VIRTUAL METROLOGY FOR SEMICONDUCTOR MANUFACTURING APPLICATIONS

SUPERVISOR: Ch.mo Prof. Beghi Alessandro

STUDENT: Bertorelle Nicola

A.A. 2009-2010

UNIVERSITY OF PADUA
DEPARTMENT OF INFORMATION ENGINEERING
TESI DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

VIRTUAL METROLOGY FOR SEMICONDUCTOR MANUFACTURING APPLICATIONS

SUPERVISOR: Ch.mo Prof. Beghi Alessandro

CO-ADVISOR: Ing. Gian Antonio Susto

STUDENT: *Bertorelle Nicola*

Padua, 28 June 2010

Alla mia famiglia

“ Dove tuona un fatto, siatene certi, ha lampeggiato un’idea.”

Ippolito Nievo

Abstract

Per essere competitive nel mercato, le industrie di semiconduttori devono poter raggiungere elevati standard di produzione a un prezzo ragionevole. Per motivi legati tanto ai costi quanto ai tempi di esecuzione, una strategia di controllo della qualità che preveda la misurazione completa del prodotto non è attuabile; i test sono effettuati su un ristretto campione dei dati originali. Il traguardo del presente lavoro di Tesi è lo studio e l'implementazione, attraverso metodologie di modellistica tipo non lineare, di un algoritmo di metrologia virtuale (Virtual Metrology) d'ausilio al controllo di processo nella produzione di semiconduttori. Infatti, la conoscenza di una stima delle misure non realmente eseguite (misure virtuali) può rappresentare un primo passo verso la costruzione di sistemi di controllo di processo e controllo della qualità sempre più raffinati ed efficienti. Da un punto di vista operativo, l'obiettivo è fornire la più accurata stima possibile delle dimensioni critiche a monte della fase di etching, a partire dai dati disponibili (includendo misurazioni da fasi di litografia e deposizione e dati di processo - ove disponibili). Le tecniche statistiche allo stato dell'arte analizzate in questo lavoro comprendono:

- multilayer feed-forward networks;

Confronto e validazione degli algoritmi presi in esame sono stati possibili grazie ai dataset forniti da un'industria di semiconduttori nell'ambito della continuazione del progetto ENIAC EU-IMPROVE (WP2). In conclusione, questo lavoro di Tesi rappresenta un primo passo verso la creazione di un sistema di controllo di processo e controllo della qualità evoluto e flessibile, che abbia il fine ultimo di migliorare la qualità della produzione.

Contents

List of Figures

List of Tables

Introduction

The semiconductor industry is one of the most technology-evolving and capital-intensive market sectors.

The speed and effectiveness with which new products are developed have an important influence on market competitiveness. So, the control of manufacturing operations is well recognized to a source of competitive advantage. In order to implement an effective control strategy, product samples (wafers) have to be measured. Unfortunately, this procedure (*Metrology*) bears high costs and produces delay in control feedback and process optimization.

The semiconductor manufacturing industry has a large-volume multistage manufacturing system. To insure the high stability and the production yield on-line a reliable wafer monitoring is required. The Advanced Process Control (APC) is currently deployed in factory-wide control of Front End Of the Line (FEOL) processing in semiconductor manufacturing. The APC tools are the main ways to ensure a continuous process improvement (see [9]). However, most APC tools strongly depend on the physical measurement provided by metrology tools. Critical wafers parameters are measured, such as, for example, the thickness or the roughness of the thin films. The physical metrology of critical parameters of wafers quality is performed after each processing step only on monitor wafers that are periodically selected by sampling in production equipment for each lot processing (usually one to four wafers per lot). This approach involves that the production wafer quality between two measures is unknown. When the equipment is out of order and the abnormality is not detected in time, many defective wafers may have been produced before the next measure. This will result in a large amount of wafer scraps and will greatly impact the cost. To overcome this problem, an efficient way is to predict the process quality of every wafer using process parameter data of production equipment without physically conducting quality measures.

In order to resolve these drawbacks of metrology, the concept of *Virtual Metrology* (acronym VM) has gathered in recent years: Chen et al.[1] (2005) defined it as “a novel technique to predict wafer performance from tool state variables”; Yung-Cheng and Cheng[3] (2005), Besnard and Toprac[6] (2006) provided more detailed definitions: “a method to conjecture operation performance of a process tool based on data sensed from the process tool and without physical metrology operation”, “the estimation of metrology values based on process data such as fault detection and classification (FDC), context, and previous metrology", respectively.

What the aforementioned definitions have in common is that the purpose of VM is to predict "*every*" wafer's metrology measurements based on data available and it consists in the definition and the application of some predictive and corrective models for metrology outputs (physical measurements) in function of the previous metrology outputs and of the equipment parameters of current and previous steps of fabrication.

Of course it is necessary to develop a new generation of sensors to improve the characterization of physical and chemical reactions occurring on the wafer surface during process steps. Their data will constitute the basis for the Statistical and Physical models that will be developed. A typical Fault Detection and Classification (FDC) system collects on-line data from process by sensors equipment for every process run. They are called process variables or FDC data. Some reliable available FDC data are essential in VM model. A first approach is to use VM for an individual process using the pre-process metrology data and the FDC data from the chosen tools that are generally collected in real time for fault detection purposes. Into a factory implementation, VM modules for individual processes can be coordinated with one another for a better prediction quality. Since upstream wafer processing affects results of the current process, the VM module for a particular process step can produce a more accurate prediction of the output by using related preprocess metrology data (predicted via VM as well as current) of the upstream processes. The objective is to develop a robust prediction that can provide estimation of metrology and which is able to handle process drifts and step function changes induced by preventive maintenance disturbances. There are several methods of prediction including:

Nonlinear methods - Neural Networks (NN):

- Back Propagation Neural Networks (BPNN);
- Radial Basis Function Neural Networks (RBFN).

This thesis will discuss and compare various techniques to estimate metrology values.

Virtual Metrology is the first step and one of the most important steps to build a control system aimed to reduce costs and delay without affecting product quality.

The main aim of this work is to present a methodology for VM module for individual process applications in semiconductor industries.

The thesis is structured as follows:

- **Chapter 1**: the chapter introduces the semiconductor manufacturing process and his fundamental steps, among them we cite the Statistical Process Control (SPC) and the Advanced Process Control (APC) techniques, the Run-to-Run (R2R) control module together the two specific control laws (EWMA and dEWMA).
- **Chapter 2**: the role of VM in semiconductor industry is presented, explaining how it may improve control strategies.
- **Chapter 3**: in this chapter we introduce a detailed report on neural networks and their state of art, underlining the aspects we will use afterwards.
- **Chapter 4**: we present our VM system design including data description, preprocessing, dimensionality reduction techniques.
- **Chapter 5**: experimental results are analyzed, underlining the performance measures used, and some relevant issues are discussed.
- **Chapter 6**: we conclude with a summary and we address a discussion of future work.

Data for testing the VM methods were provided by a worldwide semiconductor factory (partner of ENIAC EU-IMPROVE project (WP2)[1]).

[1] This thesis has been carried out as part of the research activity funded by the ENIAC EU Project "Improve"; IMPROVE (Implementing Manufacturing science solutions to increase equiPment pROductiVity and fab pErformance) is a 3-years collaborative European project, funded by the ENIAC initiative. The main objective is to improve the European semiconductor fabs efficiency by providing methods and tools. The main focus of research is on virtual metrology (VM), predictive maintenance (PM) and adaptive control planning (ACP).

Chapter 1

The Semiconductor Manufacturing Process

1.1 Introduction

This chapter will describe the steps to build Integrated Circuit (IC) chips with ULSI (Ultra Large Scale Integration - over 10 millions semiconductors devices per wafer) technology, the tests that they have to pass to be commercialized, and the Run-to-Run technique, widely used to improve the stability of the process.
Semiconductor device fabrication is the process used to create the integrated circuits (also called *silicon chips*) that are present in everyday electrical and electronic devices. It is a multiple steps sequence of photographic and chemical processing steps during which electronic circuits are gradually created on a *wafer* made of pure semiconducting material. *Silicon* is the commonly used semiconductor material today, along with various compound semiconductors.
The entire manufacturing process from start to packaged chips ready for shipment takes six to eight weeks and is performed in highly specialized facilities referred to as *fabs*.

1.2 Wafers

A wafer is a thin slice of semiconductor material, such as a silicon crystal, used in the fabrication of integrated circuits and other micro-devices. The wafer serves as the substrate for microelectronic devices built in and over the wafer and undergoes many micro-fabrication process steps that we will describe afterwards.

Wafers are formed of highly pure (99.9999% purity), nearly defect-free single crystalline material. The process for forming crystalline wafers is known as Czochralski growth (see Figure 1.1, picture taken from [27]); here, a cylindrical ingot of high purity crystalline silicon is formed by pulling a seed crystal from a melt. The ingot is then sliced with a wafer saw and polished to to obtain a very regular and flat surface. The size of wafers for photovoltaics is $100 - 200\,mm$ square and the thickness is $200 - 300\,\mu m$; in the future, $160\,\mu m$ will be the standard. Electronics use wafer sizes from $100 - 300\ mm$ diameter. The resulting thin wafers can then be doped to achieve the desired electronic properties.

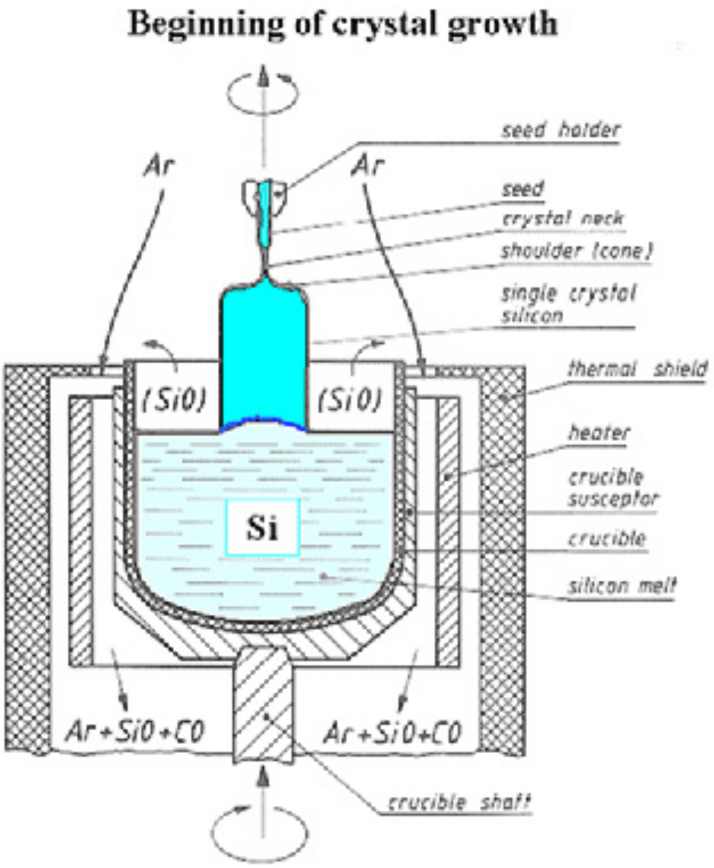


Figure 1.1: Czochralski growth process

Once the wafers are prepared, many process steps are necessary to produce the desired semiconductor integrated circuit. In general, the steps can be grouped into two main areas:

- Front-end processing;
- Back-end Processing.

1.3 Processing

In semiconductor device fabrication, the various processing steps fall into four general categories:

- deposition;
- removal;
- patterning;
- modification of electrical properties.

Deposition is any process that grows, coats, or otherwise transfers a material onto the wafer. Available technologies consist of physical vapor deposition (PVD), chemical vapor deposition (CVD), electrochemical deposition (ECD), molecular beam epitaxy (MBE) and more recently, atomic layer deposition (ALD) among others.

Removal processes are any that remove material from the wafer either in bulk or selectively and consist primarily of etch processes, either wet etching or dry etching. Chemical-mechanical planarization (CMP) is also a removal process used between levels.

Patterning covers the series of processes that shape or alter the existing shape of the deposited materials and is generally referred to as lithography. For example, in conventional lithography, the wafer is coated with a chemical called a *photoresist.* The photoresist is exposed by a *steppe*r, a machine that focuses, aligns, and moves the mask, exposing select portions of the wafer to short wavelength light. The unexposed regions are washed away by a developer solution. After etching or other processing, the remaining photoresist is removed by plasma ashing.

Modification of electrical properties has historically consisted of doping transistor sources and drains originally by diffusion furnaces and later by ion implantation. These doping processes are followed by furnace anneal or in advanced devices, by rapid thermal anneal (RTA) which serve to activate the implanted dopants. Modification of electrical properties now also extends to reduction of dielectric constant in low-k insulating materials via exposure to ultraviolet light in UV processing (UVP).

Note that the only just aforementioned steps will be repeated several times during the entire process, to produce multiple interconnected layers on the wafer. Many modern chips have eight or more levels produced in over 300 sequenced processing steps.

In next three subsections, the attention will be focused on three steps of the front end process, playing a major role in this dissertation:

- deposition and CVD, in particular;
- lithography and its sub-steps;
- etching and dry-etching, in particular.

Chemical Vapor Deposition - CVD

Chemical Vapor Deposition is the process leading to the formation of a nonvolatile solid film on a substrate by the reaction of vapor-phase chemicals (reactants) containing the required constituents. It is a material synthesis process whereby the constituents of the vapor phases react chemically near (or on) a substrate surface to form a solid product. Several steps must occur in every CVD reaction:

1. Transport of reacting gaseous species to the substrate surface
2. Absorption, or chemisorption, of the species on the substrate surface
3. Heterogeneous surface reaction catalyzed by the substrate surface
4. Desorption of gaseous reaction products
5. Transport of reaction products away from the substrate surface

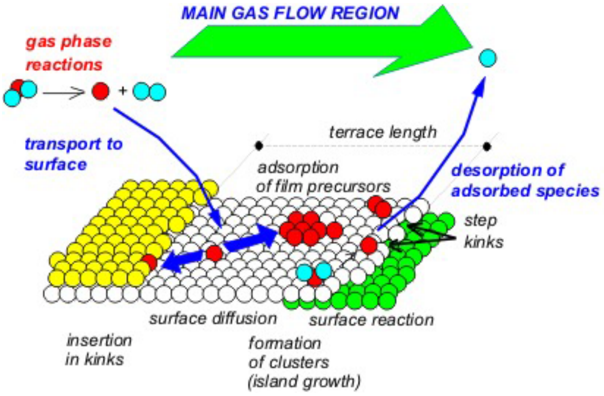


Figure 1.2: CVD steps

The sequences of reaction steps in a CVD process is illustrated in Figure 1.2 (picture taken from [28]). Heterogeneous reaction occur selectively only on the heated surface and produce good-quality films. On the other hand, homogeneous reactions are undesirable, because they form gas-phase clusters of the depositing material, resulting in low-density films with defects, and a decrease in deposition rates. The most common deposition methods are Atmospheric-Pressure CVD (APCVD), Low-Pressure CVD (LPCVD), and Plasma-Enhanced CVD (PECVD).

There are several variables to be controlled: temperature, pressure, flow rate, position and reactant ratio are all important factors for high-quality films.

Lithography

The lithographic process is composed by three main steps:

1. photosensitive polymer (resist) and Bottom Anti-Reflective Coating (BARC) deposition;

2. masking;

3. Exposure to a laser illumination to obtain the desired lithographic pattern

The BARC layer helps the adhesion to the substrate providing a better dimensional control. The commercial name of the resin is Novolac. The system is considered a positive resist system if the laser-exposed regions increase their solubility and, during the development step, the exposed regions are washed away. Otherwise, the system is a negative resist system. Optical lithography performs the formation of images with ultraviolet radiation in a photoresist using projection printing: this methodology offers high resolution because of its image formation system.

Virtually all advanced microelectronics devices are fabricated using projection lithography, shown schematically in Figure 1.3 (picture taken form [29]). In this technique, a light image of the desired pattern, transmitted through a mask, is reduced in size and precisely focused onto a resist-coated wafer using a system of lenses. Due to diffraction and slight imperfections in the optical components, a nominally square wave pattern of light intensity is presented as a sinusoidal pattern of light at the wafer plane. The minimum resolution W_{min} achievable with projection lithography, according to Rayleigh's criterion, is governed by the equation:

$$W_{min} = k_1 \, \frac{\lambda}{NA} \tag{1.1}$$

where λ is the wavelength of exposing radiation, k_1 is a process- and material-dependent parameter less than unity, and $NA = n \, sin\theta_{max}$ is the numerical aperture (n is the refractive index), equal to the refractive index of the surrounding medium (1 for air) times the sine of the angle q subtended by the objective lens of the system. Either the wavelength must be decreased or the NA of the system increased to improve tool resolution. In the most advanced production exposure systems, the exposure wavelength is 193 nm and the NA is approaching 1, the fundamental limit for imaging in air. However, if a fluid with a higher refractive index is interposed between resist film and the objective lens, this limit is eased, paving the way to improved resolution. There is today an intensive industry effort to adapt 193 nm projection exposure tools to such immersion imaging, with the goals of improved process control and ultimately improved resolution compared to dry imaging systems.

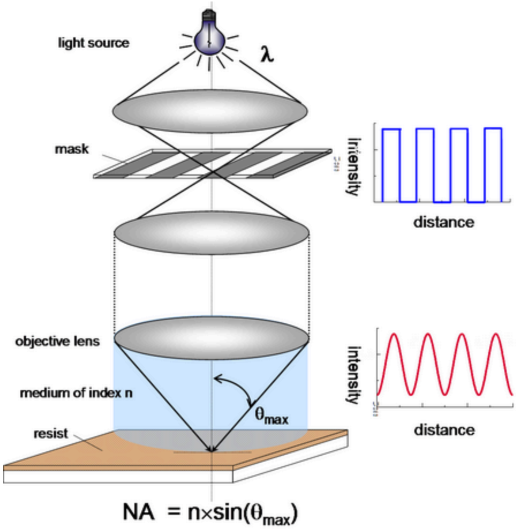


Figure 1.3: Schematic diagram of the optics of a projection exposure lithography system

Dry-etching

Plasma etching is a process in which a solid film is removed by a chemical reaction with ground state or excited state neutral species. Plasma etching is often enhanced or induced by energetic ions generated in a gaseous discharge. Here is a schematic view of microscopic processes occuring in plasma etching:

1. an RF field accelerates the electrons, that collide with gas atoms and molecules; the glow (light emission) is produced by the de-excitation of some electronically excited atoms and molecules.

2. Some Other atoms and molecules collide with high-energy electrons, and they are ionized to form radicals, atoms, and ions. The active species are transported to the wafer surface where they can be absorbed or desorbed:

   - the first ones react with the wafer surface to form etch products;
   - the second ones are desorbed from the wafer surface without reaction.

3. If etch products are volatile, they desorb to a gas phase. An inert gas is used to generate reactive species, like F or CL radicals. These radicals react with the wafer surface to form volatile products. The chemical compound, must be volatile so that it can be pumped out of the reactor.

Plasma etching comprises chemical, ion-sputter, and ion-enhanced plasma etching. All these methods are based on the generation of plasma by an RF discharge in a gas at low pressure. But two basic methods can be distinguished:

Physical etching: the surface is bombarded, at high speed, by positive ions. Physical etching can yield anisotropic profiles, but it has a low etch selectivity and a high bombardment-induced damage.

Chemical etching: the neutral reactive species generated by the plasma interact with the material surface to form volatile products. Chemical etching shows a high etch rate and a good selectivity, produces low ion bombardment-induced damage, and yields isotropic profiles

Combinations of these two methods give anisotropic etch profiles, good selectivity, and low bombardment-damage.

1.4 Front-end processing

Front-end processing refers to the formation of the transistors directly on the silicon. The raw wafer is engineered by the growth of an ultra-pure, virtually defect-free silicon layer through epitaxy. In the most advanced logic devices, prior to the silicon epitaxy step, tricks are performed to improve the performance of the transistors to be built. One method involves introducing a *straining step* wherein a silicon variant such as silicon-germanium (SiGe) is deposited. Once the epitaxial silicon is deposited, the crystal lattice becomes stretched somewhat, resulting in improved electronic mobility. Another method, called ”silicon on insulator” technology involves the insertion of an insulating layer between the raw silicon wafer and the thin layer of subsequent silicon epitaxy. This method results in the creation of transistors with reduced parasitic effects.

1.4.1 Gate oxide and implants

Front-end surface engineering is followed by: growth of the gate dielectric, traditionally silicon dioxide (SiO2), patterning of the gate, patterning of the source and drain regions, and subsequent implantation or diffusion of dopants to obtain the desired complementary electrical properties. In memory devices, storage cells, conventionally capacitors, are also fabricated at this time, either into the silicon surface or stacked above the transistor.

1.5 Back-end processing

1.5.1 Metal layers

Once the various semiconductor devices have been created they must be interconnected to form the desired electrical circuits. This *back end of line* (BEOL, the latter portion of the wafer fabrication, not to be confused with ”back end” of chip fabrication which refers to the package and test stages) involves creating metal interconnecting wires that are isolated by insulating dielectrics. The insulating material was traditionally a form of SiO2 or a silicate glass, but recently new low dielectric constant materials are being used. These dielectrics presently take the form of SiOC and have dielectric constants around 2.7 (compared to 3.9 for SiO2), although materials with constants as low as 2.2 are being offered to chipmakers.

1.5.2 Interconnect

Historically, the metal wires consisted of aluminium. In this approach to wiring often called *subtractive aluminium*, blanket films of aluminium are deposited first, patterned, and then etched, leaving isolated wires. Dielectric material is then deposited over the exposed wires. The various metal layers are interconnected by etching holes, called "*vias*", in the insulating material and depositing tungsten in them with a CVD technique. This approach is still used in the fabrication of many memory chips such as dynamic random access memory (DRAM) as the number of interconnect levels is small, currently no more than four.
More recently, as the number of interconnect levels for logic has substantially increased due to the large number of transistors that are now interconnected in a modern microprocessor, the timing delay in the wiring has become significant prompting a change in wiring material from aluminium to copper and from the silicon dioxides to newer low-K material. This performance enhancement also comes at a reduced cost via damascene processing that eliminates processing steps. In damascene processing, in contrast to subtractive aluminium technology, the dielectric material is deposited first as a blanket film, and is patterned and etched leaving holes or trenches. In *single damascene* processing, copper is then deposited in the holes or trenches surrounded by a thin barrier film resulting in filled vias or wire *lines* respectively. In *dual damascene* technology, both the trench and via are fabricated before the deposition of copper resulting in formation of both the via and line simultaneously, further reducing the number of processing steps. The thin barrier film, called copper barrier seed (*CBS*), is necessary to prevent copper diffusion into the dielectric. The ideal barrier film is as thin as possible. As the presence of excessive barrier film competes with the available copper wire cross section, formation of the thinnest continuous barrier represents one of the greatest ongoing challenges in copper processing today.

As the number of interconnect levels increases, planarization of the previous layers is required to ensure a flat surface prior to subsequent lithography. Without it, the levels would become increasingly crooked and extend outside the depth of focus of available lithography, interfering with the ability to pattern. *CMP* (Chemical Mechanical Planarization) is the primary processing method to achieve such planarization although dry *etch back* is still sometimes employed if the number of interconnect levels is no more than three.

1.6 Other steps

1.6.1 Wafer test and device test

The highly serialized nature of wafer processing has increased the demand for metrology in between the various processing steps. Wafer test metrology equipment is used to verify that the wafers have not been damaged by previous processing steps up until testing. If the number of dies, that are the integrated circuits that will eventually become chips, etched on a wafer exceeds a failure threshold (i.e. too many failed dies on one wafer), the wafer is scrapped rather than investing in further processing.

Once the front-end process has been completed, the semiconductor devices are subjected to a variety of electrical tests to determine if they function properly. The proportion of devices on the wafer found to perform properly is referred to as the yield.

The fab tests the chips on the wafer with an electronic tester that presses tiny probes against the chip. The machine marks each bad chip with a drop of dye. The fab charges for test time; the prices are on the order of cents per second. Chips are often designed with *testability features* such as *built-in self-test* to speed testing, and reduce test costs.

Good designs try to test and statistically manage corners: extremes of silicon behavior caused by operating temperature combined with the extremes of fab processing steps. Most designs cope with more than 64 corners.

1.6.2 Die preparation and packaging

Plastic or ceramic packaging involves mounting the die, connecting the die pads to the pins on the package, and sealing the die. Tiny wires are used to connect pads to the pins. In the old days, wires were attached by hand, but now purpose-built machines perform the task. Traditionally, the wires to the chips were gold, leading to a *lead frame* of copper, that had been plated with solder, a mixture of tin and lead. Lead is poisonous, so lead-free *lead frames* are now mandated by ROHS (acronym of Restriction of Hazardous Substances Directive).

Chip-Scale Package (CSP) is another packaging technology. A plastic dual in-line package, like most packages, is many times larger than the actual die hidden inside, whereas CSP chips are nearly the size of the die. CSP can be constructed for each die before the wafer is diced.

The packaged chips are retested to ensure that they were not damaged during packaging and that the die-to-pin interconnect operation was performed correctly. A laser etches the chip's name and numbers on the package.

1.7 Quality control

A quality control system aims to reach the desired product quality. In the semiconductor manufacturing process, there are multiple factors that introduce variability in the product (process variations, process drifts, external disturbance, etc.). Process variables are continuous time signals (for example pressure or temperature inside a processing chamber) and are measured by sensors.

A quality control system must be able to detect the need to adjust tool conditions, process conditions, and process inputs. Tool and process conditions are usually set at the start of manufacturing process and can be fixed after the end of the process; on the contrary, process inputs can be fixed during the process as well. Therefore, quality control applied to process inputs, can be of two types:

ex-situ control: process inputs are adjusted after process completion, if the quality variable is out of specified range. The adjustment process hinges on statistical analysis of metrology data and operator experience. This type of control is suitable if the process does not require, or cannot support, realtime changes.

In-situ control: process inputs are adjusted in run-time. This control requires run-time monitoring of the process, to be able to modify the process inputs properly.

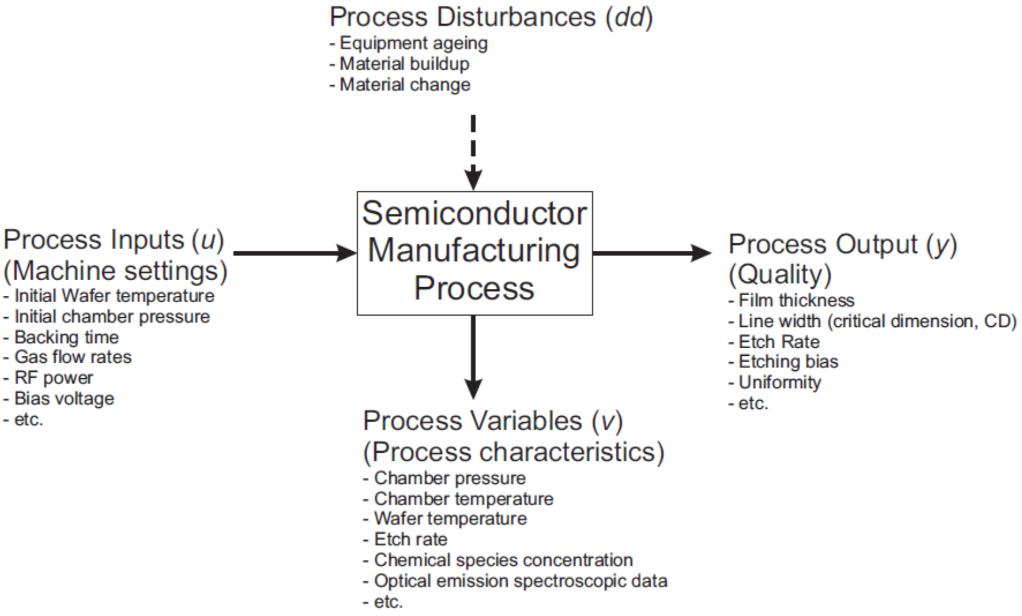


Figure 1.4: Semiconductor machining process: inputs, outputs, variables and disturbances.

Some of the inputs, outputs, process variables, and disturbances of the semiconductor manufacturing process are listed in Figure 1.4 (picture taken from [14]).

In the next subsection two class of quality control techniques will be presented: Statistical Process Control (SPC) and Advanced Process Control (APC).

1.7.1 Statistical Process Control - SPC

SPC is a control technique in which control charts are used to monitor the process output in order to detect a possible out-of-control conditions. The basic steps to build a control chart are:

- the gathering of information and data about the process;
- estimate the mean μ and the standard deviation σ;
- set the Upper Control Limit (UCL) and the Lower Control Limit (LCL); to make an example, the Shewhart's chart for individual data sets:

  $UCL = \mu + 3\sigma$, and $LCL = \mu - 3\sigma$ (where μ is the mean and σ is the standard deviation).

The adjustment of an out-of-control process follows a procedure, called OCAP (Out of Control Action Plan), that must be defined in conjunction with the design of the Control Chart.

1.7.2 Advanced Process Control - APC

The final goal of APC techniques is to keep under control multivariable process variations by means of feedback and feed-forward control methods. A general APC system has three components:

- a model based on historical process data and process knowledge;
- real-time process information;
- control and optimization algorithms.

APC methods use metrology data of product variables to feedback the control algorithm. The feedback feed-forward loop uses a process model to adjust the inputs in order to maintain the desired quality level (target) of the output (i.e. Critical Dimension (CD), thickness, and so on).

1.7.3 APC versus SPC

Traditional semiconductor manufacturing obtains accurate results depending on pre-set process recipes, whose execution is monitored and validated using Statistical Process Control (SPC). Monitoring wafer fabrication with SPC charts is still widely used; however, it cannot address the needs of the most advanced processes, which are far more susceptible to variability. Instead, a collection of methods referred to as Advanced Process Control (APC) have been introduced to improve the performance, yield, throughput, and flexibility of the manufacturing process. These methods include run-to-run, wafer-to-wafer, and within-wafer control. In contrast to traditional single variable-based SPC methods, APC systems have embraced multivariate statistical techniques such as Principal Components Analysis(PCA), in combination with feed-forward and feedback mechanisms. These advances have helped improve process yield through quick Fault Detection and Classification (FDC), as well as through dynamic recipe optimization.

1.7.4 Run-to-Run control

As we have just aforementioned, in semiconductor manufacturing processes, a most important module of APC is Run-to-Run control (R2R)[10]

Definition 1.1 *R2R control is a form of discrete process and machine control in which, with respect to a particular machine process, the product recipe (machine settings) is modified ex-situ, i.e., between machine runs*[1]*, so as to minimize process drift, shift, and variability.*

In order to modify the recipe to address the process drift, shift and other variability, the current tool and wafer states need to be estimated.

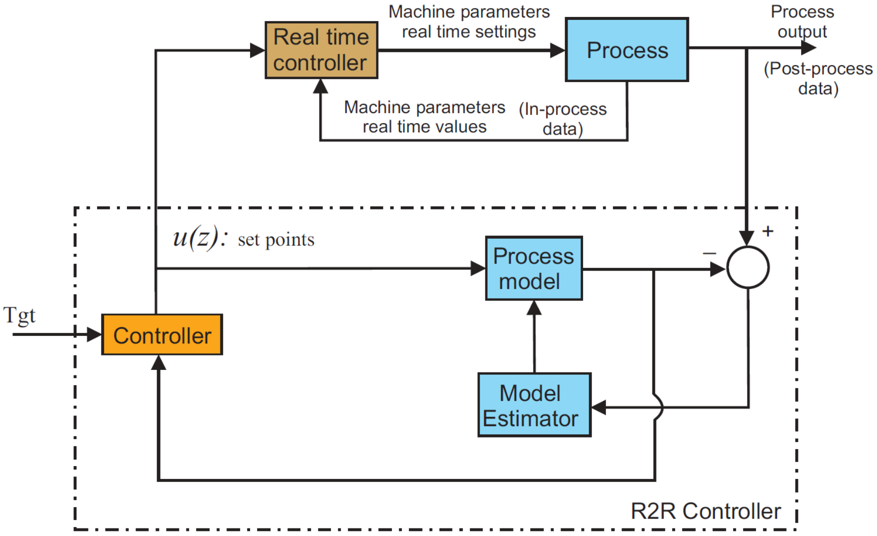


Figure 1.5: $R2R$ control in semiconductor manufacturing.

[1] In semiconductor manufacturing, a run refers to a single process performed on one or more wafers (usually a lot).

In R2R control, post-processing of data from the previous run is used to calculate new set-points of the realtime controller for the next run. A R2R controller (view Figure 1.5, picture taken by [14]), usually has three main components:

- a process model, to describe the input-output relationship;
- an estimator, to estimate the process state and the difference between observed results and the target;
- a control law.

The *R2R* control is in fact lot-to-lot (*L2L*) control since the recipe settings are kept the same for all the wafers in a lot. In Figure 1.6 (picture taken form [14]) an instance of *R2R* control is shown.

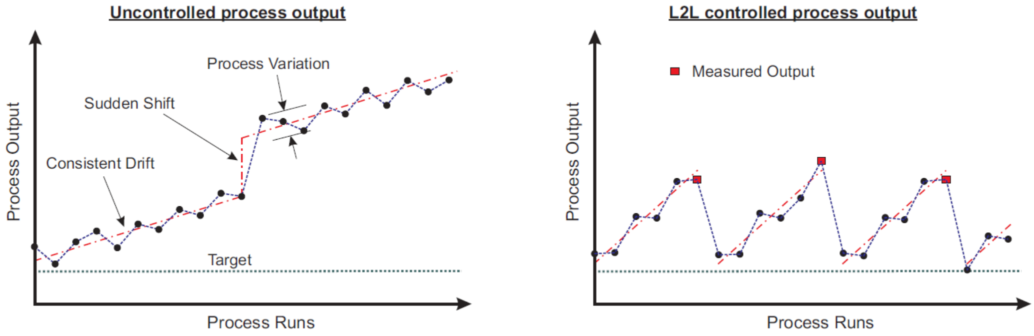


Figure 1.6: *R2R* (*L2L*) process control.

Different control laws lead to different types of R2R controllers: the *Exponentially Weighted Moving Average* (**EWMA**) and the *double Exponentially Weighted Moving Average* (**dEWMA**) are the most widely used in semiconductor manufacturing process. The next two subsections briefly present the EWMA and dEWMA methods.

EWMA

In the Exponentially Weighted Moving Average method, the process is modeled as

$$y(k) = \alpha u(k) + b(k) + \epsilon(k) \tag{1.2}$$

where:

- $y(k)$ is the output at the end of run k;
- α is the process gain;
- $u(k)$ is the input (machine/recipe setting) at the start of run k;

- $b(k)$ is the offset term;
- $\epsilon(k)$ is white noise with zero mean and variance σ^2.

The initial estimates of process gain $\hat{\alpha}(0)$, offset term $\hat{b}(0)$ and control value $u(0)$ are obtained via DOE (Design Of Experiments) techniques. The estimate of the offset term is then updated at each process run using an EWMA filter, while the process gain estimate remains unchanged:

$$\hat{b}(k) = \omega[y(k) - \hat{\alpha}u(k)] + [1 - \omega]\hat{b}(k-1) \tag{1.3}$$

where $\omega \in (0, 1)$ is a discounting factor used to dampen the influence of old data in favor of the new data $\hat{b}(k)$. Finally, the control law for the next process run is simply plant inversion or deadbeat controller:

$$u(k+1) = \frac{Tgt - \hat{b}(k)}{\hat{\alpha}} \tag{1.4}$$

where $Tgt \in R$ is the desired target value for the process output.

dEWMA

The simple EWMA control method as described above is adequate for controlling processes with small disturbances and slow changes in output variation. However, the process output result in steady state error from the Tgt if there is a consistent process drift. For the control of such processes, a predictor-corrector or double EWMA (dEWMA) control scheme is employed. In the dEWMA control method, the process is modeled as:

$$y(k) = \alpha u(k) + b(k) + \delta k \epsilon(k) \tag{1.5}$$

where δk is an average process drift for the k^{th} run.

The updating formula of the offset and drift terms uses two EWMA filters (with two discounting factors ω_1 and ω_2):

$$\hat{b}(k) = \omega[y(k) - \hat{\alpha}u(k)] + [1 - \omega_1]\hat{b}(k-1) \tag{1.6}$$

$$\hat{\delta}(k) = \omega[y(k) - \hat{\alpha}u(k) - \hat{b}(k-1)] + [1 - \omega_2]\hat{\delta}(k-1) \tag{1.7}$$

The control law for the $(k+1)^{th}$ process run is:

$$u(k+1) = \frac{Tgt - \hat{b}(k) - \hat{\delta}(k)}{\hat{\alpha}} \tag{1.8}$$

where Tgt is the usual output target.

Chapter 2

Virtual Metrology: a Survey

2.1 The Holy Grail of Metrology

The semiconductor industry is risk sensitive and reluctant to adopt novel metrology technologies unless they provably address concerns of process stability, tool contamination and ownership costs. To realize widespread adoption in high-volume semiconductor manufacturing facilities, novel metrology technologies must offer the following attributes:

Spatial and temporal wafer state: information Current in-line and in-situ metrology provide the necessary process state information to obtain wafer-to-wafer and lot-to-lot uniformity. However, as feature sizes shrink even further, within wafer uniformity becomes critical. As a result, spatially resolved wafer state information has now become necessary.

Cost effective reliable integration: the integration of metrology tools with process tools often requires significant software and hardware modifications. This can be time-consuming, expensive, and can compromise equipment reliability. It can also affect process stability and can make production tools more vulnerable to false alarms due to metrology errors. Consequently, the large initial capital investment and the increased operating costs of metrology integration must be justified by the anticipated financial gains resulting from improved process capabilities and increased yields.

Ease of deployment: in many processes periodic wafer state measurements suffice to ensure process uniformity and repeatability. In these situations, the costs associated with permanent integration of the necessary metrology cannot be justified. However, taking the tool off-line for inspection and calibration can also be prohibitively expensive. Hence, there is a need for simple and rapid metrology techniques which can be periodically deployed without making any modifications to the process tool.

No influence on process: the semiconductor industry expends considerable resources on maintaining an ultra-clean production environment to prevent

> catastrophic yield loss due to contamination. Any serious metrology choice should not contaminate the process being sensed. In addition, the metrology tool itself should not affect the process or distort the variables being sensed.

2.2 An introduction to the Virtual Metrology

Virtual metrology can be defined as the prediction of metrology variables (either measurable or non-measurable) using information about the state of the process and/or product[7]. VM can be realized by utilizing the pre-process product metrology data and more importantly the process data from the underlying machine that is generally collected in real time for FD analysis. A typical fault detection and classification (FDC) system collects equipment data (referred to as process variables in this dissertation) for every process run. This enormous amount of data (involving hundreds of variables) can be used for VM purposes. The VM data obtained for every process can then be used in a feedback control scheme to provide R2R control for every product.

In semiconductor manufacturing, equipment monitoring is a primary need in order to ensure stable production. In current practice, a wafer (or few wafers) from the processed lot are measured. Wafer metrology data are taken so representative of the whole lot, and are used to adjust process inputs for the next run ($L2L$ control). In Figure 2.1 (picture taken from [15]) an example of $L2L$ is depicted.

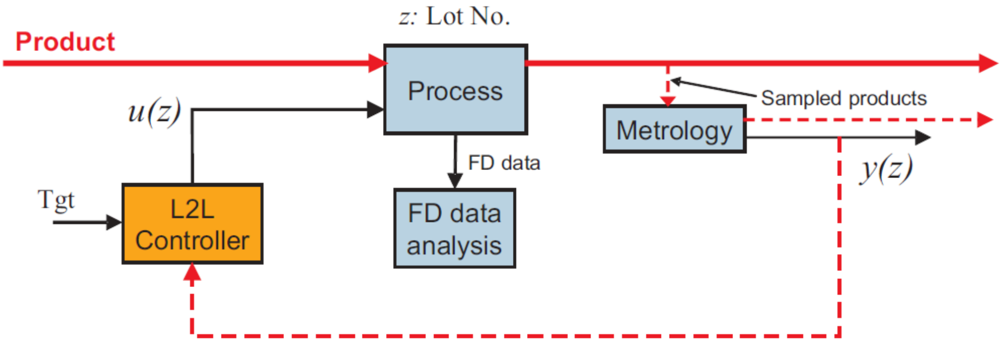


Figure 2.1: $L2L$ control without VM module.

Although metrology data alone can be used for online feedback control, a significant time delay can exist in the feedback loop due to sampled metrology, physical location of the (off-line) metrology station, time taken by metrology, and the product going through a number of processing stations. Thus, a considerable number of faulty products could be produced before a corrective action is taken. In order to reduce the time delay, metrology frequency may be reduced by selectively inspecting at a subset of the production stations. This reduction in metrology frequency is done to reduce the cost of metrology in terms of up front cost of metrology stations and lost throughput associated with the time required for metrology.

Hence the need for Virtual Metrology (VM) techniques able to estimate the missing metrology data.

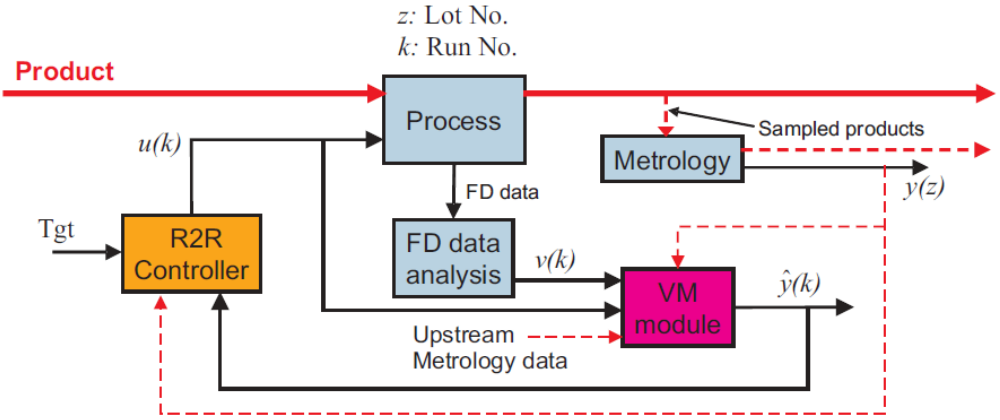


Figure 2.2: $R2R$ control using VM for a semiconductor manufacturing process.

The formulation of VM and $R2R$ control at the wafer level for the process shown in Figure 2.1 is pictured schematically in Figure 2.2 (illustration taken from [16]).

The next subsection aims to describe the state of the art in Virtual Metrology.

2.2.1 VM in semiconductor manufacturing

Virtual Metrology (VM) is a class of methods aiming to estimate metrology values, given process data and previous metrology information.

Ensuring stable wafer fabrication in semiconductor manufacturing requires periodic tool performance monitoring. Tool performance can be monitored:

- by analyzing real-time process variable signals acquired during the process;
- by measuring the wafers after the process run ends.

A typical fault detection and classification system monitors tool performance by analyzing several process variables in real-time during processing. Measurement of wafers at the metrology station provides a complimentary capability by monitoring product quality that may be related to tool or process drifts and variations. Metrology data is also used in the R2R control of the wafer quality variables[10]. In most semiconductor manufacturing processes, metrology is performed on a subset of wafers in a lot and, therefore the $R2R$ control is limited to $L2L$. In practice it is usually not feasible to measure every wafer coming out of a process, which necessitates the need for a VM framework to provide the required metrology data to enable W2W control [10, 11].

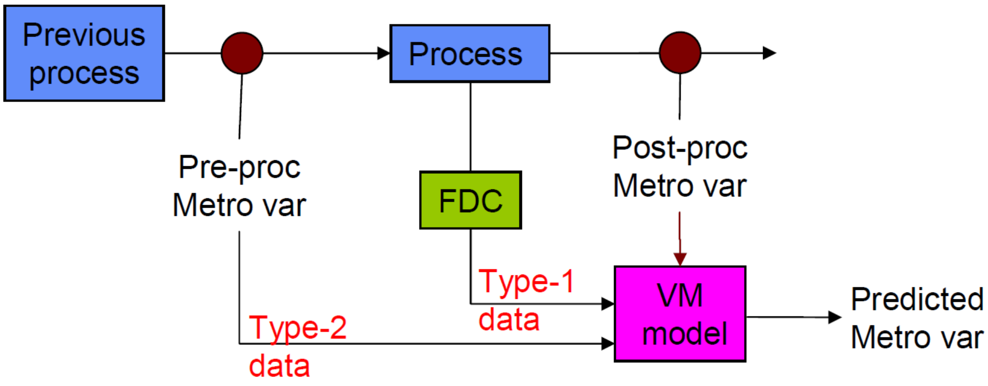


Figure 2.3: Type-1 and type-2 data for VM.

There are two types of data that can be used to enable VM: process variables and actual metrology data from upstream processes referred to as *type-1* and *type-2* data in Figure 2.3 (picture taken from [16]), respectively[14]. In addition, actual metrology data after processing of the lot can also be used in the VM module. The process variables (*type-1* data), collected for FDC purposes for every process run, are direct indicators of the state of the process, i.e., they contain information about the resulting quality variables behavior like chamber pressure, chamber temperature, wafer temperature, optical emission spectroscopic data, gas flows/concentrations and so on. On the other hand, wafer quality characteristics obtained after a process also depend on the upstream processing of that wafer , for instance wafer critical dimension (CD) after etching process strongly depends upon CD after lithography development (*type-2* data from previous operation). Generally speaking, *type-2* data are measures taken after the end of the process. Thus for a VM module to accurately predict wafer quality it needs both *type-1* and *type-2* data and, as it has been said, a well-built VM module can improve the throughput and reduce the need for actual metrology operations.

On the other hand, the VM module can also be designed to predict the CDs of every wafer in the lot. This kind of approach, that requires accurate data collection and wafer tracking, is called Wafer-to-Wafer control ($W2W$ Metrology[3, 6, 13]).

2.2.2 VM module methodology for individual process

In this section we propose a methodology for VM Module for individual process. This methodology is composed of three successive stages:

Stage 1: Data Pre-processing - The aim of this stage is to assure the quality of the data which will be the inputs of the VM models in the Stage 2. The data pre-process includes three steps:

- **Step 1: Data Sources** - To define pilot unit processes, individual process, technology, family of products. After the definition of the family of products, to select the recipes and its important steps that will be used into VM module development. The input data of VM models are collected from two sources (see section 2.2.1): sensor data of production equipment (FDC data) and measurements data of metrology equipment. To assure the quality and effectiveness of VM models it is necessary to do preliminary quality studies of process and metrology equipments. It can be the variance analysis, like as gauge capability of process and metrology equipments. In this step the choice of technology, family of products, recipes and equipments with high capability and stability is mandatory.
- **Step 2: Data Acquisition** - To define two raw data sets acquisitions both including the FDC data from production equipment (**X**) and measurements data from metrology equipment (**Y**). The two raw data sets can be collected from two different periods of production, between 2 and 6 months, for example. Another alternative is to collect a first raw data set from historical data base of production and a second raw data set from Design of Experiments (DOE).
- **Step 3: Data Consolidation** - To define the pretreatment of the two raw data sets collected during Step 2. This includes performing the data cleaning and the statistical data analysis. Data cleaning includes to identify and to remove the outliers, the missing values and the data from out of control production. Statistical data analysis include the data normalization, the data correlation studies and the data reduction. The data reduction, as stepwise regression, methods can be used to remove the redundant data and select only the critical variables. Moreover, the multivariate analysis, as principal components analysis, can be used to reduce the quantity of columns of the input matrices, **X** and **Y**. After the pretreatment of two raw data sets, we will have two off line input data sets for the Phase 2. The first one will be able to be separated in *Training Data Set* and *Running Data Set* to construct the VM prediction models. The second one will be able to be used as a *Validation Data Set* for comparison and validation of models.

Stage 2: VM Module Development - The aims of this stage are to build different prediction models, to compare them and to validate the best model to perform the VM Module. This stage includes three steps:

- **Step 4: VM Modeling** To choose the nonlinear prediction methods. To build each prediction models in two levels: the Training Level with the Training Data Set and the Running Level with the Running Data Set.

- **Step 5: Models Comparisons** - To define the performance indices from the robustness and prediction accuracy criteria. To use the Validation Data set for validation and assessment of the models from Step 4. The goal of this step is to select the best model relative to the performance indices.

- **Step 6: VM Module** To perform the VM Module with the adjustments of the best model chosen in the Step 5.

Stage 3: VM Module Implementation - The objective of this stage is to define the steps to integrate the VM module from Step 6 of Stage 2 into an industrial environment. This phase includes three steps:

- **Step 7: VM Module Tests** - The aim of this step is to perform off line tests with off line data from production in order to identify problems of the model stability, the model capability and to evaluate results of model when the process drifts. The goal is to define a prototype for off line VM Module implementation.

- **Step 8: VM Module in Production** - To define architectural guidelines for integration of real time VM Module in an industrial environment. Provide guidelines for the full integration of the VM Module into the Manufacturing System.

- **Step 9: VM Module Consolidation** - To define the Maintenance Policies for the update of real time VM Module.

2.3 Variables Selection Techniques

During semiconductor manufacturing process, a large amount of information is recorded from etching machines and various other diagnostic. This situation leads to a surplus of available data for each wafer processed. Deciding which variables are most useful to explain variations in process output (in our case the output was the CVD thickness) is a challenging task. Modeling from first principals is a complicated option and leads to computer models that take hours or days to compute seconds of process simulation. Relating process tool parameters to process parameters on a nanometre scale is a daunting task, and we often turn to statistical methods to model variations in the examined process. This work uses three different statistically-based methods for variable selection.

2.3.1 Principal Component Analysis

Principal component analysis (PCA) is a multivariate technique that analyzes a data set in which observations are described by several inter-correlated quantitative dependent variables. Its goal is to extract the important information from the set, to represent it as a set of new orthogonal variables called principal components (PCs), and to display the pattern of similarity of the observations and of the variables as points in maps. Mathematically, PCA depends upon the eigen-decomposition of positive semi-definite matrices and upon the singular value decomposition (SVD) of rectangular matrices.

The data set to be analyzed by PCA comprises n observations (rows) described by m variables (columns) and it is represented by the matrix $\mathbf{X} \in \Re^{n \cdot m}$, whose generic element is $x_{i,j}$ ($i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$).

The matrix $\mathbf{X}$ has rank l where $l \leq [n, m]$.

In general, the data set will be pre-processed before the analysis. Almost always, the columns of $\mathbf{X}$ will be centered so that the mean of each column is equal to 0. If in addition, each element of $\mathbf{X}$ is divided by $\sqrt{m}$, the analysis is referred to as a *covariance PCA* because, in this case, the matrix $\boldsymbol{X^T X}$ is a covariance matrix. In addition to centering, when the variables are measured with different units, it is customary to standardize each variable to unit norm: this is obtained by dividing each variable by its norm. In this case, the analysis is referred to as a *correlation PCA* because, then, the matrix $\boldsymbol{X^T X}$ is a correlation matrix. Scaling to unit variance gives all variables equal importance for the analysis.

The matrix $\mathbf{X}$ has the following singular value decomposition (SVD):

$$\mathbf{X} = \mathbf{P \Delta Q^T} \tag{2.1}$$

where $\mathbf{P}$ is the n x l matrix of left singular vectors, $\mathbf{Q}$ is the m x l matrix of right singular vectors, and $\mathbf{\Delta}$ is the diagonal matrix of singular values. Note that $\mathbf{\Delta}^2$ is equal to $\mathbf{\Lambda}$ which is the diagonal matrix of the (non-zero) eigenvalues of $\mathbf{X^TX}$ and $\mathbf{XX^T}$.

The *inertia of a column* is defined as the sum of the squared elements of this column and is computed as:

$$\gamma_j^2 = \sum_{i=1}^{n} x_{i,j}^2 \tag{2.2}$$

The sum of all the γ_j^2 is denoted Γ and it is called the *inertia of the data set* or the *total inertia*. Note that the total inertia is also equal to the sum of the squared singular values of the data set.

The *center of gravity of the rows* (also called centroid or barycenter), denoted $\mathbf{g}$, is the vector of the means of each column of $\mathbf{X}$. When $\mathbf{X}$ is centered, its center of gravity is equal to the row vector $0^T \in \Re^{1xm}$.

The Euclidean distance of the i-th observation to g is equal to

$$d_{i,\mathbf{g}}^2 = \sum_{j=1}^{m} (x_{i,j} - g_j)^2 \tag{2.3}$$

When the data are centered, Equation (2.3) reduces to

$$d_{i,\mathbf{g}}^2 = \sum_{j=1}^{m} x_{i,j}^2 \tag{2.4}$$

Note that the sum of all $d_{i,\mathbf{g}}^2$ is equal to Γ which is the inertia of the data set.

Goals of PCA

The goals of PCA are to:

- extract the most important information from the data set;
- compress the size of the data set by keeping only this important information;
- simplify the description of the data set;
- analyze the structure of the observations and the variables.

In order to achieve these goals, PCA computes new variables called *principal components* which are obtained as linear combinations of the original variables. The first principal component is required to have the largest possible variance (i.e., inertia and therefore this component will “explain” or “extract” the largest part of the inertia of the data table). The second component is computed under the constraint of being orthogonal to the first component and to have the largest possible inertia. The other components are computed likewise. The values of these new variables for the observations are called factor *scores*, these factors scores can be interpreted geometrically as the projections of the observations onto the principal components.

Finding the components In PCA, the components are obtained from the singular value decomposition of the data set $\mathbf{X}$. Specifically, with $\boldsymbol{X} = \boldsymbol{P} \cdot \boldsymbol{\Delta} \cdot \boldsymbol{Q^T}$ (cf. Equation (2.1)), the n x l matrix of factor scores, denoted $\mathbf{T}$ is obtained as:

$$\mathbf{T} = \mathbf{P\Delta} \tag{2.5}$$

The matrix $\mathbf{Q}$ gives the coefficients of the linear combinations used to compute the factors scores. This matrix can also be interpreted as a projection matrix because multiplying $\mathbf{X}$ by $\mathbf{Q}$ gives the values of the projections of the observations on the principal components. This can be shown by combining Equations (2.1) and (2.5) as:

$$\mathbf{T} = \mathbf{P\Delta} = \mathbf{P\Delta QQ^T} = \mathbf{XQ} \tag{2.6}$$

The components can also be represented geometrically by the rotation of the original axes. In this context, the matrix $\mathbf{Q}$ is interpreted as a matrix of direction cosines (because $\mathbf{Q}$ is orthonormal). The matrix $\mathbf{Q}$ is also called a *loading matrix.* In this context, the matrix $\mathbf{X}$ can be interpreted as the product of the factors score matrix by the loading matrix as:

$$\begin{aligned}\mathbf{X} &= \mathbf{t}_1\mathbf{q}_1^T + \mathbf{t}_2\mathbf{q}_2^T + \ldots + \mathbf{t}_l\mathbf{q}_l^T & (2.7)\\ &= \mathbf{TQ^T} & (2.8)\end{aligned}$$

with $\mathbf{T^TT} = \mathbf{\Delta}^2$ and $\mathbf{Q^TQ} = I$. This decomposition is often called the *bilinear decomposition* of $\mathbf{X}$.

Conclusions

PCA, as just seen above, is a method used to transform a set of correlated variables into new uncorrelated variables, known as principal components. The first PC is the linear combination of the m original variables that explains the greatest amount of variability ($\mathbf{t}_1 = \mathbf{Xq}_1$). In the m-dimensional variable space, the loading vector $\mathbf{q}_1$ defines the direction of the greatest variance. Overall, loadings represent how the original variables are combined to make the PCs, scores represent original data projected onto the new uncorrelated variables. For a matrix $\mathbf{X}$ of rank l , l PCs cab be calculated; however, the first k ($k < l$) of these may be sufficient to explain the bulk on the variance in the data.

PCA can be used as a variable selection technique by examining the loading vectors for the first few principal components. The variables that contribute the most variance to these components will have the highest loading values. These variables can then be selected as inputs to process output models.

2.3.2 Correlation Methodology

Another method, arguably simpler than principal component analysis, to select important variables is to analyze the linear correlations between each process chamber variable and the process output recorded that, we remember, in this work it was the CVD thickness.

The correlation between two variables is defined as:

$$\rho_{x,y} = \frac{\mathrm{cov}(x,y)}{\sigma_x \sigma_y} = \frac{\mathrm{E}((x-\mu_x)(y-\mu_y))}{\sigma_x \sigma_y} \tag{2.9}$$

where x and y are two variables, with mean μ_x and μ_y and standard deviation σ_x and σ_y, respectively. E is the expected value operator and cov denotes covariance. The correlation coefficient $\rho_{x,y}$ can not exceed 1 in absolute value and it is a measure of the degree of linear relationship between two random variables. The closer the correlation is to -1 or $+1$ the more closely the two variables are related. If $\rho_{x,y}$ is close to 0, it means there is no relationship between the variables. If $\rho_{x,y}$ is positive, it means that as one variable gets larger the other gets larger. If $\rho_{x,y}$ is negative it means that as one gets larger, the other gets smaller (often called an "inverse" correlation).

As correlations measures only the degree of the linear relationship between two variables, it is useful to pass the input variables through non-linear transforms before correlation tests, as a test for some non-linear relationships. For this variable selection technique, each input is raised to a number of powers before correlation testing (e.g. $x^{1,}, x^2, \ldots, x^n$). This increases dramatically the correlation between input and output vectors for some variables.

After all of the variables have been correlated with the output, they are ranked in order of correlation coefficients, and then the most correlated variables are used as inputs to neural-network based models.

2.3.3 Stepwise Regression

Stepwise regression is probably the most widely used variable selection technique. The procedure iteratively constructs a sequence of regression models by adding or removing variables at each step. The criterion for adding or removing a variable at any step is usually expressed in terms of a partial F-test. Let f_{in} be the value of the F-random variable for adding a variable to the model, and let f_{out} be the value of the F-random variable for removing a variable from the model. We must have $\mathrm{f}_{in} \geq \mathrm{f}_{out}$ and usually $\mathrm{f}_{in} = \mathrm{f}_{out}$. Stepwise regression begins by forming a one-variable model using the regressor variable that has the highest correlation with the response variable Y. This will also be the regressor producing the largest F-statistic. We denote with p_0 the number of variables present in the previous model and with p_j the number of variables present in the current model; for example, suppose that at this step, x_1 is selected. At the second step, the remaining $m-1$ candidate variables are examined, and the partial F-statistic cab be expressed as:

$$\mathrm{F}_j = \frac{(RSS0 - RSS_j)/(p_j - p_0)}{RSS_j/(N - p_j - 1)} \quad j = 1, 2, \ldots, m-1 \tag{2.10}$$

where RSS_j is the residual sum-of-squares for the least squares fit of the bigger model with $p_j + 1$ parameters, and $RSS0$ the same for the nested smaller model with $p_0 + 1$ parameters, having $p1 - p0$ parameters constrained to be zero. The F-statistic measures the change in residual sum-of-squares per additional parameter in the bigger model, and it is normalized by an estimate of variance σ^2. The variable for which the partial F-statistic is a maximum is added to the equation, provided that $\mathrm{f}_j > \mathrm{f}_{in}$. Suppose that this procedure indicates that $x_j = x_2$ should be added to the model. Now the stepwise regression algorithm determines whether the variable x_1 added at the first step should be removed. This is done by calculating the F-statistic

$$\mathrm{F}_1 = \frac{(RSS1 - RSS2)/(p_2 - p_1)}{RSS2/(N - p_2 - 1)} \tag{2.11}$$

where $RSS1$ is the residual sum-of-squares for the least squares fit of the model with x_1 and $p_1 + 1$ parameters, and $RSS2$ the same for the model with $x_1 - x_2$ and $p_2 + 1$ parameters.

If the calculated value $\mathrm{f}_1 < \mathrm{f}_{out}$, the variable x_1 is removed; otherwise it is retained, and we would attempt to add a regressor to the model containing both x_1 and x_2.

In general, at each step the set of remaining candidate regressors is examined, and the regressor with the largest partial F-statistic is entered, provided that the observed value of f exceeds f_{in}. Then the partial F-statistic for each regressor in the model is calculated, and the regressor with the smallest observed value of F is deleted if the observed $\mathrm{f} < \mathrm{f}_{out}$. The procedure continues until no other regressors can be added to or removed from the model.

Forward Selection

The forward selection procedure is a variation of stepwise regression and is based on the principle that regressors should be added to the model one at a time until there are no remaining candidate regressors that produce a significant increase in the regression sum of squares. That is, variables are added one at a time as long as their partial F-value exceeds f_{in}. More accurately, if a variable is not currently in the model, the null hypothesis that the term would have a zero coefficient is tested. If there is sufficient evidence to reject the null hypothesis, the term is added to the model.

Forward selection is a simplification of stepwise regression that omits the partial F-test for deleting variables from the model that have been added at previous steps. This is a potential weakness of forward selection; that is, the procedure does not explore the effect that adding a regressor at the current step has on regressor variables added at earlier steps.

Backward Selection

The backward elimination algorithm begins with all m candidate regressors in the model. Then the regressor with the smallest partial F-statistic is deleted if this F-statistic is insignificant, that is, if f < f_{out}. Next, the model with $m - 1$ regressors is fit, and the next regressor for potential elimination is found. The algorithm terminates when no further regressor can be deleted.

Some Comments on Final Model Selection

We have illustrated several different approaches to the selection of variables. The final model obtained from any model-building procedure should be subjected to the usual adequacy checks, such as residual analysis, lack-of-fit testing, and examination of the effects of influential points. We may also consider augmenting the original set of candidate variables with cross-products, polynomial terms (e.g. to multiply by $x^{1,}, x^2, \ldots, x^n$), or other transformations of the original variables that might improve the model.

The f_{in} and f_{out} value limits that are used to judge whether variables are kept or added to the model are set in correspondence with the quality of the model that we establish.

Chapter 3

Artificial Neural Networks

3.1 Introduction

An artificial neural network (*ANN*), usually called neural network (*NN*), is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Neural networks are nonlinear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

Although computing these days is truly advanced, there are certain tasks that a program made for a common microprocessor is unable to perform; even so a software implementation of a neural network can be made with their advantages and disadvantages.
Advantages:

- a neural network can perform tasks that a linear program can not;
- when an element of the neural network fails, it can continue without any problem by their parallel nature;
- a neural network learns and does not need to be reprogrammed;
- it can be implemented in any application;
- it can be implemented without any problem.

Disadvantages:

- the neural network needs training to operate;
- The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated;

- requires high processing time for large neural networks.

Another aspect of the artificial neural networks is that there are different architectures, which consequently requires different types of algorithms, but despite to be an apparently complex system, a neural network is relatively simple.

Artificial neural networks are among the newest signal-processing technologies in the engineer's toolbox. The field is highly interdisciplinary, but our approach will restrict the view to the engineering perspective. In engineering, neural networks serve two important functions: as pattern classifiers and as nonlinear adaptive filters. We will provide a brief overview of the theory, learning rules, and applications of the most important neural network models. Definitions and style of computation an artificial Neural Network is an adaptive, most often nonlinear system that learns to perform a function (an input/output map) from data. Adaptive means that the system parameters are changed during operation, normally called the training phase. After the training phase the ANN parameters are fixed and the system is deployed to solve the problem at hand (the testing phase). The Artificial Neural Network is built with a systematic step-by-step procedure to optimize a performance criterion or to follow some implicit internal constraint, which is commonly referred to as the learning rule. The input/output training data are fundamental in neural network technology, because they convey the necessary information to *discover* the optimal operating point. The nonlinear nature of the neural network processing elements provides the system with lots of flexibility to achieve practically any desired input/output map, i.e., some Artificial Neural Networks are universal mappers. There is a style in neural computation that is worth describing.

An input is presented to the neural network and a corresponding desired or target response set at the output (when this is the case the training is called supervised). An error is composed from the difference between the desired response and the system output. This error information is fed back to the system and adjusts the system parameters in a systematic fashion (the learning rule). The process is repeated until the performance is acceptable. It is clear from this description that the performance hinges heavily on the data. If one does not have data that cover a significant portion of the operating conditions or if they are noisy, then neural network technology is probably not the right solution. On the other hand, if there is plenty of data and the problem is poorly understood to derive an approximate model, then neural network technology is a good choice. This operating procedure should be contrasted with the traditional engineering design, made of exhaustive subsystem specifications and intercommunication protocols. In artificial neural networks, the designer chooses the network topology, the performance function, the learning rule, and the criterion to stop the training phase, but the system automatically adjusts the parameters. So, it is difficult to bring a priori information into the design, and when the system does not work properly it is also hard to incrementally refine the solution. But ANN-based solutions are extremely efficient in terms of development time and resources, and in many difficult problems artificial neural networks provide performance that is difficult to match with other technologies. At present, artificial neural networks are emerging as the technology of choice for many applications, such as pattern recognition, prediction, *system identification*, and control.

3.2 The biological model

Artificial neural networks emerged after the introduction of simplified neurons by McCulloch and Pitts in 1943. These neurons were presented as models of biological neurons and as conceptual components for circuits that could perform computational tasks. The basic model of the neuron is founded upon the functionality of a biological neuron. “Neurons are the basic signaling units of the nervous system” and “each neuron is a discrete cell whose several processes arise from its cell body”.

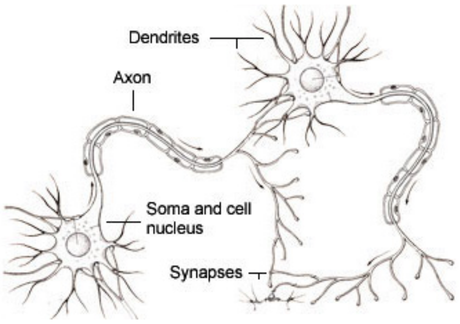


Figure 3.1: The Biological Neuron

The neuron has four main regions to its structure (see Figure 3.1, illustration taken from [30]). The cell body, called also soma, has two offshoots from it, the dendrites, and the axon, which end in presynaptic terminals. The cell body is the heart of the cell, containing the nucleus and maintaining protein synthesis. A neuron may have many dendrites, which branch out in a treelike structure, and receive signals from other neurons. A neuron usually only has one axon which grows out from a part of the cell body called the axon hillock. The axon conducts electric signals generated at the axon hillock down its length. These electric signals are called action potentials. The other end of the axon may split into several branches, which end in a presynaptic terminal. Action potentials are the electric signals that neurons use to convey information to the brain. All these signals are identical. Therefore, the brain determines what type of information is being received based on the path that the signal took. The brain analyzes the patterns of signals being sent and from that information it can interpret the type of information being received. Myelin is the fatty tissue that surrounds and insulates the axon. Often short axons don’t need this insulation. There are uninsulated parts of the axon. These areas are called Nodes of Ranvier. At these nodes, the signal traveling down the axon is regenerated. This ensures that the signal traveling down the axon travels fast and remains constant (i.e. very short propagation delay and no weakening of the signal). The synapse is the area of contact between two neurons. The neurons do not actually physically touch. They are separated by the synaptic cleft, and electric signals are sent through chemical interaction. The neuron sending the signal is called the presynaptic cell and the neuron receiving the signal is called the postsynaptic cell. The signals are generated by the membrane potential, which is based on the differences in concentration of sodium and potassium ions inside and outside the cell membrane. Neurons can be classified by their number of processes (or appendages), or by their function. If they are classified by the number of processes, they fall into three categories. Unipolar neurons have a single process (dendrites and axon are located on the same stem), and are most common in invertebrates. In bipolar neurons, the dendrite and axon are the neuron's two separate processes. Bipolar neurons have a subclass called pseudo-bipolar neurons, which are used to send sensory information to the spinal cord. Finally, multipolar neurons are most common in mammals. Examples of these neurons are spinal motor neurons, pyramidal cells and Purkinje cells (in the cerebellum). If classified by function, neurons again fall into three separate categories. The first group is sensory, or afferent, neurons, which provide information for perception and motor coordination. The second group provides information (or instructions) to muscles and glands and is therefore called motor neurons. The last group, interneuronal, contains all other neurons and has two subclasses. One group called relay or projection interneurons have long axons and connect different parts of the brain. The other group called local interneurons are only used in local circuits.

3.3 The mathematical model

The fundamental building block in an Artificial Neural Network is the mathematical model of a neuron as shown in Figure 3.2 (image taken from [31]). The three basic components of the artificial neuron are:

1. the synapses or connecting links that provide weights, w_j, to the input values, x_j for $j = 1, \ldots, m$.

2. An adder that sums the weighted input values to compute the input to the activation function $v = w_0 + \sum_{j=1}^{m} w_j \, x_j$, where w_0 is called the bias (not to be confused with statistical bias in prediction or estimation) and is a numerical value associated with the neuron. It is convenient to think of the bias as the weight for an input x_0 whose value is always equal to one, so that $v = \sum_{j=0}^{m} w_j \, x_j$.

3. An activation function g, also called a squashing function, that maps v to $g(v)$ the output value of the neuron. This function is a monotone function and an acceptable range of output is usually between 0 and 1, or -1 and 1.

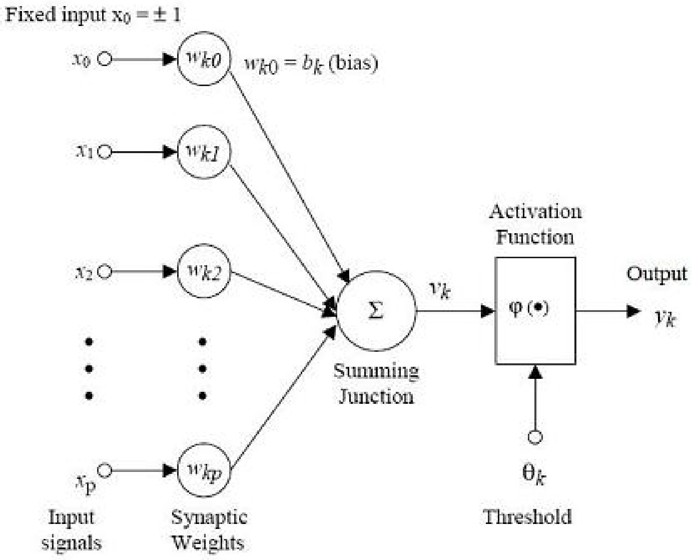


Figure 3.2: The Mathematical Neuron

3.3.1 Activation functions

As mentioned previously, the activation function acts as a squashing function, such that the output of a neuron in a neural network is between certain values (usually 0 and 1, or -1 and 1). In general, there are three types of activation functions, denoted by $\sigma(\cdot)$. First, there is a threshold function which takes on a value of 0 if the summed input is less than a certain threshold value (v), and the value 1 if the summed input is greater than or equal to the threshold value.

$$\sigma(v) = \begin{cases} 0 & \text{if } v < 0\,; \\ 1 & \text{if } v \geq 0 \end{cases}$$

Secondly, there is a piecewise linear function. This function again can take on the values of 0 or 1, but can also take on values between that depending on the amplification factor in a certain region of linear operation.

$$\sigma(v) = \begin{cases} 0 & \text{if } v \leq -\frac{1}{2}\,; \\ v & \text{if } -\frac{1}{2} < v < \frac{1}{2}\,; \\ 1 & \text{if } v \geq \frac{1}{2} \end{cases}$$

Thirdly, there is the sigmoid function:

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

This function is depicted in Figure 3.3 (picture taken from [32]), it can range between 0 and 1, but it is also sometimes useful to use the -1 to 1 range. An example of the sigmoid function is the hyperbolic tangent function:

$$\sigma(v) = \tanh(v).$$

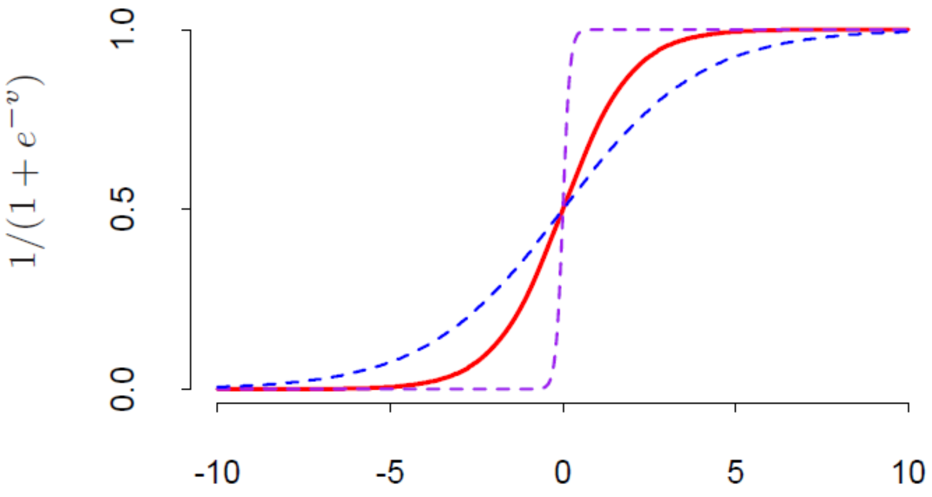


Figure 3.3: Plot of the sigmoid function $\sigma(v) = 1/(1+e^{-v})$ (red curve), commonly used in the hidden layer of a neural network. Included are $\sigma(kv)$ for $k = 1/2$ (blue curve) and $k = 10$ (purple curve). The scale parameter k controls the activation rate, and we can see that large k amounts to a hard activation at $v = 0$. Note that $\sigma(s(v - v_0))$ shifts the activation threshold from 0 to v_0.

3.4 A framework for distributed representation

The artificial neural networks which we describe are all variations on the parallel distributed processing idea. The architecture of each neural network is based on very similar building blocks which perform the processing.

An ANN consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections. A set of major aspects of a parallel distributed model can be distinguished:

- a set of processing units ("neurons", "cells");
- a state of activation y_k for every unit, which equivalent to the output of the unit;
- connections between the units, generally each connection is defined by a weight w_{jk} which determines the effect which the signal of unit j has on unit k;
- a propagation rule, which determines the effective input s_k of a unit from its external inputs;
- an activation function $\sigma(\cdot)_k$, which determines the new level of activation based on the effective input $s_k(t)$ and the current activation $y_k(t)$ (i.e., the update);
- an external input (bias, offset) w_0 for each unit;

- a method for information gathering (the learning rule);
- an environment within which the system must operate, providing input signals and, if necessary, error signals.

3.4.1 Processing units

Each unit performs a relatively simple job: receive input from neighbours or external sources and use this to compute an output signal which is propagated to other units. Apart from this processing, a second task is the adjustment of the weights. The system is inherently parallel in the sense that many units can carry out their computations at the same time. Within neural systems it is useful to distinguish three types of units: input units (indicated by an index **i**) which receive data from outside the neural network, output units (indicated by an index **o**) which send data out of the neural network, and hidden units (indicated by an index **h**) whose input and output signals remain within the neural network. During operation, units can be updated either synchronously or asynchronously. With synchronous updating, all units update their activation simultaneously; with asynchronous updating, each unit has a (usually fixed) probability of updating its activation at a time t, and usually only one unit will be able to do this at a time. In some cases the latter model has some advantages.

3.5 Neural network topologies

Now we discussed the properties of the basic processing unit in an artificial neural network. This section focuses on the pattern of connections between the units and the propagation of data. As for this pattern of connections, the main distinction we can make is between:

- **feed-forward neural networks**, where the data from input to output units is strictly feed-forward. The data processing can extend over multiple (layers of) units, but there are not feedback connections, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.
- **Recurrent neural networks**[1] that contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the neural network will evolve to a stable state in which these activations do not change anymore. In other applications, the change of the activation values of the output neurons are significant, such that the dynamical behaviour constitutes the output of the neural network (for more details see [22]).

[1] For the aim of this work, that is a regression problem, the use of this type of neural networks was not considered.

Classical examples of feed-forward neural networks are the *Perceptron* and *Adaline*. Examples of recurrent networks have been presented by Anderson (1977), Kohonen (1977) and Hopfield (1982).

3.6 Feed-forward neural networks

3.6.1 Single layer networks

Let us begin by examining neural networks with just one layer of neurons (output layer only, no hidden layers). The simplest network consists of just one neuron with the function $\sigma(\cdot)_k$ chosen to be the identity function, $\sigma(v) = v \ \forall v$. In this case notice that the output of the network is $v = \sum_{j=0}^{m} w_j \, x_j$, a linear function of the input vector x with components x_j . If we are modeling the dependent variable y using multiple linear regression, we can interpret the neural network as a structure that predicts a value $\hat{y}$ for a given input vector x with the weights being the coefficients. If we choose these weights to minimize the mean square error using observations in a training set, these weights would simply be the least squares estimates of the coefficients. The weights in neural nets are also often designed to minimize mean square error in a training data set. There is, however, a different orientation in the case of neural nets: the weights are *learned*. The network is presented with cases from the training data one at a time and the weights are revised after each case in an attempt to minimize the mean square error. This process of incremental adjustment of weights is based on the error made on training cases and is known as training the neural net. The almost universally used dynamic updating algorithm for the neural net version of linear regression is known as the Widrow-Hoff rule or the least-mean-square (LMS) algorithm.

It is simply stated: let $x(i)$ denote the input vector x for the i^{th} case used to train the network, and the weights before this case is presented to the net by the vector $w(i)$. The updating rule is $w(i+1) = w(i)+\eta[y(i)-\hat{y}(i)]x(i)$ with $w(0) = 0$ and where η is the learning rate. It can be shown that if the network is trained in this manner by repeatedly presenting test data observations one-at-a-time then for suitably small (absolute) values of η the network will learn (converge to) the optimal values of w. Note that the training data may have to be presented several times for $w(i)$ to be close to the optimal w. The advantage of dynamic updating is that the network tracks moderate time trends in the underlying linear model quite effectively.

A single-layer network has severe restrictions: the class of tasks that can be accomplished is very limited. Moreover a two layer feed-forward network can overcome many restrictions, but do not present a solution to the problem of how to adjust the weights from input to hidden units.

3.6.2 Multilayer feed-forward networks

While there are numerous different ANN architectures that have been studied by researchers, the most successful applications in data mining of neural networks have been *multilayer feed-forward networks*. These are networks that have a layered structure in which there is an input layer consisting of nodes that simply accept the input values and successive layers of nodes that are neurons as depicted in Figure 3.2. The outputs of neurons in a layer are inputs to neurons in the next layer. The last layer is called the output layer. Layers between the input and output layers are known as hidden layers. Figure 3.4 is a diagram for this architecture.

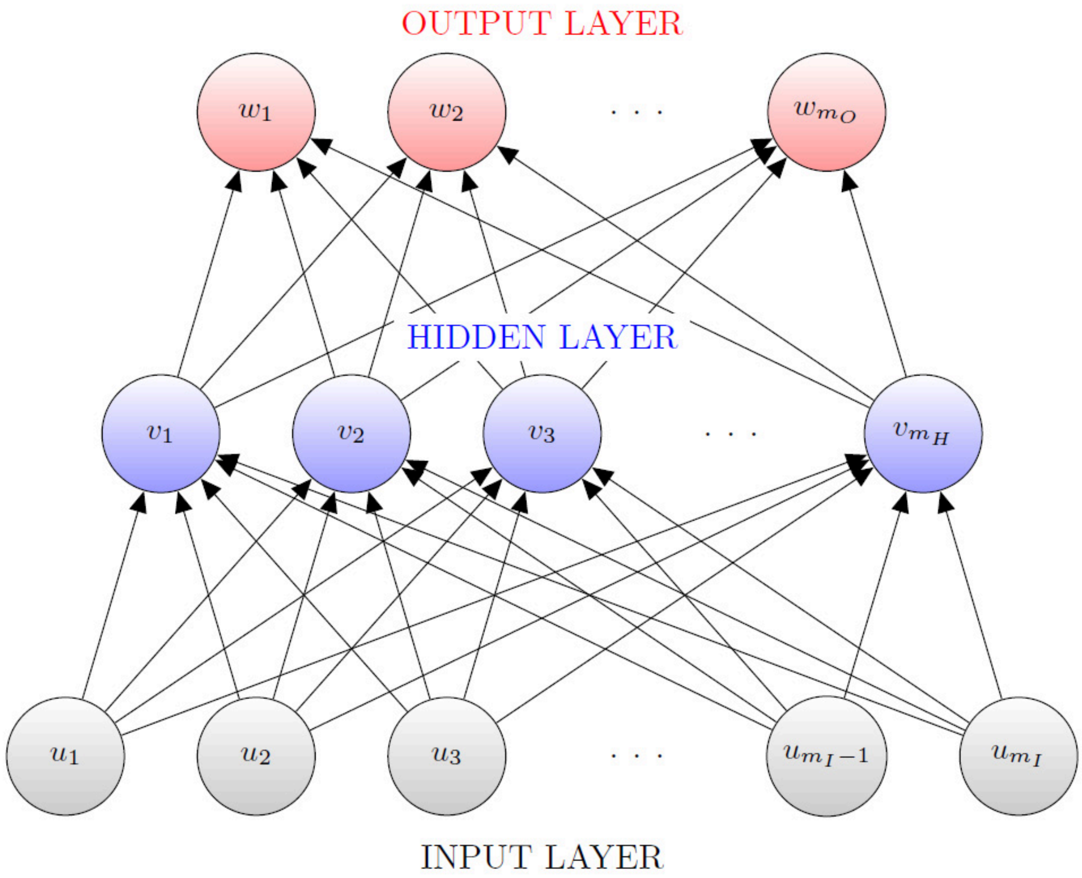


Figure 3.4: An example of a multilayer feed-forward networks.

Specifically, each layer consists of units which receive their input from units from a layer directly below and send their output to units in a layer directly above the unit. There are no connections within a layer. The m_I inputs are fed into the first layer of $m_{H,1}$ hidden units. The input units are merely *fan-out* units; no processing takes place in these units. The activation of a hidden unit is a function $\sigma_k(\cdot)$ of the weighted inputs plus a bias, as given in in equation 3.1

$$y_{k+1}(t) = \sigma_k(s_k(t)) = \sigma_k(\sum_j w_{jk}(t) y_j(t) + \theta_k(t)) \tag{3.1}$$

The output of the hidden units is distributed over the next layer of $m_{H,2}$ hidden units, until the last layer of hidden units $m_{H,L}$, of which the outputs are fed into a layer of m_O output units .

3.7 Learning and training of artificial neural networks

A neural network has to be configured such that the application of a set of inputs produces (either directly or via a relaxation process) the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a priori knowledge. Another way is to *train* the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule.

The possibility of learning is the thing that has attracted the most interest in neural networks. Given a specific task to solve, and a class of functions F, learning means using a set of observations to find f^* in F which solves the task in some optimal sense.

This entails defining a cost function $C : F \rightarrow \mathbb{R}$ such that, for the optimal solution f^*, $C(f^*) \leq C(f)\ \forall f \in F$ (i.e., no solution has a cost less than the cost of the optimal solution).

The cost function C is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost.

For applications where the solution is dependent on some data, the cost must necessarily be a function of the observations, otherwise we would not be modeling anything related to the data. It is frequently defined as a statistic to which only approximations can be made.

3.7.1 Choosing a cost function

While it is possible to define some arbitrary, ad hoc cost function, frequently a particular cost will be used, either because it has desirable properties (such as convexity) or because it arises naturally from a particular formulation of the problem (e.g., in a probabilistic formulation the posterior probability of the model can be used as an inverse cost). Ultimately, the cost function will depend on the task we wish to perform. The three main categories of learning tasks are overviewed below.

We can classify the learning situations in three distinct sorts. These are:

- supervised learning (also called associative learning) in which the network is trained by providing it with input and matching output patterns. These input-output pairs can be provided by an external teacher or by the system which contains the neural network (self-supervised).

- Unsupervised learning, or self-organization, in which an (output) unit is trained to respond to clusters of pattern within the input. In this paradigm the system is supposed to discover statistically salient features of the input population. Unlike the supervised learning paradigm, there is no a priori set of categories into which the patterns are to be classified; rather the system must develop its own representation of the input stimuli.

- Reinforcement learning: this type of learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a feedback response from the environment. The learning system grades its action good (rewarding) or bad (punishable) based on the environmental response and accordingly adjusts its parameters. Generally, parameter adjustment is continued until an equilibrium state occurs, following which there will be no more changes in its parameters. The self-organizing neural learning may be categorized under this type of learning.

3.7.2 Supervised learning

In supervised learning, we are given a set of example pairs (x, y), $x \in X$, $y \in Y$ and the aim is to find a function $f : X \to Y$ in the allowed class of functions that matches the examples. In other words, we wish to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error which tries to minimize the average squared error between the network's output, $f(x)$, and the target value y over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called Multi-Layer Perceptrons, one obtains the common and well-known back-propagation algorithm for training neural networks.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a *teacher*, in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

3.7.3 Unsupervised learning

In unsupervised learning we are given some data x and the cost function to be minimized, that can be any function of the data x and the network's output, $f(\cdot)$.

The cost function is dependent on the task (what we are trying to model) and our a priori assumptions (the implicit properties of our model, its parameters and the observed variables). Furthermore, we remember that the cost function can be rather complicated.

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

3.7.4 Reinforcement learning

In reinforcement learning, data x are usually not given, but generated by an agent's interactions with the environment. At each point in time t, the agent performs an action y_t and the environment generates an observation x_t and an instantaneous cost c_t, according to some (usually unknown) dynamics. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost; i.e., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally, the environment is modeled as a Markov decision process with states $s_1, ..., s_n \in S$ and actions $a_1, ..., a_m \in A$ with the following probability distributions: the instantaneous cost distribution $P(c_t|s_t)$, the observation distribution $P(x_t|s_t)$ and the transition $P(s_t + 1|s_t, a_t)$, while a policy is defined as conditional distribution over actions given the observations. Taken together, the two define a Markov chain. The aim is to discover the policy that minimizes the cost; i.e., the MC for which the cost is minimal.

ANNs are frequently used in reinforcement learning as part of the overall algorithm.

Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

3.8 The back-propagation algorithm

3.8.1 Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation.

Most of the algorithms used in training artificial neural networks employ some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods, simulated annealing, and expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks (for more details see also [25]).

Temporal perceptual learning relies on finding temporal relationships in sensory signal streams. In an environment, statistically salient temporal correlations can be found by monitoring the arrival times of sensory signals. This is done by the perceptual network.

3.8.2 The Delta rule

Both learning paradigms supervised learning and unsupervised learning result in an adjustment of the weights of the connections between units, according to some modification rule. Virtually all learning rules for models of this type can be considered as a variant of the Hebbian learning rule suggested by Hebb (see also [19]). The basic idea is that if two nodes j and k are active simultaneously, their interconnection must be strengthened. If j receives input from k, the simplest version of Hebbian learning prescribes to modify the weight w_{jk} with:

$$\Delta w_{jk} = \eta y_i y_k$$

where η is a positive constant of proportionality representing the learning rate. Another common rule does not use the actual activation of unit k but the difference between the actual and desired activation for adjusting the weights:

$$\Delta w_{jk} = \eta y_i (d_k - y_k)$$

in which d_k is the desired activation provided by a teacher. This is often called the Widrow-Hoff rule or the delta rule.

3.8.3 The algorithm

Although the back-propagation algorithm can be used very generally to train neural networks, it is most famous for applications to layered feed-forward networks, or multilayer perceptrons. We know that simple perceptrons are very limited in their representational capabilities (for example, they can not represent the XOR function).

We will consider multilayer perceptrons with L layers of synaptic connections and $L+1$ layers of neurons. This is sometimes called an L-layer network, and sometimes an $L+1$-layer network. We will generally follow the convention that a network with a single layer can approximate any function, if the hidden layer is large enough.

Let's diagram the network as:

$$x_0 \xrightarrow{W_1,b_1} x_1 \xrightarrow{W_2,b_2} \ldots \xrightarrow{W_L,b_L} x_L$$

where $x_l \in \Re^{n_l}$ for all $l = 0, \ldots, L$ and W_l is an matrix $\Re^{n_l \cdot n_{l-1}}$ for all $l = 0, \ldots, L$. There are $L+1$ layers of neurons, and L layers of synaptic weights. We would like to change the weights W and biases b so that the actual output x_L becomes closer to the desired output d. The back-propagation algorithm consists of the following steps (for more references see [24]):

1. **Forward pass - Computation of outputs of all the neurons in the network:** the input vector x_0 is transformed into the output vector x_L, by evaluating the equation:

$$x_i^l = \sigma(s_l^i) = \sigma(\sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} + b_i^l) \quad for\ l = 1, \ldots, L\ . \tag{3.2}$$

   The algorithm starts with the first hidden layer using as input values the independent variables from the training data set. The neuron outputs are computed for all neurons in the first hidden layer by performing the relevant sum and activation function evaluations. These outputs are the inputs for neurons in the second hidden layer. Again the relevant sum and activation function calculations are performed to compute the outputs of second layer neurons. This continues layer by layer until we reach the output layer and compute the outputs for this layer. These output values constitute the neural net's guess at the value of the dependent variable. The values of W_{ij} are initialized to small (generally random) numbers in the range 0.00 ± 0.05. These weights are adjusted to new values in the backward pass as described below.

2. **Error computation:** the difference between the desired output d and the actual output x_L is computed:

$$\delta_i^L = \frac{\partial \sigma(s_i^L)}{\partial s}(d_i - x_i^L)\ . \tag{3.3}$$

3. **Backward pass - Propagation of error and adjustment of weights:** the error signal at the output units is propagated backwards through the entire network, by evaluating:

$$\delta_j^{l-1} = \frac{\partial \sigma(s_j^{l-1})}{\partial s} \sum_{i=1}^{n_l} \delta_i^l W_{ij}^l \quad for\ l = L, \ldots, 1\ . \tag{3.4}$$

4. **Learning updates:** the synaptic weights and biases are updated using the results of the forward and backward passes:

$$\Delta W_{ij}^l = \eta \delta_i^l x_j^{l-1} \tag{3.5}$$

$$\Delta b_i^l = \eta \delta_i^l \tag{3.6}$$

where η is the learning rate and these are evaluated for $l = 1$ to L (the order of evaluation does not matter). This phase begins with the computation of error at each neuron in the output layer. These errors are used to adjust the weights of the connections between the last-but-one layer of the network and the output layer. The adjustment is similar to the simple Widrow-Hoff rule that we saw earlier. The new value of the weight W_{ij} of the connection from node i to node j is given by:

$$W_{ij}^{l,new} = W_{ij}^{l,old} + \Delta W_{ij}^{l} \tag{3.7}$$

always for $l = 1$ to L. Here η is an important tuning parameter that is chosen by trial and error by repeated runs on the training data. Typical values for η are in the range 0.1 to 0.9. Low values give slow but steady learning, high values give erratic learning and may lead to an unstable network. The process is repeated for the connections between nodes in the last hidden layer and the last-but-one hidden layer. The backward propagation of weight adjustments along these lines continues until we reach the input layer. At this time we have a new set of weights on which we can make a new forward pass when presented with a training data observation.

Now we will show that this is gradient descent on a cost function.

3.8.4 Back-propagation as gradient descent

Let's define the cost function:

$$E(\mathbf{W}, \mathbf{b}) = \frac{1}{2} \sum_{i=1}^{n_L} (d_i - x_i^L)^2 \tag{3.8}$$

where x_L is a function of $\mathbf{W}$ and $\mathbf{b}$ arises through the equations of the forward pass. This cost function measures the squared error between the desired and actual output vectors. We are going to prove that back-propagation is gradient descent on this cost function. In other words, the back-propagation weight updates are equivalent to:

$$\Delta W_{ij}^{l} = -\eta \frac{\partial E}{\partial W_{ij}^{l}} \tag{3.9}$$

$$\Delta b_{i}^{l} = -\eta \frac{\partial E}{\partial b_{i}^{l}} \tag{3.10}$$

3.8.5 Properties of the algorithm

The back-propagation algorithm has a number of interesting features:

1. the forward and backward passes use the same weights, but in the opposite direction

$$x_j^{l-1} \xrightarrow{W_{ij}^l} x_i^l \tag{3.11}$$

$$\delta_j^{l-1} \xleftarrow{W_{ij}^l} \delta_i^l \tag{3.12}$$

2. The update for a synapse depends on variables at the neurons to which it is attached. In other words, the learning rules are local, once the forward and backward passes are complete.

3. As we will see later, the back-propagation algorithm is gradient descent on the squared error cost function between the desired and actual outputs. In general, it takes $\mathcal{O}(N)$ operations to compute the value of the cost function, where N be the number of synaptic weights. Naively, it should take $\mathcal{O}(N^2)$ operations to compute the N components of the gradient. In fact, the back-propagation algorithm finds the gradient in $\mathcal{O}(N)$ steps, which is much shorter.

3.8.6 Derivation with the chain rule

We need to compute the gradient of E with respect to $\mathbf{W}$ and $\mathbf{b}$. The technical difficulty is that the dependence on $\mathbf{W}$ and $\mathbf{b}$ is implicit, buried inside x_L. The standard way of dealing with this difficulty is to apply the chain rule to the equations of the forward pass, which describe the dependence of the output layer x_L on the input x_0.

What is the meaning of the quantity δ_i^l?

It is the sensitivity of the cost function to changes in the bias of neuron i in layer l.

$$\delta_i^l = -\frac{dE}{db_i^l} \tag{3.13}$$

In gradient learning for a single-layer perceptron, the weight update is the product of presynaptic activity, and an error term that is proportional to the difference between the desired and actual outputs. In gradient learning for a multilayer perceptron, no desired output for the hidden neurons is available. But the back-propagated error serves as a proxy. What replaces the error term is the sensitivity of the cost function to input to the postsynaptic neuron.

3.8.7 Derivation with Lagrange multipliers

Another method is to use Lagrange multipliers. This method is closely related to dynamic programming and optical control. We will need to define the function $\phi(y) = \sigma'(\sigma^{-1}(y))$. This is the slope of σ, considered as a function of the neural output. Equivalently, we can write $\phi(\sigma(x)) = \sigma'(x)$ or $\phi(y) = 1/\sigma^{-1'}(y)$.

The equations of the forward pass can be written as

$$\sigma^{-1}(x_i^l) = \sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} + b_i^l \tag{3.14}$$

Now we define a Hamiltonian

$$H(x, \delta, \mathbf{W}, \mathbf{b}) = \frac{1}{2}\sum_{i=1}^{n_L}(d_i - x_i^L)^2 + \sum_{l=1}^{L}\sum_{i=1}^{n_{l-1}} \delta_i^l \Big[\sigma^{-1}(x_i^l) - \sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} - b_i^l\Big] \tag{3.15}$$

Suppose that the Hamiltonian is stationary with respect to x and y, meaning that the derivatives with respect to x and y vanish. Then

$$\frac{\partial H}{\partial \delta_i^l} = \sigma^{-1}(x_i^l) - \sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} - b_i^l \tag{3.16}$$

vanishes. This implies that the equations of the forward pass are satisfied, and furthermore that $H = E$.

$$E(\mathbf{W}, \mathbf{b}) = stat_{x,y} H(x, \delta, \mathbf{W}, \mathbf{b}) \tag{3.17}$$

Therefore, minimizing H with respect to $\mathbf{W}$ and $\mathbf{b}$ at a stationary point with respect to x and y is equivalent to minimizing E with respect to $\mathbf{W}$ and $\mathbf{b}$ subject to the constraint that the x_i^l i satisfy the equations of the forward pass.

The other requirement for a stationary point is that the partial derivatives

$$\begin{aligned} \frac{\partial H}{\partial x_i^L} &= -(d_i - x_i^L) + \frac{\delta_i^L}{\phi(x_i^L)} && (3.18) \\ \frac{\partial H}{\partial x_j^{l-1}} &= \frac{\delta_j^{l-1}}{\phi(x_j^{l-1})} - \sum_{i=1}^{n_l} \delta_i^l W_{ij}^l && (3.19) \end{aligned}$$

must also vanish. Setting these to zero yields the error computation and the backward pass. Therefore, the backward pass is a way of solving the equation $\partial H/\partial x = 0$. We can quantify the change in E by looking at the change in H

$$dE = \frac{\partial H}{\partial x}dx + \frac{\partial H}{\partial y}dy + \frac{\partial H}{\partial \mathbf{W}}d\mathbf{W} + \frac{\partial H}{\partial \mathbf{b}}d\mathbf{b} \tag{3.20}$$

In this expression we assume that the changes dx and dy are slaved to $d\mathbf{W}$ and $d\mathbf{b}$, since both x and y are both implicitly functions of $\mathbf{W}$ and $\mathbf{b}$. By our cunning, we have defined H so that the first two terms vanish, and we are left with

$$dE = \frac{\partial H}{\partial \mathbf{W}}d\mathbf{W} + \frac{\partial H}{\partial \mathbf{b}}d\mathbf{b} \tag{3.21}$$

Therefore we can compute the gradients of E as

$$\frac{\partial E}{\partial W_{ij}^{l}} = \frac{\partial H}{\partial W_{ij}^{l}} = -\delta_i^l x_j^{l-1} \tag{3.22}$$

$$\frac{\partial E}{\partial b_i^l} = \frac{\partial H}{\partial b_i^l} = -\delta_i^l \tag{3.23}$$

Only the explicit dependence of H on $\mathbf{W}$ and $\mathbf{b}$ matters in the gradient computation. The implicit dependence does not matter because we are at a stationary point.

3.8.8 Some issues in training neural networks

There is quite an art in training neural networks. The model is generally overparametrized, and the optimization problem is non-convex and unstable unless certain guidelines are followed. In this section we summarize some of the important issues.

Starting Values

We note that if the weights are near zero, then the operative part of the sigmoid (Figure 3.3) is roughly linear, and hence the neural network collapses into an approximately linear model. Usually starting values for weights are chosen to be random values near zero. Hence the model starts out nearly linear, and becomes nonlinear as the weights increase. Individual units localize to directions and introduce nonlinearities where needed. Use of exact zero weights leads to zero derivatives and perfect symmetry, and the algorithm never moves. Starting instead with large weights often leads to poor solutions.

Number of Hidden Units and Layers

While for the input and output layer the choice regarding the amount of neurons is strictly related to the number of inputs and outputs of the model, there is not a systematic way for deciding the number $m_{H,l}$ $(l = 1, \ldots, L)$ of hidden nodes.

Generally speaking it is better to have too many hidden units than too few. With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data; with too many hidden units, the extra weights can be shrunk toward zero if appropriate regularization is used.

Typically the choice of number of hidden neurons is:

$$m_O \leq m_{H,l} \leq m_I$$

where m_O is the number of the output layer and m_I the one of the input, with the number increasing with the number of inputs and number of training cases. It is most common to put down a reasonably large number of units and train them with regularization. We can use cross-validation to estimate the optimal number, but this seems unnecessary if cross-validation is used to estimate the regularization parameter. Choice of the number of hidden layers is guided by background knowledge and experimentation. Each layer extracts features of the input for regression or classification. Use of multiple hidden layers allows construction of hierarchical features at different levels of resolution.

Multiple Local Optima and Epochs

The back-propagation algorithm is a version of the steepest descent optimization method applied to the problem of finding the weights that minimize the error function of the network output. Due to the complexity of the function and the large numbers of weights that are being *trained* as the network *learns*, there is no assurance that the back-propagation algorithm (and indeed any practical algorithm) will find the optimum weights that minimize error, the procedure can get stuck at a local minimum. It has been found useful to randomize the order of presentation of the cases in a training set between different scans. It is possible to speed up the algorithm by batching, that is updating the weights for several exemplars in a pass. However, at least the extreme case of using the entire training data set on each update has been found to get stuck frequently at poor local minima. A single scan of all cases in the training data is called an epoch. Most applications of feed-forward networks and back-propagation require several epochs before errors are reasonably small. A number of modifications have been proposed to reduce the epochs needed to train a neural net. One commonly employed idea is to incorporate a momentum term that injects some inertia in the weight adjustment on the backward pass. This is done by adding a term to the expression for weight adjustment for a connection that is a fraction of the previous weight adjustment for that connection. This fraction is called the momentum control parameter. High values of the momentum parameter will force successive weight adjustments to be in similar directions. Another idea is to vary the adjustment parameter η so that it decreases as the number of epochs increases. Intuitively this is useful because it avoids over-fitting that is more likely to occur at later epochs than earlier ones.

Overfitting and the choice of training epochs

A weakness of the neural network is that it can be easily overfitted, causing the error rate on validation data to be much larger than the error rate on the training data. It is therefore important not to overtrain the data. A good method for choosing the number of training epochs is to use the validation data set periodically to compute the error rate for it while the network is being trained. The validation error decreases in the early epochs of back-propagation but after a while it begins to increase. The point of minimum validation error is a good indicator of the best number of epochs for training and the weights at that stage are likely to provide the best error rate in new data.

Chapter 4

Pre-Processing Results

The semiconductor manufacturing process data-sets analyzed for the experimental part of this dissertation were provided by a worldwide semiconductor factory. This chapter will describe:

- format and contents of data from several process steps;
- data extraction and organization tools;

The data we have used for this work is concerned Chemical Vapor Deposition and the purpose we have prearranged is to find a predicted model of the CVD deposited layer.

4.1 Data description

The data files provided by a semiconductor fabrication plant were related to CVD and were collected over a period of seven months. The analysis is carried out on production data for a well-controlled, CVD thickness process and all the data is obtained from the same CVD machine, consisting in three chambers and where every chamber has two distinct sub-chambers. Every processed lot is composed by fifty wafers divided in two subsets: in each of these wafers are numbered from 1 to 25. CVD thickness measurements were usually taken from wafers 3 and 4 approximately twice in every lot, leading to measurements of approximately 8% of wafers.

The data available present about twenty different recipes: due to proprietary reasons, we are not going to indicate here the exact name of them.

The available data are organized as:

- **APC data**: a **.mat** file collects the measurements of APC. The format is:
  - **- Created** : start time of the current processed wafer;
  - **- Equipment** : deposition equipment name;
  - **- Equipment_ID** : ID of equipment;

- **Batch_ID** : number of the batch[1];
- **Run_ID** : number of the processed wafer;
- **LotLog_ID** : ID of current lot;
- **WafLog_ID** : ID of current wafer;
- **Technology** : set of different products;
- **Operation** : operation performed;
- **Lot** : name of processed lot;
- **Slot** : number of processed slot (between 1 and 25);
- **Carrier** : it distinguishes in which subset of the same lot the processed wafer is.

Usually, 2 wafers per lot.

- **R2R data**: a **.mat** file collects measurements of APC. The format is:

  - **Material_ID** : name of processed lot;
  - **Material_Level** : number of processed wafer;
  - **ProcessGroup_ID** : recipe's name used to process current wafer. Recipes can be considered as machine's settings;
  - **Product** : kind of measurement;
  - **Route_ID**
  - **BasicType_ID**
  - **EPA_D** : kind of measurement;
  - **ProcessTool_ID** : deposition equipment name;
  - **ParentMaterial_ID** : lot name;
  - **DepoTime_used** : deposition time.

  Usually, 2 wafers per lot.

- **rawtab data**: in a **.mat** file, for almost every wafer, internal process state variables are collected as time series uniformly samples on thirty points and stored using a *raw text* format. Generally, these signals describe internal signals (for example gas flows, pressures, temperatures, voltages,...) on a wafer-by-wafer bases.

  The complete list of process variables is displayed in Table 4.1.

- **y data**: a **.mat** file collects measurements of CVD thickness in nine equidistant different points of the processed wafer.

[1] A batch is a subset of wafers processed by the same sub-chamber of the machine.

ID	**Variable Name**
1	BY_divert_Flow
2	He_Hi_Flow
3	NF3_Flow
4	O2_Flow
5	O3_Flow
6	O3_gen_concentration
7	PRESS_FORELINE
8	TEB_Flow
9	TEOS_Flow
10	TEPO_Flow
11	chord_lengh_error_left
12	chord_lengh_error_right
13	heater_AO_row_value
14	hi_manometer
15	leading_edge_error_left
16	leading_edge_error_right
17	lo_manometer
18	pressure_reading
19	susceptor_temp
20	throttle_valve_step

Table 4.1: List of initial variables.

The main data issues are:

- **partial information**: not every wafer is measured, in fact metrology in performed on a very small percentage of wafers;
- **sampling changes**: there is no guarantee that the same wafer is measured during different process steps;
- **no wafer tracking**: it is impossible to know which wafer (in a lot) is measured (except for process data).

When the data was stored and properly organized, the pre-processing analysis and the following neural network analysis could began.

4.2 Statistical modeling and analysis

The Figures in this section show some notable examples of data variability between lots[2]:

CVD data (Figure 4.1) : 4 deposition measurements on 4 different wafers.

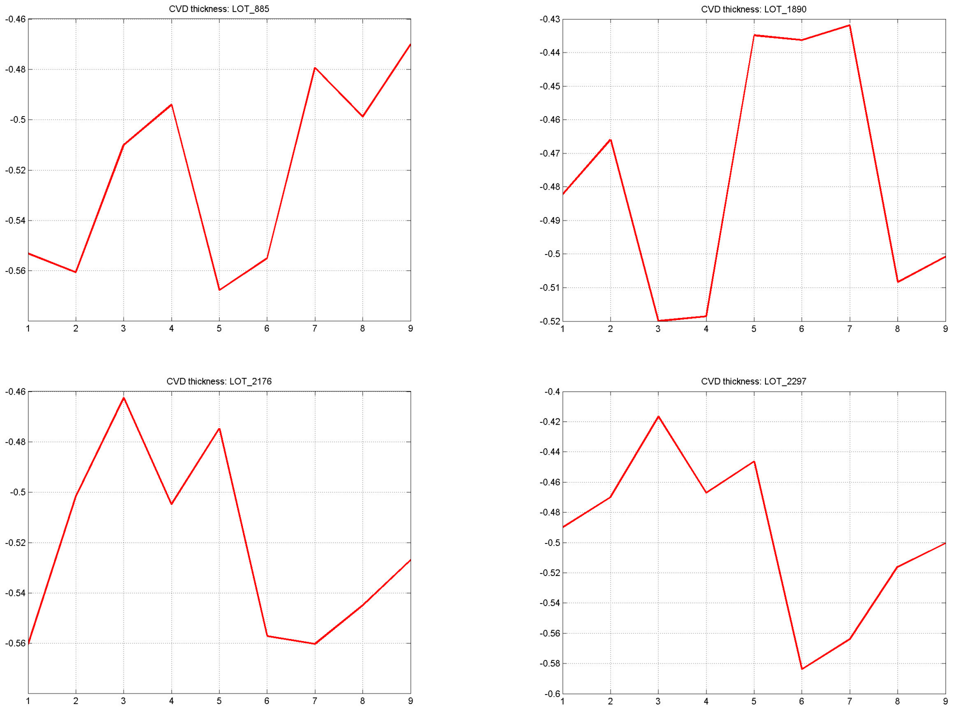


Figure 4.1: Examples of CVD data in four different wafers.

[2] Data plotted are normalized as $\frac{value-\mu}{\sigma}$ where μ is the mean and σ is the standard devaition of the data, respectively.

The first modeling steps were:

- data modeling in order to describe a whole lot and its inner relationships;
- process variables reduction to discard the useless ones;
- data clustering to find out possible similar behaviours among different recipes and therefore to consider more than one recipe as the only one for future modeling;
- explorative correlation analysis to find out important parameters and discard useless variables;
- explorative principal component analysis analysis, used as variable extraction method, to reduce dimensionality;
- explorative stepwise linear regression analysis, used as variable selection method, to choose meaningful variables.

In Figures 4.2, 4.3 and 4.4 examples of time series of different sub-chambers involving in the study are depicted.

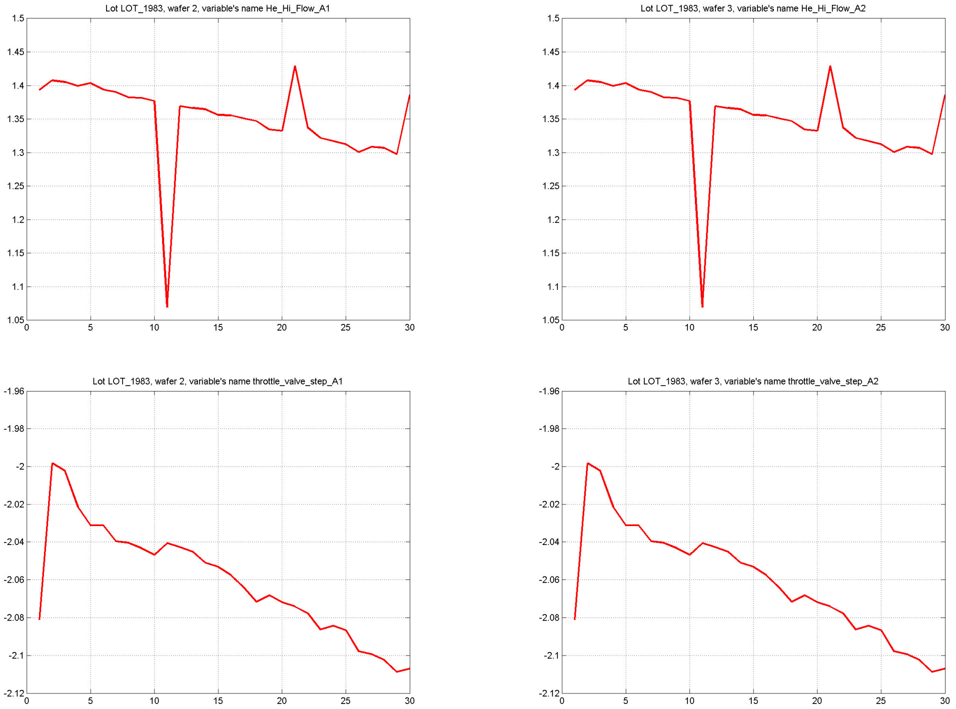


Figure 4.2: Example of time series of different sub-chambers (A_1 and A_2) of Lot # 1983.

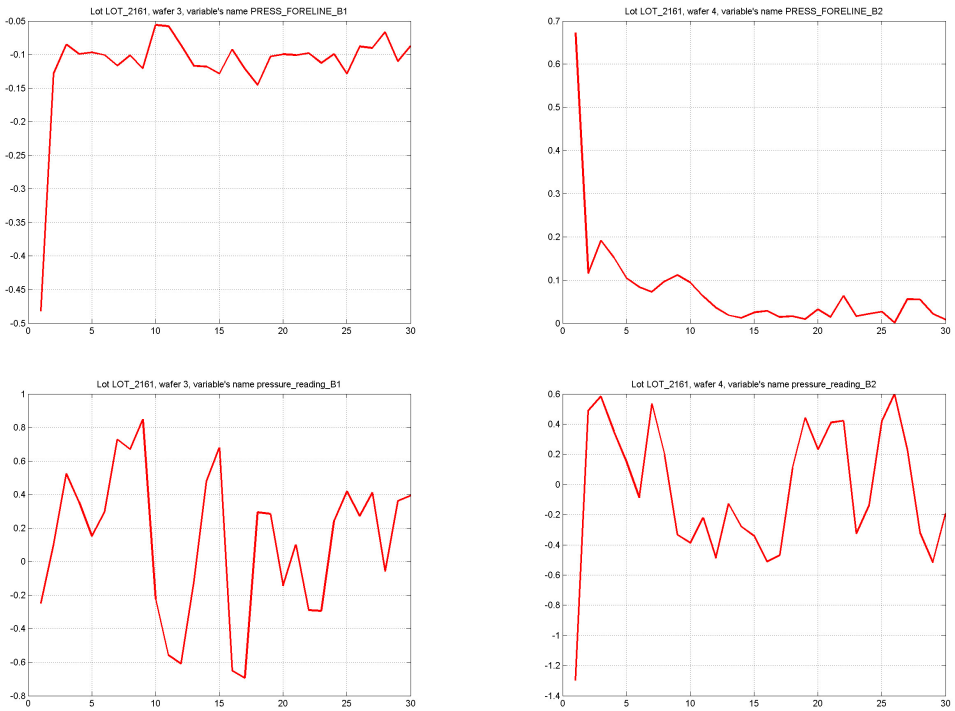


Figure 4.3: Example of time series of different sub-chambers (B_1 and B_2) of Lot # 2161.

4.2.1 Data modeling

According to data structure, the modeling procedure was performed as follows:

CVD data modeling : 2 (rarely 4) wafers per lot were tested. Because of the lack of wafer tracking, the mean of the provided measures was chosen to represent the entire lot:

$$CVD_{lot} = \frac{\sum_{i=1}^{TOT_{CVD\_samples}} DepositionValue_i}{TOTAL_{CVD\_samples}} \tag{4.1}$$

Process data modeling: Process data time series were measured for (almost) every wafer in every lot. Unfortunately, the lack of precise referrals from the upstream process renders nearly useless this large amount of data; being stationary signals, the time series were reduced to statistical parameters: mean (or median, in case of frequent outliers) and variance.

More precisely, the model for process data time series is:

1. calculate mean and standard deviation of the m^{th} process time series for every j^{th} wafer processed where m is the number of the process variables:

$$\begin{aligned} mean^j &= [mean_1^j \cdots mean_m^j] & (4.2)\\ std^j &= [std_1^j \cdots std_m^j] & (4.3) \end{aligned}$$

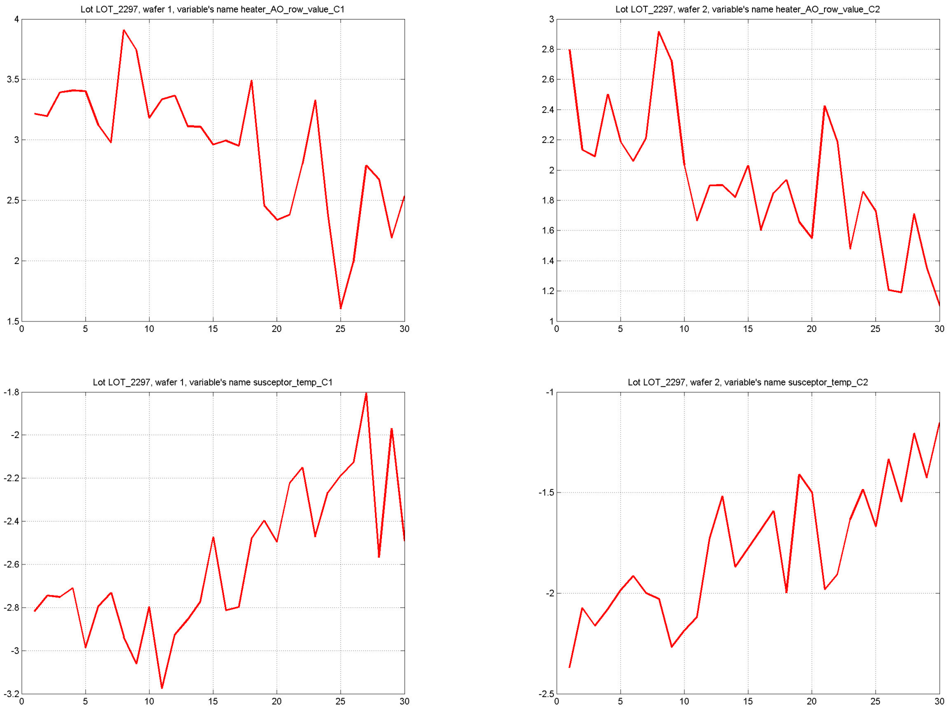


Figure 4.4: Example of time series of different sub-chambers (C_1 and C_2) of Lot # 2297.

2. calculate median to reduce outliers influence (if necessary):

$$M^j = Median(mean^j) \quad (4.4)$$

$$STD^j = Median(std^j) \quad (4.5)$$

3. iterate the procedure $\forall j = 1, 2, \ldots, N$ where N denotes the number of processed wafers:

$$M = [M^1 \cdots M^N] \quad (4.6)$$

$$STD = [STD^1 \cdots STD^N] \quad (4.7)$$

4.2.2 Process variables reduction

CVD equipment is equipped with a considerable number of sensors. However, not all sensor data are required to produce the VM data. Also, only certain critical stage's sensor data are necessary in the whole wafer process. In this research, the sensor selection is based on the the following rules:

Rule 1: exclude the turned-off sensors in the process;

Rule 2: eliminate sensors with constant values in the temporal chart, which shows sensor data versus time, because they do not affect the actual metrology value;

Rule 3: among the sensors measuring the same physical property, only one is selected;

Rule 4: delete the sensors that are the linear combination of selected sensors;

Rule 5: exclude the sensors that only relate to the processing step number because they are irrelevant to the process.

According to the above rules, among 20 CVD equipment sensors, only 15 comparatively important ones are selected for the analysis, as shown in Table 4.2.

ID	Variable Name
1	BY_divert_Flow
2	He_Hi_Flow
3	NF3_Flow
4	O2_Flow
5	O3_Flow
6	O3_gen_concentration
7	PRESS_FORELINE
8	TEB_Flow
9	TEOS_Flow
10	TEPO_Flow
11	heater_AO_row_value
12	hi_manometer
13	pressure_reading
14	susceptor_temp
15	throttle_valve_step

Table 4.2: List of recommended variables.

4.2.3 Clustering

First, we have divided the entire body of data in accordance with the process group used to process the current wafer.

According to the above-mentioned subdivision, for every process group a comprehensive matrix $X_{ProcessGroup} \in \Re^{n \cdot m}$ is built as follows:

$$\tag{4.8}$$

$$X_{ProcessGroup} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ x_{n,1} & \cdots & \cdots & x_{n,m} \end{bmatrix} \tag{4.9}$$

where n is the number of wafers processed by the same process group, m is the number of the process variables and $x_{i,j}$ $(i = 1, \ldots, n;\ j = 1, \ldots, m)$ represents the time series of the process variables.

A parallel thing is made for the output:

$$Y_{ProcessGroup} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,t} \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ y_{n,1} & \cdots & \cdots & y_{n,t} \end{bmatrix} \tag{4.10}$$

where $Y_{ProcessGroup} \in \Re^{n \cdot t}$, n is the number of wafers processed by the same process group, t is the number of the outputs and $y_{i,j}$ $(i = 1, \ldots, n;\ j = 1, \ldots, t)$ represents the measures of CVD thickness corresponding to the same process group.

Once we have made this partition, through principal component analysis of either all $X_{ProcessGroup}$ or $Y_{ProcessGroup}$, we obtain five main clustering that are depicted in Figures 4.5, 4.6 and 4.7.

From now on, the results we reported referred with the largest clustering, named clustering *GAMMA* showed in Figure 4.6 . By the way, we named X_{GAMMA} and Y_{GAMMA} the input data matrix $X_{ProcessGroup}$ and the output data matrix $Y_{ProcessGroup}$ corresponding to the mentioned clustering, respectively.

It is important to outline that this procedure is a qualitative analysis and therefore a more mathematical and correct statistical method would be the measurement of the distance among the various distributions, seeing that we have dealt with them, and a possible solution would be the use of *Kullback-Leibler distance* (for more details see [34, 35]).

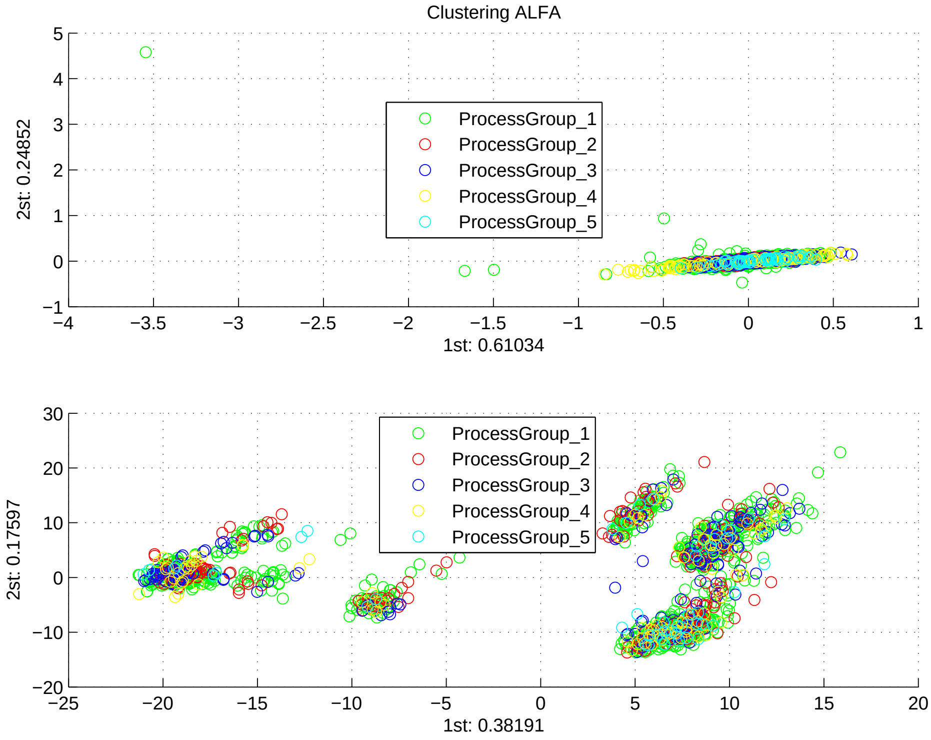


Figure 4.5: Main clustering: clustering ALFA.

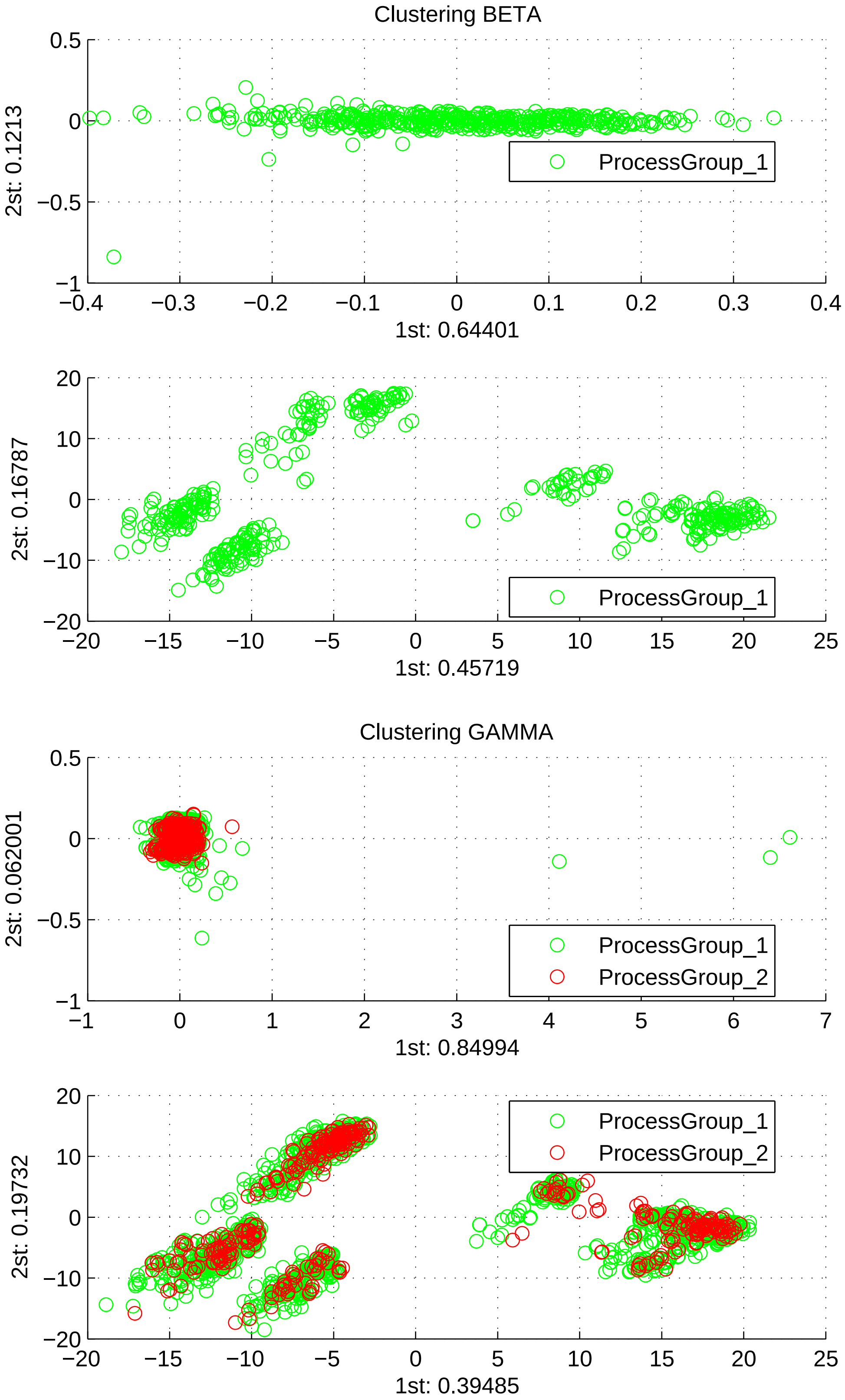


Figure 4.6: Main clustering: BETA and GAMMA.

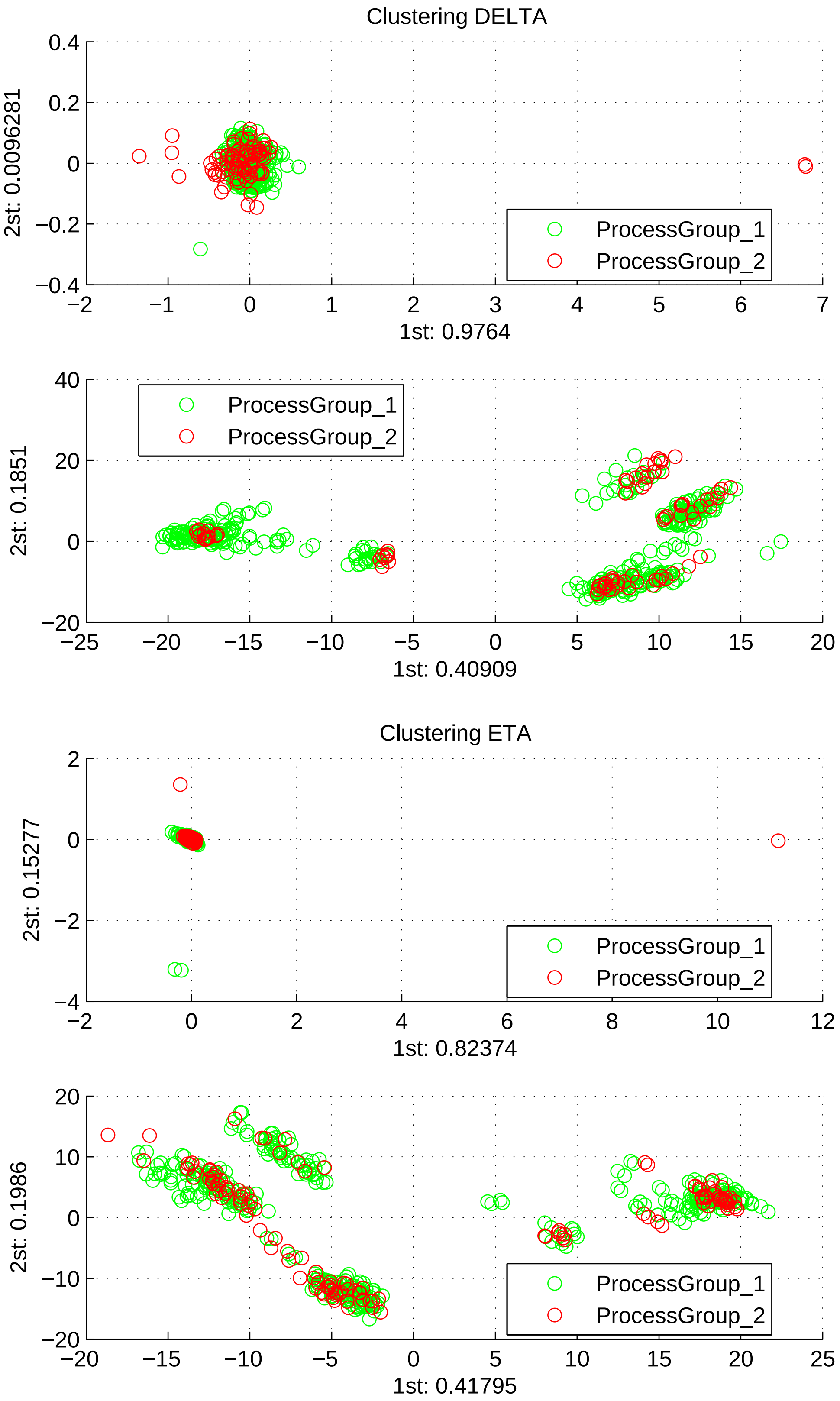


Figure 4.7: Main clustering: DELTA and ETA.

4.2.4 Data correlation analysis

The purpose of this kind of analysis if to find out exploitable latent relationships between variables. Using $X_{ProcessGroup}$ and $Y_{ProcessGroup}$ defined above, the correlation matrix R_X for $X_{ProcessGroup}$ is then computed:

$$R_X = \begin{bmatrix} r^x_{1,1} & \cdots & r^x_{1,m} \\ \vdots & \ddots & \vdots \\ r^x_{m,1} & \cdots & r^x_{m,m} \end{bmatrix} \tag{4.11}$$

where $r^x_{i,j}$ is the correlation coefficient between the i^{th} and the j^{th} variables.

While for $Y_{ProcessGroup}$ the correlation matrix R_{XY} is:

$$R_{XY} = \begin{bmatrix} r^{xy}_{1,1} & \cdots & r^{xy}_{1,t} \\ \vdots & \ddots & \vdots \\ r^{xy}_{m,1} & \cdots & r^{xy}_{m,t} \end{bmatrix} \tag{4.12}$$

where $r^{xy}_{i,j}$ is the correlation coefficient between the i^{th} variable and the j^{th} output.

A graphic representation of R_X and R_{XY} is shown in Figures 4.8 and 4.9 where:

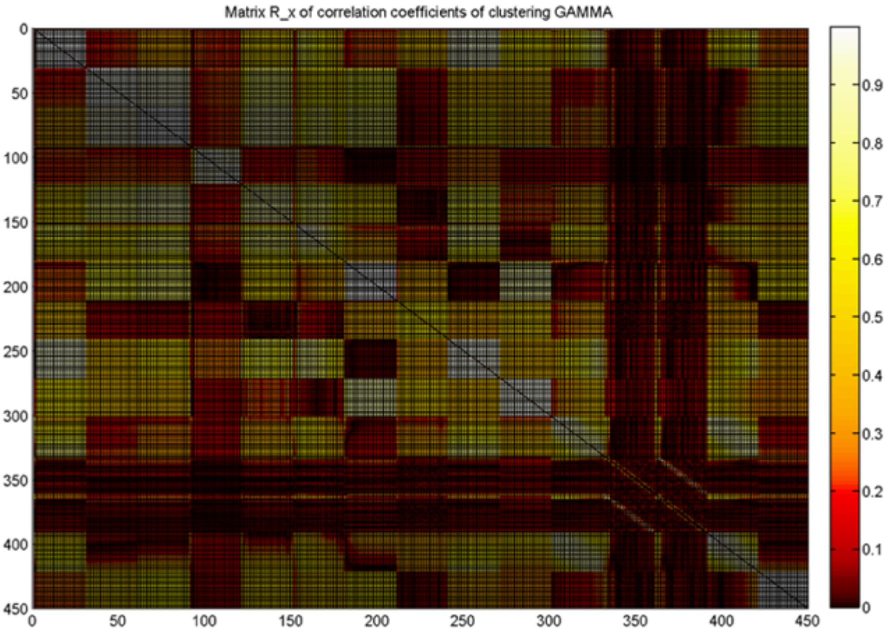


Figure 4.8: Matrix R_X of correlation coefficients.

- white dots describe both positive and negative correlation where the level of confidence (LoC) is equal at 95%;
- dark red dots describe no correlation.

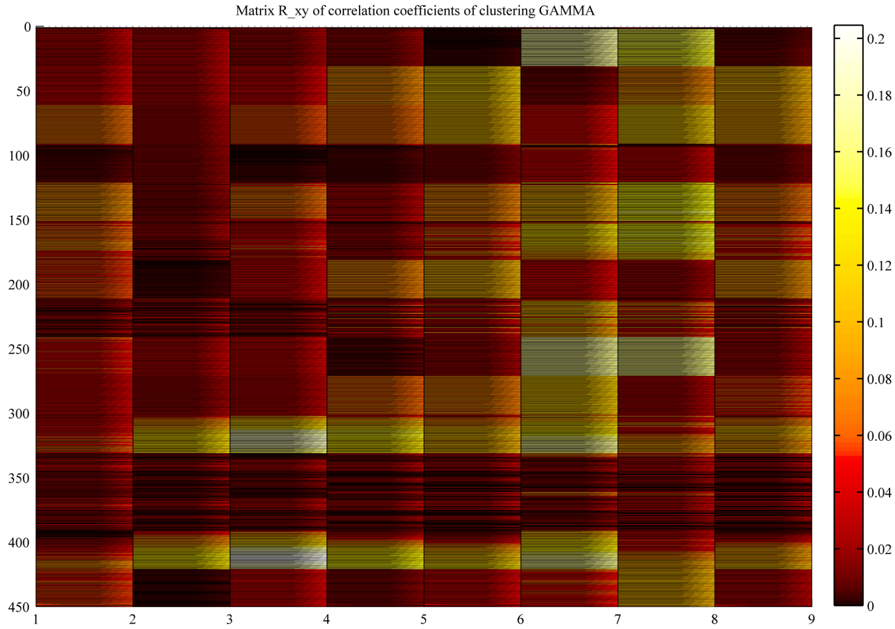


Figure 4.9: Matrix R_{XY} of correlation coefficients.

It can be noticed that:

R_X:

- in R_X some variables are highly correlated ($r^x_{i,j} \approx \pm 1$); in particular, there are inner relationships between first process variable and ninth one;
- some of the variables are almost constant ($\sigma^2 \approx 0$).

R_{XY}: there are low inner relationships between metrology variables and process variables.

4.2.5 Principal components analysis

In this section we explain the results we have got after principal component analysis.

As we can see in Figure 4.10(a), a PCA of the data demonstrates that the individual variables have very little correlation between them. This is seen as the overall variance for the data-set can not be explained using a small number of principal components. In fact, starting with the whole set of $m = 450$ variables, if we want to have a level of confidence (LoC) of 95%, we have to take the first $p = 35$ PCs; while if we choose that the level is equal at 99%, the number of PCs rises to $p = 102$ (see Figure 4.10(b)).

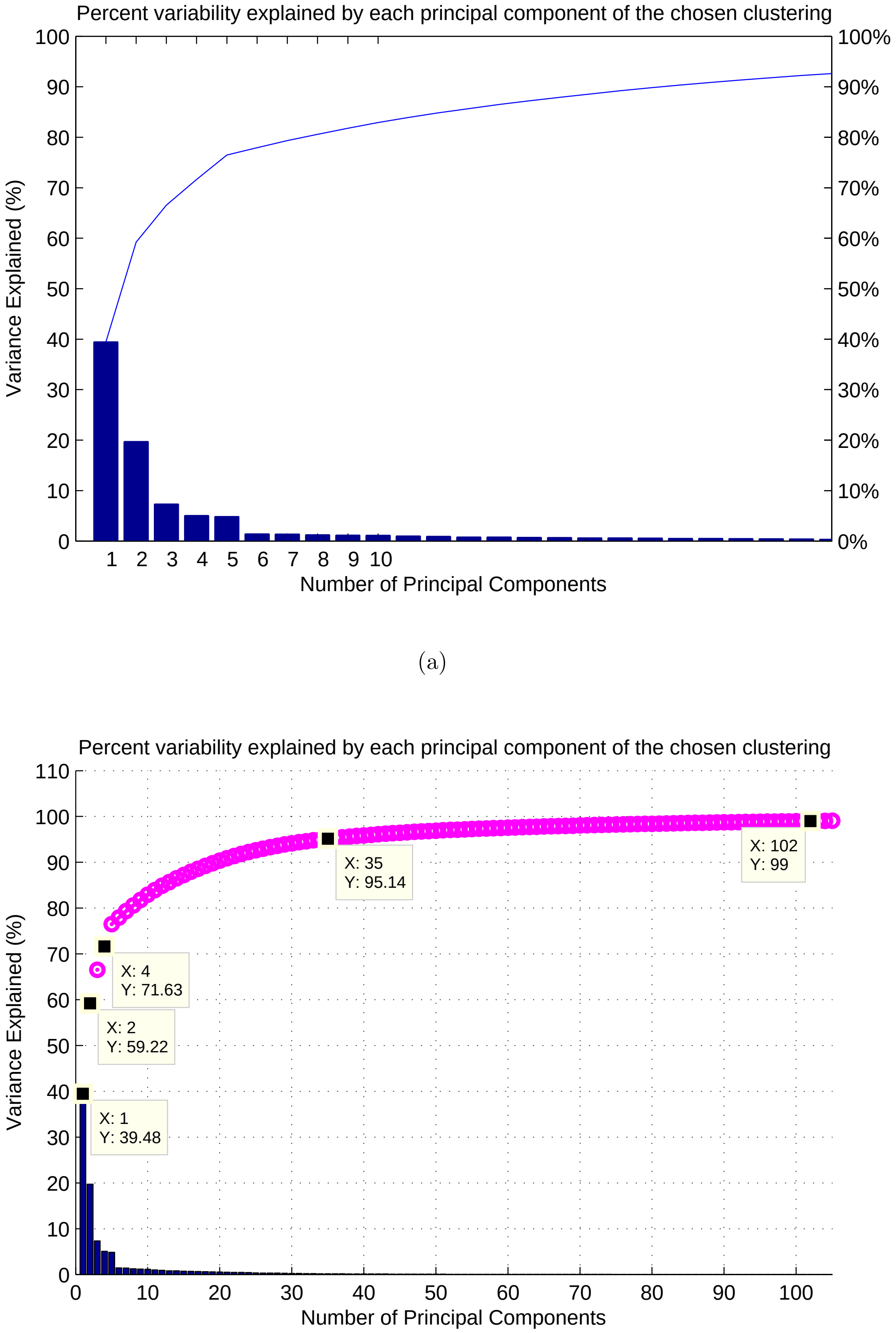
In Figure 4.11 the scores obtained by PCA are depicted:

- in 4.11(a) the plot $3D$ of the first three PCs is shown. We can observe the layout of the scores according to the sub-chamber in which the relative wafer is processed:

  there are three distinct *clouds* corresponding to the three chambers A, B, C and we can state that the behaviour of the sub-chambers of the same chamber is very similar.

- Figure 4.11(b) is a version of 4.11(a) one in which we have set the view along the y-axis, with the x-axis extending horizontally and the z-axis extending vertically. Here, we can notice something on the stability of the various chambers:

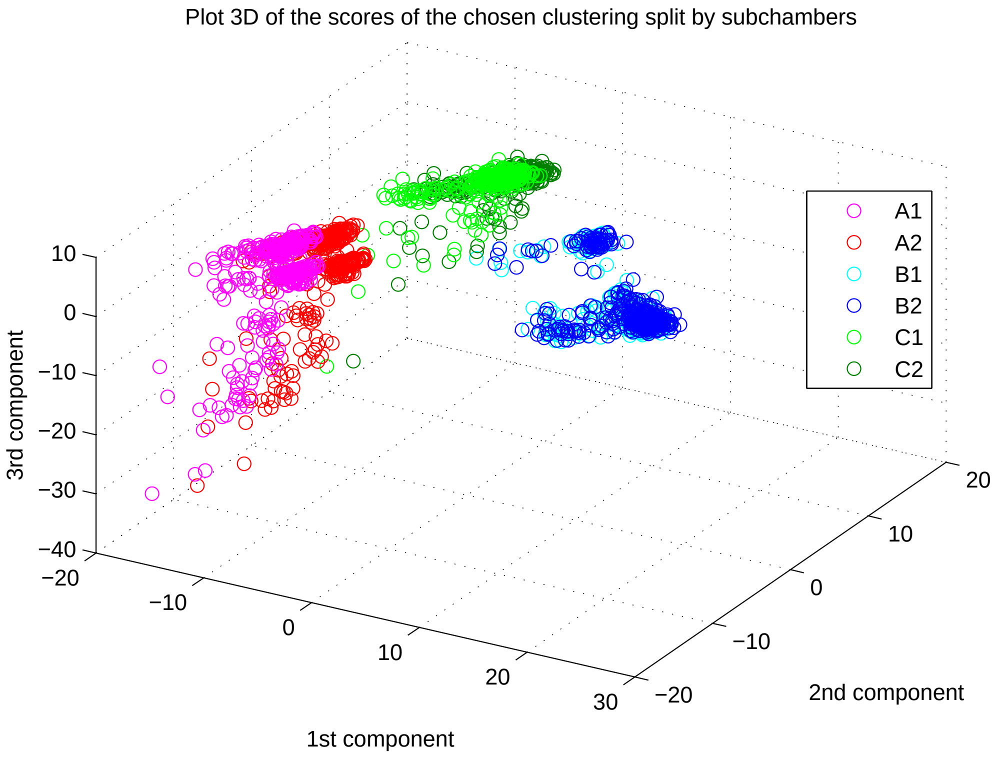
  the behaviour of chamber B appears stabler than the others ones where the trends is more irregular.



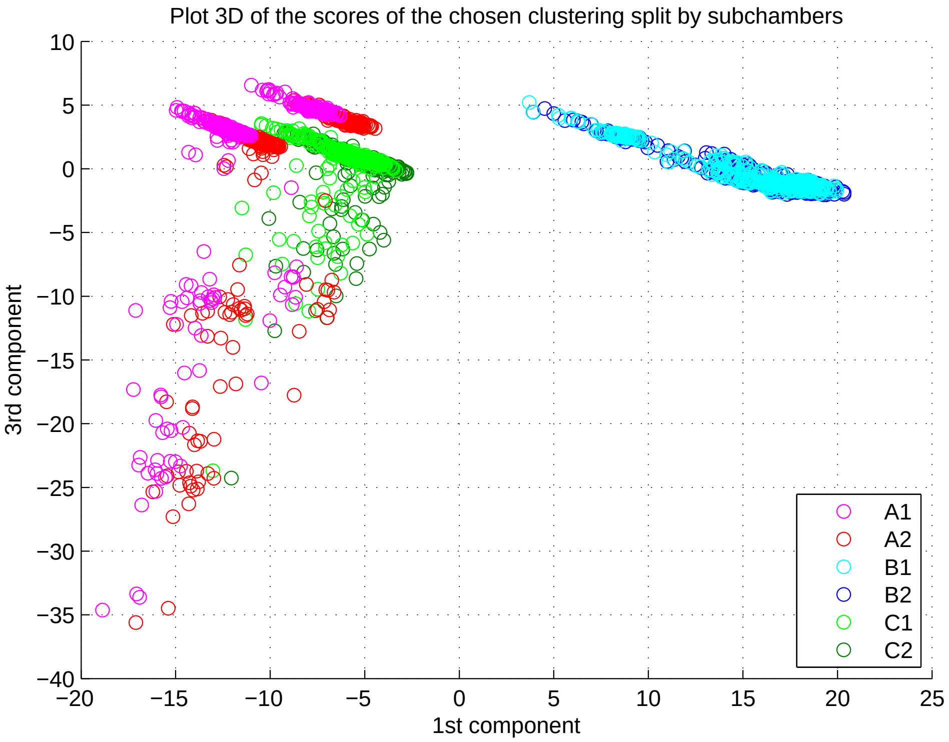
(a)

(b)

Figure 4.10: Variance explained as a function of Principal Components for input data.



(a)



(b)

Figure 4.11: Principal Components Scores for input data.

4.2.6 Stepwise selection analysis

Now, we will take a look at the results we have obtained performing stepwise selection analysis.

First, for the purpose of this work $f_{in} = 0.05$ and $f_{out} = 0.10$ are used for addition and removal of regression variables respectively; therefore the level of confidence for both the techniques is settled at 95%.

Starting with $m = 450$ process variables, first we have chosen as principal output the central[3] measurement of CVD thickness, named $Y_{GAMMA}(:, j)$ where j is the corresponding column index; then, we have performed this analysis in two different ways:

- in the first manner, the input variables have coincided with the matrix X_{GAMMA} and we have picked out $p = 21$ regression variables. In Table 4.3 the correspondence among these resulting 21 comparatively important regressors and the number of recommended process variables listed in Table 4.2 is explained.
- In the second one, we have implemented stepwise regression introducing in input the scores we have got after principal component analysis. Here, the number of regressors selected by the algorithm is been $p = 32$ and in Table 4.4 a chart of analogous meaning as above is depicted.

ID	Variable Name	# Selected Regressor
1	BY_divert_Flow	3
2	He_Hi_Flow	1
3	NF3_Flow	
4	O2_Flow	
5	O3_Flow	
6	O3_gen_concentration	
7	PRESS_FORELINE	
8	TEB_Flow	7
9	TEOS_Flow	2
10	TEPO_Flow	
11	heater_AO_row_value	5
12	hi_manometer	
13	pressure_reading	
14	susceptor_temp	1
15	throttle_valve_step	2

Table 4.3: List of 21 selected regressors via stepwise selection method.

[3] We could choose either one of the nine possible outputs or the mean of these ones.

ID	**Variable Name**	**# Selected Regressor**
1	BY_divert_Flow	5
2	He_Hi_Flow	1
3	NF3_Flow	1
4	O2_Flow	
5	O3_Flow	2
6	O3_gen_concentration	1
7	PRESS_FORELINE	1
8	TEB_Flow	2
9	TEOS_Flow	3
10	TEPO_Flow	3
11	heater_AO_row_value	1
12	hi_manometer	1
13	pressure_reading	5
14	susceptor_temp	4
15	throttle_valve_step	2

Table 4.4: List of 32 selected regressors via stepwise selection method after PCA.

Figures 4.12 and 4.13 show the correlation structure between the variables chosen and the entire Y_{GAMMA} in the two manners.

We have already seen in Subsection 4.2.4 that there was a little correlation between X_{GAMMA} and Y_{GAMMA}; now, about these pictures, in both cases there has been a smaller correlation coefficient among the process variables and the Y_{GAMMA} in correspondence of the selected output to implement stepwise regression.

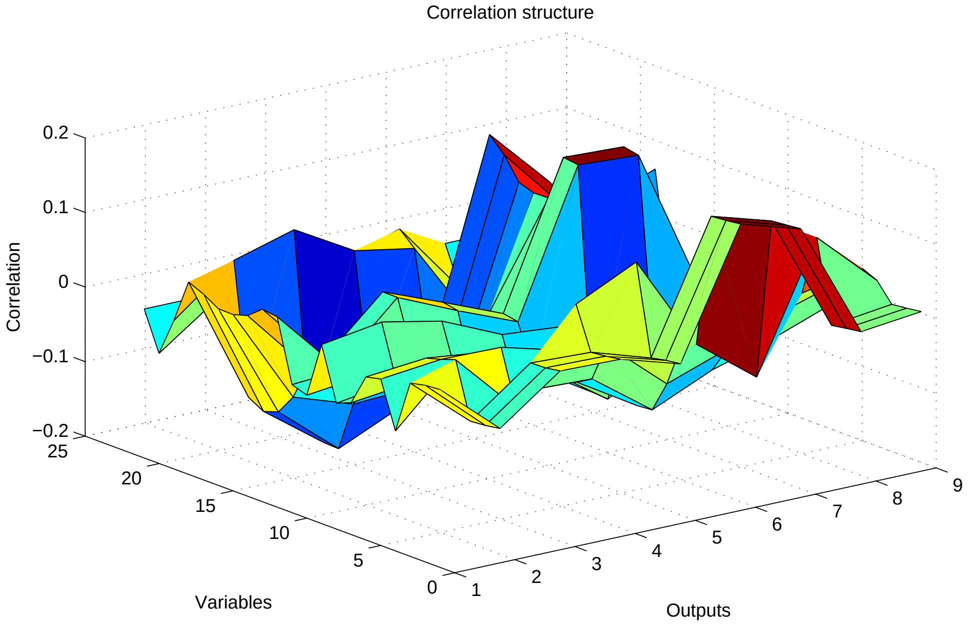


Figure 4.12: Correlation structure for variables chosen with stepwise method: 1^{st} way.

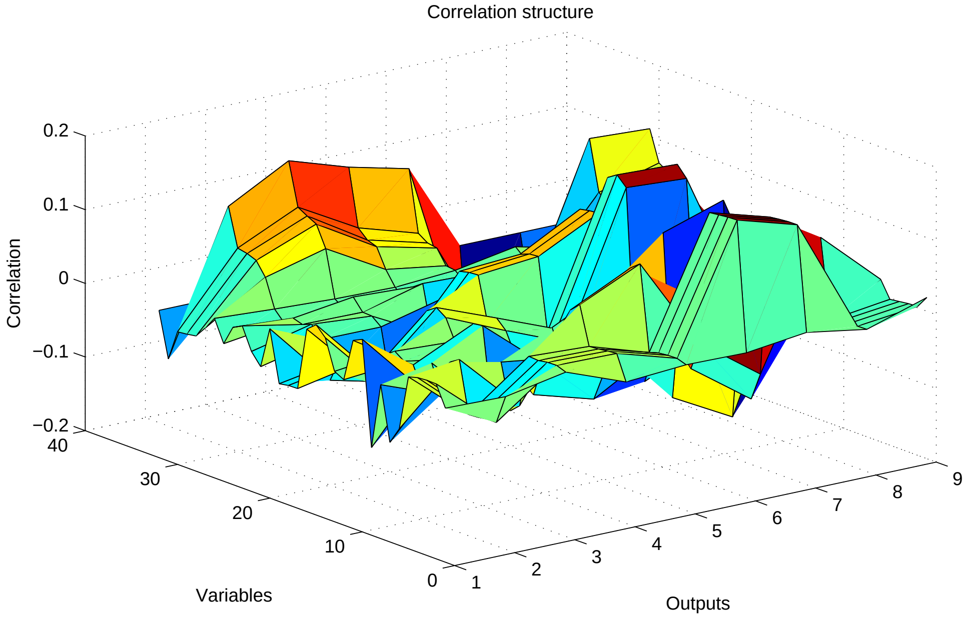


Figure 4.13: Correlation structure for variables chosen with stepwise method: 2^{nd} way.

4.2.7 Comparison among selection techniques

In this section, we give a roundup of considerations about merits and faults of the selection techniques we have utilized.

First, the main disadvantages of the correlation and PCA based variable selection methods is that there is a high probability of the algorithms choosing predictor variables that are correlated with one another. These extra variables are added to the prediction model, but they do not add much extra information or value to the prediction accuracy.

This phenomenon arises in the correlation selection algorithm form signals such as two different flows.

In the case of PCA, all of the variables from the same principal component are likely to be similar as they are used to describe the same component of the dataset variance. Hence, selecting some variables from the same principal component may actually add very little new information to the model.

Another complication to this selection technique is that the PCs are calculated without any reference to the output. The variable selected by principal component model may best explain the largest variance in the input data, but may be poor predictors of the system output.

The stepwise regression method of variable selection has the advantage that the selected predictor variables are unlikely to be highly correlated. Each variable is added to the model only if it contributes to the accuracy of the prediction. Adding a variable that is highly correlated to an existing variable in the model will not contribute significantly and so there is a low probability of many correlated variable existing in the final model structure. As variable are assessed during the algorithm and removed if they no longer contribute, stepwise regression should lead to the most parsimonious model.

Chapter 5

Experimental Modeling Results

As shown before, several techniques are available for building a VM environment. This chapter will describe in more detail some solutions capable of providing an accurate estimation of CVD thickness.

We recall that the following results refer to clustering *GAMMA* and that we chose it because it is the largest one.

In particulary, the chapter will present and discuss:

- mathematical and statistical methods applied to real data;
- some modifications in order to improve performances;
- advantages and weaknesses of different techniques.

In order to evaluate how well a VM model fits the relation between input data and metrology target values, the following accuracy indicators have been used:

Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.1}$$

Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{\sum_{i=1}^{n}\frac{|(y_i - \hat{y}_i)|}{y_i}}{n} \cdot 100 \tag{5.2}$$

where n is the number of processed wafers, y_i and $\hat{y}_i$ are the actual target and predicted value of i^{th} test wafer, respectively.

Before reporting and debating results we have obtained, we have seen in Section 4.2 that data correlation analysis has not brought satisfactory process variables reduction, therefore for this reason we have not related about it in the pre-processing phase.

To try out the goodness of the behaviour of neural networks as compared with another identification methods, first, we have modeled the process in question with linear regressions, in particular we have used Partial Least Squares regression (for more references see [15]) and the results concerning the central CVD target are summarized in Table 5.1 where a global model related to the CVD producer and a individual model for each sub-chamber of this machine were made.

Partial Least Squares Regression		
Method	**TYPE**	**MSE**
Full set	Producer	0.0543
PCA		0.0674
SS		0.0663
Full set	sub-chamber A1	0.0200
PCA		0.1234
SS		0.1266
Full set	sub-chamber A2	0.0184
PCA		0.1205
SS		0.1213
Full set	sub-chamber B1	0.0088
PCA		0.0553
SS		0.0517
Full set	sub-chamber B2	0.0022
PCA		0.0108
SS		0.0113
Full set	sub-chamber C1	0.0011
PCA		0.0050
SS		0.0062
Full set	sub-chamber C2	0.0020
PCA		0.0090
SS		0.0115

Table 5.1: Partial least squares regression performance with different variable selection techniques.

Our simulations, in addition to PCA and SS, made also use of the entire data-set; we see that the performance with full set is better than the others ones but this involves a bigger computational time cost. This is the typical engineering trade-off between computational time and estimation performances: if the computational time is not a critical issue it is preferable to exploit the entire data-set, on the other hand if the on-line implementation requires fast computation we can save time modeling on a smaller regressors data-set.

5.1 NN for VM

In Section 3.8 we presented the back-propagation algorithm of Neural Network technique we used to estimate regression coefficients for CVD thickness. In succession we explain how we have projected the Virtual Metrology Module trough neural networks structure.

First of all, we followed these guiding lines:

- for each sub-chamber, building two models in correspondence with reduction method that we have performed: the first one utilizing Principal Component Analysis; whereas the second one using Stepwise Selection.
- constructing a global model that it does not make a distinction among the various sub-chambers.
- expanding the basis with a approach which also considers the variability of sub-chamber within the whole available data-set.

According to the selected model, we divided the the starting data-set X_{GAMMA} into two sets using random indices: the first one, with a percentage of 70% of X_{GAMMA} is the training set while the second one (30% of X_{GAMMA}) is the validation set.

For training NN we have made use of Levenberg-Marquardt algorithm that we explain below.

5.1.1 Levenberg-Marquardt Algorithm

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed-forward networks), then the Hessian matrix can be approximated as

$$\mathbf{H} = \mathbf{J}^T\mathbf{J}$$

and the gradient can be computed as

$$g = \mathbf{J}^T\mathbf{e}$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard back-propagation technique that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e} \tag{5.3}$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm.

The main drawback of the Levenberg-Marquardt algorithm is that it requires the storage of some matrices that can be quite large for certain problems.

The memory space, computing the Jacobian matrix, can be a critical issue; even if there are some techniques of memory reduction, the Levenberg-Marquardt algorithm will always compute the approximate Hessian matrix, which has dimensions n x n: if the network is very large, then we might run out of memory. The reader can refer to [33] for more detailed explanations.

Fore more references, the application of Levenberg-Marquardt to neural network training is described in [36, 37].

5.1.2 Choosing number of hidden units and layers, and starting values

We have learnt that with neural networks there is not a only and exact rule for choosing:

- number of hidden layers;
- number of hidden neurons (size of hidden layers);
- starting values.

About the number of hidden layers, it has been proved that a NN composed of a input layer, one or two hidden layers and one output layer, can approximate arbitrarily well any linear or non linear function (for more references see [32]). Consequently, for this work we have decided for a structure with two hidden layers ($l = 2$) where the last one is made of one only unit ($n_2 = 1$).

In relation with the others two drawbacks, for each model we wished to implement, we made several trials, varying the number n_1 of hidden neurons of the first hidden layer among 1 and 30 e changing the starting values among a thousand of different initial conditions. This action is an attempt bunch for having the right intuition but it does not mean that it is the best possible choice. The architecture of the neural network that we considered is shown in Figure 5.1 .

To get homogeneous conditions for having an homogeneous comparison in according to the selection method we have used, we have taken the best starting value related to the central thickness measurement.

For instance, in Figure 5.2 is shown the trend of Validation Error of the entire X_{GAMMA} where we can observe that the best number of hidden neurons is $n_l = 6$.

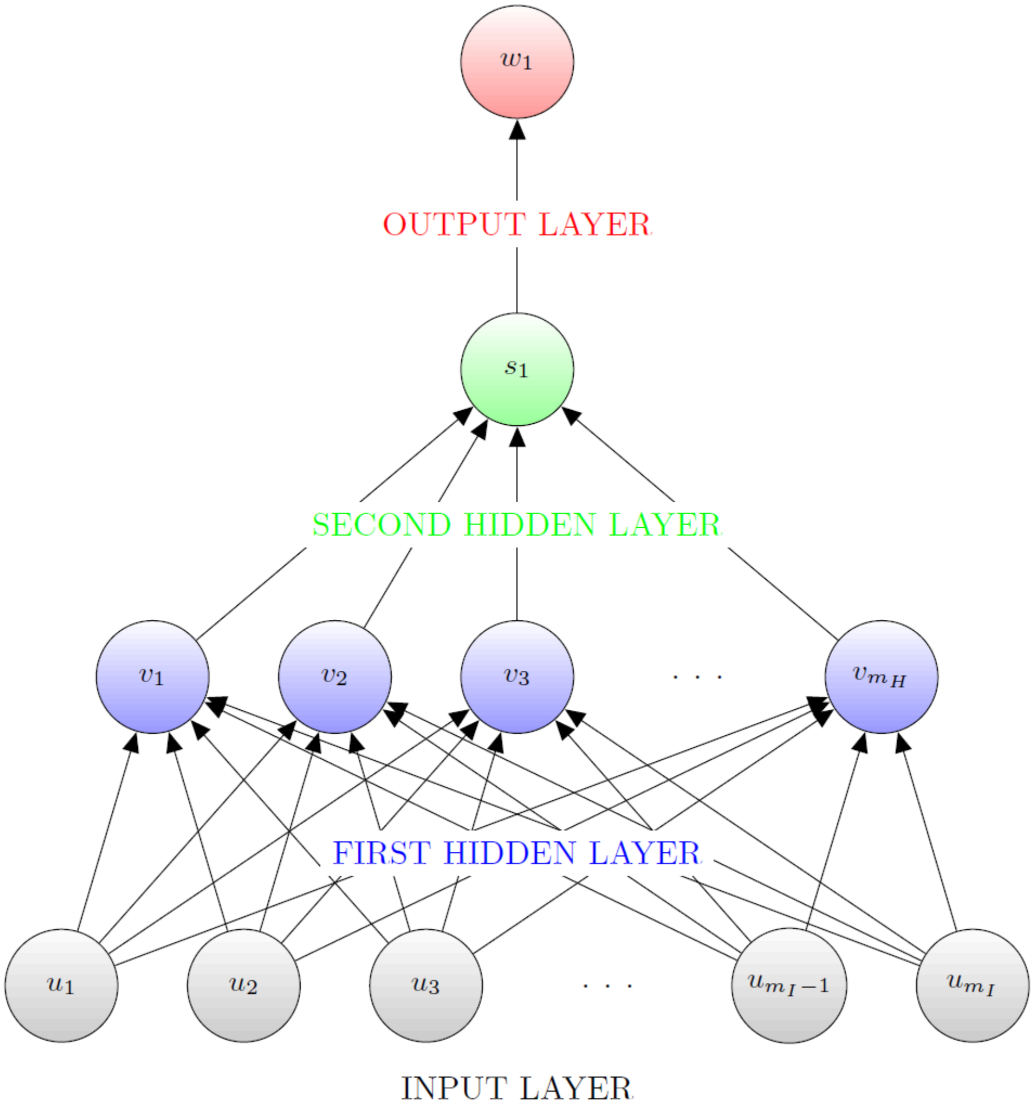


Figure 5.1: Structure of the considered NN.

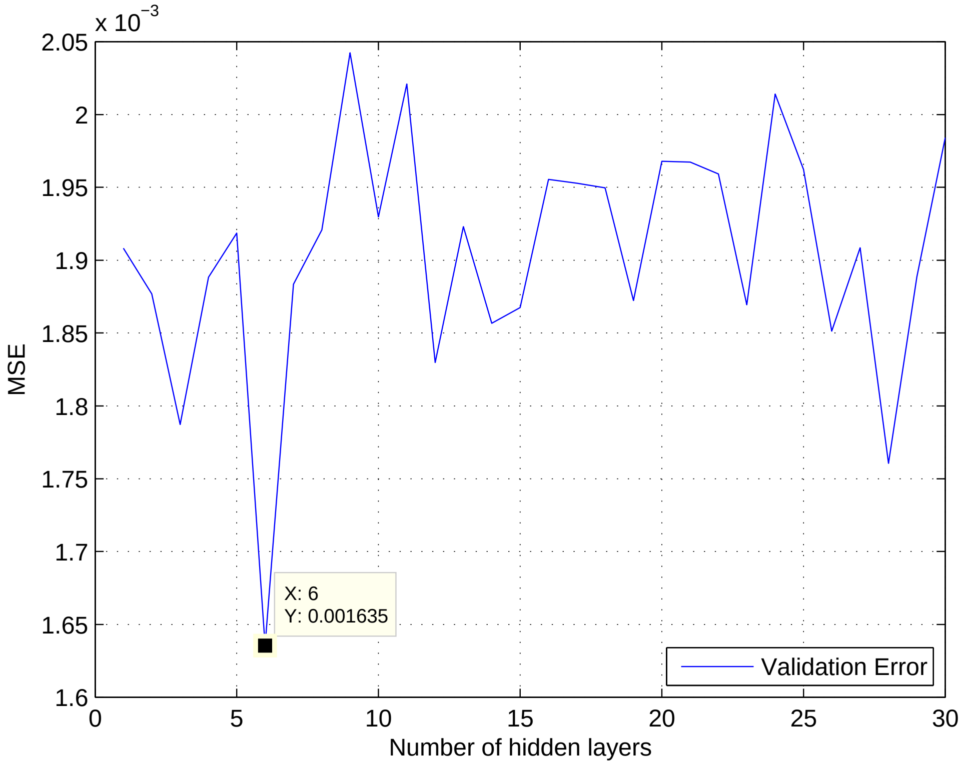


Figure 5.2: Trend of Validation Error changing the number n_l of neurons of the first hidden layer.

Before depicting and commenting upon the results we have got, in Table 5.2 the number of predictor variables taken according to the type of model and variable selection technique is shown where the level of confidence is settled on 95%. It is important to note that the counter relating to the principal component analysis is unique while that one of the stepwise selection is dependent to the output we want to select[1].

Technique	**Model Type**							
	Global	**Basis Expansion**	$\mathbf{A_1}$	$\mathbf{A_2}$	$\mathbf{B_1}$	$\mathbf{B_2}$	$\mathbf{C_1}$	$\mathbf{C_2}$
PCA	35	35	31	28	31	32	78	83
SS - 1st	2	9	5	7	11	21	12	13
SS - 2nd	19	22	6	4	18	13	18	7
SS - 3rd	34	15	4	2	17	23	25	5
SS - 4th	23	24	1	7	7	12	15	12
SS - 5th	21	17	4	6	13	10	20	8
SS - 6th	19	10	6	3	14	39	18	14
SS - 7th	3	18	5	12	8	19	18	14
SS - 8th	23	15	6	7	11	13	13	6
SS - 9th	16	18	3	7	12	15	26	13

Table 5.2: Number of components taken for type of model and according to variable selection technique with LoC = 95%.

Tables 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8 collect MSE indices for each sub-chamber.

We can make the following remarks about:

1. method applicated:
   - on the whole, neural networks provide better performance than partial least squares regression;
   - only for chamber C, especially for the sub-chamber $C2$, MSE of linear systems is on the order of that one of the feed-forward networks and sometimes it is better. A possible explanation is because chamber C is stabler and then also a linear model is able to fit in a good manner.
2. Variable selection technique performed:
   - In general stepwise selection combined with either NN or PLS provide better results than principal component analysis, this confirm that we mentioned in 4.2.7 where we said that predictor variables through stepwise regression technique were calculated with references to the output.

[1] We remember that PCA technique is performed without any reference to the output while in Stepwise Selection a predictor variable is added or removed in accordance to the output.

Technique	**CVD thickness 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00111	0.00098	0.00079	0.00077	0.00054	0.00083	0.00122	0.00061	0.00087
NN based SS	0.00439	0.00114	0.00120	0.00087	0.00088	0.00190	0.00115	0.00059	0.00096
PLS based PCA	0.01680	0.01380	0.01330	0.01260	0.01290	0.01510	0.01320	0.01260	0.01290
PLS based SS	0.01700	0.01400	0.01370	0.01340	0.01330	0.01520	0.01340	0.01280	0.01330

Table 5.3: MSE of different techniques for sub-chamber A1.

Technique	**CVD thickness 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00184	0.00087	0.00141	0.00122	0.00125	0.00145	0.00136	0.00097	0.00112
NN based SS	0.00199	0.00093	0.00120	0.00117	0.00119	0.00170	0.00155	0.00091	0.00105
PLS based PCA	0.01360	0.01320	0.01350	0.01350	0.01460	0.01330	0.01270	0.01290	0.01310
PLS based SS	0.01370	0.01350	0.01380	0.01320	0.01460	0.01390	0.01250	0.01290	0.01280

Table 5.4: MSE of different techniques for sub-chamber A2.

Technique	**CVD thickness 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00177	0.00134	0.00110	0.00064	0.00062	0.00073	0.00076	0.00110	0.00118
NN based SS	0.00120	0.00105	0.00102	0.00060	0.00064	0.00063	0.00062	0.00100	0.00105
PLS based PCA	0.00890	0.00290	0.00370	0.00760	0.00820	0.00220	0.00800	0.00770	0.00610
PLS based SS	0.00820	0.00250	0.00270	0.00720	0.00750	0.00190	0.00790	0.00750	0.00580

Table 5.5: MSE of different techniques for sub-chamber B1.

Technique	**CVD thickness 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00110	0.00063	0.00149	0.00134	0.00133	0.00142	0.00112	0.00044	0.00054
NN based SS	0.00091	0.00076	0.00143	0.00129	0.00119	0.00097	0.00118	0.00039	0.00045
PLS based PCA	0.00310	0.00081	0.00160	0.00130	0.00190	0.00130	0.00140	0.00064	0.00085
PLS based SS	0.00084	0.00075	0.00130	0.00130	0.00180	0.00094	0.00120	0.00060	0.00078

Table 5.6: MSE of different techniques for sub-chamber B2.

Technique	**CVD thickness 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00102	0.00064	0.00102	0.00058	0.00044	0.00045	0.00072	0.00062	0.00077
NN based SS	0.00071	0.00056	0.00080	0.00050	0.00035	0.00057	0.00067	0.00059	0.00056
PLS based PCA	0.00067	0.00055	0.00065	0.00050	0.00050	0.00041	0.00056	0.00049	0.00057
PLS based SS	0.00077	0.00057	0.00068	0.00055	0.00053	0.00042	0.00065	0.00058	0.00056

Table 5.7: MSE of different techniques for sub-chamber C1.

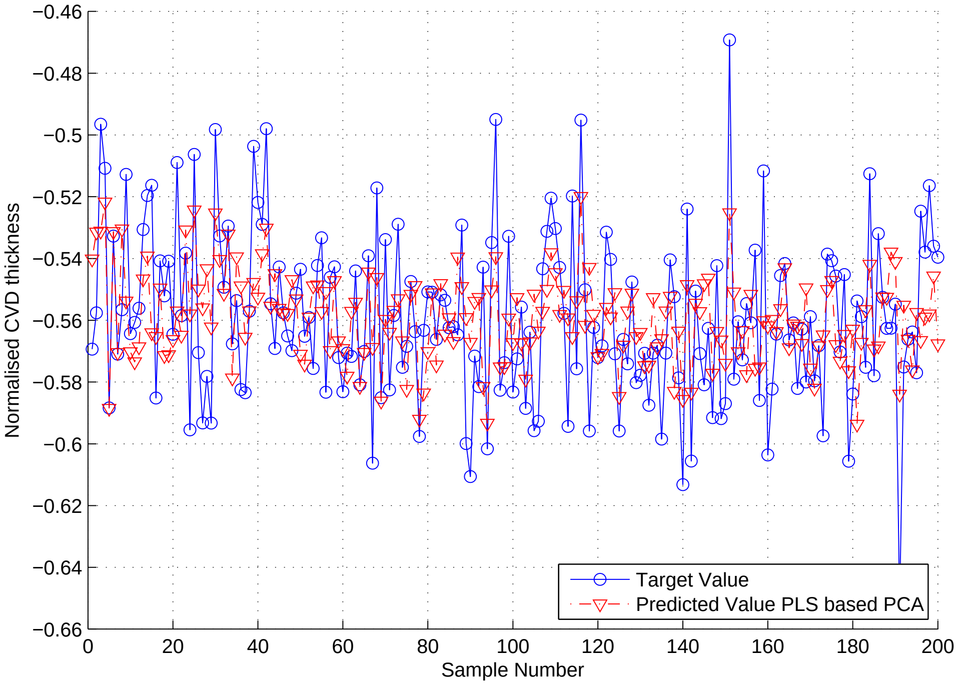
Technique	**CVD thickness 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00130	0.00077	0.00175	0.00129	0.00125	0.00173	0.00176	0.00073	0.00109
NN based SS	0.00100	0.00067	0.00106	0.00095	0.00120	0.00153	0.00169	0.00071	0.00094
PLS based PCA	0.00095	0.00062	0.00130	0.00081	0.00140	0.00120	0.00130	0.00052	0.00064
PLS based SS	0.00110	0.00074	0.00160	0.00097	0.00170	0.00130	0.00150	0.00065	0.00076

Table 5.8: MSE of different techniques for sub-chamber C2.

Figures 5.3 and 5.4 show the actual target values and the predicted values for the 5^{th} output of sub-chamber $C1$ for the different combinations of method-selection technique.

We note that the prediction of the neural network based PCA is worse than the others ones, it can not forecast in a good manner the output maintaining approximately on the mean of it.

In Figures 5.5 and 5.6 the corresponding residuals and the cumulative MSE of the different couples method-variable selection technique are shown, respectively.



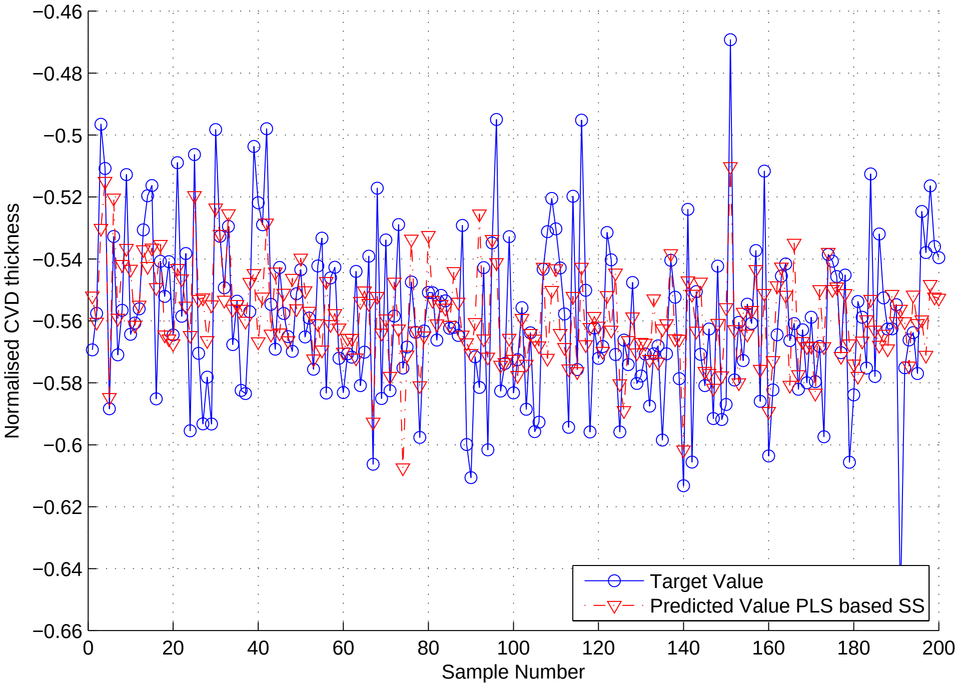


Figure 5.3: Prediction results of CVD thickness in sub-chamber C1: PLS based Principal Component Analysis and PLS based Stepwise Selection, respectively.

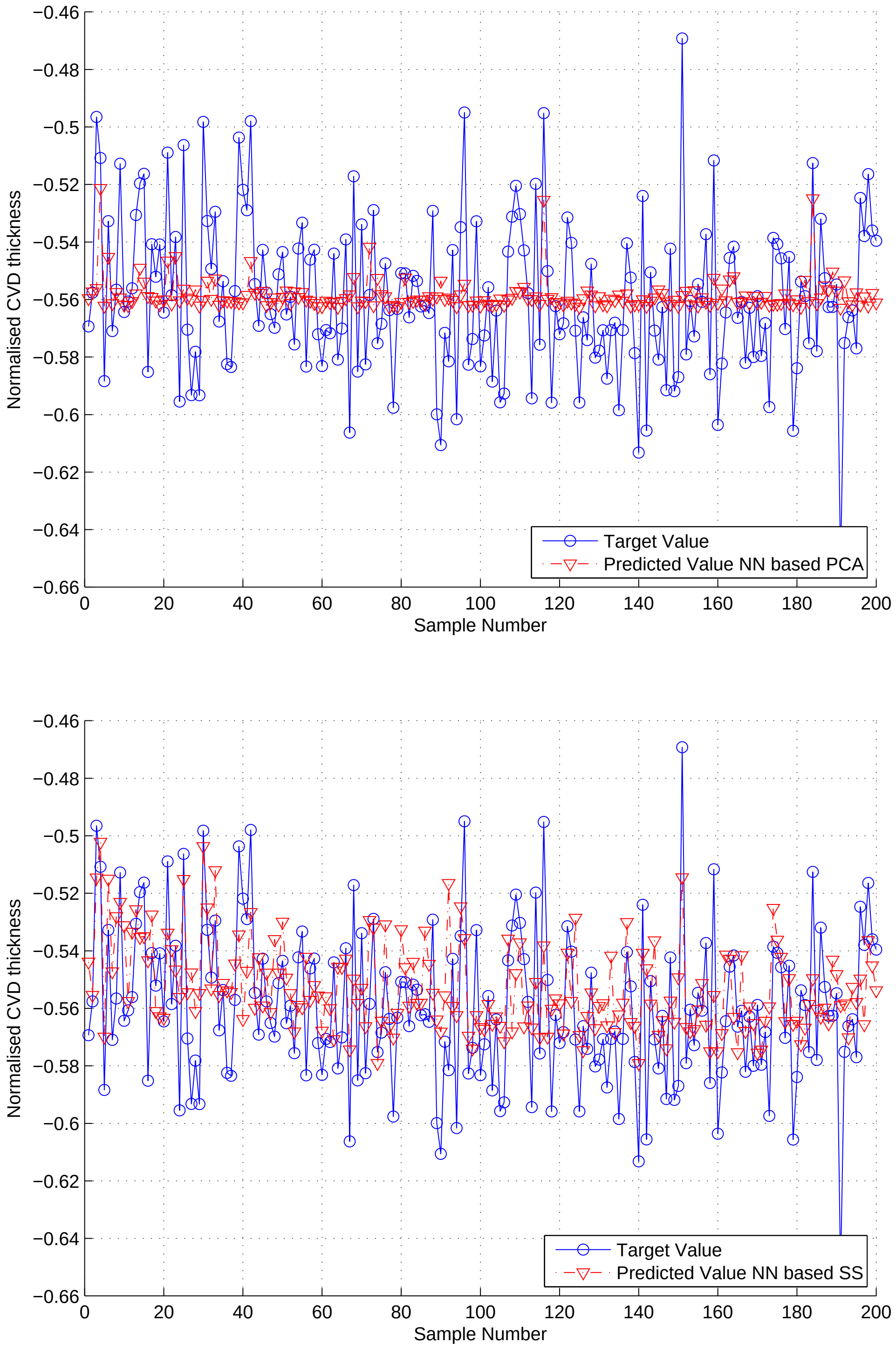


Figure 5.4: Prediction results of CVD thickness in sub-chamber C1: NN based Principal Component Analysis and NN based Stepwise Selection, respectively.

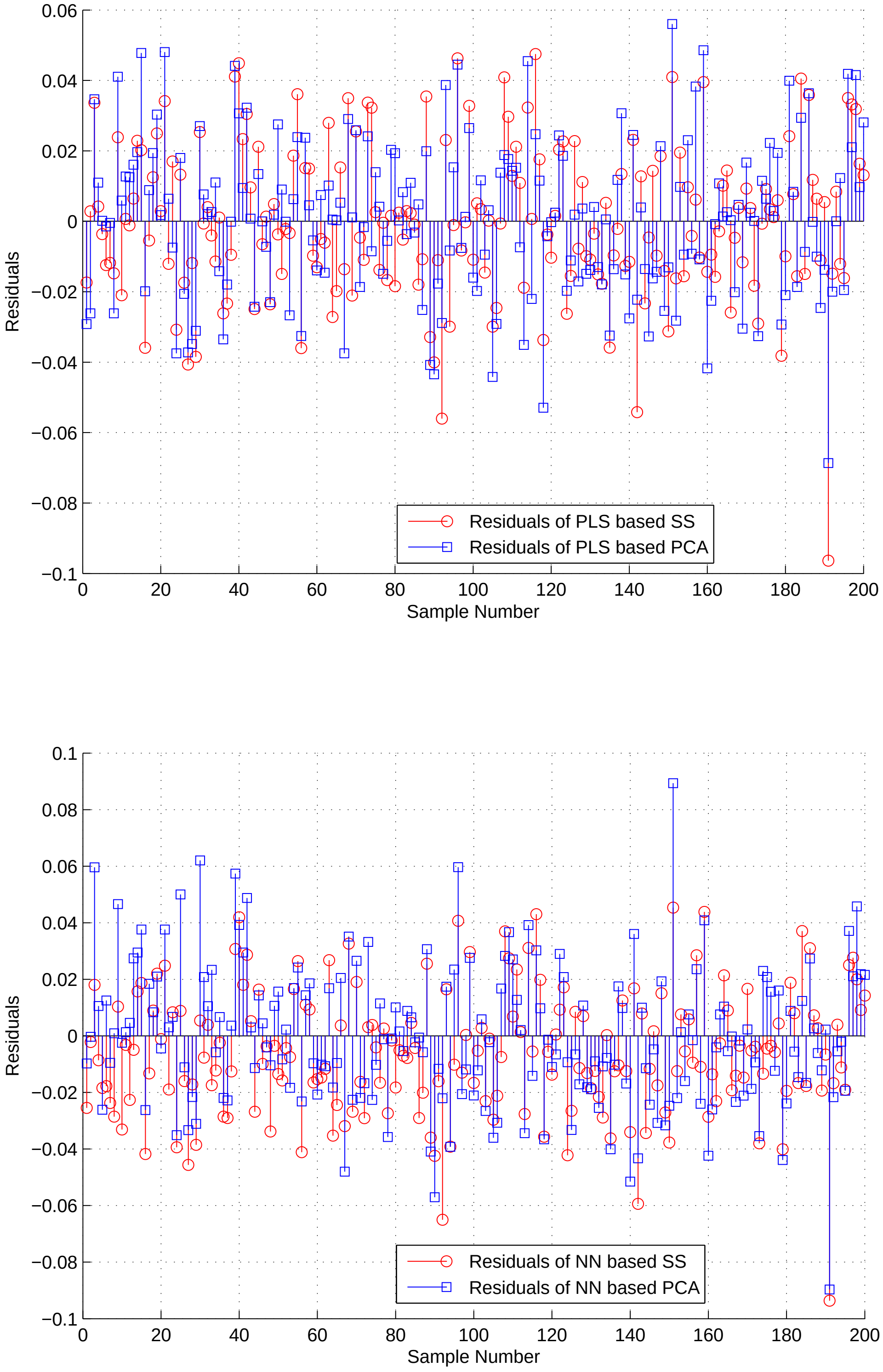


Figure 5.5: Residuals of prediction results of Figures 5.3 and 5.4, respectively.

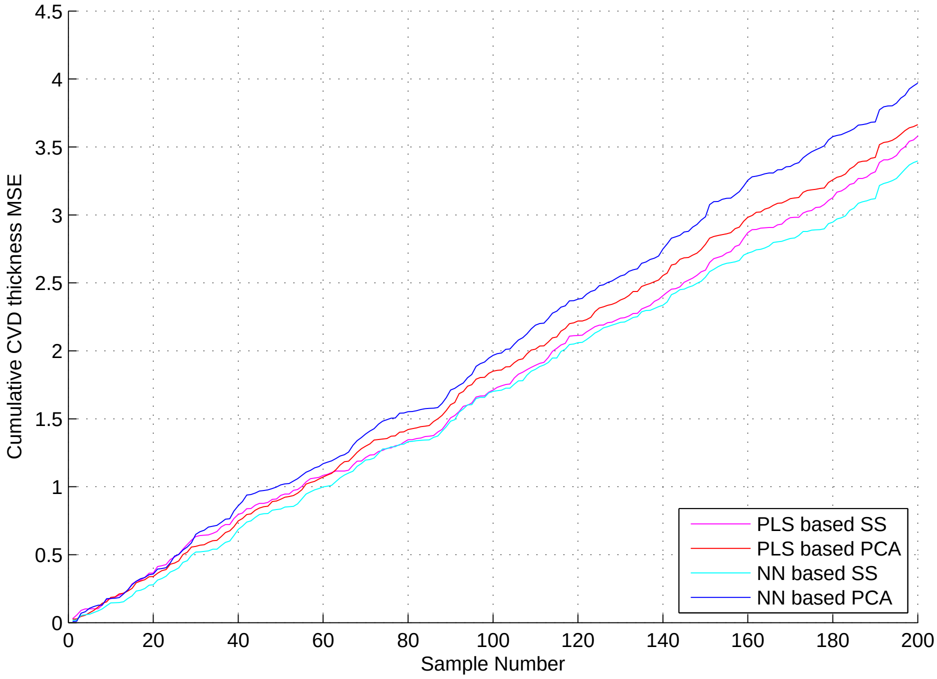


Figure 5.6: Cumulative MSE of different couples method-selection technique used.

5.1.3 Global model versus basis expansion approach

We have built a global model of the CVD producer that does not take in regard of the sub-chambers it has composed by. From the beginning we already knew that this model would work worse than the others. This is due to the fact that the sub-chambers, just observing the first three PCs (return to Section 4.2 for more details), have some completely different behaviours among them. But our aim is to compare it with a basis expansion model where we have added the variability of sub-chamber within the starting whole available data-set $\mathbf{X}_{GAMMA}$.

Model	**Input Data**	**Output Data**
Global	$\mathbf{X}_{GAMMA}$	$\mathbf{Y}_{GAMMA}$
Basis Expansion	$\tilde{\mathbf{X}}_{\Gamma}$	$\mathbf{Y}_{GAMMA}$

Table 5.9: Input and output data for global and basis expansion models.

The input and output data for both models are shown in Table 5.9 where $\tilde{X}_{\Gamma}$ is built as follows:

$$\tilde{\mathbf{X}}_{\Gamma} = \left[\mathbf{X}_{GAMMA}\, \mathbf{X}_{\Gamma}^{A1}\, \mathbf{X}_{\Gamma}^{A2}\, \mathbf{X}_{\Gamma}^{B1}\, \mathbf{X}_{\Gamma}^{B2}\, \mathbf{X}_{\Gamma}^{C1}\, \mathbf{X}_{\Gamma}^{C2} \right] \tag{5.4}$$

where $\mathbf{X}_{\Gamma}^{sub-chamber} \in \Re^{n \cdot 1}$ (sub-chamber = A1, A2, B1, B2, C1, C2) is a vector which has the element

$$i^{th} = \begin{cases} 1 & \text{if the current wafer is processed inside the corresponding sub-chamber} \\ 0 & \text{otherwise} \end{cases}$$

In Tables 5.10 and 5.11 the results (referred to performance MSE) we obtained are shown where we followed the same procedure we saw for a single sub-chamber. We can see, as aforementioned, that the global model really works worse than the corresponding basis expansion one, indiscriminately at the model and selection technique we used, therefore, the information of each sub-chamber we put inside the whole data-set $\mathbf{X}_{GAMMA}$ brings more improvement in the global model.

However, even now, in all two cases, it is proved that neural networks allow better results than linear techniques and the gap between the performance of these nonlinear systems and those ones of partial least squares regression is markedly larger than that one of individual sub-chamber model. This can mean that, although the relative and poor meaning of a global model, NN are trained sufficiently well and know to generalize in a good manner a bigger set of heterogeneous data.

Techinque	**CVD thickness - 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00224	0.00134	0.00183	0.00229	0.00167	0.00206	0.00211	0.00200	0.00213
NN based SS	0.00166	0.00110	0.00170	0.00115	0.00161	0.00157	0.00184	0.00164	0.00170
PLS based PCA	0.00810	0.00620	0.00710	0.00760	0.00800	0.00690	0.00790	0.00780	0.00770
PLS based SS	0.00820	0.00610	0.00670	0.00740	0.00780	0.00670	0.00810	0.00750	0.00740

Table 5.10: MSE of different techniques for entire CVD producer.

Techinque	**CVD thickness - 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	0.00161	0.00110	0.00158	0.00181	0.00170	0.00186	0.00224	0.00188	0.00175
NN based SS	0.00155	0.00106	0.00145	0.00112	0.00128	0.00128	0.00142	0.00105	0.00105
PLS based PCA	0.00810	0.00620	0.00700	0.00760	0.00800	0.00690	0.00790	0.00780	0.00770
PLS based SS	0.00790	0.00600	0.00660	0.00690	0.00750	0.00650	0.00710	0.00680	0.00650

Table 5.11: MSE of different techniques for basis expansion model of CVD producer.

In Tables 5.12 and 5.13 the MAPE indexes, relating only to neural networks, are reported and we can note that the level of performance of both the models using PCA is rather similar, while about that ones that makes use of SS, the basis expansion model yields better outcomes than the global pattern, with the exception of the first and second targets.

In Figure 5.7 the comparisons of the trend of cumulative MSE split for models we used are illustrated.

Techinque	**CVD thickness - 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	3.13	2.63	3.34	3.87	3.21	3.46	3.67	3.33	3.88
NN based SS	2.93	2.35	3.47	2.75	3.22	3.18	3.50	3.25	3.17

Table 5.12: MAPE of different techniques for entire CVD producer.

Techinque	**CVD thickness - 9 points**								
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
NN based PCA	3.16	2.50	3.61	3.81	3.36	3.45	3.77	3.94	3.71
NN based SS	3.00	2.55	3.06	2.73	2.76	2.68	2.92	2.52	2.41

Table 5.13: MAPE of different techniques for basis expansion model of CVD producer.

We can make these considerations:

- on the whole, indepentently of global and basis expansion models and variable selection techniques, neural networks produces better results than linear regression and this confirms what we expected during this work;
- in correspondence of samples number 558, 650 and 1616 there are some evident increases that are probably due to outliers;
- within PLS, basis expansion model based on stepwise selection has the best performance while the others ones have similar trends;
- about NN, also now the basis expansion model based on SS implements the smaller cumulative MSE, in this case followed by global one based SS. Instead, the last two patterns based on PCA have similar behaviours but with worse performances.

5.2 Conclusions

In this chapter we reported a case study of virtual metrology module in a semiconductor fab concerning to the thickness prediction in a CVD process.

The point of the matter is that the engineer has to choose the best trade-off between a more flexible system, capable of predicting the entire data-set exploiting the basis expansion approach, and a system more performing on a single sub-chamber but which estimations cannot be extend to the other sub-chambers.

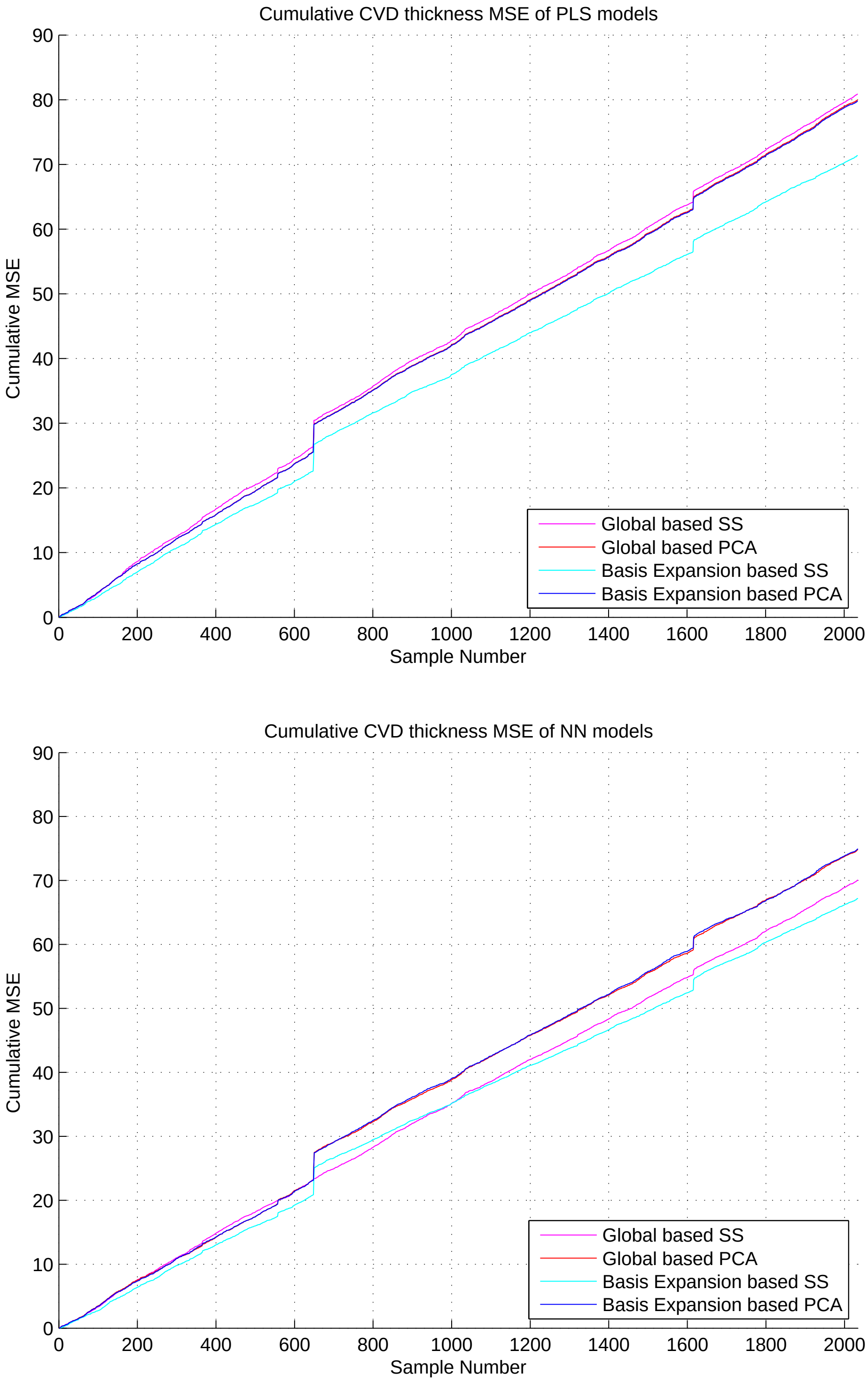


Figure 5.7: Comparison of Cumulative MSE between global and basis expansion approaches: PLS models and NN models, respectively.

Chapter 6

Summary, Conclusions and Future Work

6.1 Summary

In this thesis, some Virtual Metrology techniques for semiconductor manufacturing process control were presented and tested. It is worth noting that standard estimation techniques may experience difficulties in dealing with very large dataset, like semiconductor manufacturing process data. In order to estimate metrology data of a CVD producer, various alternative methods that select a subset of key-variables were considered:

- Partial Least Squares (PLS) based Stepwise Selection: predictor variables are selected by a stepwise procedure and then are transformed in PLS-components that are sorted by statistical importance.
- PLS based Principal Component Analysis: predictor variables are selected by a PCA method and then are transformed in PLS-components that are sorted by statistical importance.
- Neural Networks (NN) based Stepwise Selection: variables are selected by a stepwise procedure and next we used back-propagation algorithm as modeling method.
- NN based PCA: variables are selected by a PCA technique and subsequently we used back-propagation algorithm to model CVD thickness.

Experimental tests showed that neural networks (being a non linear system that models a non linear process) allow better results than linear techniques. Furthermore, concerning only to NN, the use of stepwise selection method to select main predictor variables supplies better performance than principal component analysis.

6.2 Conclusions

The manufacturing industry continues to strive for cost effective and higher quality products. The research work in this dissertation is aimed at improving product quality by providing process control at every run (i.e. for every product) and by providing product quality data to the process controller without employing extra sensors. Towards that end a VM based process control solution is proposed and developed for manufacturing processes. The approach provides a mechanism to update the quality prediction model using R2R diagnostic data and periodic metrology data.

In conclusion, the results of the present thesis suggest the flexibility of a run-to-run control system encompassing a VM module (see Figure 6.1).

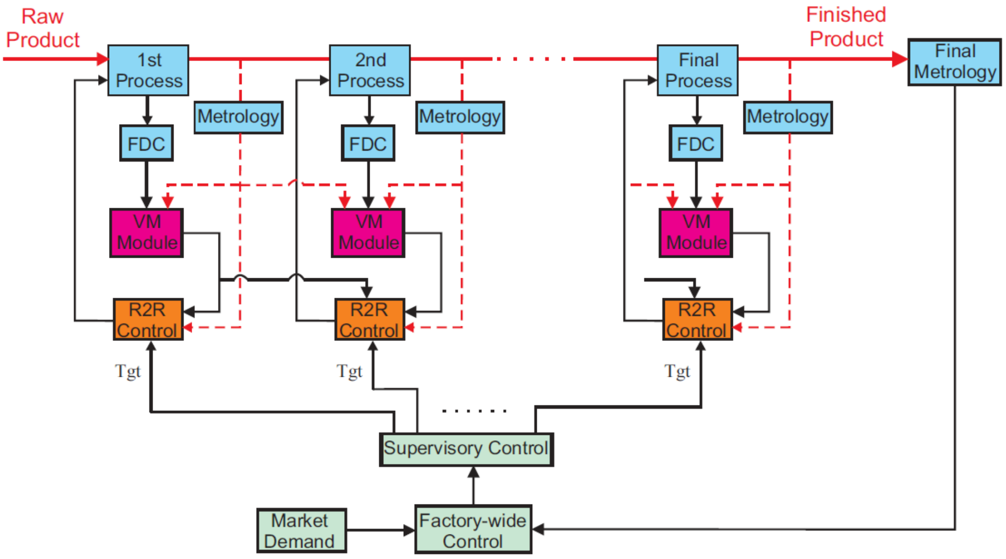


Figure 6.1: VM control system.

6.3 Future Work

The field of VM and its subsequent use for feedback control in the manufacturing industry is still evolving. This dissertation provides an introduction to the VM topic and emphasizes its importance for product quality improvement so that the manufacturing industry can meet today's stringent market demands. Necessary concepts and methods have been studied in this research to realize VM for individual processes, which opens the door for new research topics in this field. There are also a few issues that must be addressed before VM can be successfully implemented for individual processes and subsequently made part of production planning on the factory level:

new metrology strategies based on VM data: improving the strategies, both concerning the phase of pre-processing and modelling's one, here presented and testing new techniques for the same reason;

multi-recipe and multi-step processes: Many processes in semiconductor manufacturing consist of multiple process steps conducted in a sequence before a physical measurement is made on the quality attributes (or metrology variables). For example, the contact etch process has two similar steps: an oxide etch and a subsequent nitride etch. Both processes are performed in similar processing tools one after the other, but wafer CD is measured only after the second process. This situation is schematically shown in Figure 6.2.

In such cases it becomes difficult to correlate two sets of process variables with one set of metrology variables. A multi-process multi-recipe approach is therefore needed for the formulation of VM problem. In the example case of contact etch, the second process, nitride etch, determines the final CD. It seems logical to use process data from this process only in building the VM module. However, in general, data from all process steps should be used in the VM module and each step components should be weighted according to the corresponding level of their influence on the final quality variables.

In addition, the R2R controller must be capable of generating two sets of control values for both steps of the process.

Note that this is a future research topic.

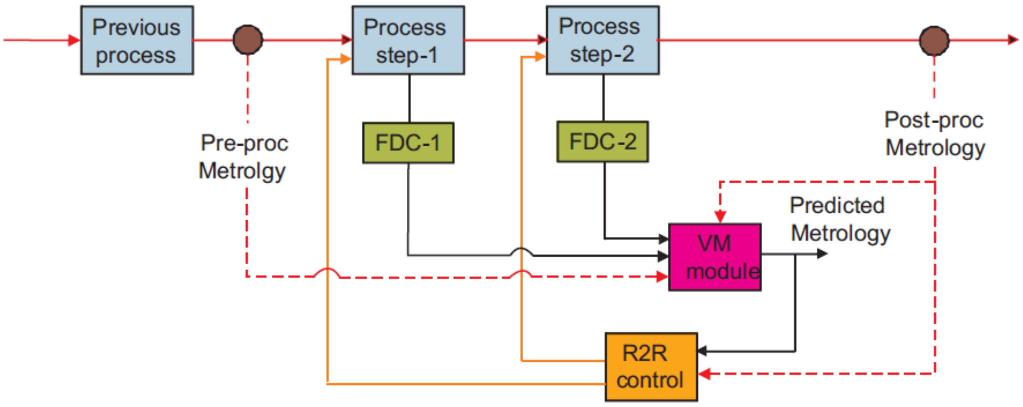


Figure 6.2: VM module for multi-step process.

Bibliography

[1] P. Chen , S. Wu, J. Lin, F. Ko, H. Lo, J. Wang, C.-H Yu & M.-S. Liang, *Virtual metrology: a solution for wafer to wafer advanced process control*, in Proceedings of IEEE International Symposium on Semiconductor Manufacturing (ISSM 2005), pp.155-157. September 2005, San Jose, CA, USA.

[2] Y.-J. Chang, Y. Kang, C.-L. Hsu, C.-T. Chang & T.-Y. Chan, *Virtual Metrology Technique for Semiconductor Manufacturing*, in Proceedings of the Conference on Neural Networks Sheraton Vancouver Wall Center Hotel, 2006, pp. 5289-5293.

[3] J.-C. Yung-Cheng & F.-T. Cheng, *Application development of virtual metrology in semiconductor industry*, in Proceedings of the 32nd annual conference of IEEE Industrial Electronics Society (IECON 2005). Los Alamitos, CA, USA.

[4] F.-T. Cheng, H.-C. Huang & W.-M. Wu, *Dual-Phase Virtual Metrology Scheme*, in IEEE Transactions on Semiconductor Manufacturing, 20(4), November 2007, pp. 566-571.

[5] F.-T. Cheng, *Researching Strategy and Development Proposal of e-Manufacturing*, in Automation Division of National Science Council. Taiwan, R.O.C, October 2004.

[6] J. Besnard & A. Toprac, *Wafer-to-wafer virtual metrology applied to run-to-run control*, in Proceedings of the 3rd ISMI symposium on manufacturing effectiveness. Austin, Texas (USA), 2006.

[7] T.-H. Lin, M.-H. Hung, R.-C. Lin & F.-T. Cheng, *A virtual metrology scheme for predicting CVD thickness in semiconductor manufacturing*, in Proceedings of of IEEE International Conference on Robotics and Automation, May 2006, pp. 1054-1059.

[8] Y.-C. Su, T.-H. Lin, F.-T. Cheng & W.-M. Wu, *Accuracy and Real-Time Considerations for Implementing Various Virtual Metrology Algorithms*, in IEEE Transactions on Semiconductor Manufacturing, 21(3), August 2008, pp. 426-434.

[9] J. Moyne, *Making the move to fab-wide APC*, in Solid State Technology, 47(9), September 2004, pp. 47.

[10] J. Moyne, E. del Castillo, and A. M. Hurwitz, *Run-to-Run Control in Semiconductor Manufacturing.* CRC Press, 2001.

[11] T.-F. Edgar, S.-W. Butler, W.-J. Campbell, C. Pfeiffer, C. Bode, S.-B. Hwang, K.-S. Balakrishnan & J. Hahn, *Automatic control in microelectronics manufacturing: Practices, challenges, and possibilities*, in Automatica, 36(11), 2000, pp. 1567-1603.

[12] M.-H. Hung, T.-H. Lin, F.-T. Cheng & R.-C Lin, *A novel virtual metrology scheme for predicting CVD thickness in semiconductor manufacturing* in IEEE/ASME Transactions on Mechatronics, 12(3), June 2007, pp. 308-316.

[13] P. Kang, H. Lee, S. Cho, D. Kim, J. Park, C.-K. Park & S. Doh, *A virtual metrology system for semiconductor manufacturing*, in Expert Systems With Applications, 2009.

[14] A.A. Khan, *Predictive Inspection Based Control using Diagnostic Data for Manufacturing Processes*, PhD thesis, The University of Michigan, 2007.

[15] A.A. Khan, J.R. Moyne & D.M. Tilbury, *Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares*, in Journal of Process Control, 18(10), 2008, pp. 961-974.

[16] A.A. Khan, J.R. Moyne & D.M. Tilbury, *An Approach for factory-wide control utilizing virtual metrology*, in IEEE Transactions on Semiconductor Manufacturing, 20(4), November 2007, pp. 364-375.

[17] A. Ferreira, A. Roussy & L. Conde, *Virtual metrology models for predicting physical measurement in semiconductor manufacturing*, in Proceedings IEEE/SEMI Advanced Semiconducor Manufacturing Conference (ASMC), Berlin, May 2009, pp. 149-154.

[18] S. Lynn, J. Ringwood, E. Ragnoli, S. McLoone & N. MacGearailt, *Virtual metrology for plasma etch using tool variables*, in Proc. IEEE/ SEMI Adv. Semicond. Manuf. Conf. (ASMC), Berlin, May 2009, pp. 143-148.

[19] Donald O. Hebb, *The organization of behavior: a neuropsychological theory.* Wiley, New York, 1949.

[20] G. Dreyfus, *Neural Networks Methodology and Applications.* Hardcover, 2002.

[21] S.-J. Qin, G. Cherry, R. Good, J. Wang & C.-A. Harrison, *Semiconductor manufacturing process control and monitoring : a fabwide framework*, in Journal Process Control, 16(3), 2006, pp. 179-191.

[22] B.-A. Pearlmutter, *Dynamic recurrent neural networks*, in Technical Report CMU-CS-90-196, Carnegie Mellon University School of Computer Science, Pittsburgh, PA, December 1990.

[23] A.-C. Rencher, *Methods od multivariate Analysis.* Hardcover 2002.

[24] C.-M Bishop, *Neural Networks for Pattern Recognition.* Oxford, 1995.

[25] C.-M Bishop, *Pattern Recognition and Machine Learning.* Springer , 2006.

[26] C.-D. Himmel & G.-S. May, *Advantages of Plasma Etch Modeling Using Neural Networks Over Statistical Techniques*, in in IEEE Transactions on Semiconductor Manufacturing, 6(2), May 1993, pp. 103-111.

[27] Electronic lectures of R.-V Jones, *the Robert L. Wallace Professor of Applied Physics Emeritus* in the School of Engineering and Applied Sciences, Harvard University, that can be consulted at http://people.seas.harvard.edu/∼jones/es154/lectures/lecture_2/materials .

[28] Electronic lectures that can be consulted at http://www.lpe-epi.com .

[29] Electronic lectures that can be consulted at http://www.almaden.ibm.com/st/chemistry/lithography/immersion .

[30] Electronic lectures of Ray McBride, that can be consulted at http://raymcbride.com/2010/01/15/artificial-neural-networks .

[31] Electronic lectures that can be consulted at http://www.learnartificialneuralnetworks.com .

[32] T. Hastie, R. Tibshirani & J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd Edition, Springer, 2008.

[33] H. Demuth, M. Beale & M. Hagan, *Neural Network Toolbox^{TM} 6, User's Guide*, The MathWorks, Inc.

[34] S. Kullback, *Information theory and statistics.* John Wiley and Sons (NY), 1959.

[35] S. Kullback, *Letter to the Editor: The Kullback-Leibler distance*, in The American Statistician 41(4), pp. 340-341, 1987.

[36] M.-T. Hagan & M. Menhaj, *Training feed-forward networks with the Marquardt algorithm*, in IEEE Transactions on Neural Networks, 5(6), November 1999, pp. 989-993.

[37] M.-T. Hagan, H.-B. Demuth & M.-H. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

Acnowledgments

Finally, it's time to write the last page of my thesis. Indeed, maybe the most important page, since friendship and love of all people who knowingly or unknowingly have contributed to this work is something that will last in time and will never abandon me.

First of all, I want to thank my parents and my sisters, in other words all my family. No amount of gratitude to them would be sufficient. I am very fortunate to be where I am today because of their constant support and encouragement given that sometimes I like to take it easy.

Many thanks to all my friends, from those ones who are no more to those ones who have the fortune to be here, from my "thick as thieves" friends to those ones I had the chance to know in these nearly "7" amazing years of university, someone of them have also become closer friends. With them I shared, I share and I hope to share a lot of moments of real happiness and amusement. I planned to give a full list of names, but I realized it would have been too long!

I would like to thank my thesis advisor Alessandro Beghi for the chance that he's granted me to work in an charming and interesting semiconductor's world.

I am grateful to Ph.D. student Gian Antonio Susto for his invaluable support and mentorship in research-related and research-unrelated issues, and for getting me acquainted with the world of research.

To all these people, and to those who unintentionally I forgot, an only thing I have to say:

THANK YOU.

Vicenza, June 2010 *Nicola Bertorelle*

Ringraziamenti

Finalmente, è giunto il momento di scrivere l'ultima pagina della mia tesi. In effetti, forse la pagina più importante, perchè l'amicizia e l'amore di tutte le persone che consciamente o inconsciamente hanno contribuito a questo risultato è qualcosa che durerà nel tempo e che non mi abbandonerà mai.

Prima di tutto, ringrazio i miei genitori e le mie sorelle, in poche parole tutta la mia famiglia. Nessuna espressione di gratitudine nei loro confronti sarebbe sufficiente. Sono molto fortunato a essere dove sono ora, grazie al loro costante supporto e incoraggiamento dato che qualche volta mi piace prendermela con calma.

Grazie di cuore a tutti i miei amici, da quelli che non ci sono più a tutti quelli che hanno la fortuna di esserci, dagli amici più stretti a quelli che ho avuto la fortuna di conoscere in questi quasi "7" bellissimi anni di università, alcuni di loro son diventati anch'essi amici per la pelle. Con loro ho condiviso, condivido e spero di condividere in fututo molti momenti di vera felicità e vero divertimento. Avevo pensato di scrivere una lista con i loro nomi, ma mi sono accorto che sarebbe stata troppo lunga!

Voglio ringraziare il mio relatore Alessandro Beghi per l'oppurtunità concessami di lavorare nel mondo affascinante e interessante dei semiconduttori.

Sono grato al dottorando Gian Antonio Susto per il suo prezioso supporto negli argomenti di ricerca (e non), e per avermi fatto conoscere da vicino il mondo della ricerca.

A tutte queste persone, e a quelle che involontariamente ho dimenticato, una solo cosa ho da dire:

GRAZIE.

Vicenza, Giugno 2010 *Nicola Bertorelle*