Autom. and Test in Europe Conf. and Exhibit. 2009, pp. 375–380.

[78] H. Chang and S. S. Sapatnekar. "Statistical Timing Analysis Considering Spatial Correlations using a Single Pert-Like Traversal". In: Proc. IEEE/ACM Intl. Conf. Comp.-Aided Design. 2003, pp. 621–625.

[79] B. Stine, D. Boning, and J. Chung. "Analysis and Decomposition of Spatial Variation in Integrated Circuit Processes and Devices". In: *IEEE Trans. Semicond. Manuf.* 10.1 (1997), pp. 24–41.

[80] A. Argarwal et al. "Statistical Delay Computation Considering Spatial Correlations". In: Design Auto. Conf. 2003. Proc. ASP-DAC 2003. Asia and South Pacific. 2003, pp. 271–2760.

[81] F-L. Chen and S-F. Liu. "A Neural-Network Approach to Recognize Defect Spatial Pattern in Semiconductor Fabrication". In: *IEEE Trans. Semicond. Manuf.* 13.3 (2000), pp. 366–373.

[82] P. K. Mozumder and L. M. Loewenstein. "Method for Semiconductor Process Optimization Using Functional Representations of Spatial Variations and Selectivity". In: *IEEE Trans. On Comp., Hybrids, Manuf. Tech.* 15.3 (1992), pp. 311–316.

[83] M. Orshanky, L. Milor, and C. Hu. "Characterization of Spatial Intrafield Gate CD Varibaility, Its Impact on Circuit Performance, and Spatial Mask-Level Correction". In: *IEEE Trans. Semicond. Manuf.* 17.1 (2004), pp. 2–11.

[84] S. P. Cunningham and S. MacKinnon. "Statistical Methods for Visual Defect Metrology". In: *IEEE Trans. Semicond. Manuf.* 11.1 (1998), pp. 48–53.

[85] Y. Qiao, K. Qian, and C. J. Spanos. "Variability-Aware Compact Model Characterization for Statistical Circuit Design Optimization". In: vol. 8327. Proc. SPIE. 2012, 83270J–1–9.

[86] J. Cain and C. J. Spanos. "Electrical linewidth metrology for systematic CD variation characterization and causal analysis". In: vol. 5038. Proc. SPIE. 2003, pp. 350–361.

[87] K. Qian and C. J. Spanos. "A Comprehensive Model of Process Variability for Statistical Timing Optimization". In: vol. 6925. Proc. SPIE. 2008, 69251G–1–11.

[88] Mark A. Davenport. "Error Control for Support Vector Machines". MA thesis. Houston: Rice University, 2007.

[89] Dit-Yan Yeung, Hong Chang, and Guang Dai. "Learning the Kernel Matrix by Maximizing a KFD-based Class Separability Criterion". In: *Pattern Recogn.* 40.7 (July 2007), pp. 2021–2028.

[90] Joachim Dahl, Lieven Vandenberghe, and Vwani Roychowdhury. "Covariance Selection for Nonchordal Graphs via Chordal Embedding". In: *Optimization Methods Software* 23.4 (Aug. 2008), pp. 501–520.

[91] Gary M. Weiss, Kate McCarthy, and Bibi Zabar. "Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?" In: *DMIN'07*. 2007, pp. 35–41.

[92] Zhenxing Qin et al. "Cost Sensitive Classification in Data Mining". In: *Advanced Data Mining and Applications*. Vol. 6440. Lecture Notes in Computer Science. 2010, pp. 1–11.

[93] Maytal Saar-Tsechansky and Foster Provost. "Active Sampling for Class Probability Estimation and Ranking". In: *Machine Learning* 54.2 (2004), pp. 153–178.

[94] Martin S. Andersen and Lieven Vandenberghe. "Support vector machine training using matrix completion techniques". unpublished.

[95] Francesca Bovolo, Lorenzo Bruzzone, and Lorenzo Carlin. "A Novel Technique for Subpixel Image Classification Based on Support Vector Machine". In: *Trans. Img. Proc.* 19.11 (Nov. 2010), pp. 2983–2999.

[96] David H. Wolpert and William G. Macready. "No Free Lunch Theorems for Optimization". In: *IEEE Trans. Evol. Comput.* 1.1 (1997), pp. 67–82.

[97] Niklas Lavesson. "Evaluation and Analysis of Supervised Learning Algorithms and Classifiers". PhD thesis. Blekinge Institute of Technology, 2006.

Appendix A

Background on Support Vector Machines

We briefly go over the theoretical background behind support vector machine (SVM) classifiers in this appendix. The invention of this algorithm is credited to Vapnik *et al.* in 1995. [46] A complete derivation can be found in the references at the back of this chapter. [46, 47, 48, 34, 49]

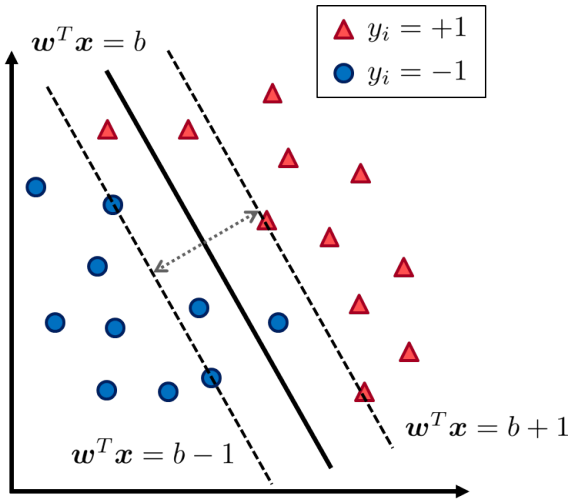
The training set $\mathcal{T}$ consists of n data points, $\mathcal{T} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n) \mid \boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \{-1, +1\}\}$, where $\boldsymbol{x_i}$ is a p-dimensional feature vector with a dependent categorical variable y_i. The labels $\{-1, +1\}$ correspond to each "class" in a binary classification problem. The objective of a linear SVM classifier is to construct a p-dimensional hyperplane that separates data points in the two classes such that it maximizes the distance between two margin hyperplanes. Figure A.1a shows an example with $\boldsymbol{x}_i \in \mathbb{R}^2$. The blue and red points can be thought of as samples with $y_i = -1$ and $+1$, respectively.

A.0.1 Hard-margin SVMs

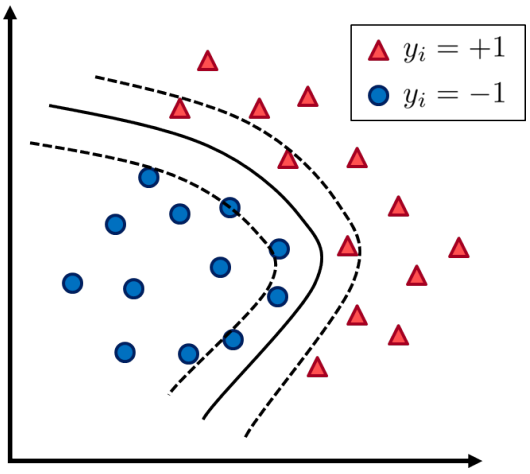
Hard-margin SVMs assume that the data is linearly separable without any misclassified points. Specifically, the objective is to construct a decision hyperplane $\boldsymbol{w}^T\boldsymbol{x} = b$, and two margin hyperplanes $\boldsymbol{w}^T\boldsymbol{x} = b + 1, \boldsymbol{w}^T\boldsymbol{x} = b - 1$ such that the distance between the two margin hyperplanes is maximized while achieving separation. Take $\boldsymbol{x}_{min}$ to be nearest data point to hyperplane $\boldsymbol{w}^T\boldsymbol{x} - b = 0$ and normalize $\boldsymbol{w}, b$ such that $|\boldsymbol{w}^T\boldsymbol{x}_{min} - b| = 1$. The following theorem from [47] is very useful.

Theorem 3. ***Arbitrary-Norm Projection on a Plane.*** *[47] Let $\boldsymbol{q} \in \mathbb{R}^n$ be any point in $\mathbb{R}^n$ not on the plane:*

$$P := \{\boldsymbol{x} \mid \boldsymbol{w}^T\boldsymbol{x} = b\}, 0 \neq w \in \mathbb{R}^n, b \in \mathbb{R}. \tag{A.1}$$



(a) Soft-margin classification.



(b) Classification with non-linear boundaries.

Figure A.1: (a) Soft-margin SVMs minimize the summed distance from the corresponding hyperplane for misclassified points. (b) Most cases require a non-linear boundary for proper classification.

A projection $p(\boldsymbol{q}) \in P$ *using a general norm* $||\cdot||$ *on* $\mathbb{R}^n$ *is given by:*

$$p(\boldsymbol{q}) = \boldsymbol{q} - \frac{\boldsymbol{w}^T\boldsymbol{q} - b}{||\boldsymbol{w}||'}\boldsymbol{z}(\boldsymbol{w}), \tag{A.2}$$

where $||\cdot||'$ *is the dual norm to* $||\cdot||$ *and:*

$$\boldsymbol{z}(\boldsymbol{w}) \in \underset{||\boldsymbol{z}||=1}{arg\ max}\ \boldsymbol{w}^T\boldsymbol{z}. \tag{A.3}$$

Consequently, the distance between q and its projection p(q) is given by:

$$||\boldsymbol{q} - p(\boldsymbol{q})|| = \frac{|\boldsymbol{w}^T\boldsymbol{q} - b|}{||\boldsymbol{w}||'}. \tag{A.4}$$

By Theorem 3, the Euclidean distance from any point on the decision hyperplane $\boldsymbol{w}^T\boldsymbol{x} = b$ to either hyperplanes $\boldsymbol{w}^T\boldsymbol{x} = b+1$ and $\boldsymbol{w}^T\boldsymbol{x} = b-1$ is $\frac{|b-(b\pm1)|}{||\boldsymbol{w}||_2} = \frac{1}{||\boldsymbol{w}||_2}$. Note that the dual norm of the L_2 norm is itself. Our problem now turns into

$$\begin{aligned} \max_{\boldsymbol{w},b} \quad & \frac{1}{||\boldsymbol{w}||_2} \\ \text{subject to} \quad & \min_{i=1,2,\ldots,n} |\boldsymbol{w}^T\boldsymbol{x}_i - b| = 1. \end{aligned} \tag{A.5}$$

Formulating this as a linear program (LP), this is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{A.6}$$

From hereon, we derive the dual of the problem in (A.6). We denote the dual variables associated with the inequality constraints as α_n for each training sample. The dual problem is now

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w},b} \quad & \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{w}, b) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) - 1) \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{A.7}$$

Taking $\nabla_{\boldsymbol{w},b}\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{w}, b) = 0$, we get

$$\begin{aligned} & \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{w}, b) = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0 \\ & \frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \therefore \ \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i, \quad \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{A.8}$$

Substituting the results in (A.8) into (A.14), the dual problem now becomes

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0 \quad \text{for} \quad i = 1, 2, \ldots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{A.9}$$

which is a quadratic program. We can rewrite (A.16) as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T \begin{bmatrix} y_1 y_1 \boldsymbol{x}_1^T \boldsymbol{x}_1 & \cdots & y_1 y_n \boldsymbol{x}_1^T \boldsymbol{x}_n \\ \vdots & \ddots & \vdots \\ y_n y_1 \boldsymbol{x}_n^T \boldsymbol{x}_1 & \cdots & y_n y_n \boldsymbol{x}_n^T \boldsymbol{x}_n \end{bmatrix} \boldsymbol{\alpha} + (-1)^T \boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha} \leq 0, \\ & \boldsymbol{\alpha}^T \boldsymbol{y} = 0. \end{aligned} \tag{A.10}$$

Given optimal solutions $\boldsymbol{\alpha}^*, \boldsymbol{w}^*, b^*$, we see that the complementary slackness from the KKT conditions gives us

$$\begin{aligned} & \alpha_i^*(y_i(\boldsymbol{w}^{*T}\boldsymbol{x}_i - b*) - 1) = 0, \quad i = 1, 2, \ldots, n \\ \Rightarrow & \text{either } \alpha_i^* = 0 \text{ or } y_i(\boldsymbol{w}^{*T}\boldsymbol{x}_i - b^*) = 1. \end{aligned} \tag{A.11}$$

For values of $\alpha_i > 0$, the corresponding x_i's are on the hyperplane $\boldsymbol{w}^T\boldsymbol{x}_i = b \pm 1$. These are called the ***support vectors***. We denote the set $\mathcal{U}$ to contain the indices of an equal number of support vectors from each class. After solving for $\boldsymbol{\alpha}^*$ in (A.14) through any optimization package, the variables $\boldsymbol{w}^*$ and b^* are given by

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i, \quad b^* = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \boldsymbol{w}^{*T} \boldsymbol{x}_u = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i^T \boldsymbol{x}_u. \tag{A.12}$$

For an incoming sample $\boldsymbol{x}_j$, the classifier is given by $h_j = \boldsymbol{w}^{*T}\boldsymbol{x}_j + b^*$, and the classification is given by $\hat{y}_j(\boldsymbol{x}_j) = \text{sign}(h_j)$.

A.0.2 Soft-margin SVMs

We now turn to soft-margin SVMs. Soft-margin SVMs are used when the data is not completely separable. During the training period, data points are allowed to cross their corresponding margin hyperplanes, but of course, the number of these, as well as how far they cross over the hyperplane should be minimized. This is done by relaxing the constraints on the hyperplanes given in (A.6). The new primal problem now becomes

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{A.13}$$

More details can be found in references, but ξ_i is 0 for non-support vectors, and is a positive number representing the distance of a data point from its margin hyperplane for misclassified points. The farther away the point is from the margin hyperplane, the greater

the value of ξ_i. C is a parameter controlling the tradeoff between maximizing the margin and decreasing the number of misclassified points. If misclassified points are penalized too much, overfitting will occur; on the contrary, if the margin is too wide, the classifer may not be flexible enough and underfitting will occur. The optimal C value is find through cross-validation techniques as will be mentioned in the next section. Similarly to Section A.0.1, we find out the Lagrangian is

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\boldsymbol{w},b,\xi} \quad & \mathcal{L}(\boldsymbol{\alpha},\boldsymbol{w},b) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \\ \text{subject to} \quad & \alpha_i \geq 0, \\ & \beta_i \geq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{A.14}$$

Taking $\nabla_{\boldsymbol{w},b,\xi}\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w},b,\xi) = 0$, we get

$$\begin{aligned} \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{w},b) &= w - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0 \\ \frac{\partial\mathcal{L}}{\partial b} &= \sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial\mathcal{L}}{\partial\xi_i} &= C - \alpha_i - \beta_i = 0. \end{aligned} \tag{A.15}$$

Substituting the results, the dual problem now becomes

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \quad \text{for} \quad i = 1, 2, \ldots, n, \\ & \sum_{i=1}^{n}\alpha_i y_i = 0. \end{aligned} \tag{A.16}$$

We now look at the complementary slack conditions. Contrary to the hard-margin case, there are two types of support vectors.

- $\xi_i^* = 0 \Rightarrow \beta_i^* > 0 \Rightarrow 0 < \alpha_i^* < C \Rightarrow y_i(\boldsymbol{w}^{*T}\boldsymbol{x}_i - b^*) = 1$
  $\Rightarrow$ Corresponds to support vectors on the margin hyperplanes, similar to those seen in (A.11). These are denoted as *unbounded support vectors*, since α_i^* is between 0 and C.
- $\xi_i^* > 0 \Rightarrow \beta_i^* = 0 \Rightarrow \alpha_i^* = C \Rightarrow y_i(\boldsymbol{w}^{*T}\boldsymbol{x}_i - b^*) < 1$
  $\Rightarrow$ Corresponds to support vectors that are on the wrong side of their corresponding margin hyperplane. These are denoted as *bounded support vectors*, since α_i^* is equal to the bound C.

Similarly to the hard-margin case, $\boldsymbol{w}^*$ and b^* can be found by (A.12), where $\mathcal{U}$ is now the index set of an equal number of unbounded support vectors.

A.0.3 SVMs for Non-linear Data

One problem that remains is separating a data set with non-linear boundaries. Figure A.1b shows an example of a data set that is best separated with a non-linear boundary. SVM can be applied to such problems by mapping the current feature space into another one that will provide linear boundaries. Consider the following equations for an ellipse in $\mathbb{R}^2$:

$$\begin{aligned} &c(x_1 - a)^2 + d(x_2 - b)^2 - 1 = 0 \\ &cx_1^2 + dx_2^2 - 2acx_1 - 2bdx_2 + ca^2 + db^2 - 1 = 0 \end{aligned} \tag{A.17}$$

If we take the following mapping into some space $\mathbb{Z}$,

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \Longrightarrow \begin{bmatrix} x_1^2 & x_2^2 & x_1 & x_2 & x_1 x_2 & 1 \end{bmatrix}$$

we find that the original equation is now a linear function of the new features, and we can now apply our original linear SVM models. One can see that the optimization problem and solution all require only the inner product $\boldsymbol{x}_i^T \boldsymbol{x}_j$. We do not have to explicitly transform each $\boldsymbol{x}_i$ into the new feature space, all we need is a function for the inner product in that space. For two vectors $\boldsymbol{z}_i, \boldsymbol{z}_j \in \mathbb{Z}$ mapped from $\boldsymbol{x}_i, \boldsymbol{x}_j$,

$$\begin{aligned} \boldsymbol{z}_i^T \boldsymbol{z}_j &= x_{1i}^2 x_{1j}^2 + x_{2i}^2 x_{2j}^2 + \ldots + x_{1i} x_{2i} x_{1j} x_{2j} + 1 \\ &= (1 + \boldsymbol{x}_i^T \boldsymbol{x}_j)^2 \\ &= \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{A.18}$$

where we have defined the "kernel function" $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the inner product between two feature vectors in the transformed space. Some well-known kernel functions are

- Linear: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Ellipsoid: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^T \boldsymbol{x}_j)^2$
- Radial Basis Function (RBF): $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\beta ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$
- ARD-Gaussian: $\kappa(\boldsymbol{x}_i^d, \boldsymbol{x}_j^d) = \exp(-\sum_{k=1}^{p} \beta_k ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$.

The RBF kernel maps the feature space into an infinite dimension space, and is one of the most flexible kernel functions for non-linear classification. The optimal kernel parameter β is found through cross-validation techniques, discussed in the subsequent section. The ARD-Gaussian kernel is an extension of the RBF but has a separate parameter for each feature dimension. This allows each dimension to be scaled differently avoiding underfitting and overfitting. To use a certain kernel function in the construction of a classifier, $\boldsymbol{x}_i^T \boldsymbol{x}_u$ is replaced with $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_u)$ in (A.12).

A.0.4 Cross Validation and Empirical Loss

There are many parameters in the SVM that are not an optimization or dual variable. For example, C is a parameter that chooses the tradeoff between margin width and misclassified points. β in the RBF kernel determines the shape of the classifier. These kinds of parameters in learning are often called hyperparameters. As well-known in statistical learning theory, the optimal classifier is chosen by cross-validation. Usually, the data set is divided into training, validation, and testing sets. The training set is used to induce classifiers with different sets of hyperparameters. These models are then applied to an unseen validation set, where the hyperparameter combination that minimizes the loss function is chosen as the optimal value. The accuracy of the validated model on the testing set is reported. In many cases, the loss is averaged over many partitions of the data set. As before, we denote the training set as $\mathcal{T}$, with $|\mathcal{T}_-| = n_-$ and $|\mathcal{T}_+| = n_+$, and the validation set as $\mathcal{Z}$, with $|\mathcal{Z}_-| = m_-$ and $|\mathcal{Z}_+| = m_+$.

The classifier will induce a so called generalization cost (*risk* in other literature), which is the expected value of the loss function, for an unseen data set $z = (\boldsymbol{x}, y)$. The most widely used loss function for classification models is the 0-1 loss, where

$$l(\hat{y}, y) = \begin{cases} 1 & \text{if } \hat{y} \neq y \\ 0 & \text{if } \hat{y} = y. \end{cases} \tag{A.19}$$

Since the ground truth distribution $P(\boldsymbol{x}, y)$ is never known, the expected cost is usually estimated with an empirical average on the data set. For the validation data, this is given by

$$\hat{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}) = \frac{1}{2m} \sum_{j \in \mathcal{Z}} [1 - y_j \operatorname{sign}(h_j)], \tag{A.20}$$

where $\boldsymbol{\phi}$ is a vector of hyperparameters. Note that for each validation sample $(\boldsymbol{x}_j, y_j)$, $1 - y_j \operatorname{sign}(h_j)$ is 2 if the true label y_j is different from the classification estimate $\operatorname{sign}(h_j)$, and 0 if the labels coincide. Finally, the problem of hyperparameter selection becomes

$$\boldsymbol{\phi}^* = \underset{\phi}{\operatorname{argmin}} \quad \hat{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}). \tag{A.21}$$

As mentioned before, an exhaustive search across the hyperparameter space is needed to validate the optimal $\boldsymbol{\phi}^*$ that minimizes the empirical generalization cost.

A.1 The ν-SVM Classifier

So far, we have demonstrated the foundation of a SVM classifier: the primal and dual problems, support vectors, and its hyperparameters. One disadvantage of the C-SVM is that the optimal C can range anywhere between 0 to ∞. An alternate model is the ν-SVM,

where the parameter $\nu \in (0, 1)$. The primal problem is given by

$$\begin{aligned} \min_{\boldsymbol{w},b,\rho,\xi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \nu\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq \rho - \xi_i \quad \text{for} \quad i = 1, 2, \ldots, n \\ & \xi_i \geq 0 \\ & \rho \geq 0. \end{aligned} \tag{A.22}$$

Scholkopt *et al.* [50, 51, 52] proposed that if the optimization result has $\rho > 0$,

- ν is an upper bound on the fraction of margin errors (hence the fraction of training errors).
- ν is also a lower bound on the fraction of support vectors.
- If the data was generated i.i.d. from a certain distributsion, and the kernel is analytic and non-constant, with probability 1, asymptotically, ν equals both the fraction of support vectors and the fraction of margin errors.

The Lagrangian of (A.22) is given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, \xi, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) =& \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \nu\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) - \rho + \xi_i) \\ & - \sum_{i=1}^{n}\beta_i\xi_i - \delta\rho. \end{aligned} \tag{A.23}$$

where $\alpha_i, \beta_i, \delta \geq 0$. Taking $\nabla_{\boldsymbol{w},b,\rho}\mathcal{L}(\boldsymbol{w}, b, \xi, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) = 0$, we get

$$\begin{aligned} \nabla_{\boldsymbol{w}}\mathcal{L} &= w - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0 \\ \frac{\partial\mathcal{L}}{\partial b} &= \sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial\mathcal{L}}{\partial\rho} &= -\nu + \sum_{i=1}^{n}\alpha_i - \delta = 0 \\ \frac{\partial\mathcal{L}}{\partial\xi_i} &= \frac{1}{n} - \alpha_i - \beta_i = 0. \end{aligned} \tag{A.24}$$

Plugging everything in the Lagrangian gives us

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\delta} \quad & \mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j - \rho\sum_{i=1}^{n}\alpha_i - \delta + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ & - b\sum_{i=1}^{n}\alpha_i y_i + \rho\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i - \delta\rho \\ \text{subject to} \quad & \boldsymbol{\alpha},\boldsymbol{\beta},\delta \geq 0 \quad \text{for} \quad i = 1,2,\ldots,n.\end{aligned} \tag{A.25}$$

This is equivalent to the dual problem

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\delta} \quad & \mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{n} \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & \sum_{i=1}^{n}\alpha_i \geq \nu.\end{aligned} \tag{A.26}$$

A.2 Cost-Sensitive SVM Classifiers

Traditional classifiers like C-SVM and ν-SVM penalize errors in both classes equally. That is, if the training data has a disproportionate number of samples for one class, the above formulations will favor reducing the error on the class with larger size. In this section, we explore the 2ν-SVM, which penalizes each category in a cost-sensitive manner. The 2ν-SVM (or *dual-*ν) was originally proposed by Chew *et al.* [53] in 2001. As a variation of the ν-SVM, the cost-sensitive version introduces an additional parameter $\gamma \in [0,1]$ that decides the relative weight between the two classes errors in constructing the classifier. [53, 54, 55] Similarly to the ν-SVM, ν controls the upper bound of margin errors, and a small or large value of ν may cause over or under-fitting, respectively. The primal problem of the 2ν-SVM is given by

$$\begin{aligned}\min_{\boldsymbol{w},b,\rho,\xi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \nu\rho + \frac{\gamma}{n}\sum_{i\in I_+}\xi_i + \frac{1-\gamma}{n}\sum_{i\in I_-}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq \rho - \xi_i \quad \text{for} \quad i = 1,2,\ldots,n \\ & \xi_i \geq 0 \\ & \rho \geq 0.\end{aligned} \tag{A.27}$$

The dual for the 2ν-SVM is just

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\delta} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ \text{subject to} \quad & 0 \le \alpha_i \le \frac{\gamma}{n} \quad \text{for} \quad i \in I_+ \\ & 0 \le \alpha_i \le \frac{1-\gamma}{n} \quad \text{for} \quad i \in I_- \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \sum_{i=1}^{n} \alpha_i \ge \nu. \end{aligned} \tag{A.28}$$

Following notation from [55], we note parameters

$$\nu_+ = \frac{\nu n}{2\gamma n_+}, \quad \nu_- = \frac{\nu n}{2(1-\gamma)n_-}. \tag{A.29}$$

where

- ν_+ is an upper bound on the fraction of margin errors in class +1.
- ν_- is an upper bound on the fraction of margin errors in class -1.
- ν_+ is a lower bound on the fraction of support vectors from class +1.
- ν_- is a lower bound on the fraction of support vectors from class -1.

Moreover, (A.28) is feasible if and only if

$$\nu \le \frac{2\min(\gamma n_+, (1-\gamma)n_-)}{n}, \tag{A.30}$$

or in other words,

$$\nu_+ \nu_- \le \min(\nu_-, \nu_+). \tag{A.31}$$

Assuming we have different costs for false and missed alarms, the expected generalization cost, for example, can be written as

$$\begin{aligned} \Psi_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}) &= E[l(\hat{y}(\boldsymbol{x}) \neq y)] \\ &= c_1 P(\hat{y}(\boldsymbol{x}) = +1 | y = -1) P(y = -1) + c_2 P(\hat{y}(\boldsymbol{x}) = -1 | y = +1) P(y = +1) \end{aligned} \tag{A.32}$$

where c_1 is the false alarm cost and c_2 is the missed alarm cost. Note that the generalization cost can have other forms than (A.32). For instance, either c_1, c_2 can be functions of the false and missed alarm rates, and not just constants.

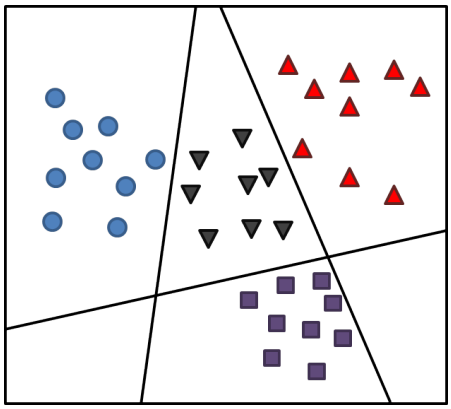


Figure A.2: Multicategorical support vector machine classification.

Similarly to (A.20), we can define an empirical generalization cost as

$$\hat{\Psi}_{\mathcal{T},\mathcal{Z}}(\boldsymbol{\phi}) = \frac{c_1}{2m_-} \sum_{j \in \mathcal{Z}_-} [1 - y_j \operatorname{sign}(h_j)] \frac{m_-}{m} + \frac{c_2}{2m_+} \sum_{j \in \mathcal{Z}_+} [1 - y_j \operatorname{sign}(h_j)] \frac{m_+}{m} \tag{A.33}$$

where $\frac{1}{2m_-} \sum_{j \in \mathcal{Z}_-} [1 - y_j \operatorname{sign}(h_j)]$ and $\frac{1}{2m_+} \sum_{j \in \mathcal{Z}_+} [1 - y_j \operatorname{sign}(h_j)]$ are unbiased estimates of false and missed alarms, and $\frac{m_-}{m}$ and $\frac{m_+}{m}$ are the estimated probability of occurrences for the two classes.

A.2.1 Multicategorical SVMs

Some problem settings require multiple categories for classification. An obvious example is the distinct modes of failure in scatterometry, such as positive/negative defocus, necking, bridging, and unexposed fields. It would be useful if we could differentiate not only faults but also the distinct types of faults. [56] Figure A.2 shows a 2-D example where the objective is to differentiate between black samples and samples of three different colors. This is done by constructing three separate hyperplanes, leading to a polyhedral classifier. Assuming a total of r categories, we denote each "out-of-spec" category with a subscript k, and the "in-spec" category with a subscript 0. For the supervised case, we assume that we exactly know the value of r, and construct the following optimization problem:

$$
\begin{aligned}
\min_{\boldsymbol{w}_1 \ldots \boldsymbol{w}_r, b_1 \ldots b_r, \boldsymbol{\xi}^{(1)} \ldots \boldsymbol{\xi}^{(r)}} \quad & \frac{1}{2} \sum_{k=1}^{r} \boldsymbol{w}_k^T \boldsymbol{w}_k + \sum_{k=1}^{r} C_1 \sum_{i=1}^{n_k} \xi_i^{(k)} + C_2 \sum_{i}^{n_0} \xi_i^{(0)} \\
\text{subject to} \quad & \boldsymbol{w}_k^T \boldsymbol{x}_i - b_k \geq 1 - \xi_i^{(k)} \quad \text{for} \quad i = 1, \ldots, n_k, \; k = 1, 2, \ldots, r \\
& - \boldsymbol{w}_0^T \boldsymbol{x}_i - b_0 \geq 1 - \xi_i^{(0)} \quad \text{for} \quad i = 1, \ldots, n_0 \\
& \xi_i^{(k)} \geq 0 \quad \text{for} \quad i = 1, \ldots, n_k, \; k = 1, 2, \ldots, r \\
& \xi^{(0)} \geq 0 \quad \text{for} \quad i = 1, \ldots, n_0.
\end{aligned}
\tag{A.34}
$$

where n_k denotes the number of samples in category k. The idea behind (A.34) is to simply construct r hyperplane classifiers between class 0 and class k, where each classifier has the form $\boldsymbol{w}_k^{*T} \boldsymbol{x} = b_k^*$.

Appendix B

Scatterometry Grating SEM Images and Bossung Curves

B.0.2 Focus-Exposure Matrix CD-SEM Images

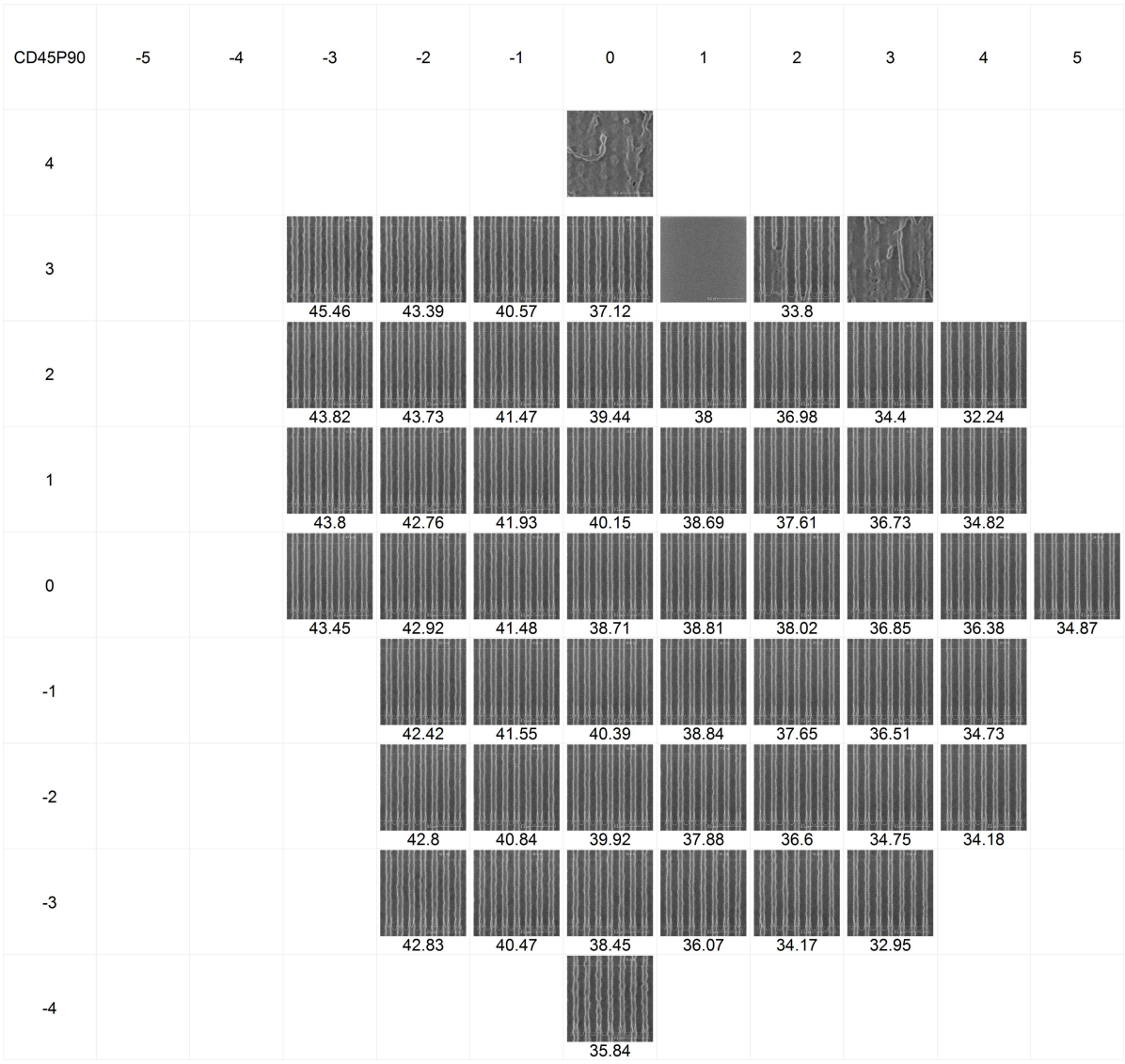


Figure B.1: CD-SEM images for P90CD45.

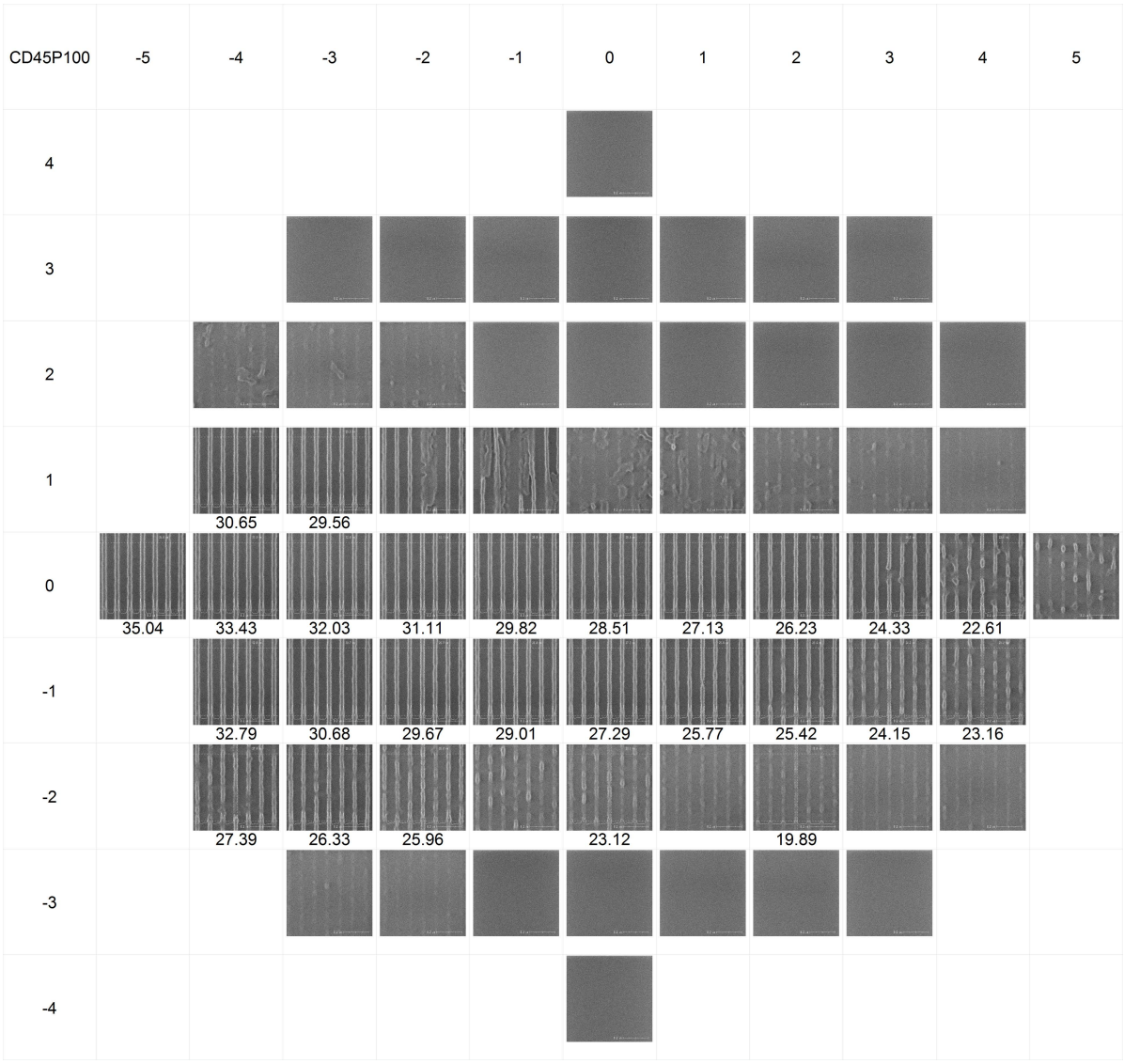


Figure B.2: CD-SEM images for P100CD45.

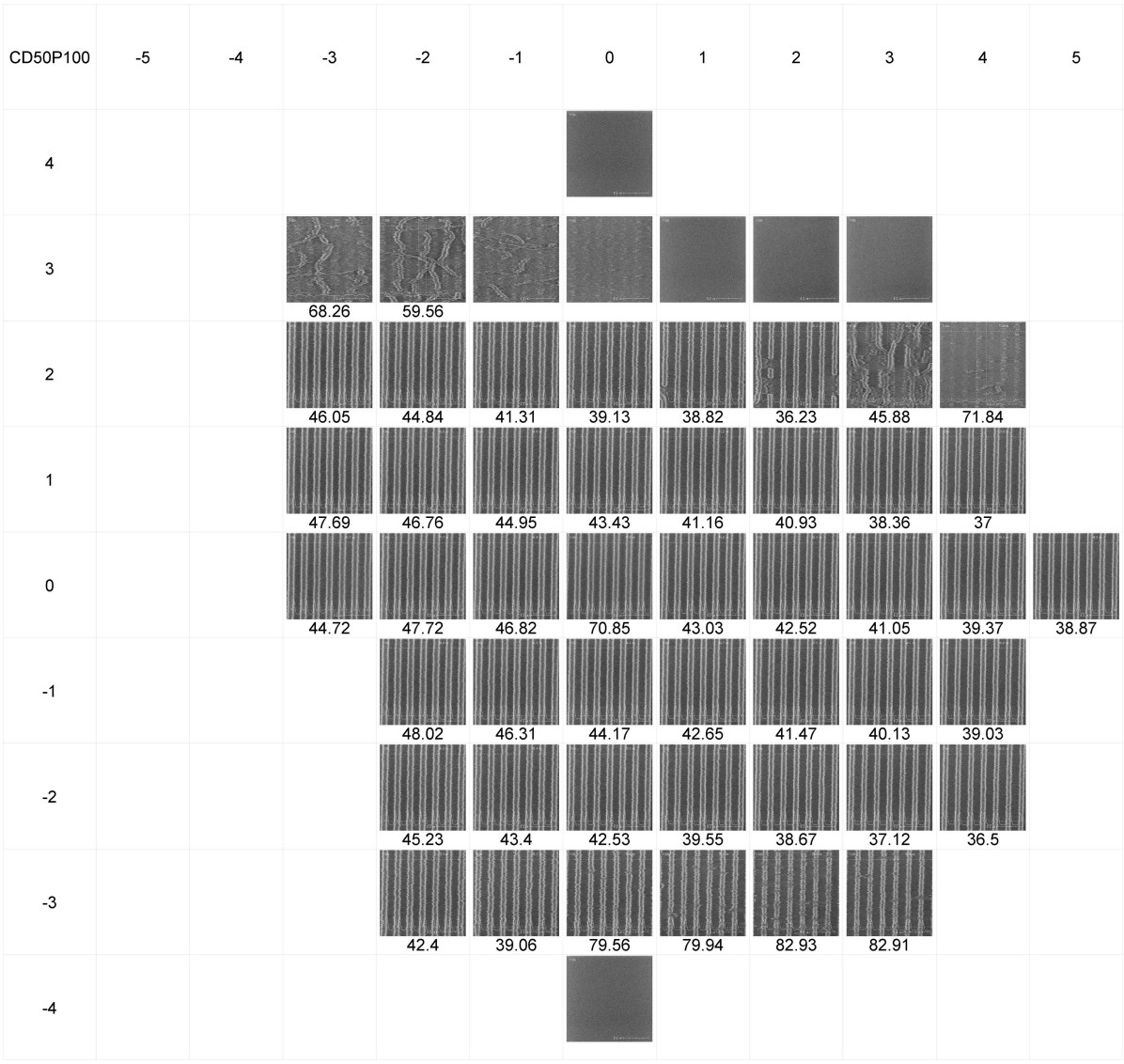


Figure B.3: CD-SEM images for P100CD50.

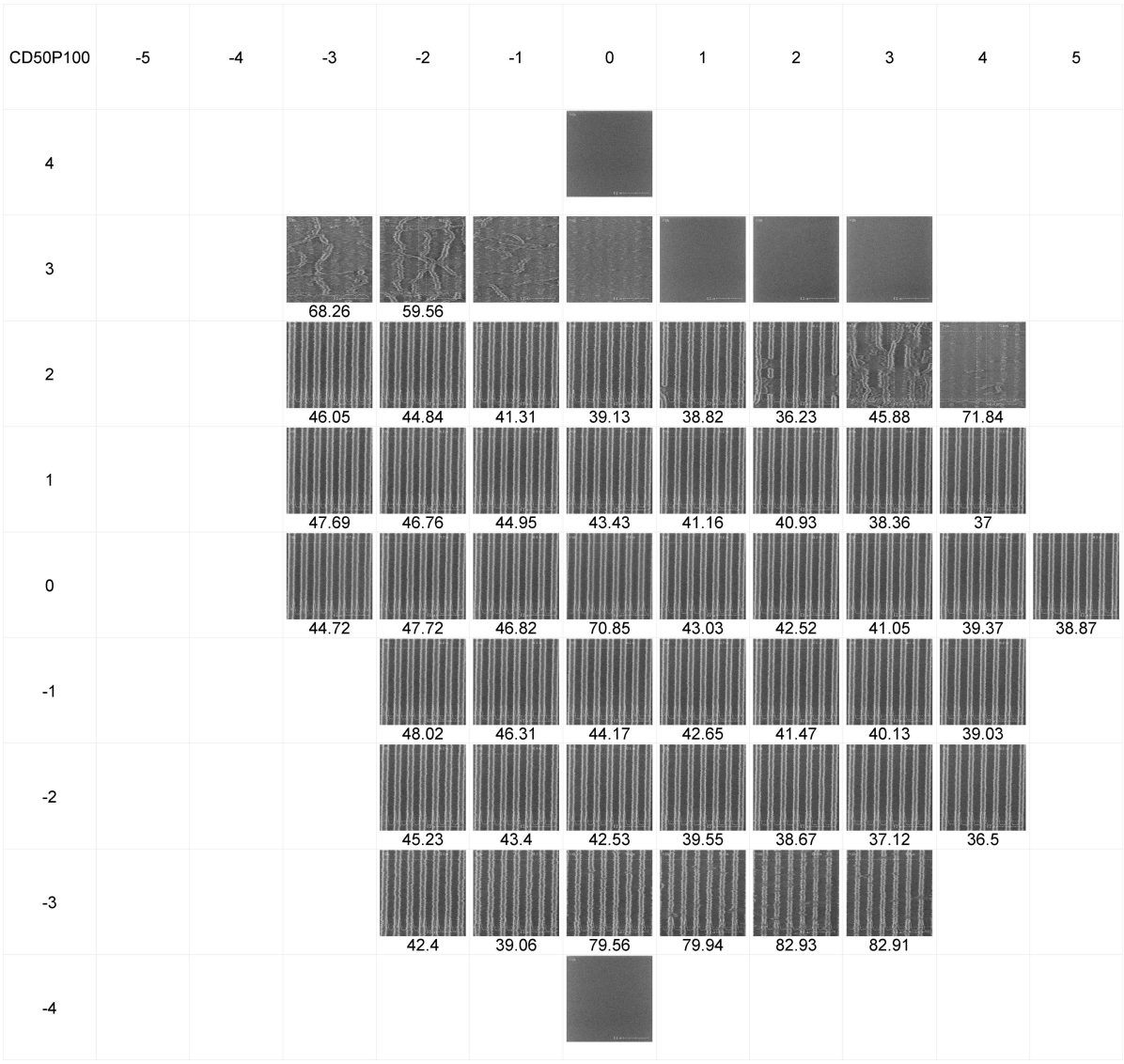


Figure B.4: CD-SEM images for P110CD55.

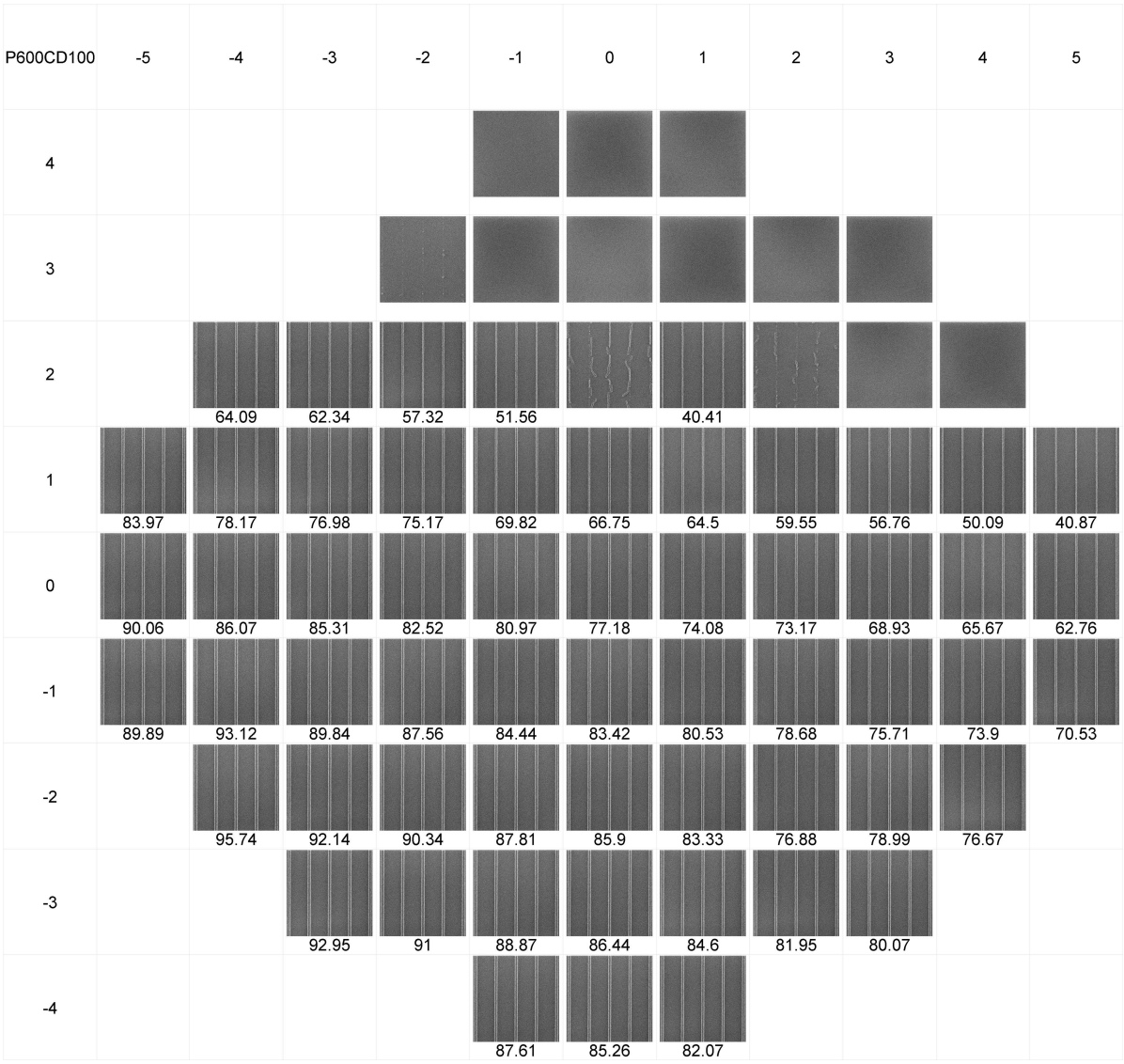


Figure B.5: CD-SEM images for P600100.

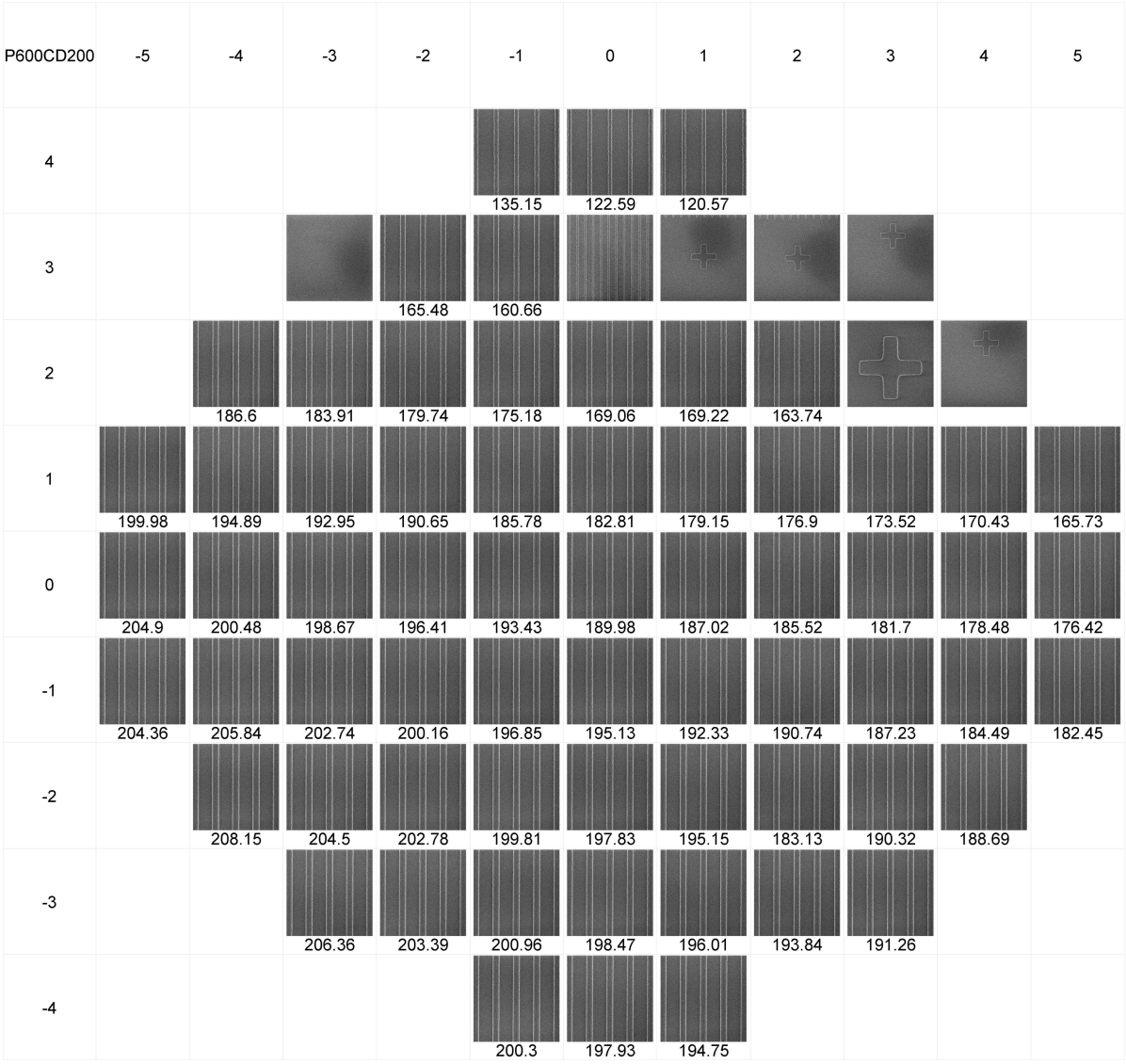


Figure B.6: CD-SEM images for P600CD200.

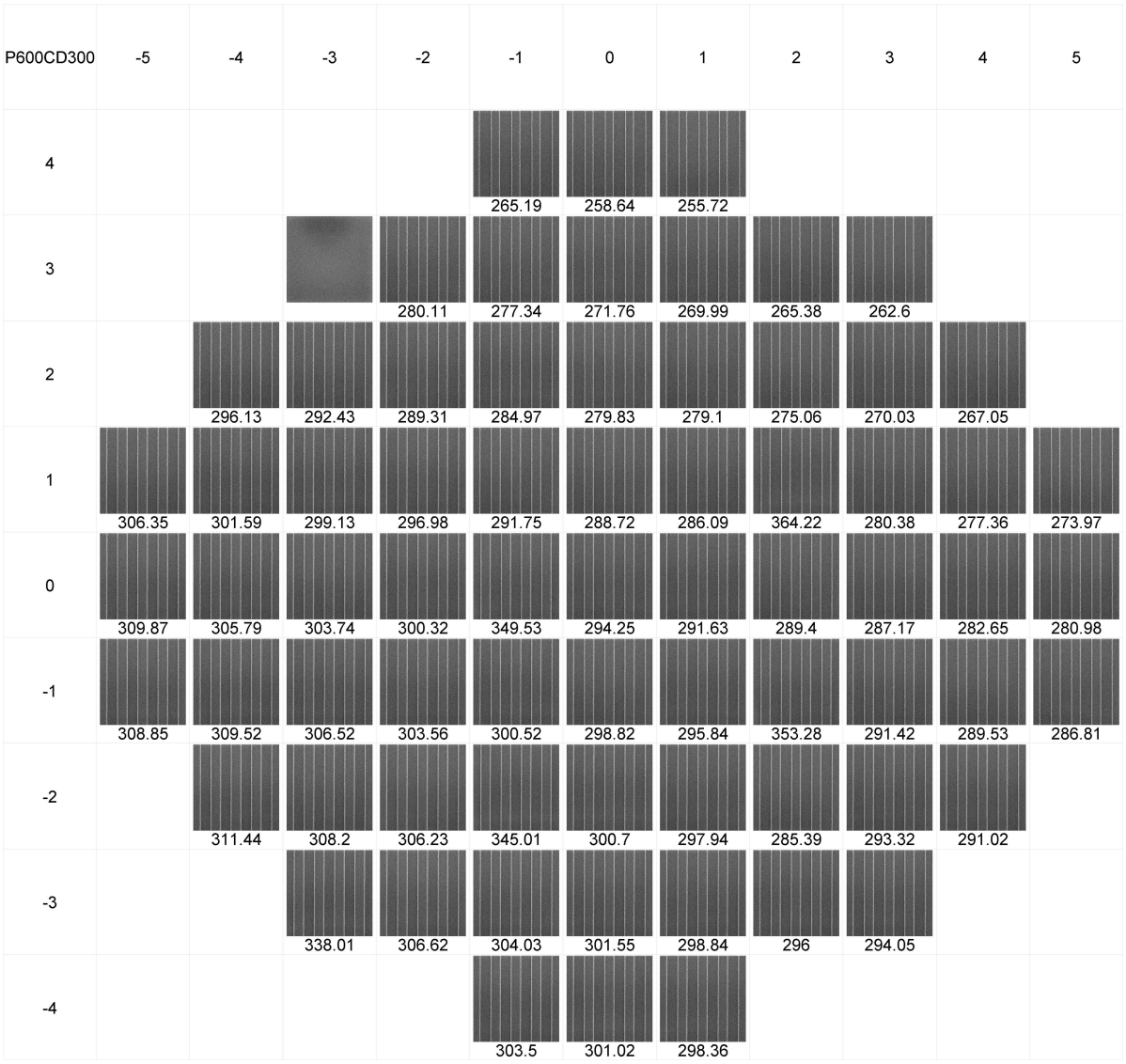


Figure B.7: CD-SEM images for P600CD300.

B.0.3 Bossung Curves from YieldStar Measurements

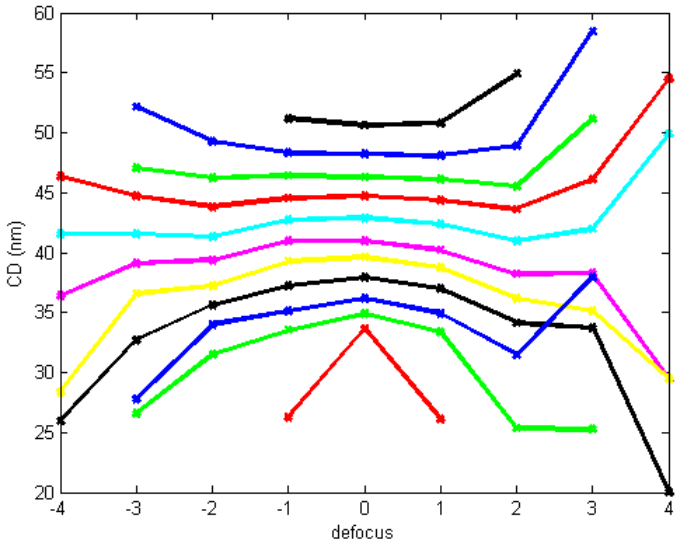


Figure B.8: YieldStar Bossung curves for P90CD45.

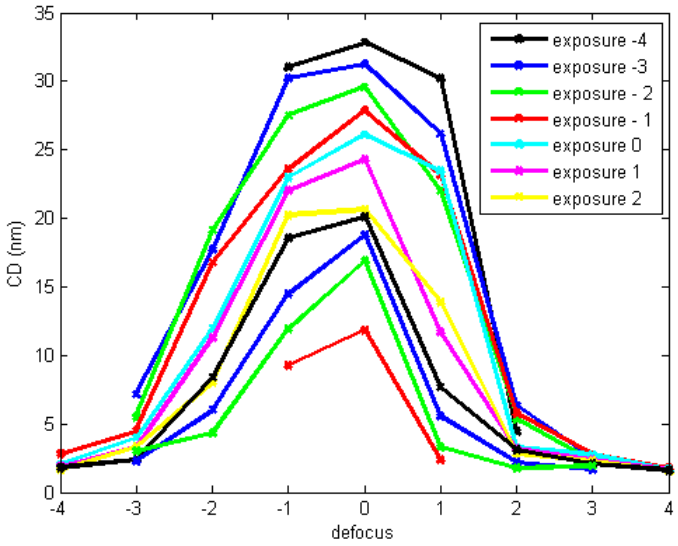


Figure B.9: YieldStar Bossung curves for P100CD45.

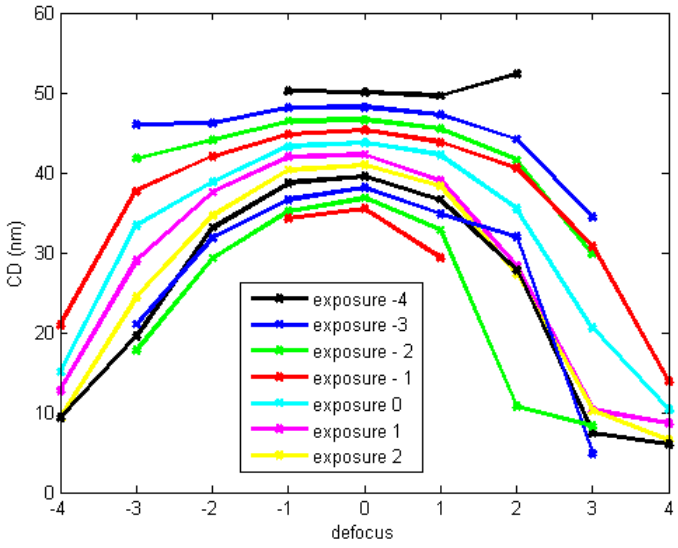


Figure B.10: YieldStar Bossung curves for P100CD50.

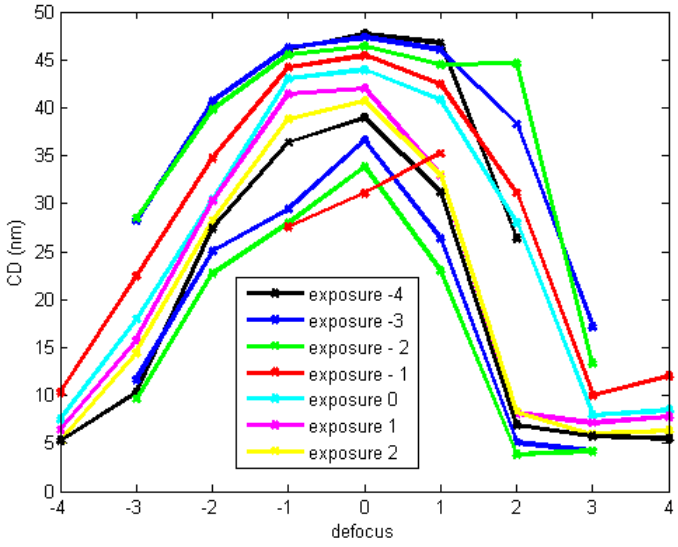


Figure B.11: YieldStar Bossung curves for P110CD55.

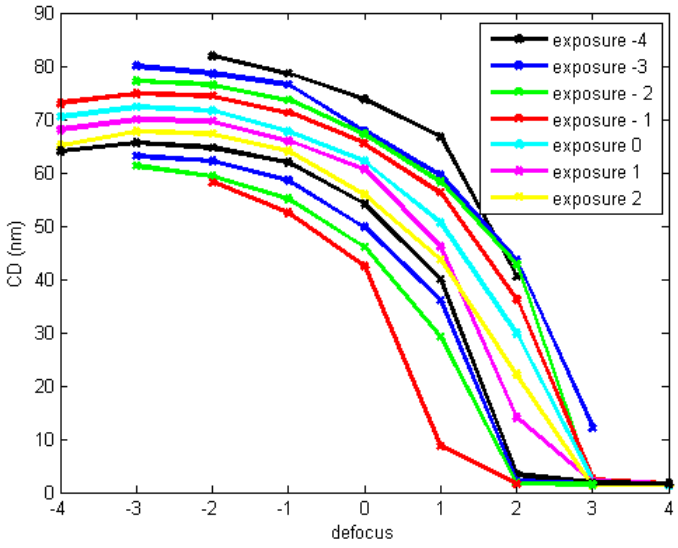


Figure B.12: YieldStar Bossung curves for P600CD100.

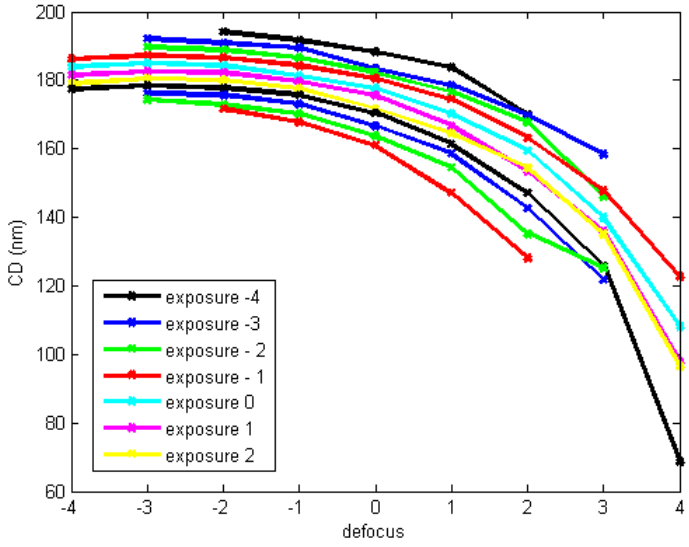


Figure B.13: YieldStar Bossung curves for P600CD200.

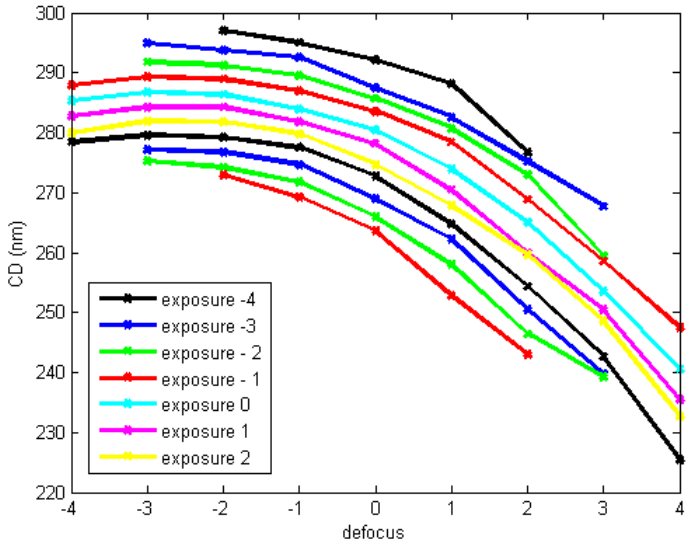


Figure B.14: YieldStar Bossung curves for P600CD300.

Appendix C

Proofs for Fast Model Selection for Grating Classification

As a reminder, $\boldsymbol{y}$ is the ordered labels, $\boldsymbol{Q}_{\boldsymbol{\beta}}$ is the augmented kernel matrix, $\boldsymbol{C}_{\mathcal{A}}$ the rows of active constraints, and $\boldsymbol{P}_{\boldsymbol{\beta}}$ defined in 4.4, we further define

$$\boldsymbol{G}_{\boldsymbol{\beta}} = \frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}. \tag{C.1}$$

In addition, we denote the number of non-support vectors $s_0 = |\mathcal{S}_0|$, bounded support vectors $s_b = |\mathcal{S}_b|$, and unbounded support vectors $s_{ub} = |\mathcal{S}_{ub}|$.

C.0.4 Solution of over-determined system

Lemma 1. *If the solution $\boldsymbol{\alpha}^*(\boldsymbol{\phi})$ is non-degenerate, with $\boldsymbol{C}_{\mathcal{A}}$ and $\boldsymbol{y}$ defined previously, the over-determined system for the variable $\boldsymbol{\xi}$*

$$\boldsymbol{C}_{\mathcal{A}}^{\alpha\,T}\boldsymbol{\xi} = \boldsymbol{y} \tag{C.2}$$

does not have a solution.

Proof. Note that the rows of $\boldsymbol{C}^{\alpha}$ only contains n dimensional standard basis $\{-e_1^T, -e_2^T, \ldots, -e_n^T\}$, $\{e_1^T, e_2^T, \ldots, e_n^T\}$ and $[-1, \ldots, -1]^T$. The active constraints are induced by non-support vectors in the first block, and bounded support vectors in the second block. These $s_0 + s_b = n - s_{ub}$ active constraints must be orthogonal and span a subspace of $\mathbb{R}^n$. This is because α_i^* cannot be 0 and γ/n or $(1-\gamma)/n$ at the same time. Ignoring the last inequality constraint, the rows of $\boldsymbol{C}_{\mathcal{A}}^{\alpha}$ consist of $n - s_{ub}$ positive or negative standard basis. For any $k \in \mathcal{A}^C$, the basis $\pm e_k$ are not in $\boldsymbol{C}_{\mathcal{A}}^{\alpha}$, hence the k_{th} column vectors of $\boldsymbol{C}_{\mathcal{A}}^{\alpha}$ must be all zero. When the solution of SVM is non-degenerate, there exist at least two samples in $\mathcal{A}^C$, with one corresponding to $y_k = 1$ and another corresponding to $y_k = -1$, by definition. Now consider two cases for the last inequality constraint $\sum_{i=1}^{n} \alpha_i^* \geq \nu$.

1. $\sum_{i=1}^{n} \alpha_i^* > \nu$ – Since $\boldsymbol{C}_{\mathcal{A}}^{\alpha T}$ has at least two zero rows, the linear system $\boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\xi} = \boldsymbol{y}$ results in two equations $0 = -1$ and $0 = +1$.

2. $\sum_{i=1}^{n} \alpha_i^* = \nu$ – $(\boldsymbol{C}_{\mathcal{A}}^{\alpha})^T$ now has an additional last column of $[-1, \ldots, -1]$. We again consider two indices $k, k' \in \mathcal{A}^C$ such that $y_k = -1$ and $y_k = +1$. The k^{th} and k'^{th} row of $\boldsymbol{C}_{\mathcal{A}}^{\alpha T}$ are zeros except the last element -1. This results in two equations $-\boldsymbol{\xi}_{n-s_{ub}} = -1$ and $-\boldsymbol{\xi}_{n-s_{ub}} = +1$.

We see that for both cases, $\boldsymbol{C}_{\mathcal{A}}^{\alpha T}\boldsymbol{\xi} = \boldsymbol{y}$ is an inconsistent system with no solutions. □

C.0.5 Properties of matrix $\boldsymbol{G}_{\boldsymbol{\beta}}$

Lemma 2. *$\boldsymbol{G}_{\boldsymbol{\beta}}$ has rank $n-1$.*

Proof. By the Rank-nullity theorem, we know that $\text{rank}(\boldsymbol{G}_{\boldsymbol{\beta}}) + \text{nul}(\boldsymbol{G}_{\boldsymbol{\beta}}) = n$. We now consider the nullspace of $\boldsymbol{G}_{\boldsymbol{\beta}}$. First, note that if a vector $\boldsymbol{v} \in \mathbb{R}^n$ is in the nullspace of $\boldsymbol{G}_{\boldsymbol{\beta}}$,

$$\boldsymbol{G}_{\boldsymbol{\beta}}\boldsymbol{v} = 0 \iff (\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1})\boldsymbol{v} = (\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y})\boldsymbol{v}. \tag{C.3}$$

Since $\text{rank}(\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}) = 1$, this means $\boldsymbol{v}$ is the eigenvector corresponding to the only non-zero eigenvalue of $\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}$. We can verify $\boldsymbol{v} = \boldsymbol{y}$ and $\text{null}(\boldsymbol{G}_{\boldsymbol{\beta}}) = \boldsymbol{y}$, proving that $\text{rank}(\boldsymbol{G}_{\boldsymbol{\beta}}) = n-1$. □

Lemma 3. *$\boldsymbol{G}_{\boldsymbol{\beta}}$ is a symmetric negative semi-definite matrix.*

Proof. We look at $\boldsymbol{v}^T\boldsymbol{G}_{\boldsymbol{\beta}}\boldsymbol{v} \;\; \forall \boldsymbol{v} \in \mathbb{R}^n$.

$$\begin{aligned} \boldsymbol{v}^T\boldsymbol{G}_{\boldsymbol{\beta}}\boldsymbol{v} &= \boldsymbol{v}^T\left(\frac{\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}} - \boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\right)\boldsymbol{v} \\ &= \frac{(\boldsymbol{v}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y})^2 - (\boldsymbol{v}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{v})(\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y})}{\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}}. \end{aligned} \tag{C.4}$$

Since $\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}$ is assumed to be a positive definite matrix, it has a unique Cholesky decomposition $\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1} = \boldsymbol{L}\boldsymbol{L}^T$ where $\boldsymbol{L}$ is a lower triangular matrix. Defining $\tilde{\boldsymbol{v}} \triangleq \boldsymbol{L}^T\boldsymbol{v}$, and $\tilde{\boldsymbol{y}} \triangleq \boldsymbol{L}^T\boldsymbol{y}$ the numerator of $\boldsymbol{v}^T\boldsymbol{G}_{\boldsymbol{\beta}}\boldsymbol{v}$ becomes

$$\begin{aligned} &(\boldsymbol{v}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y})^2 - (\boldsymbol{v}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{v})(\boldsymbol{y}^T\boldsymbol{Q}_{\boldsymbol{\beta}}^{-1}\boldsymbol{y}) \\ &=(\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{y})^2 - (\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{z})(\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{y}) \\ &=(\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{y}})^2 - (\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{v}})(\tilde{\boldsymbol{y}}^T\tilde{\boldsymbol{y}}) \\ &=|\langle\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{y}}\rangle|^2 - ||\tilde{\boldsymbol{v}}||^2||\tilde{\boldsymbol{y}}||^2. \end{aligned} \tag{C.5}$$

By the Cauchy-Schwarz inequality, $|\langle \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{y}} \rangle|^2 \leq ||\tilde{\boldsymbol{v}}||^2 ||\tilde{\boldsymbol{y}}||^2$, and the numerator of $\boldsymbol{v}^T \boldsymbol{G}_{\boldsymbol{\beta}} \boldsymbol{v}$ is ≤ 0. Therefore, $\boldsymbol{G}_{\boldsymbol{\beta}}$ is a negative semi-definite matrix. □

C.0.6 Proof of Theorem 1

Proof. $\boldsymbol{P}_{\boldsymbol{\beta}}$ was defined as $\boldsymbol{P}_{\boldsymbol{\beta}} = \boldsymbol{C}_{\mathcal{A}}^{\alpha} \boldsymbol{G}_{\boldsymbol{\beta}} \boldsymbol{C}_{\mathcal{A}}^{\alpha T}$. We look at $\boldsymbol{\xi}^T \boldsymbol{P}_{\boldsymbol{\beta}} \boldsymbol{\xi} \;\; \forall \boldsymbol{\xi} \in \mathbb{R}^{n_0}$, where $n_0 = n - s_{ub} + 1$.

$$\begin{aligned} \boldsymbol{\xi}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha} \boldsymbol{G}_{\boldsymbol{\beta}} \boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi} &= \boldsymbol{\xi}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha} \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi} \\ &= \boldsymbol{\xi}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha} \boldsymbol{U} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{U}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi} \\ &= \boldsymbol{\xi}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha} \boldsymbol{R} \boldsymbol{R}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi} \\ &= \tilde{\boldsymbol{\xi}}^T \tilde{\boldsymbol{\xi}} \preceq 0 \end{aligned} \tag{C.6}$$

where we've used the spectral theorem to decompose $\boldsymbol{G}_{\boldsymbol{\beta}}$ and defined $\tilde{\boldsymbol{\xi}} \triangleq \boldsymbol{R}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi}$. If $\exists \tilde{\boldsymbol{\xi}} \mid \tilde{\boldsymbol{\xi}}^T \tilde{\boldsymbol{\xi}} = 0$, this implies $\boldsymbol{R}^T \boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi} = 0$. Notice that $\text{rank}(\boldsymbol{R}^T) = n - 1$ and $\text{null}(\boldsymbol{R}^T) = 1$, since $\boldsymbol{G}_{\boldsymbol{\beta}} = \boldsymbol{R}\boldsymbol{R}^T$ and by Lemma 2, $\text{rank}(\boldsymbol{G}_{\boldsymbol{\beta}}) = n - 1$. Moreover, the nullspace of $\boldsymbol{R}^T$ is solely composed of $\boldsymbol{y}$. Thus, in order for $\boldsymbol{\xi}^T \boldsymbol{P}_{\boldsymbol{\beta}} \boldsymbol{\xi}$ to be 0, we need $\boldsymbol{C}_{\mathcal{A}}^{\alpha T} \boldsymbol{\xi} = \boldsymbol{y}$, and by Lemma 1 this does not have a solution. Therefore, $\boldsymbol{P}_{\boldsymbol{\beta}}$ is a negative definite matrix. □

C.0.7 Proof of Proposition 1

Proof. The proof of this proposition relies on the strict convexity of the dual. Since the boundary of any two regions belongs to both closures, and the optimum is unique for all hyperparameters in the feasible set, the solution across the boundary is continuous.

Because any feasible configuration of $\boldsymbol{\phi}$ admits a solution, it must be contained in at least one region. Suppose a configuration of $\boldsymbol{\phi}$ belongs to 2 or more regions. Since the set of active constraints are different in any 2 regions, the optimum cannot be the same except at the boundary. However, by assumption, if $\boldsymbol{\phi}$ belongs to two regions, and this means the dual has at least two solutions. This is contradictory to the uniqueness of the solution.

Finally, by construction, the number of critical regions should be upper bounded by the number of all combinations of active constraints. In the non-degenerate case, this is just $|\mathcal{T}_-| \cdot |\mathcal{T}_+| \cdot 2^{|\mathcal{T}|-2}$. □

C.0.8 Proof of Proposition 3

Proof. Take two arbitrary points $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ in r^{th} region $\mathcal{R}_r$. Let $h = \boldsymbol{\phi}_1 - \boldsymbol{\phi}_2$, $J[\cdot]$ the jacobian matrix of a vector value function, $\|\cdot\|_2$ the $L2$ norm, and $\|\cdot\|_I$ the induced $L2$

norm of a matrix, i.e. $\|A\|_I = sup\{\|Ax\|_2 \mid \|x\|_2 = 1\}$, then

$$\begin{aligned}
\|g(\boldsymbol{\phi}_2) - g(\boldsymbol{\phi}_1)\|_2 &= \|g(\boldsymbol{\phi}_1 + h) - g(\boldsymbol{\phi}_1)\|_2 \\
&= \left\| \int_0^1 J\left[g(\boldsymbol{\phi}_1 + th)\right] \cdot h \, dt \right\|_2 \\
&\leq \int_0^1 \left\| J\left[g(\boldsymbol{\phi}_1 + th)\right] \cdot h \right\|_2 dt \\
&\leq \int_0^1 \left\| J\left[g(\boldsymbol{\phi}_1 + th)\right] \right\|_I \cdot \|h\|_2 dt \\
&\leq M\|h\|_2 = M\|\boldsymbol{\phi}_2 - \boldsymbol{\phi}_1\|_2
\end{aligned}$$

The second inequality is a direct application of matrix norm inequality $\|Ax\|_2 \leq \|A\|_I \cdot \|x\|_2$, and the third inequality holds since in a continuous bounded region the induced norm of the Jacobian is bounded by some scalar M. □