THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : AUTOMATIQUE - PRODUCTIQUE

Arrêté ministériel : 25 mai 2016

Présentée par

Zilong ZHAO

Thèse dirigée par **Nicolas MARCHAND,** Directeur de recherche au CNRS,
co-dirigée par **Bogdan ROBU,** Maître de Conférences à l'Université Grenoble-Alpes et
co-dirigée par **Louis JOB,** Professeur émérite de l'Institut d'Études Politiques de Grenoble, agrégé de Sciences Économiques

Préparée au sein du **GIPSA-lab**
dans **l'École Doctorale Électronique, Électrotechnique, Automatique et Traitement du Signal**

Extracting Knowledge from Macroeconomics, Images and Unreliable data

Thèse soutenue publiquement le **10 décembre 2020**,
devant le jury composé de :

Monsieur NICOLAS MARCHAND
DIRECTEUR DE RECHERCHE AU CNRS, Directeur de thèse
Madame MARY-FRANÇOISE RENARD
PROFESSEUR, UNIVERSITÉ DE CLERMONT-AUVERGNE, AGRÉGÉE DE SCIENCES ÉCONOMIQUES, Rapportrice
Monsieur EDOUARD LAROCHE
PROFESSEUR, UNIVERSITÉ DE STRASBOURG, Rapporteur
Monsieur DIDIER GEORGES
PROFESSEUR, UNIVERSITÉ GRENOBLE-ALPES, Président
Madame LYDIA Y. CHEN
ASSOCIATE PROFESSEUR, TU DELFT, Examinatrice
Monsieur GUILLAUME MERCÈRE
MAÎTRE DE CONFÉRENCES HDR À L'UNIVERSITÉ DE POITIERS, Examinateur
Monsieur LOUIS JOB
PROFESSEUR ÉMÉRITE DE L'INSTITUT D'ÉTUDES POLITIQUES DE GRENOBLE, AGRÉGÉ DE SCIENCES ÉCONOMIQUES, Invité
Monsieur BOGDAN ROBU
MAÎTRE DE CONFÉRENCES À L'UNIVERSITÉ GRENOBLE-APLES, Invité

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

At the time that I finish this thesis, I have spent 21 years of my life in the school. Most of time, I learn from previous studies. But for a PhD candidate, it demands to make original contribution to knowledge: create something new and add some important pieces to the sum of human understanding. Even though I cannot be perfect for lots of work, but I have never forgotten these requirements, and it makes me not dare to relax in scientific research. The road to scientific research has never been smooth sailing, just like anything in life. But I am very fortunate to have many mentors and a supporting family. Therefore, I want to dedicate this chapter of thesis to show my gratitude to them.

In the first place, I would like to thank my advisors Bogdan, Louis, Nicolas, Luc and Ioan for their guidance, encouragement, dedication and fellowship. Even though Luc and Ioan are not my official advisors, but from the bottom of my heart, they have no difference from my other three advisors. At the beginning of my PhD, I was just a master student with computer science background. I cannot forget the hundreds of nights that I read the books of economics and control theory, I always have the endless questions. But through all the exchanges of emails and the weekly meetings, I end up as a student that can publish scientific research papers in all above areas. If not for their dedication, motivation and energy, this thesis would undoubtedly never achieved fruition.

I would also like to thank all my collaborators during my PhD. I have the great opportunities to collaborate with Lydia Y. Chen, Robert Birke, Sara Bouchenak, Sonia Ben Mokhtar, Rui Han and Sophie Cerf. They all spent their times to explain their ideas and contributed their works to part of my publications. Without their help, that thesis will never happen.

This work would not have been qualified as a PhD contribution without careful reviews and examinations. I thank the jury members Mary-Françoise Renard, Edouard Laroche, Didier Georges, and Guillaume Mercère for the time and expertise they gave during this process.

Research environment is also important to the work. I want to thank all my colleagues in GIPSA-Lab. All the talks we had, all the ideas we shared have all contributed to this thesis. The great help of technical and administrative services plays also an important role for my research in GIPSA-Lab, I want to thank for their help.

PhD is not only a challenge for intelligence, but also a psychological test. I want to thank all my friends for their caring, and of course my parents and my wife for their love and companies. They may never understand what Machine Learning or Macroeconomics talks about, but without their unconditional and unreserved support and encouragement, I would never have had a chance to start and finish this PhD.

These were the happiest three years I have had, challenging but undoubtedly exciting. Thanks for everyone I met along this journey.

Part I

Generalities

Chapter 1

Introduction

The first chapter gives a broad outline of this thesis, we introduce the concerned research domains and the topics of the interdisciplinary studies between them. The motivations and contributions are discussed. A reading road-map of this manuscript is provided in the end.

1.1 Context, Motivation and Applications

System identification and machine learning are two similar concepts independently used in automatic/econometrics and computer science community. System identification uses statistical methods to build mathematical models of dynamical systems from measured data [124]. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so [74]. Except the final prediction accuracy of the model, converging speed and stability are another two key factors to evaluate the training process, especially in the on-line learning scenario, and these properties have already been well studied in control theory. Therefore, this thesis will implement the interdisciplinary researches between these areas.

The conducted studies can be divided into three topics, Fig. 1.1 illustrates the main concerned domains for each topic of studies. In general, the process of system identification / machine learning, can be summarized to three steps: 1) data preparation, 2) parameter estimation / learning or training and 3) system validation / testing. The second step is just one concept, two statements, and the main difference is in the third point. For some applications of system identification, especially the model-driven applications, use the same training data to valid the model (examples in chapter 6 in [80]), while machine learning algorithm specifically prepares a testing dataset to evaluate the prediction accuracy. The main contributions of this thesis are on the first two steps, where we incorporate control theory into these steps to improve the performance under certain circumstances.

The first topic is optimal control on economic model. Obviously, this topic consists of two parts: 1) system identification and 2) optimal control. Economists develop economic models to explain consistently recurring relationships [19]. E.g., the optimal growth model: Solow-Swan model [125, 127], which studies long-run economic growth by looking at capital accumulation, labour or population growth, and increases in productivity (commonly referred to at technological progress). And Ramsey-Cass-Koopmans model [109, 22, 73] that analyses the consumer optimization, endogenizing the saving rate while take interest rate and discount rate into consideration. Econometrics is the tool to conduct these analysis (dynamic optimization technique is used). Econometrics uses economic theory, mathematics, and statistical inference to quantify economic phenomena. In first part of this topic, we first present an analysis which uses only method from econometrics on China macroeconomic data. This work is not only a good practice for us (who with the background in automatic and computer

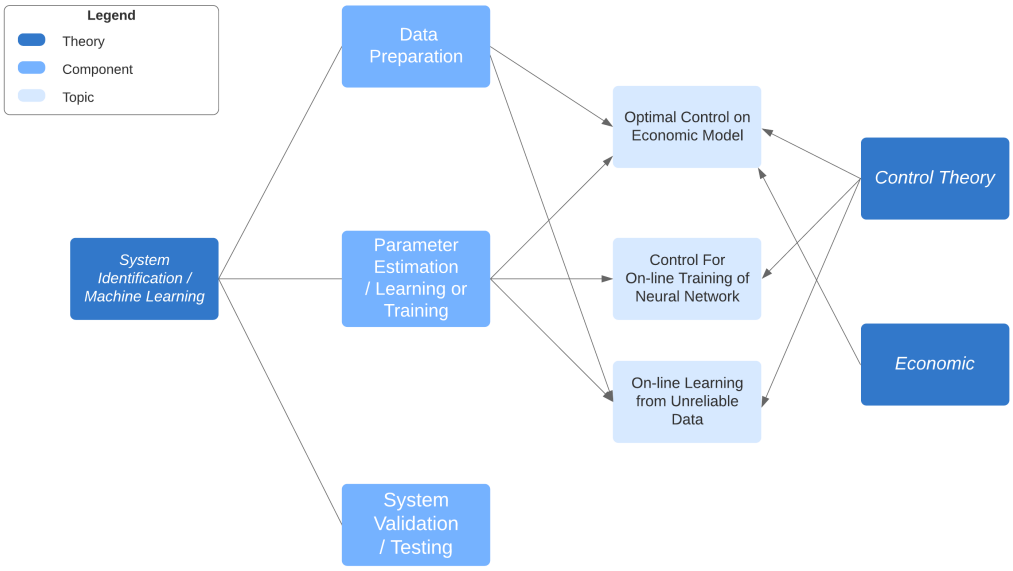


Figure 1.1 – Mainly concerned domains for the topics of study

science) to understand the theories used in econometrics, but also make the contributions in economic area, to reveal the trend of China's economic growth transition: from export-oriented to consumption-oriented. In second part of this topic, we perform the system identification on France macroeconomic data with Vector AutoRegressive (VAR) framework with Least Squares (LS) estimation, the identified model is presented in the favour of control, which means the model is represented on state space form. In economic area, there are studies to discuss extending VAR framework to analyze scenarios with unobservable explanatory variables by using state space models [78, 136, 84], because when explanatory variables are not observable, LS estimation is not usable. However, one can apply likelihood based inference, since the so-called *Kalman filter* allows to construct the likelihood function associated with a state space model. Except dynamic linear models with unobserved components, many other dynamic time series models in economics can also be represented in state space form, for instance, 1) autoregressive moving average models or 2) time varying parameter models [40]. In our case, the reason to use state space form instead of autoregressive equation is because it is easier to design the optimal controller via Linear Quadratic Regulator (LQR) solution. The LQR algorithm is essentially an automated way of finding an appropriate state-feedback controller. Once the model is estimated, the control system can apply the control law to bring the system to the desired state (e.g. a 3% yearly constant growth rate of Gross Domestic Production (GDP) in the macroeconomic model). We can also impose perturbations on outputs and constraints on inputs. This simulation can closely emulate the real world situation of economic crisis (e.g. 2008 financial crisis and Covid-19 pandemic). And economists can observe the recovery trajectory of economy with limited resources, which gives meaningful implications for policy-making. Control theory has been successfully implemented in economic domain to cope with problems such as: monetary policy [20, 62], fiscal policy [90] and resource allocation [29, 66]. Our work makes the contributions to expand it to macroeconomic models.

The second topic talks about using control theory to improve the on-line training of neural network. For on-line training scenario, training data comes in batches. One challenge of this setting is that the time interval between two data batches may be short, so we need the model to learn as quickly as possible. Another uncertainty is that the data distribution between different data batches may be very different, and we need to ensure that the model continues to improve. Learning rate and gradient are two main factors to control the converging speed, the state of the art learning rate algorithm can

be grouped into two categories: 1) time-based and 2) adaptive gradient. The time-based method prefixes a trajectory of learning rate before training process, no matter it is monotonous decreasing [26], decreasing with sine wave oscillation [2] or cyclical changing [123] over time. The advantage is easy to adopt, but the disadvantage is that the learning rate can not be adjusted even for some data batches, we could (should) certainly have a bigger (smaller) learning step. Adaptive gradient method such as Adam [72] is recent state of the art, but research [138] suggested that adaptive gradient methods do not generalize as well as stochastic gradient descent (SGD). These methods tend to perform well in the initial portion of training but are outperformed by SGD at later stages of training. Even though the algortihms such as AmsGrad [111] and AdaBound [89] claimed they fixed the defects in Adam by their method (see Sec. 8.4.2), but their results tested under on-line learning scenario do not show advantages (see Sec. 10.3). Therefore, we propose our performance-based learning rate algorithm: E (Exponential) / PD (Proportional Derivative) feedback control. Training a neural network is the process to minimize the cost function, i.e. minimize the loss value, in our designed control system, we represent the Convolutional Neural Network (CNN) as plant, the learning rate as control signal and the loss value as error signal. The experiments are based on CIFAR10 [76] and Fashion-MNIST [139] datasets, the results show that not only our algorithm outperforms the comparisons in final accuracy, final loss and converging speed, the result curve of accuracy and loss are also extremely stable near the end of training.

Still for the second topic, one observation from E/PD experiments is that when the loss value continuously decreases, our learning rate decreases too. But as the loss continuously decreases, it means we are in the good direction of training, we should not decrease the learning rate at that time. To prevent this sudden drop of learning rate at PD phase, we propose an event-based learning rate algorithm based on E/PD: Event-Based Learning Rate. Results show that event-based E/PD improves original E/PD in final accuracy, final loss and converging speed. Even the new algorithm introduces small oscillations to the training process, but the influence is minor.

Another observation from E/PD experiment is that on-line learning fixes a training epoch number for each data batch. But as E/PD converges really fast, the significant improvement only comes from the beginning epochs for each data batch, the latter ones do not have much contributions for the training. Therefore, we propose another event-based control based on E/PD: Event-Based Learning Epochs, which inspects the historical loss value, when the progress of training is lower than a certain threshold, we pass the current data batch, turn to welcome the next batch. The experiments are based on CIFAR10 and CIFAR-100 [76] datasets. Results show that event-based learning epochs can save up to 67% epochs on CIFAR-10 without degrading much model performance.

The third topic focus on noisy (dirty) label data learning problem. This topic retraces to a fundamental assumption in system identification and machine learning: the data source is clean, i.e., features and labels are correctly set. But big data are everywhere in research now, and the data sets are only getting bigger, which makes it challenging to ensure the data quality. In this part, we only consider the noise on data labels instead of on data features. Considering noisy data in classification algorithms is a problem that has been explored in the machine learning community as discussed in [44, 14, 101]. And our motivating case studies (Sec. 13.2 and . 14.2) also show that noisy label data can degrade the performance of machine learning algorithms. To tackle this problem, we propose a generic framework: Robust Anomaly Detector (RAD), RAD is a framework that model within RAD can continuously learn from dirty label data. It is called anomaly detector is because we implement this framework for anomaly detection purpose in our analysis, but its usage is not limited to anomaly detection. The data selection part of RAD is a two-layer framework, where the first layer is primarily used to filter out the suspicious data, and the second layer mainly detects the anomaly patterns from the remaining data. With our designed *ensemble prediction*, predictions from both layers contribute to the final anomaly detection decision. Two experiments are conducted on two datasets: 1) IoT device

attack detections 2) Google Cluster task failure predictions. And the training scenario is also setting to on-line learning scenario, a small difference is that the on-line learning in this part trains the model with all accumulated data instead of only latest data. Results show that RAD can continuously improve model's performance under the presence of noise on labels, comparing to the scenario without filtering any noisy label data. And the experiment with varying noise level shows that RAD can resist on high noise level.

RAD indeed improves the result with noisy label data, but it is not flawless. One shortcoming is that we use only first layer model to do the data selection, meanwhile we train two models simultaneously. We should include second layer to do the data selection too. As for that, we propose a new framework RAD Voting, the difference is that RAD Voting selects training data based on the conflict of predictions from two models. Also as we observed from RAD experiment, the models are improving over time, RAD Voting uses thus current model to re-select the training data from old data batches to improve training data quality. Results show that RAD Voting outperforms RAD in final accuracy.

In the end of topic three, we introduce another extension of RAD: RAD Active Learning, which we introduce an expert in the framework, the data selection part is similar as in RAD Voting, but when there is conflict on the prediction of two models, these uncertain data will send to expert, and expert will return the ground true labels of these data. The experiment results are very promising, the RAD Active Learning performs almost as good as the case where there is no noise on labels.

This section briefly introduced the motivations and main contributions in this thesis, the detail of related work for each topic are given at the beginning of Part. II III and IV, respectively. The very detailed motivations and contributions are presented in Sec. 2.4 and 3.2. Part. V draws conclusions on this work and provides insights of future works that would worth investigating on.

1.2 Main Results and Collaborations

1.2.1 Publications

The work developed in this thesis has lead to several contributions which have been published in various venues, both the control and computing systems communities. All my works have been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.

International Journals

- Zilong Zhao, Sophie Cerf, Bogdan Robu, Nicolas Marchand. **Event-Based Control for Online Training of Neural Networks** IEEE Control Systems Letters (*L-CSS*), vol. 4, no. 3, pp. 773-778, July 2020. [145].
- *Minor revision:* Zilong Zhao, Sophie Cerf, Bogdan Robu, Nicolas Marchand, Sara Bouchenak. **Enhancing Robustness of On-line Learning Models on Highly Noisy Data.** IEEE Transactions on Dependable and Secure Computing (*TDSC*), Special Issue: Artificial Intelligence/Machine Learning for Secure Computing
- *Submitted:* Zilong Zhao, Louis Job, Luc Dugard, Bogdan Robu. **Modelling and dynamic analysis of the domestic demand influence on the economic growth of China.** Review of Development Economics
- *TBD:* Zilong Zhao, Bogdan Robu, Ioan Landau, Nicolas Marchand, Luc Dugard, Louis Job **Modelling and Optimal Control of MIMO System - French Macroeconomic Model Case.** TBD

International Conferences

- Zilong Zhao, Louis Job, Luc Dugard, Bogdan Robu. **Modelling and dynamic analysis of the domestic demand influence on the economic growth of China.** Conference on International Development Economics, Nov. 2018, Clermont-Ferrand, France [146]
- Zilong Zhao, Sophie Cerf, Robert Birke, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, Lydia Y. Chen. **Robust Anomaly Detection on Unreliable Data.** 49th IEEE/IFIP International Conference on Dependable Systems and Networks (*DSN 2019*), June 2019, Portland, Oregon, USA [142]. **(acceptance rate: 21.4%)**
- Zilong Zhao, Sophie Cerf, Bogdan Robu, Nicolas Marchand. **Feedback Control for On-line Training of Neural Networks.** 3rd IEEE Conference On Control Technology And Applications (*CCTA 2019*), Aug. 2019, Hong Kong, China [143].
- Amirmasoud Ghiassi, Taraneh Younesian, Zilong Zhao, Robert Birke Valerio Schiavoni, Lydia Y. Chen. **Robust (Deep) Learning Framework Against Dirty Labels and Beyond.** 2019 IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (*TPS-ISA*), Dec. 2019, Los Angeles, California, United States. [48].

1.2.2 Collaborations

Those works has been conducted thanks to fruitful collaborations:

- Pr. Ioan D. Landau (Gipsa-lab, Univ. Grenoble-Alpes) on system identification and control,
- Pr. Luc Dugard (Gipsa-lab, Univ. Grenoble-Alpes) on system identification and control,
- Dr. Lydia Y. Chen (TU Delft) on dirty label data learning and active learning,
- Dr. Robert Birke (ABB Research) on dirty label data learning,
- Pr. Sara Bouchenak (LIRIS lab, INSA-Lyon) on dirty label data learning,
- Dr. Sonia Ben Mokhtar (LIRIS lab, INSA-Lyon) on dirty label data learning,
- Dr. Rui Han (Beijing Institute of Technology) on dirty label data learning,
- Dr. Sophie Cerf (INRIA) on dirty label data learning and event-based control,

1.2.3 Technical contributions

The contributions of the above publications are not only on theoretical area, technical advances are also made.

- During the preparation for the economic method, we have developed a Ramsey–Cass–Koopmans model simulator[1] based on a matlab project[2]. All the code are realised in matlab, the interface is developed by Simulink.
- The learning rate algorithm proposed in [143] and [145] are published as open-source code in github[3]. The E/PD control and the event-based learning rate and event-based learning epoch controls are all packaged as learning rate scheduler in Keras1[26] library, which make users very easy to adopt to their own projects.

[1] video demo: https://www.youtube.com/watch?v=iXP0kQ9hig0

[2] project link: https://www.mathworks.com/company/newsletters/articles/simulating-the-ramsey-cass-koopmans-model-using-matlab-and-simulink.html

[3] project link: https://github.com/zhao-zilong/Event-Based-Control-Learning-Rate

- The framework RAD which is proposed to learn from unreliable data is developed in python with scikit-learn [105] and Keras libraries. The code of vanilla RAD is published on github[4]. As the paper for extension of RAD is still under review, we will publish the code later under this git[5].

- The optimal control framework developed for french macroeconomic model will also be published in the same git account as RAD. The framework is coded in Simulink. Since the paper is not published yet, the code remain confidential for now.

1.3 Thesis Outline

This thesis consists of five parts. The first part gives a general overview of the background, motivation and contributions of the works. From the second to the forth part, we elaborate the conducted studies in three topics: 1) System identification and optimal control on Macroeconomic data, 2) Control theory for on-line training of neural networks and 3) On-line learning from highly unreliable Data demonstrated on the use case of anomaly detection. The fifth part concludes on the whole contributions of this thesis, and gives possible directions for future works. Each part is divided in chapters, the content of each chapter is briefly described as follows.

Part 1 - Generalities.

Chapter 1 - Introduction. First chapter introduces the context of the all three research topics, their background, and an general picture of the main results achieved. Publications, collaborations and technical contributions through all the works in this thesis are also given in this chapter. The outline of this thesis and the read road-map for readers from different academic background are provided in the end.

Chapter 2 - Background and Motivation. This chapter elaborates the basic ideas of control theory, system identification and machine learning, which are the three research domains we explored in this thesis. It shows the objective of control system and how it works, and provides the comparisons between system identification and machine learning. Motivating cases are showed in this chapter, which leads to our following research in Part. II III and IV.

Chapter 3 - Objectives and Contributions. This chapter sets forth the objectives of the research topics in Part. II III and IV, which is mainly the responses to the motivating cases discussed in Chapter. 2, and explains the theoretical and technical contributions of the works from each topics.

Part 2 - System Identification and Optimal Control on Economic Data.

Chapter 4 - Background on System Identification and Control. This chapter first introduces two optimal growth models: 1) Solow-Swan model and 2) Ramsey-Cass-Koopmans model. These two models are not used in later context, but they are giving the ideas how do economists apply optimal control on economic problem. Then we introduce the background of system identification methods used in econometrics, such as Augmented Dickey-Fuller test for verifying the stationarity of time series, or Vector Autoregressive Exogenous (VARX) model to modelize the macroeconomic data. The background of system identification in automatic is also introduced, such as state space representation. And one optimal control solution: LInear-Quadratic Regulator (LQR) is also presented.

Chapter 5 - Modelling and Dynamic Analysis of the Domestic Demand Influence on the Economic. This chapter presents an economic study on China macroeconomic data. Comparing

[4]project link: https://github.com/zhao-zilong/RAD

[5]project link: https://github.com/zhao-zilong

to previous studies, we include the time series from recent years, and add more variables into the models. Econometrics methods are implemented to identify the economic model, Granger causality tests are conducted between the economic growth, household final consumption, inward foreign direct investment and export in China. The experiments are realised on Eviews.

Chapter 6 - Modelling and Optimal Control of MIMO System - French Macroeconomic Model Case. After identifying the macroeconomic model of China with econometrics methods, this chapter uses France macroeconomic data, and represent the study in an automatic way. We regard the model as a Multiple-Input and Multiple-Output (MIMO) system, and represent the model in state-space form, which can help to design the controller. The optimal controller is designed via Linear-Quadratic Regulator (LQR). The experiments are conducted with different parameters of LQR, and perturbations on outputs and constraints on inputs. The system identification experiments are realised on Gretl and Matlab. The control system is illustrated on Simulink.

Chapter 7 - Conclusion on Economic Model Control. This chapter summarizes Part. II, concludes their results and gives perspectives on future works.

Part 3- Control Theory For On-line Training of Neural Networks.

Chapter 8 - Neural Network and Learning Rate: Background and Related Works. This chapter introduces the Convolutional Neural Network structures, the performance metrics to evaluate the neural network and Continual (On-line) learning scenario for Part. III. It details different type of time-based learning rate algorithms, and elaborates the state of the art adaptive-gradient methods.

Chapter 9 - Exponential / Proportional-Derivative Control of Learning Rate. This chapter introduces the performance-based learning rate algorithm: Exponential (E) / Proportional-Derivative (PD) control. This study is mainly compared with time-based learning rate methods. It first shows some motivation cases from fixed learning rate scenario, and gradually presents the P, PD and E/PD control. The experiments are on two image datasets: CIFAR-10 and Fashion-MNIST. All the experiments are realised based on Keras (tensorflow backend) library, with the help of GPU from google cloud compute engine.

Chapter 10 - Event-Based Control for Continual Training of Neural Networks. Based on E/PD control, this chapter introduces two event-based control to improve E/PD in continual learning scenario: i) Event-Based Learning Rate and ii) Event-Based Learning Epochs. The experiments are conducted on CIFAR-10 and CIFAR-100, and the results are compared with four state of the art adaptive gradient method: 1) Adam 2) Nadam 3) AMSGrad and 4) AdaBound.

Chapter 11 - Conclusion on Learning Rate Control. This chapter concludes the Part. 9, gives the possible direction to extend this work.

Part 4- On-line Learning from Highly Unreliable Data: Anomaly Detection.

Chapter 12 - Noisy Data Learning and Anomaly Detection: Background and Related Works. This chapter introduces the problems of noisy data learning and anomaly detection. It details the features of the anomaly detection datasets that we will deal with. The different continual learning setting for different datasets is presented, and the previous state of the art algorithms that we will compare with are illustrated with formula and schema.

Chapter 13 - Robust Anomaly Detection on Unreliable Data. Robust Anomaly Detector (RAD) is a generic framework to deal with the dirty label data learning problem, it is called anomaly detector is because it is first applied to deal with anomaly detection problem. This chapter presents the RAD with our designed ensemble prediction method. The evaluations are based on two dataset: 1) IoT thermostat device attack detection and 2) Google cluster task failure prediction. In the end of evaluation, the limitation of RAD is discussed. The experiments are implemented with Scikit-Learn library.

Chapter 14 - Extension of RAD framework for On-line Anomaly Detection for Nosiy Data. Due to the limitation of RAD presented in Chapter. 13, we propose several extensions of RAD to improve these deficiencies. RAD Voting and RAD Active Learning are two enhanced version of RAD based on the additional features of conflicting opinions of classifiers, repetitively cleaning, and oracle knowledge. To show the broad applicability of RAD (and its extensions) framework, we extend the evaluation with a new dataset FaceScrub, which is to recognize 100 celebrity faces. RAD Slim is an adapted version of RAD Active Learning for image dataset. The experiments are implemented with Keras and Pytorch libraries in this chapter.

Chapter 15 - Conclusion on Noisy Data Learning. This chapter concludes the studies for the topic of noisy data learning, and provides insights for future works.

Part 5- Conclusions and Perspectives.

The ending part of this thesis is dedicated to conclude the whole contributions from Part. II III and IV. By analysing the results, we point out some theoretical improvement directions for certain studies, and possible experimental extensions for current work.

1.4 Read Roadmap

Since this thesis touches various research domains, this section guides the readers from different backgrounds or with different interests.

For the reader with interest in Economics. The whole Part. II is dedicated to discuss the economic problem. If one has economic background, there is no difficulty to understand the Chapter. 5 which studies the economic transition of China. Chapter. 6 also focuses to analyze economic data, it incorporates french macroeconomic model into the control system to conduct optimal control. Therefore for the reader without control knowledge, one has to at least read the Sec. 2.1 and Sec. 4.2 to understand the basic of control system and optimal control.

For the reader with interest in applications of Control Theory. Chapter. 6 applies optimal control (designed via LQR) on economic model, simulation with 1) varying parameter of LQR, 2) constraints on inputs and 3) perturbations on outputs are implemented. Part. III explores to integrate control theory into learning rate algorithm of machine learning. Chapter. 9 proposes a performance-based feedback control E/PD, which updates the learning rate based on the evolution of historical loss value. Chapter. 10 two event-based controls: i) event-based learning rate and ii) event-based learning epochs. First is used to improve E/PD, second one is used to reduce the inefficient training epochs. Even though these controls are used to improve machine learning algorithms, all the studies in Part. III have been published in conference and journal of control community, therefore their presentations are in favor of the reader with control background. E/PD and event-based learning rate are also involved in the experiments in Chapter. 14 for RAD Slim algorithm, but they are only implemented to accelerate the training process, not theoretical innovations of these algorithms in that part.

For the reader with interest in Machine Learning. Part. III is dedicated to improve the machine learning algorithm performance under continual learning scenario. Since the achievement is realised with control theory. It is better for the reader without control background to read Sec. 2.1 first, to have an impression of the control system and feedback control. Part. IV is quite independent, readers who are interested with dirty label data training problem, can skip other chapters.

Chapter 2

Background and Motivation

This chapter gives a general overview of Control Theory, System Identification and Machine Learning. First, basic concepts are explained as an attempt to depict the global picture of the field, as well as to give a rapid introduction for readers unfamiliar with the terminology. Then, motivation cases associated with control theory, system identification and machine learning are discussed.

2.1 Basics of Control Theory

Control theory is a theory that studies how to adjust the characteristics of dynamic systems. The process under study should evolve over time and be causal (only past and present events impact the future). Fig. 2.1 presents a system (physical, biological, economic, etc., referred usually as *plant*) in control community. In this schema, at least one of the outputs should be able to measure, and at least one of the inputs can influence the outputs through the configuration of the plant. The plant can be characterised using the mathematical model which contains a set of parameters that describes its behaviors. For time-variant system, the parameters of the model can change over time, and for time-invariant system, we assume these parameters are fixed during the whole time. Depending on number of inputs and outputs, a system can be either SISO (Single-Input, Single-Output), MIMO (Multi-Inputs, Multi-Outputs) or a combination of the two.

To sum-up, a system eligible for control should be dynamic, causal and be configurable by at least one input signal, and observable with at least one output signal [53].

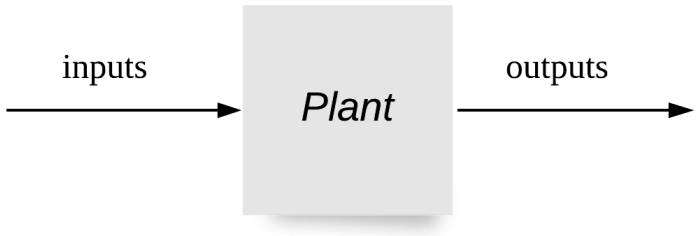


Figure 2.1 – Representation of system in control

Given such a system, three complementary goals can be achieved when using control theory [23]:

- **Stability.** There are three type of stability in control theory: 1) The stability of a general dynamical system with no input can be described with Lyapunov stability criteria. That is, any trajectory with initial conditions around $x(0)$ can be maintained around $x(0)$; 2) A linear system

is called bounded-input bounded-output (BIBO) stable if its output will stay bounded for any bounded input; 3) Stability for nonlinear systems that take an input is input-to-state stability (ISS), which combines Lyapunov stability and a notion similar to BIBO stability. In this thesis, we mainly focus on the second case BIBO stable. Control can be used to stabilize an unstable system, but most of all it should ensure to keep stable an originally stable one.

- **Tracking.** Once the system is ensured to be stable, one may consider the behavior of the plant. One main objective of control system is to let the output of plant follow our desired trajectory. There are several indexes to evaluate a controller. Assume the reference is a constant signal, one index is the *precision*, which means how close the output and reference. Another index is the *responsetime* which represents the time from beginning to the moment output is stable. *Overshoot* is also an important index, it measures the maximum value above reference.

- **Disturbance Rejection.** As no system evolves in a perfectly mastered environment, a control strategy should be able to deal with exogenous influences. Whether these disturbances can be measured or not, modeled or not; control theory provides ways to reject them, i.e. minimize their impact on the plant outputs. This aspect of control is called a perturbation rejection problem.

2.2 Feedback Control

A control system can be regarded as a system with four functions: (i) Measurement, (ii) Comparison, (iii) Calculation and (iv) Correction. As open loop control strategies are not used in this thesis, we focus on closed loop (also referred to as feedback) control system in this section. A high-level representation of a feedback control system can be illustrated in Fig. 2.2. It introduces three extra elements comparing to system in Fig. 2.1: (1) a reference signal; (2) a controller (containing a control algorithm); (3) a feedback from output to input. The output (i.e. measurement) signal is compared to its reference, to which the plant should tend. The difference between the reference and the output is used as input (referred to as *error* in Fig. 2.2) for controller to generate a control signal. The idea behind this design is easy to understand, if the error is big, we need to largely adjust our control signal to stimulate the plant, so that the output quickly approaches to the reference. If the error is small, we should carefully adjust the control signal. The objective of this control system is to minimize the difference between the reference and the output.

The control engineer usually uses the imperfect knowledge of the plant (represented as a state space or transfer function) to compute the controller algorithm which will make the closed loop plant comply with the functionality and technical specification.

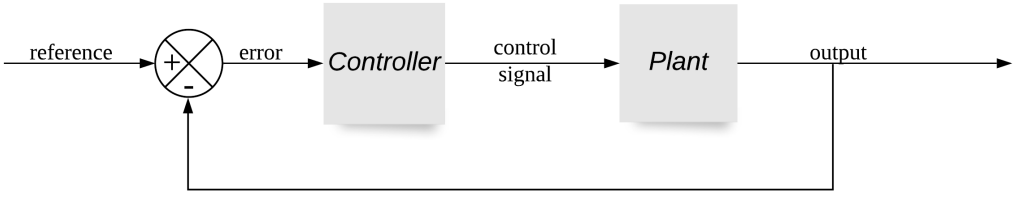


Figure 2.2 – High-level representation of a feedback control system

2.3 Basics of System Identification & Machine Learning.

The field of system identification uses statistical methods to build mathematical models of dynamical systems from measured data [124]. The term *system identification* is mainly used in the field of control engineering.

Machine learning (ML) is the study of computer algorithms that improve automatically through experience [97]. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so [74, 17]. The term *machine learning* is mainly used in computer science field.

We can see that both notions build mathematical models, they all try to use available historical data to approximate the function (relation) between input and output. The main difference between these two notions reside in the usage. While some of system identification algorithms care more if the estimated model fits well the training data (e.g. model-driven estimation), in order to build an efficient algorithm. Machine learning cares more if the estimated model can predict well in the new data. That is why machine learning is also referred to as predictive analytics.

The procedure of performing the system identification and machine learning are almost same. The high-level steps to perform system identification and machine learning on a dataset can be summarized as showed in Fig. 1.1: 1) Data Preparation, 2) Parameter Estimation / Learning or Training, and 3) System Validation / Testing.

- **Data Preparation.** When we deal with a dataset, we almost never use the original data to estimate the model. For instance, the stationarity is important if we do the regression on time series. If the time series are non-stationary, the regression gives spurious results (see Part. II). The cleanness of training data is vital for training machine learning model, if the collected data are polluted, the true output y_i of input x_i is not reliable, the trained model can be corrupted (see Part. IV). To solve above problems, we need to deal with the data before feeding data to algorithm.

- **Parameter Estimation / Learning or Training.** The parameter estimation in automatic control is similar to the term of Learning or Training for machine learning. It is the process to approximate the functions between input and output by mathematical methods. For linear regression, Ordinary Least Square (OLS) and Maximum Likelihood Estimationis (MLE) are widely used. And for more complex task such as face recognition, neural network is the common choice, then convolution and gradient (Sec. 8.1) are necessary.

- **System Validation / Testing .** The main difference between System Identification in automatic and Machine Learning in computer science is in this step. For system identification, to validate the estimated model, we use the data which is used to train the model to test the model. The aim is to check if the model has perfectly fitted on the data. If the residuals from the validation are white noise, we believe the estimated model is good. In machine learning, before training the model, we need to split the whole dataset into one training dataset and one testing dataset. The ratio between testing and training dataset is often 1:2 or 1:3. Only training dataset is used to train the model, and only testing dataset is used to test the model.

Above three points are only high-level skeleton of the pipeline, there are also other details when we implement. For example, if we perform the auto-regression on time series, before estimating parameters of model in parameter estimation step, we need to estimate the order of model first, this part is discussed in Sec. 5.4 and Sec. 6.2.2. Estimation of model's order can help to build a model with least parameters but still valid. This step as well as model order reduction usually does not appear in machine learning, one reason can be the interpretability of the model, from the highly interpretable

lasso regression to impenetrable neural networks, but they generally sacrifice interpretability for predictive power. Since many of the model's structures are unexplainable, it will be difficult to justify any reduction of the model.

2.4 Motivation Cases

In this section, we provides some real world cases which motivate us to perform our studies in following three parts.

2.4.1 Control Theory on Macroeconomic Model

First, let us check the quarterly GDP growth rate from 2005Q1 to 2018Q1 in Fig. 2.3. EA denotes to Euro Area. EU and US denote to European Union and United State. One can observe a big drop around 2008Q4 and 2009Q1, which is the period of the 2008 financial crisis. If we see the macroeconomic system as the plant we introduced in Sec. 2.1, then GDP can be modelized as one of the outputs. In macroeconomic, we know there are several economic variables we can control, for instance the interest rate in central bank, the oil price, the public expenditure from government, etc. We can see these variables as the inputs of the plant, then we can easily modelize the economic crisis problem as control problem. We can also add perturbations on outputs, constraints on inputs. By simulating the system, we can emulate different kind of crisis on different outputs. Observing the reaction of inputs during the system recovery process provides very good political implications for economists to study. This study is explored in Part. II.

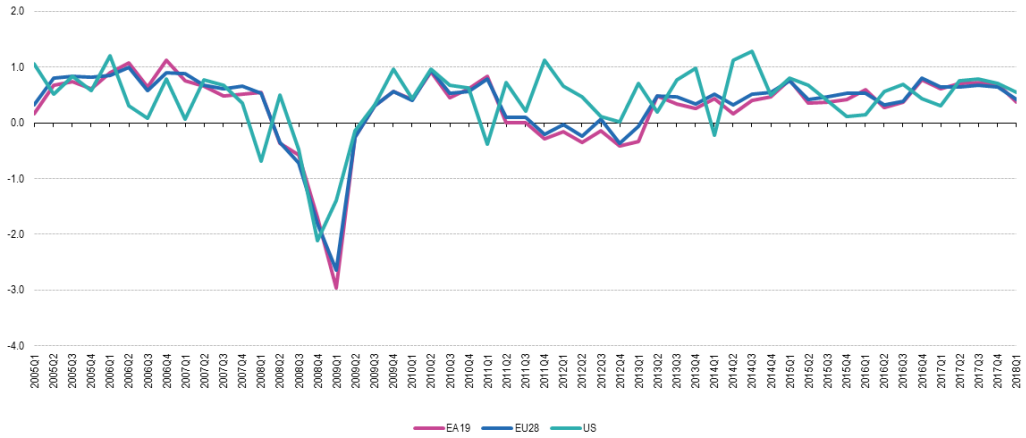


Figure 2.3 – Quarterly GDP Growth Rate from 2005Q1 to 2018Q1 (Source: Eurostat)

2.4.2 On-line Training of Neural Network

On-line training (or continual learning) is widely used in many scenarios, when there is not much available data at beginning. For example shopping site or Netflix, when a new user comes in, there is no historical information of the user, then the recommended products or films are not well fitted to user's taste. As the user generates more data, the recommendations are more accurate.

There are two scenarios of on-line learning setting: 1) the size of data is relatively small, every time a user generates new data, we can retrain the model with all accumulated historical data. 2) if the size of data is large, when there is newly generated data available, we can only use the new data to train the model instead of re-learning with all the accumulated data.

Apart from the data size, time interval of data generation is another factor to consider, it challenges the convergence speed of algorithm. Learning rate and Gradient are two key parts which influence the convergence speed of machine learning algorithms. After studying the previous strategies, they can be summarized as: (1) time-based learning rate, (2) adaptive gradient. The problem of time-based method is that its learning rate is prefixed before training process, which makes it impossible to react, even though sometimes it can accelerate the learning speed as we are far from optimum, or slow down the learning speed as we are close to the optimum in case we do not skip it. For adaptive gradient methods, there are studies show that adaptive gradient methods do not generalize as well as stochastic descent gradient (SGD). These methods tend to perform well in the initial portion of training but are outperformed by SGD at later stages of training.

Above studies drive us to think: If we can develop a performance-based learning rate algorithm using SGD? So that we can overcome the shortcomings in time-based learning rate and adaptive gradient methods. The performance-based controller can be designed to solve this problem.

Actually the following thoughts come out after we finished the performance-based controller. For a typical machine learning setting, we need to fix a number of training epochs before we launch the training process. Since we developed the performance-based controller, the converging speed of the model is largely accelerated, and as the algorithm is already converged after several epochs in the beginning, the following training epochs becomes less useful. This phenomenon drives us to propose an event-based control to decide when we stop the learning, that will shapely reduce the training epochs while maintaining a good model. Above studies are performed in Part. III.

2.4.3 Dirty Label Data Learning

When we try to build a specific image classification model, the first step is to prepare a labelled dataset. There are several ways to get high-quality labelled data, for example, explore pre-labelled public datasets or leveraging the crowd-sourcing service. But in most case, we need to harvest our own training data and labels from free sources. Here we illustrate the struggles one can experience when construct their own dataset and explain why we need to solve this problem.

Imagine one wants to build an image classification model to distinguish if an image is airplane, and we assume that there is no free available pre-labelled public datasets. Then the fastest way we can think about is to use google. If we search keyword "airplane" in Google image search, the first pages will show the images as in Fig. 2.4. There are different type of airplanes, and this is what we want to collect.

(a) (b)

Figure 2.4 – Correct Airplane Images.

In reality, if we want to quickly gather as many as possible the images, one useful tool is the web crawler, it will help us to thousands of images by running a few lines of code instead of downloading them one by one by clicking. However, if we check the later pages of google results, other relevant images pop up. For instance, the images showed in Fig. 2.5 are the cabinet of airplane and a cartoon image of an airplane. To solve this problem, one can manually check all the images. But when the dataset is huge, that can be really time consuming or even impossible. That motivates us to propose a proper algorithm to address this problem to mitigate the influence of dirty labelled data on final classification model.

(a) (b)

Figure 2.5 – Undesired Airplane Images.

Chapter 3

Objectives and Contributions

In this context of association of control theory, system identification and machine learning, this thesis explores the problems which can be grouped into three subjects which will be presented in Part. II, III and IV. The main objective of this thesis are detailed as follows:

3.1 Objectives

Part. II focuses on the problem of system identification and optimal control of the macroeconomic model, the study is based on the economic data from China and France. In this study, we will show the methods used in Economics and Automatic to perform the system identification. After that, we will show two use cases on how economists achieve optimal growth for economic models (Solow-Swan model and Ramsey-Koopmans-Cass model). And then we will introduce our designed optimal control model, one advantage of our algorithm is that it can take into account of constraints on inputs and the state of the system as well as perturbations on outputs .

Part. III incorporates control theory into learning rate algorithms, aiming to accelerate the machine learning process on continual learning scenario. The state of the art algorithms can roughly be divided into two categories: i) time-based learning rate, and ii) adaptive gradient learning rate. The algorithms from these two groups all have their own drawbacks. Time-based learning rate can not adjust its learning rate regards to the training data and training phase, which leads to a slow converge speed of the training process. Adaptive gradient methods performs better in converge speed comparing to time-based learning rate, but its final accuracy tends to be worse than the method based on the stochastic gradient descent (SGD). Therefore, we propose a performance-based learning rate algorithms, which not only converge as fast as (or even faster) the adaptive gradient methods, but also ensure no compromise on final accuracy as it is based on SGD. Besides the performance-based learning rate, we also propose an event-based control algorithm which aims to automatically decide when the training process stops and forwards to next training batch in continual learning scenario. To massively cut off inefficient training epochs but still accelerate the learning process.

Part. IV sheds light on dirty label data problem, which means that, as the training data quality is difficult to be guaranteed, we design the algorithms to alleviate the influences of dirty label data on the model. The algorithms are compared with several state of the art, in respect of final accuracy, final loss value, converging speed and stability of the performance. The experiments are implemented on three datasets: 1) IoT device attack detection, 2) Cluster task failure prediction and 3) face recognition, in order to show the board applicability on different type of data.

3.2 Contributions of the Thesis

In this section, we will summarize the contributions of this thesis from each chapter that presented in Part. II, III and IV.

3.2.1 Dynamic Analysis of China Macroeconomic Model

This study uses the classical economic methods to perform system identification, comparing to previous researches, we extends the time series with recent year data, include new variable: Household Consumption into the macroeconomic model. We also implement tests to show the Granger causalities between variables. Many studies on the early stage of Open Door Policy have shown that there are two-way Granger causalities between Export, FDI (Foreign Direct Investment) and GDP (Gross Domestic Production), but our study shows that with recent year data, the growth of FDI and the growth of GDP have no two-way Granger causalities any more, but the growth of Consumption does.

3.2.2 Optimal Control on France Macroeconomic Model

This study is based on France's macroeconomic data, we perform the system identification as we typically do in Automatic. After estimating the model, the optimal control solution: LQR is designed for our problem which is to maintain a constant GDP increasing ratio. And once the system is stable, we introduce the perturbations on all outputs (i.e. IMP. EXP, GDP). Introducing shock on variables in VAR has been well studied in economic area by impulse response function (IRF). IRF traces the effects of an innovation shock to one variable on the response of all variables in the system. Comparing to IRF, our approach not only observes the changes after the shock, but also intervenes the process of recovery. Our objective is to use all the available resources to help the model regain the stability, return to the level before the shock. We also impose the constraints on inputs, all these factors together help us to well emulate the real world situation (e.g. 2008 financial crisis and Covid-19 pandemic). The whole control system is realised with Simulink (a handy simulating environment), we think this tool can help economist to better estimate the recover trajectory of the economy.

3.2.3 Exponential / Proportional-Derivative Control of Learning Rate

When performing image classification tasks with neural networks, often comes the issue of on-line training, from sequential batches of data. The interval between data batch can be short and the data distribution from one batch to another can vary a lot. All these problems lead us to design a better algorithm, which can make the training process converge faster so that we can reduce training time, and the algorithm should also be stable when we change the data batch from one to another. In addition to that, our algorithm should not sacrifice its final accuracy and final loss for maintaining stable and faster converging speed. That is why we propose a performance-based learning rate algorithm - Exponential (E) / Proportional-Derivative (PD) control, it converges faster than the compared state of the art algorithm, it's much stable than others near its end of training, and its final accuracy and loss is lower than the comparisons.

3.2.4 Event-Based Control for Continual Training of Neural Networks

This work is an extension work based on E/PD control we mentioned in last section. We first propose an enhanced version of E/PD - Event-Based E/PD. It can prevent the learning rate to decrease when loss value continuously decreases. The result shows that this small change indeed improves the model in all the index we mentioned in last section.

We also propose a second event-based control based on E/PD, this control is based on our observation that when we implement E/PD on continual learning scenario, the significant improvement in the learning only occurs at the beginning when loading a new batch, the accuracy and loss value evolve slowly afterwards. Therefore, we implement an event-based control to inspect the record of the loss value. If the loss record has the tendency to increase, showing little learning efficiency, we will drop the rest learning epochs for current data batch. In the experiment with dataset CIFAR10, it could save up to 67% training epochs.

3.2.5 Robust Anomaly Detection (RAD) on Unreliable Data

As dataset gets bigger and bigger, the data quality becomes much difficult to control than before. Dirty label data learning is an important topic for machine learning in recent years, And in this work, we use anomaly detection task as an example to address this problem. The learning scenario is setting on continual learning, we propose a two-layer framework for Robust Anomaly Detection - RAD. The first layer (called label quality model) mainly aims at differentiating the label quality, i.e., noisy v.s. true labels, for each batch of new data and only "clean" data points are fed in the second layer. The major job of second layer (called anomaly classifier) is to predict the coming-out event, that can be in multiple classes of (non)anomalies, depending on the specific anomaly use case. Both the prediction from label model and anomaly classifier contribute to the final decision of anomaly detection, using our designed ensemble prediction technique. The results show that RAD outperforms the case when there is no data selection for training. The experiment with varying noise level on labels shows that RAD can resist high noise level in the data label.

3.2.6 Extension of RAD framework for On-line Anomaly Detection for Noisy Data

This work is an extension of last section. The problem studied and experimental setting remain the same for same the datasets. we extend RAD with additional features of conflicting opinions of classifiers and repetitively cleaning as RAD Voting, and with oracle knowledge as RAD Active Learning. To deal with complex data such as image, we also propose another version of RAD Active Learning, namely RAD Slim, which instead of use two-layer framework, RAD Slim is reduced to one layer, and delegate the role of second layer to oracle. To test RAD Slim, we introduce a new face recognition dataset: FaceScrub with 100 celebrity faces.

RAD Voting solved the problem that two models converge over time in RAD, because we let second layer to join the data selection process, and that injects more diversities of our chosen training data. And with the help of oracle, our results show that with our framework, it can reach almost the same accuracy as there is no noise (under 30% noise level). By comparison with the designed experiment (i.e. PreSelect Oracle), we clearly show that if we randomly choose the data to ask oracle, the performance is much worse than RAD Active Learning and RAD Slim.

Part II

System Identification and Optimal Control on Economic Data: Applications on China and France

There are many concepts both used in economic and automatic domain to perform system identification, such as: using VARX (Vector Autoregressive Exogenous) model to represent system, using the Akaike information criterion (AIC) for estimating model order, and using the Ordinary Least Square (OLS) for parameter estimation. In this part, we perform the system identification in two studies, one uses only the methods which commonly used in economic area, another will introduce some methods merged from automatic area. Most of their steps are the same, but there are still some differences (e.g. different way of identifying model order). In second study, we also apply the optimal control via linear–quadratic regulator (LQR) on the estimated model, introducing constraints on inputs, perturbation on outputs, to show that our designed controller can really adopt to the real world problem.

In Chapter. 4, we introduce some basic concepts in details which we will use in the later analysis, such as the Augmented Dickey–Fuller(ADF) Test which is used to determine if a time series is stationary, or Linear–Quadratic Regulator (LQR) which is the technique we will adapt to design our controller for optimal control. In Chapter. 5, we perform an analysis of the trend of China's economic transition with the system identification methods that are widely used in economic domain. This study is performed with China's macroeconomic data. Comparing to previous studies, we include data from recent years, also take new variables into consideration.

As the research progresses, we want to introduce more variables and richer data (e.g. data in quarterly not yearly), but some Chinese data (e.g. total investment or public expenditure) are very limited, therefore Chapter. 6 implements a study with the macroeconomic data of France, we first perform the system identification, then transform the model to state space representation and carry out the optimal control via LQR. Constraints on inputs and perturbations on outputs are considered during the control in this study, to adopt to real world situation. Chapter. 7 concludes these two studies, and also indicates the possible future extensions.

Chapter 4

Background on System Identification and Optimal Control in Economics and Automatic

In this chapter, we first introduce two optimal growth models from macroeconomic. These two models are not used in the later analysis, but they clearly show how the economists use dynamic optimization to solve macroeconomic problem. Then introduce the methods used during the procedure of system identification in economic and automatic domain. This procedure consists of data preparation and system estimation. In addition to that, the Granger Causality (which is a statistical concept of causality that is based on prediction) is studied. That will help us to better understand the models. We will also present the technique to formalize the system on state space representation. And the algorithms used in automatic to perform optimal control on system.

4.1 Economic Area

4.1.1 Optimal Growth Model

Before introducing two optimal growth models, all the symbols used for the two models are listed in Tab. 4.1.

4.1.1.1 Solow-Swan Model

The basic version of Solow-Swan model [125, 127] considers a closed economy producing one single good using both *labour* and *capital*. it assumes there is perpetual full employment of labour, all the savings are used to invest. It takes the technology progress as given and the constant saving rate as exogenous. Assume all the firms have access to the same production function, the aggregate production function (Cobb Douglas production function) for the unique final good is:

$$Y(t) = A(t)K(t)^{\alpha}L(t)^{1-\alpha} \tag{4.1}$$

It can be written as output per effective unit of labour $y(t)$, which is a measure for wealth creation:

$$y(t) = \frac{Y(t)}{A(t)L(t)} = k(t)^{\alpha} \tag{4.2}$$

where $k(t) = K(t)/L(t)$, it is the capital intensity, the capital stock per unit of effective labour. The key equation of the Solow-Swan model is:

$$\dot{k} = sk(t)^{\alpha} - (n+g+\delta)k(t) \tag{4.3}$$

Table 4.1 – Symbol description

Symbol	Description
n	labor growth rate
s	saving rate
g	technology growth rate
δ	depreciation rate
r	interest rate
ρ	discount rate (Appendix .2)
H	number of household
$Y(t)$	total production in Solow-Swan model
$F(t)$	total production in RCK model
$L(t) = L(0) \cdot e^{nt}$	total labour
$K(t)$	total capital
$A(t) = A(0) \cdot e^{gt}$	total factor productivity
$C(t)$	consumption per worker
$u(C(t))$	instantaneous utility function
W	wage per worker
B	total family wealth
U	household lifetime utility function

the term $sk(t)^{\alpha}$ corresponds to the same time the amounts of save and investment, it is the actual investment per unit of effective labour: the fraction s of the output per unit of effective labour $y(t)$ that is saved and invested. The second term, $(n+g+\delta)k(t)$ is the "break-even investment": the amount of investment that must be invested to prevent $k(t)$ from falling. In the steady state where $k(t)$ = $k(t)^*$, one can easily to notice that $k(t)^*$ can only reside on where $\dot{k}(t) = 0$, which means investment (i.e. saving) equals depreciation, this is also called equilibrium point. Fig. 4.1 clearly shows this relation, it represents $y(t) = f(k(t)) = k(t)^{\alpha}$, the corresponding k^* where makes the two curves cross is where $s(k(t)^*)^{\alpha} = (n+g+\delta)k(t)^*$. When $k(t) = k_1(t)$, $s(k_1(t))^{\alpha} > (n+g+\delta)k_1(t)$, then $\dot{k}(t) > 0$, k^* will move towards to $k(t)^*$. When $k(t) = k_2(t)$, $s(k_2(t))^{\alpha} < (n+g+\delta)k_2(t)$, then $\dot{k}(t) < 0$, k will also move towards to $k(t)^*$. Therefore in Solow-Swan model, once all the parameters are fixed, the k is constant. From Fig. 4.2, we can see that once saving rate changes, the equilibrium point changes.

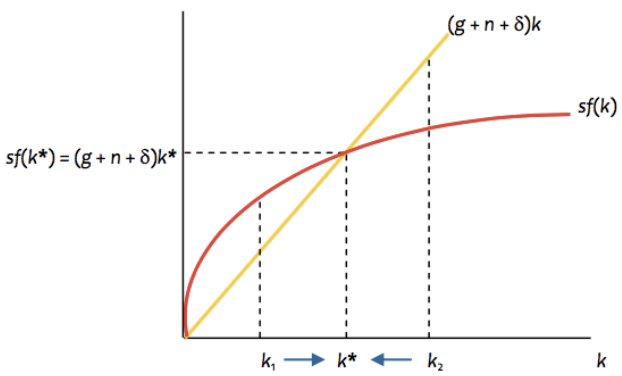


Figure 4.1 – Solow-Swan model equilibrium.

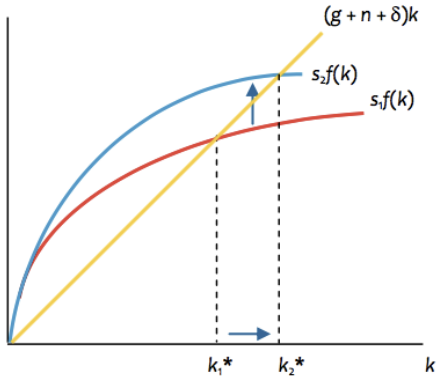


Figure 4.2 – Solow-Swan model changing equilibrium with different saving rate.

4.1.1.2 Ramsey Cass Koopmans (RCK) Model

Solow-Swan model assumes that aggregate consumption is simply a linear function of aggregate output, so that the fraction of output devoted to investment (=saving in a closed economy) is also constant. This is a strong assumption, and in reality, the household consumption is much more complex than simply a fixed proportion of income. Our consumption is definitely influenced by our anticipation to future incomes. Therefore, RCK adds one layer to the Solow-Swan model: it allows households to make optimal consumption/saving decisions at the microeconomic level, given the environment they are facing. As a result, the evolution of the capital stock will reflect the interactions between utility-maximizing households (supplying savings) and profit-maximizing firms (demanding investment). In this model, the saving rate is not constant anymore.

- **Assumption**: There are some background assumptions for RCK model. 1) The technology is the same as in Solow-Swan model, it grows exogenously at rate $\dot{A}(t)/A(t) = g$. 2) There is no depreciation: $\delta = 0$. 3) Firms hire workers at real wage $W(t)$ at time t and rent capital at rate r(t) to maximize profits. 4) $K(0)$, $A(0)$ and $L(0)$ all given and all > 0. 5) Each household grows at rate n. 6) Each member of household provides 1 unit of labour. 7) Each household receives income from the following sources: labour income (wages of the household members) and capital income (from renting out capital to the firms) [54].

- **Firm**: Here in RCK model, we consider to use the same production model as Eq. (4.1). Therefore:

$$F(t) = A(t)K(t)^{\alpha}L(t)^{1-\alpha} \tag{4.4}$$

  Since firms pay a price $r(t)$ for renting a unit of capital and there is no depreciation, they will equate the marginal product of capital: r(t) = $\partial F(K,AL)/\partial K$. Since $\partial F(K,AL)/\partial K = f'(k)$, where $f(k) \equiv F(k,1)$ and $k = K/(AL)$, firms will rent capital up to the point where:

$$f'(k(t)) = r(t) \tag{4.5}$$

- **Labour**: Firms will hire workers up to the point where the real wage W(t) equals the marginal product of labor $\partial F(K,AL)/\partial L$. To express the marginal product of labor in terms of $f(\cdot)$,

observe that:

$$\begin{aligned}\frac{\partial F(K,AL)}{\partial L} &= \frac{\partial}{\partial L} ALf(K/(AL)) \\ &= Af(k) - ALf'(k)(\frac{K}{AL^2}) \\ &= A[f(k) - kf'(k)] \end{aligned} \tag{4.6}$$

we define the wage per effective unit of labour:

$$w(t) = W(t)/A(t) \tag{4.7}$$

then $w(t)$ satisfies:

$$\begin{aligned} w(t) &= f(k) - kf'(k) \\ f(k) &= kf'(k) + w \end{aligned} \tag{4.8}$$

- **Household**: Each household has to decide how much to consume and how much to save to maximize the lifetime utility:

$$U = \int_{t=0}^{\infty} e^{-\rho t} u(C(t)) \frac{L(t)}{H} dx \tag{4.9}$$

subject to the following dynamic budget constraint:

$$\dot{B}(t) = r(t)B(t) + W(t)L(t) - C(t)L(t) \tag{4.10}$$

which means that the family wealth increase comes from the interest of family wealth plus total wages minus total consumption. Of course, in equilibrium, we will need to have B(t) = K(t), since in a closed economy, the stock of savings by households must be equal to the stock of physical capital. The household takes B(0) = K(0) > 0 as given.

For unility function, RCK model chooses a *constant relative risk aversion* [110], more details of risk aversion is given at Appendix .3 which:

$$u(C(t)) = \frac{C(t)^{1-\theta}}{1-\theta}; \theta > 0; \rho - n - (1-\theta)g > 0 \tag{4.11}$$

where θ is the *coefficient of relative risk aversion*, defined as $-Cu''(C)/u'(C)$. $1/\theta$ is the *inter-temporal elasticity of substitution* (inter-temporal elasticity of substitution is explained in Appendix .1). θ governs the curvature of utility function.

Since we have considered technology progress, we call AL is the *effective supply of labour*. Then we have:

$$c(t) = C(t)/A(t) \tag{4.12}$$

where c(t) denotes the consumption per unit of effective labour. Same calculations for B to have the *unit of effective labour*:

$$b(t) = B(t)/(A(t)L(t)) \tag{4.13}$$

Now, instantaneous utility function u(C(t)) can be written as:

$$u(C(t)) = \frac{(A(t)c(t))^{1-\theta}}{1-\theta} \tag{4.14}$$

$$= A(0)^{1-\theta} e^{(1-\theta)gt} u(c(t)) \tag{4.15}$$

and U can be written as:

$$\begin{aligned} U &= \int_{t=0}^{\infty} e^{-\rho t} u(C(t)) \frac{L(t)}{H} dx \\ &= \frac{A(0)^{1-\theta} L(0)}{H} \int_{t=0}^{\infty} e^{-(\rho - n - (1-\theta)g)t} u(c(t)) dx \\ &\propto \int_{t=0}^{\infty} e^{-\beta t} u(c(t)) dx \end{aligned} \tag{4.16}$$

where $\beta = \rho - n - (1-\theta)g > 0$ by assumption, which is to make sure the U do not diverge. From Eq. 4.13, we could calculate the relation between $\dot{B(t)}$ and $\dot{b(t)}$ (Inference process is provided in Appendix .4):

$$\dot{B}(t) = \dot{b}(t)AL + b(g+n)AL \tag{4.17}$$

From Eq.(4.10), (4.12), (4.7), (4.13) and (4.17). Eq.(4.10) can be rewritten as :

$$\dot{b}(t) = (r(t) - g - n)b(t) + w(t) - c(t) \tag{4.18}$$

In this problem, $c(t)$ is the *control* variable, while $b(t)$ is the *state* variable.

Now we will talk about the maximum principle to solve the problem defined in Eq. (4.16) [121, 54]. Let $c^*(t)$ be a consumption sequence that solves this problem. Then there exists a co-state variable $\lambda(t) >= 0$ such that the Hamiltonian:

$$\mathscr{H}(c(t), b(t), \lambda(t)) = u(c(t)) + \lambda(t)[(r(t) - g - n)b(t) + w(t) - c(t)] \tag{4.19}$$

is maximized at $c^*(t)$ given $\lambda(t)$ and $b(t)$:

$$\frac{\partial \mathscr{H}}{\partial c}(c^*(t), b(t), \lambda(t)) = 0 \tag{4.20}$$

at all times. Furthermore, the co-state variable satisfies the following differential equation:

$$\dot{\lambda}(t) = \beta\lambda(t) - \frac{\partial \mathscr{H}}{\partial c}(c^*(t), b(t), \lambda(t)) \tag{4.21}$$

Finally, the co-state variable $\lambda(t)$ satisfies the Transversality Condition (TC):

$$\lim_{t \to \infty} b(t)\lambda(t)e^{-\beta t} \leq 0 \tag{4.22}$$

From Eq. 4.20, we have equation:

$$u'(c(t)) = c(t)^{-\lambda} = \lambda(t) \tag{4.23}$$

From Eq. 4.21, we have equation:

$$\dot{\lambda}(t) = \beta\lambda(t) - \lambda(t)(r(t) - g - n) \tag{4.24}$$

with condition (4.11), (4.23) and (4.24). We could get following equation (Inference process is provided in Appendix .5):

$$\frac{\dot{\lambda}(t)}{\lambda(t)} = -\theta \frac{\dot{c}(t)}{c(t)} = \beta - (r(t) - g - n) \tag{4.25}$$

then we could replace β with its definition in Eq. (4.16).

$$\begin{aligned}\frac{\dot{c}(t)}{c(t)} &= \frac{r(t)-g-n-\beta}{\theta} \\ &= \frac{r(t)-\rho-\theta g}{\theta}\end{aligned} \tag{4.26}$$

Until now, we have shown that the optimal consumption/saving problem satisfies the following two equations:

$$\begin{aligned}\frac{\dot{c}(t)}{c(t)} &= \frac{f'(k(t))-\rho-\theta g}{\theta} \\ \dot{b}(t) &= (r(t)-g-n)b(t)+w(t)-c(t)\end{aligned} \tag{4.27}$$

Remember that equilibrium on the asset market requires that family wealth equals the stock of capital: b(t) = k(t). With condition in (4.5) and (4.8) , substituting these expressions, we obtain a dynamic system:

$$\begin{aligned}\frac{\dot{c}(t)}{c(t)} &= \frac{f'(k(t))-\rho-\theta g}{\theta} \\ \dot{k}(t) &= f(k(t))-c(t)-(g+n)k(t)\end{aligned} \tag{4.28}$$

Figure. 4.3 shows the dynamic of the system. g(c) and g(k) mean the gradient of c and k. The area below the $g(k)=0$ curve indicates the area where $g(k)>0$, the area above the $g(k)=0$ curve indicates the area where $g(k)<0$. The left area the line $g(c)=0$ indicates the area where $g(c)>0$, the right area the line $g(c)=0$ indicates the area where $g(c)<0$.

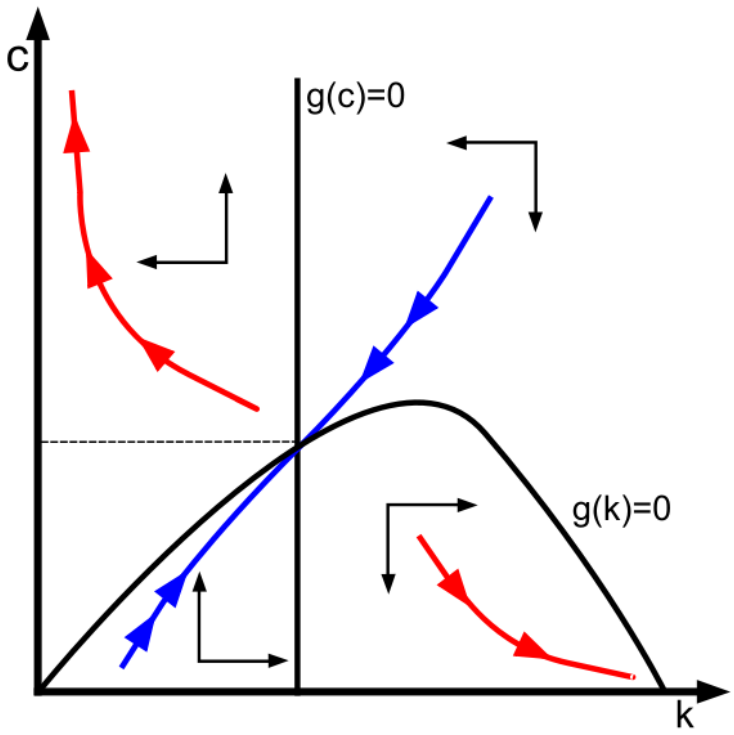


Figure 4.3 – Phase space diagram of the Ramsey-Cass-Koopmans model. The blue line represents the dynamic adjustment (or saddle) path of the economy in which all the constraints present in the model are satisfied. It is a stable path of the dynamic system. The red lines represent dynamic paths which are ruled out by the transversality condition.

Above section introduced two optimal growth models, it gives an overview of how to achieve optimal growth under different economic conditions. Their assumptions and solutions are well-established and well-studied, and they focus on the entire economic cycle. In the following sections for Economic Area, we will focus on details of process of identifying the mathematics relations between macroeconomic variables, the method can be used to modelize the entire economic system, but can also be used to modelize part of it, using real world data to reveal their relations (There may exist or not).

4.1.2 Augmented Dickey–Fuller (ADF) Test

In statistics and econometrics, an augmented Dickey–Fuller test (ADF) tests the null hypothesis that if a unit root is present in a time series sample. If one time series does not contain unit root, we say the time series is stationary. The stationarity is important if we do the regression on time series. If the time series are non-stationary, the regression gives spurious results. [55].

As the name suggest, the ADF test is an 'augmented' version of the Dickey Fuller test. The Dickey-Fuller (DF) test can be written as:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \varepsilon_t \tag{4.29}$$

where α is constant, β is the coefficient on the time trend, y_{t-1} is the value of y at time $t-1$, $\Delta y_t = y_t - y_{t-1}$. The null hypothesis (H0) here is $\gamma = 0$, which means time series is non-stationary.

The The ADF test expands the Dickey-Fuller test equation to include high order regressive process in the model. The H0 holds the same, the ADF test can be written as:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} \cdots + \delta_{p-1} \Delta y_{t-p+1} + \varepsilon_t \tag{4.30}$$

As the null hypothesis assumes the presence of unit root, the p-value obtained should be lower than a significant level (e.g. 0.05) in order to reject the null hypothesis. Thereby, inferring that the series is stationary. ADF test is implemented in Sec. 5.4 and Sec. 6.2.

4.1.3 Vector Autoregession (VAR)

Vector autoregression (VAR) is a stochastic process model used to capture the linear interdependencies among multiple time series. It is useful when one is interested in predicting multiple time series variables using a single model. VAR extends the idea of univariate autoregression to k time series regressions. Eq. (4.31) shows a VAR(p) model of k variables, where p is the lag order of model, y_t is a vector of length k and each A_i is a k $\times$ k matrix.

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t \tag{4.31}$$

Eq. (4.31) can be expanded to eq. (4.32) by the equation of regression notation. The matrix notion of eq. (4.31) is illustrated in eq. (4.33).

$$\begin{aligned}
y_{1,t} &= c_1 + a^1_{1,1} y_{1,t-1} + a^1_{1,2} y_{2,t-1} + \cdots + a^1_{1,k} y_{k,t-1} + \cdots + a^p_{1,1} y_{1,t-p} + a^p_{1,2} y_{2,t-p} + \cdots + a^p_{1,k} y_{k,t-p} + e_{1,t} \\
y_{2,t} &= c_2 + a^1_{2,1} y_{1,t-1} + a^1_{2,2} y_{2,t-1} + \cdots + a^1_{2,k} y_{k,t-1} + \cdots + a^p_{2,1} y_{1,t-p} + a^p_{2,2} y_{2,t-p} + \cdots + a^p_{2,k} y_{k,t-p} + e_{2,t} \\
&\vdots \\
y_{k,t} &= c_k + a^1_{k,1} y_{1,t-1} + a^1_{k,2} y_{2,t-1} + \cdots + a^1_{k,k} y_{k,t-1} + \cdots + a^p_{k,1} y_{1,t-p} + a^p_{k,2} y_{2,t-p} + \cdots + a^p_{k,k} y_{k,t-p} + e_{k,t}
\end{aligned} \tag{4.32}$$

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \\ \vdots \\ y_{k,t} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} + \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & \cdots & a^1_{1,k} \\ a^1_{2,1} & a^1_{2,2} & \cdots & a^1_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ a^1_{k,1} & a^1_{k,2} & \cdots & a^1_{k,k} \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ \vdots \\ y_{k,t-1} \end{bmatrix} + \cdots + \begin{bmatrix} a^p_{1,1} & a^p_{1,2} & \cdots & a^p_{1,k} \\ a^p_{2,1} & a^p_{2,2} & \cdots & a^p_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ a^p_{k,1} & a^p_{k,2} & \cdots & a^p_{k,k} \end{bmatrix} \begin{bmatrix} y_{1,t-p} \\ y_{2,t-p} \\ \vdots \\ y_{k,t-p} \end{bmatrix} + \begin{bmatrix} e_{1,t} \\ e_{2,t} \\ \vdots \\ e_{k,t} \end{bmatrix} \tag{4.33}$$

4.1.4 Cointegration Test

Before introducing cointegration, let us first check the notion of **Order of Integration**. In statistics, the order of integration, denoted I(d), of a time series is a summary statistic, which reports the minimum number of differences required to obtain a stationary series. I(0) is the series that are already stationary without any differencing.

Cointegration is a statistical property of time series introduced in economic analysis, by Engle and Granger (1987) [39]. In simple terms, cointegration makes it possible to detect the long-term relationship between two or more time series. For a set of time series, if all of the series are integrated of order d, if there exists a linear combination of this set of time series, which is integrated of order less than d, then the collection is said to be co-integrated. For instance when X_t and Y_t are I(1), and if there is a θ such that $Y_t - \theta X_t$ is I(0), then X_t and Y_t are cointegrated. Cointegration test is implemented in Sec. 5.4.2.

4.1.5 Vector Error Correction Model (VECM)

Following the example given in the end of Sec. 4.1.4, if X_t and Y_t are I(1) and cointegrated, their differences are stationary and can be modeled in a VAR which is augmented by the regressor $Y_{t-1} - \theta X_{t-1}$. This is called a vector error correction model (VECM) and $Y_t - \theta X_t$ is called the **error correction term**. Lagged values of the error correction term are useful for predicting ΔX_t and/or ΔY_t. The VECM model makes it possible to study short-term fluctuations around the long-term equilibrium. It is implemented in Sec. 5.4.3.

4.1.6 Granger Causality Test

The Granger causality [56] is a statistical concept of causality that is based on prediction. Granger causality test is for determining whether one time series is useful in forecasting another. Since the question of "true causality" is deeply philosophical, we should understand the Granger causality more like "predictive causality", and we should avoid to use the term "X_1 cause X_2", but "X_1 granger-cause X_2" if it can be shown.

G-causality is normally tested in the context of linear regression models. For illustration, consider a bivariate linear autoregressive model of two variables:

$$\begin{aligned} X_1(t) &= \sum_{j=1}^{p} A_{11,j} X_1(t-j) + \sum_{j=1}^{p} A_{12,j} X_2(t-j) + E_1(t) \\ X_2(t) &= \sum_{j=1}^{p} A_{21,j} X_1(t-j) + \sum_{j=1}^{p} A_{22,j} X_2(t-j) + E_2(t) \end{aligned} \tag{4.34}$$

where p is the order of model, A is the coefficient matrix, E_1 and E_1 are residuals for each estimation. If the variance of E_1 (or E_2) is reduced by inclusion of X_2 (or X_1) term in the first (or second) equation, then it is said that X_2 (or X_1) granger-cause X_1 (or X_2).

Granger causality is used at Sec. 5.4.4.

4.1.7 Vector Autoregressive Exogenous (VARX) model

Recall the eq. (4.31), the VARX equation is like:

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + B_1 u_{t-1} + B_2 u_{t-2} + \cdots + B_q u_{t-q} + e_t \tag{4.35}$$

where u_i is the exogenous inputs and B_i are their coefficient matrix. It is basically the VAR model with external inputs. VARX is implemented in Sec. 6.2.3.

4.2 Automatic Area

4.2.1 State Space Representation

State space representation is commonly used in control engineering, and it has also become widespread in macroeconomics and finance over last decades [40]. Some common examples are autoregressive moving average (ARMA) models, time varying regression models, and dynamic linear models with unobserved components. A state space is the set of all possible configurations of a system. We use state space representation because it is easier to implement a control algorithm. In this thesis, we only

focus on discrete-time systems. The most general state-space representation of a linear system with p inputs, q outputs and n state variables is written in the following form:

$$\begin{aligned} x(k+1) &= A \cdot x(k) + B \cdot u(k) \\ y(k) &= C \cdot x(k) + D \cdot u(k) \end{aligned} \tag{4.36}$$

where $k \in \mathbb{Z}$, $\mathbf{x}(\cdot)$ is called state vector, $\mathbf{x}(\cdot) \in \mathbb{R}^n$; $\mathbf{y}(\cdot)$ is called output vector, $\mathbf{y}(\cdot) \in \mathbb{R}^q$; $\mathbf{A}(\cdot)$ is the "state (or system) matrix", $\dim[\mathbf{A}(\cdot)] = \mathbb{R}^{n \times n}$; $\mathbf{B}(\cdot)$ is the "input matrix", $\dim[\mathbf{B}(\cdot)] = \mathbb{R}^{n \times p}$; $\mathbf{C}(\cdot)$ is the "output matrix", $\dim[\mathbf{C}(\cdot)] = \mathbb{R}^{q \times n}$; $\mathbf{D}(\cdot)$ is the "feedthrough (or feedforward) matrix" (in cases where the system model does not have a direct feedthrough, $\mathbf{D}(\cdot)$ is the zero matrix); $\dim[\mathbf{D}(\cdot)] = \mathbb{R}^{q \times p}$. For the control system, there are three important properties to verify:

4.2.1.1 Stability

For the discrete-time state space system, the system is stable if the magnitude of the eigenvalues of the system matrix A is less than 1.

4.2.1.2 Controllability

Controllability is an important property of a control system. The state controllability condition implies that it is possible – by admissible inputs – to steer the states from any initial value to any final value within some finite time window. Mathematically, a system is controllable if for all the time span $[k_i, k_j]$, for all the state $x_i, x_j \in X$ where $x(k_i) = x_i$, there exists a control u applied during time span $[k_i, k_j]$ to have $x(k_j) = x_j$.

In practice, to verify the controllability for a state space system, it is to check if the controllability matrix is full rank:

$$\operatorname{rank}(M_c) = \operatorname{rank}\begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} = n \tag{4.37}$$

where n is the number of the state variables.

4.2.1.3 Observability

Observability is a measure for how well internal states of a system can be inferred by knowledge of its external outputs. A system is said to be observable if the observation of its inputs and outputs during a finite time interval $[k_i, k_j]$, allows to determine the initial state $x(k_i)$, and therefore, by integration of the equation of state, to know $x(\cdot)$ at any time belonging to the interval $[k_i, k_j]$.

$$\operatorname{rank}(M_b) = \operatorname{rank}\begin{bmatrix} C \\ CA \\ \dots \\ CA^{n-1} \end{bmatrix} = n \tag{4.38}$$

The initial condition is determined if the so-called *observability matrix* M_b has rank n, where n is the number of the state variables.

State space representation is used in Sec. 6.2.4.

4.2.2 Linear–Quadratic Regulator (LQR)

The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic function is called the LQ problem [4]. One of the main results in the theory is that the solution is provided by the linear–quadratic regulator (LQR), which is a feedback controller. For a discrete-time linear system:

$$\mathrm{X(k+1)} = A \cdot \mathrm{X(k)} + B \cdot \mathrm{u(k)}$$

the cost function of discrete time LQR in finite horizon can be presented as follows:

$$J = X(N)^T QX(N) + \sum_{k=0}^{N-1} (X(k)^T QX(k) + u(k)^T Ru(k) + 2x(k)^T Nu(k)) \tag{4.39}$$

where Q, R are the weight matrices for state and input, the cross term matrix N is in general set to 0.

The feedback control law that minimizes the cost function J is:

$$u(k) = -K \cdot X(k) \tag{4.40}$$

where K is given by:

$$K = (R + BP^T B)^T (B^T PA + N^T) \tag{4.41}$$

and P is the unique positive definite solution to the discrete time algebraic Riccati equation (DARE):

$$P = A^T PA - (A^T PB + N)(R + B^T PB)^{-1}(B^T PA + N^T) + Q \tag{4.42}$$

This control law will let output converge to 0 if $Reference$ vector is 0. The typical structure of LQR control system is showed in Fig. 4.4. More details of LQR is discussed in Sec. 6.4.1.

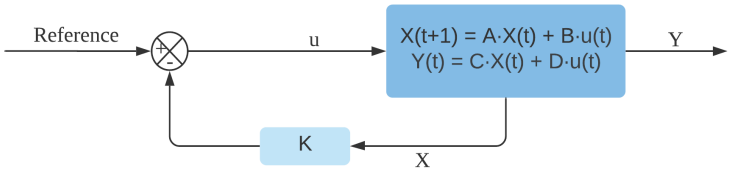


Figure 4.4 – Typical LQR Control System.

Chapter 5

Modelling and Dynamic Analysis of the Domestic Demand Influence on the Economic Growth of China

In this chapter, we examines the Granger causality between economic growth, household final consumption (henceforth referred to as “Consumption"), inward foreign direct investment (FDI) and Export from China using annual data from 1985 to 2014. Cointegration test and vector error correction model (VECM) are used to identify long-term and short-term relationships. VECM is also used to find causal links between variables. The result shows there are bidirectional causalities between Consumption, FDI and GDP. Unidirectional causality from FDI to Export, from Export to Consumption and GDP. The change in the composition of FDI also reveals that as Consumption increases, FDI in high-tech domain goes up, especially in the high-tech service industry, which can have an effect on long-run economic growth and change the structure of China’s Export.

Above analysis including data cleaning, system estimation and Granger causality identification are all carried out with economic methods presented in Sec. 4.1. This chapter is structured as follows: Sec. 5.1 introduces the interests of performing this analysis, and the focus of recent researches. Sec. 5.2 shows several data which motivate this study. Sec. 5.3 introduces the data and methodology, Sec. 5.4 presents the empirical result and discussion, Sec. 5.5 concludes.

5.1 Introduction

Since the implementation of the Open Door Policy in 1978, China has experienced over 40 years economic growth and is known as an export-oriented growth model.The huge economic growth attracts many foreign investments, in 2017, there are 144 billion dollars FDI inflow in China, after USA, ranked second in the world. Also in 2017, total retail sales in China hit 5781.43 billion dollars, which is almost even to USA. Considering that China’s population is more than four times that of the United States, and consumption growth (10.2% in 2017) is also higher than in the United States (4.2% in 2017)[6]. There is no doubt that China will become the world’s largest market in next years. But there is limited literature that analyse Consumption’s impact on FDI, Export and GDP. As China’s middle class has grown, Consumption will play an important role in the transition of China’s economic structure. This study is not only beneficial to understand the development strategies in China, but also provides ideas for other developing countries.

[6]National Bureau of Statistics of China and US Census Bureau

Previous research on this subject mainly focuses on the causal relationship between FDI, trade and GDP using time series or panel data, different periods will lead to different conclusions. Liu [87] finds that there are two-way causalities between Export, FDI and GDP from 1981 to 1997. Tang [129] examines the interactions between FDI, DI (Domestic Investment), and GDP, and finds only one-way causality from FDI to GDP using time series from 1978 to 2003. Li [86] indicates that Export is a long-term and short-term source of GDP growth of west China during the period 1985 to 2008, and there is bidirectional causality between Export and GDP. At the first stage of the Open Door Policy, China's inner market is very small, they have to use FDI to build factories and introduce technologies, then export low-end products to accumulate wealth. At that moment FDI and Export are the two most important factors to China's economy. As the average salary increased from 2711 RMB in 1992 to 74318 RMB in 2017[7], we can no longer ignore the importance of domestic demand, some recent studies begin to concern this point. In Eichengreen [38]'s study of the slow down of fast-growing economies, they conclude that an exceptionally low consumption share of GDP is positively associated with the probability of a slowdown in growth. Guilhot [58] analyses China's new development strategy to overcome the middle-income trap, she summarizes that if China wants to cross the middle-income trap, it must develop domestic demand and high value-added manufacturing. Lardy [81] lists several reasons of why China's leadership wishes to transition to a more consumption-driven growth: the overinvestment and the excess production slows the growth of factor productivity; extensive pattern of economic development has impeded the growth of personal consumption and generated very modest gains in employment; burgeoning energy consumption makes detrimental effects on the environment. Renard [113] indicates that the reason that China changes its commercial structure is twofold: i) the 2007 financial crisis, 2) the increasing cost of production. The slow-down of the economic growth of Europe and USA reduces the external demand of China products, therefore it influences the exportation and importation (e.g., the importation of materials to be used to assemble and export). That forces China to focus more on its internal market. The rebalancing towards consumption, in particular of services, results in a lower demand for foreign goods. As all the cited papers above indicate the importance of Consumption in China today, we will show in this chapter the interaction between GDP, FDI, Consumption and Export with the data including recent years.

5.2 Motivation

In recent years, China has rapidly transitioned from a predominantly lower-middle class society to a middle, upper-middle and affluent class society over the past decade [10]. Fig. 5.1 shows that we rank per capita disposable income (henceforth referred to as "PCDI") of nationwide households from high to low, evenly divided in 5 group (so each group contains 20% households). We can observe that during 2013-2017, PCDI of households in all the levels are increasing. Obviously, higher groups have a sharper increasement, but that doesn't mean there is no improvement in lower group, In Fig. 5.2, we could see that the differences of growth rate between groups are not enormous, but when we consider the big gap of bases between groups, higher groups indeed have a bigger absolute augmentation.

As the PCDI increase in all groups, a huge demand is being created for all level of consumptions, especially the emerging middle, upper-middle class, they will demand more better quality product, the most obvious field is the automobile. In 2009, China surpassed USA to become the world's biggest automobile market. In 2017, it selled 28.8 million cars, of which more than 1.2 million are imported (BMW, Mercedes, Lexus ranked in the top 3). Also in 2017, Chinese tourists spent 258 billion U.S. dollars on international tourism, almost one fifth of the world's total tourism spending [37]. Insurance policies per capita increased from 1.96 (2010) to 2.89 (2017) [30].

[7] Ministry of Human Resources and Social Security, China

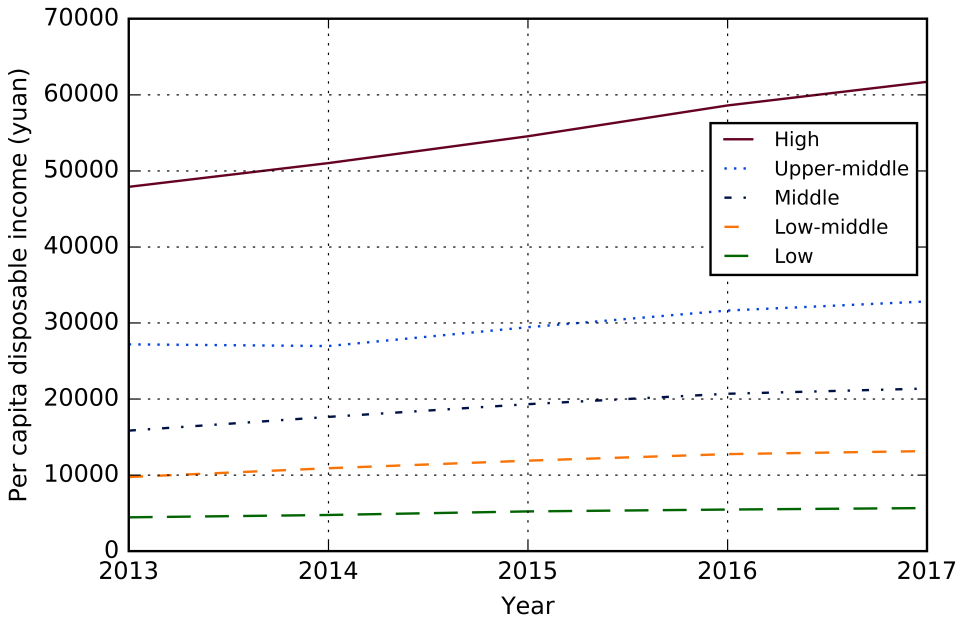


Figure 5.1 – Per Capita Disposable Income of Natiaonwide Households by Income Quintile

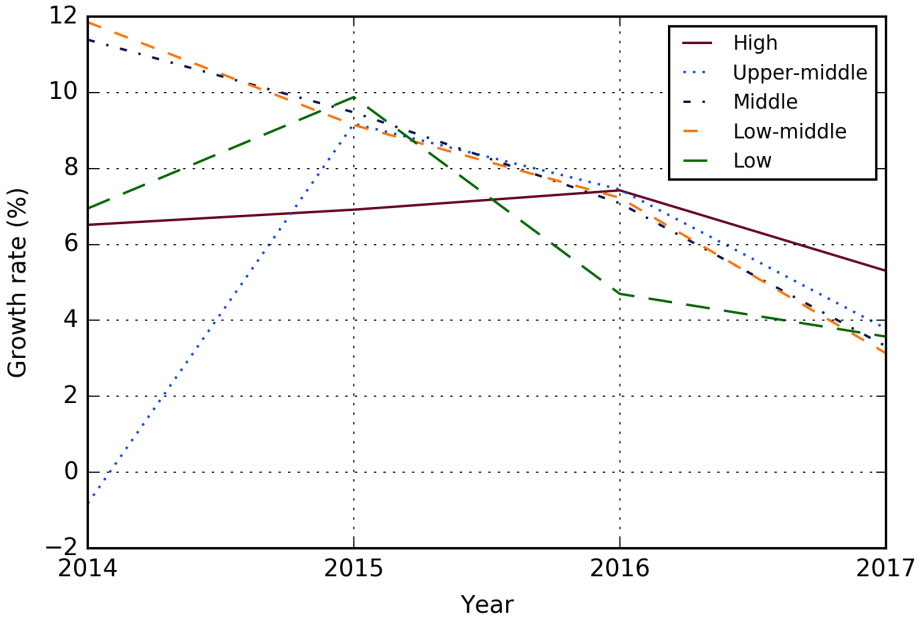


Figure 5.2 – Growth Rate of Per Capita Disposable Income of Natiaonwide Households by Income Quintile

While the consumption booms with income, after reaching the top in 2006, the proportion of exports to GDP begins to decrease. Due to the cost reasons, the low-end manufacturing is partially replaced by Southeast Asian countries, and as the high-end manufacturing are controlled by very few countries in the world, it is almost impossible to convince other countries to share jobs in this area. But the huge consumer market is an important leverage for China, this is why we can see the factories of BMW and Mercedez, or even of Airbus and Boeing in China. In the following sections, we will show the importance of consumption to China's GDP, and explain how China's consumption reshape the country's economic growth.

5.3 Data and Methodology

Annual Export, Consumption and GDP of China from 1985 to 2014 are obtained from World Bank. Export consists of goods and services. Annual FDI inflow data for the same period are acquired from various sources, including IMF (International Monetary Fund) from 1985 to 1989 and UN Comtrade from 1990 to 2014. All the series are deflated by GDP deflator (2010 = 100), GDP deflator data is also obtained from World Bank. And all the variables are expressed on logarithm[8].

At first, in order to use the Least Squares regressor and to estimate a VAR model, we will test the stationarity of each variable and their first difference by ADF (Augmented Dickey-Fuller) unit root test. Then, we will estimate a VAR model with level value to select lag. A VAR (p) model (VAR model with p lags) can be carried out via the following equations:

$$y_t = c + A_1 y_{t-1} + \cdots + A_p y_{t-p} + e_t, \quad t = 0,1,2,...,N \tag{5.1}$$

where $y_t = (y_{1,t}, y_{2,t}, ..., y_{k,t})^T$ is a $(K \times 1)$ vector, K is the dimension of y_t. A_i is a $(K \times K)$ coefficient matrix, $c = (c_1, c_2, ..., c_k)^T$ is a $(K \times 1)$ intercept vector and $e_t = (e_{1,t}, e_{2,t}, ..., e_{k,t})^T$ is a k-dimensional *white noise*.

Thirdly, the Johansen cointegration test will be used to find the cointegration rank and the cointegration equation. Fourthly, we will estimate the VECM (Vector Error Correction Model) to find various Granger causal relations, then use it for further weak exogeneity tests. We will also run other pairwise Granger causality tests (ΔLEXPORT, ΔLCONSUMPTION), (ΔLEXPORT, ΔLGDP) and (ΔLEXPORT, ΔLFDI) to better explain our model.

VAR need to be implemented with stationary variables, which means there are no trends or shifts in the mean or in the covariances, but trends are quite common in practice, for example Fig. 5.3. We can see that there are strong increasing trends for all four variables, so in this situation, we need to check if all the variables are integrated of order one, if so, we could run Johansen cointegration test to find the cointegration rank and cointegration equations, then use them to fit a VECM. Supposed that all the variables are I(1): integrated of order one, and cointegration rank equals to 1, then VECM with one lag can be written in form:

$$\Delta y_t = \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \\ \alpha_{41} \end{bmatrix} \left[\beta_{11}\beta_{12}\beta_{13}\beta_{14}\right] y_{t-1} + \Gamma_1 \Delta y_{t-1} + c + \varepsilon_t, \quad t = 0,1,2,...,N \tag{5.2}$$

where Δ is the first difference operator, $y_t = (y_{1,t}, ..., y_{k,t})^T$ is a $(K \times 1)$ vector (in our case, K = 4), $\Delta y_t = (y_{1,t} - y_{1,t-1}, ..., y_{k,t} - y_{k,t-1})^T$ is a $(K \times 1)$ vector of first difference of y_t, Γ_1 is a $(K \times K)$ coefficient matrix, $c = (c_1, c_2, ..., c_k)^T$ is a $(K \times 1)$ intercept vector and $\varepsilon_t = (\varepsilon_{1,t}, \varepsilon_{2,t}, ..., \varepsilon_{k,t})^T$ is a

[8]Logarithm of Export, Consumption, FDI and GDP referred to as LExport, LConsumption, LFDI and LGDP

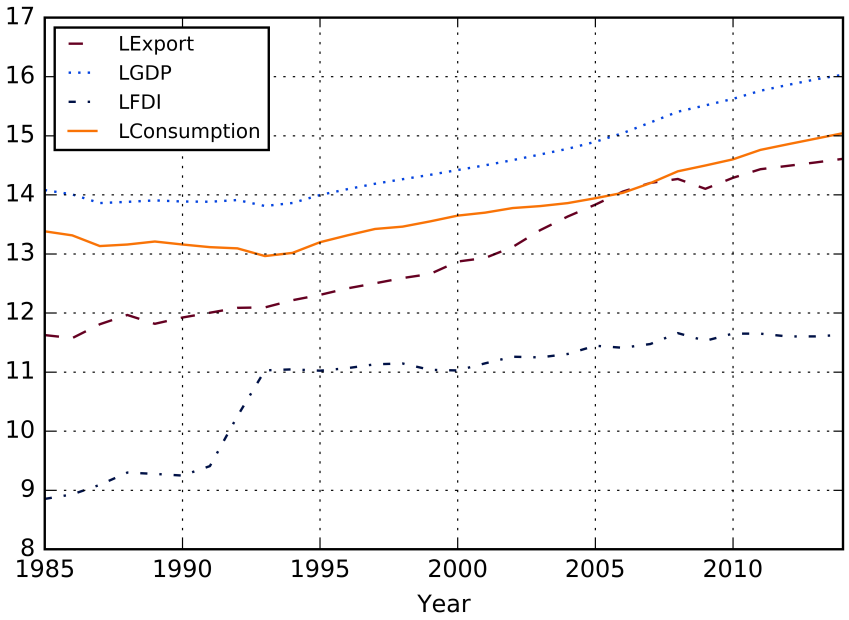


Figure 5.3 – Trends of variables

k-dimensional *white noise*, dimension of $\beta_{i,j}$ and $\alpha_{i,j}$ is just (1×1), matrix with all $\beta_{i,j}$ is called *cointegrating matrix* and matrix with all $\alpha_{i,j}$ is referred to as the *loading matrix*. Note that if cointegration equation has a constant, the part $\left[\beta_{11}\beta_{12}\beta_{13}\beta_{14}\right] y_{t-1}$ should become $\left[\beta_{11}\beta_{12}\beta_{13}\beta_{14}\beta_{15}\right] \begin{bmatrix} y_{t-1} \\ 1 \end{bmatrix}$. Weak exogeneity test [69] will be implemented with the VECM. It is also called restricted VECM where we put restriction on $\alpha_{i,j}$ and $\beta_{i,j}$.

5.4 Empirical result and discussion

5.4.1 Preparation for cointegration test and VECM

Tab. 5.1 gives the result of ADF unit root test. The selection of optimal lag is chosen by downward search from a pre-chosen lag (here we set to 5) length based on the Akaike information criterion (AIC). From the result, we cannot reject the null hypothesis for the variables in level, but the values of their first difference are stationary, so all the variables are integrated of order one.

Since all the variables are integrated of order one, we could fit a VAR and select lag. Notice that, while we have considered specifying the VAR order p, but our objective is to choose lagged differences for our VECM model, the criterion of VAR order is also applicable for choosing the number of lagged differences in a VECM. $p-1$ lagged differences in a VECM correspond to a VAR order p. Thus once we know p, we know the number of lagged differences. If some of the variables are known to be integrated, the VAR order must be at least 1 [91].

We firstly fit a VAR model, then run a VAR lag order selection test with a maximum lag equals to 3. In Tab. 5.2, column from 3 to 7, each column represents a test of order selection, the result clearly shows that lag = 2 is the best choice by the criterion in all the methods. Therefore, the lagged differences in VECM equal to 1.

Table 5.1 – ADF test for unit root

Null hypotheses: LExport, LGDP, LConsumption and LFDI contain a unit root

Variables	ADF Level	ADF First Difference
LExport	-1.08678(0)	-4.95614***(0)
LGDP	-0.779635(1)	-4.44409***(0)
LConsumption	0.881906(1)	-2.83766**(0)
LFDI	-2.11473(2)	-6.20035***(1)

Notes: (1) ***, ** and * denote significance at the 1%, 5%, and 10% levels, respectively.
(2) Figures in parentheses are the number of lags used.

Table 5.2 – VAR lag order selection criterion

Endogenous variables: LExport, LGDP, LConsumption and LFDI
Exogenous variables: C

Lag	LogL	LR	FPE	AIC	SC	HQ
0	9.181270	NA	8.01e-10	-0.383798	-0.191822	-0.326713
1	164.4898	253.0954	2.69e-10	-10.70295	-9.743069	-10.41753
2	194.5407	40.06793*	1.04e-10*	-11.74376*	-10.01598*	-11.23000*
3	209.4034	15.41312	1.44e-10	-11.65951	-9.163826	-10.91741

* indicates lag order selected by the criterion
LR: sequential modified LR test statistic(each test at 5%)
FPE: Final prediction error
AIC: Akaike information criterion
SC: Schwarz information criterion
Hq: Hannan-Quinn information criterion

5.4.2 Cointegration test

The Johansen cointegration test is used to find the cointegration rank and cointegration equations among the variables before causality testing using the VECM model. Tab. 5.3 shows that both Trace and Maximum eigenvalue test indicate 1 cointegrating equation at the 0.05 level. Four variables for one cointegrating equation, therefore, one normalization condition and tree exclusion restrictions are introduced; this study normalizes on LGDP and excludes LExport, LFDI and LConsumption, which leads the cointegrating equation below:

$$LGDP = 0.049LConsumption + 0.08LFDI + 0.81LExport + 2.47 \tag{5.3}$$

The coefficient reveals that Consumption, FDI and Export all have positive impacts on GDP, as Figure 5.3 reflects that there are common trends of all four variables, the slope of LExport is obviously steeper than others before 2008 crisis, that's why it contributes more (coefficient is much bigger than other two) in LGDP; another trend is also apparent: after 2009, growth of LExport became weaker, LFDI keeps its trend, only LConsumption follows the speed of growth of LGDP.

Table 5.3 – Johansen cointegration test

Series: LExport, LGDP, LConsumption and LFDI
Unrestricted Cointegration Rank Test (Trace)

Hypothesized No. of CE(s)	Eigen-value	Trace Statistic	0.05 Critical Value	Prob. **
None *	0.841705	80.66832	47.85613	0.0000
At most 1	0.511876	29.05613	29.79707	0.0607
At most 2	0.256716	8.974912	15.49471	0.3677
At most 3	0.023573	0.667953	3.841466	0.4138

Series: LExport, LGDP, LConsumption and LFDI
Unrestricted Cointegration Rank Test (Maximum eigenvalue)

Hypothesized No. of CE(s)	Eigen-value	Max-Eigen Statistic	0.05 Critical Value	Prob. **
None *	0.841705	51.61219	27.58434	0.0000
At most 1	0.511876	20.08122	21.13162	0.0696
At most 2	0.256716	8.306959	14.26460	0.3483
At most 3	0.023573	0.667953	3.841466	0.4138

* denotes rejection of the hypothesis at the 0.05 level
**MacKinnon-Haug-Michelis(1999) p-values

5.4.3 Weak Exogeneity and Granger Causality of VECM

According to the cointegration equation, the order of lagged difference obtained in the previous test, a VECM model in form of Eq. (5.2) is implemented, the equation which normalizes $\Delta LGDP$ on the left-hand-side is introduced below:

$$\begin{aligned}\Delta(LGDP) =& C(1)*(LGDP(-1)-0.049*LConsumption(-1)\\ &-0.08*LFDI(-1)-0.81*LExport(-1)\\ &-2.47)+C(2)*\Delta(LGDP(-1))\\ &+C(3)*\Delta(LConsumption(-1))+C(4)*\Delta(LFDI(-1))\\ &+C(5)*\Delta(LExport(-1))+C(6)\end{aligned} \tag{5.4}$$

and Tab. 5.4 gives the values of all coefficients and probabilities of T-test. The column of Probability shows that except C(5), all the coefficients are statistically significant. The most important coefficient is C(1), it must be statistically significant and negative, otherwise our Eq. (5.4) will not converge to long-term relation.

To conduct weak exogeneity test, we need to successively set each row of cointegrating matrix($\alpha_{i,j}$ in Eq. (5.2)) to zero as a restriction before we implement VECM model. Tab. 5.5 presents the weak exogeneity test result, the test for $\Delta LGDP$ $\Delta LFDI$ and $\Delta LConsumption$ massively reject the null hypothesis, but $\Delta LExport$ cannot reject that it is weakly exogenous to system. According to Fig. 5.3,

Table 5.4 – T-test of coefficient in VECM model

Method: Least Squares (Gauss-Newton / Marquardt steps)

Null hypothesis: $C(x)$ equals to 0

Coefficient	Value	Std. Error	t-Statistic	Prob.
C(1)	-0.319553	0.039229	-8.145882	0.0000
C(2)	-0.990592	0.320478	-3.090988	0.0053
C(3)	0.579396	0.230062	-2.518436	0.0196
C(4)	-0.125740	0.026865	-4.680368	0.0001
C(5)	-0.002919	0.057463	-0.050793	0.9599
C(6)	0.118864	0.015039	7.903478	0.0000

we can see that there is a big chute of LExport in 2009, but the LGDP and LConsumption are not influenced, although there is a small vibration for LFDI too, it's not as strong as LExport. This result, therefore, implies that there is bi-directional Granger causality between GDP, FDI and Consumption, but not Export. This conclusion is reinforced by Granger causality test result in Tab. 5.6. Wald test result reveals that there is bi-directional causality between GDP, FDI and Consumption, but there is only one-way causality from FDI to Export. Liu [87] and Wong [133] have all mentioned that there is bi-directional causality between GDP and Export of China using annual time-series data over the period from 1978 to 2002. After including recent annual data until 2014, this relation disappeared. However if we run 3 other pairwise Granger Causality test, Tab. 5.7 reflects that Export Granger causes GDP and Consumption at the 5% and 10% level respectively, but not FDI. Therefore, the Export does not jointly cause GDP and Consumption with other variables, but it causes GDP and Consumption pairwisely.

Table 5.5 – Weak exogeneity test

Weak exogeneity test on restricted $\alpha\beta^T$: $X^2(1)$	LR test	p-value
ΔLGDP weakly exogenous to system	28.31301	0.000000
ΔLFDI weakly exogenous to system	7.859995	0.005054
ΔLConsumption weakly exogenous to system	16.10339	0.000060
ΔLExport weakly exogenous to system	2.456090	0.117070

5.4.4 Further discussion

China has long been considered a model for an export-oriented economy, and because of tariffs and subsidies, it has led to today's trade disputes with other countries, but our results show that there is no longer two-way causality between Export and GDP. After peaking in 2006, the proportion of exports to gdp has been decreasing [9]. Since 2009, the contribution of Export to GDP growth is negative except 2012 and 2014[10].

[9] World Bank national accounts data, and OECD National Accounts data files

[10] NBS, Statistical Yearbook, 2016

Table 5.6 – VECM Granger Causality test

Dependent variables	Wald test statistics				Causality inference
	ΔLGDP	ΔLFDI	ΔLConsumption	ΔLExport	
ΔLGDP		21.90584***	6.342517***	0.002580	FDI⇒GDP Consumption⇒GDP
ΔLFDI	9.148019***		8.946369***	0.121575	GDP⇒FDI Consumption⇒FDI
ΔLConsumption	7.149010***	6.155757***		0.003784	GDP⇒Consumption FDI⇒Consumption
ΔLExport	0.000503	6.675678***	1.169216		FDI⇒Export
Bi-directional Causality					FDI⇔GDP Consumption⇔GDP Consumption⇔FDI

Notes: (1) ***, ** and * denote significance at the 1%, 5%, and 10% levels, respectively.
(2) Null hypothesis: In one row, dependent variable does not caused by other column variables.

Table 5.7 – Pairwise Granger Causality test

Null Hypothesis	**F-Statistic**	**Prob.**
ΔLGDP does not Granger Cause ΔLExport	0.61579	0.4400
ΔLExport does not Granger Cause ΔLGDP	4.96100	0.0352
ΔLConsumption does not Granger Cause ΔLExport	1.26041	0.2722
ΔLExport does not Granger Cause ΔLConsumption	4.12415	0.0530
ΔLFDI does not Granger Cause ΔLExport	1.37615	0.2518
ΔLExport does not Granger Cause ΔLFDI	0.00713	0.9334

FDI still shows bi-directional causality with GDP, but this relation needs to be explored. Due to China's regulation of FDI in the financial industry, foreign investors were not allowed to hold more than 50% of the shares of Chinese securities companies, fund management and futures companies, so most of the FDI flowed into manufacturing, until 2015, 49.97% of total accumulated FDI has been used in manufacturing [96], the technology spillover brought by foreign invested companies is a key factor in long-run economic growth [115], FDI can be seen as a proxy of this factor. We also noticed that there is only unidirectional causality from FDI to Export, that implies Export is no longer able to stimulate FDI, China's low-end manufacturing has been taken over by southeast Asian countries, the structure of China's economy is changing. In 2011, it's the first time that FDI inflow in service industry is higher than manufacturing, and in 2015, 67.61% of FDI is used in service industry, but the export of service is at its beginning (15.4% of total export, comparing to 32.1% of US in 2015), we can imagine, service industry will become more and more important in China's export.

The Consumption has two-way causality with GDP and FDI, and only one-way causality from Export to Consumption. Due to the tariffs, the price of goods imported into China is more expensive and less competitive, but no one wants to lose the market of 1.4 billion people. Then the companies choose to fabricate locally, that attracts more FDI putting into high-end manufacturing domain, which will also make technology spillover effect and result in long-term economic growth. Renard [113]'s study also indicates that China is enduring a rising deficit in service, mostly caused by transport and tourism, and IMF predicts that this deficit will aggravate in the following years. This trend can also attract FDI to invest into high-end service area. Moreover, the household final consumption expenditure which represents 40% of GDP in China is still much less than the average level (60%) of the OECD countries. Consumption will thus play an important role in the China's GDP growth in the next years.

5.5 Conclusion and Perspective

This study reveals the trend of China's economic transition. Many studies on the early stage of Open Door Policy have shown that there are two-way relations between Export, FDI and GDP. At that time, FDI mainly invested in low-end labor-intensive manufacturing, which helped China solve the poverty problem and completed China's early industrialization. At the moment, China's domestic market is still very small, and the main consumption is basic necessities. But as workers' wages increase, our result shows that Chinese consumers change not only the China's industrial structure but also the structure of FDI and path of growth of China's economy. The emerging middle class is no longer content to buy basic necessities, they need cars, travels, health care and financial products. The huge consumer market and the tariff faced by imports to China attracts and forces foreign companies to produce high-end products and provide high quality services in China, which directly changed the structure of FDI. Although the pairwise causality test shows Export still has one-way causality to GDP, its impact is not strong enough to show up in multivariable VECM causality test. Because of the environmental pressure, China is gradually stripping the highly polluting heavy industry, we can also sense this change through the structure of FDI, the proportion of investment in the service industry has increased year by year, especially in the high-tech service industry. Although the proportion of investment in manufacturing industry is continuously declining, the proportion of high-end manufacturing in total manufacturing investment continues to increase [96]. We can say that China has accumulated wealth with an export-oriented economy, and now, China is using its domestic market to adjust its economic structure.

In the future, we are more optimistic about China's domestic market than exports. After all, China's consumption still accounts for a smaller proportion of GDP than the average in OECD countries. FDI and Consumption will continue to support GDP, but with the development of the economy, China will be hard to take advantage of the status of a developing country on the issue of tariffs, once involved into a trade war, exports may be more difficult.

After implementing this study, one factor which holds us back to broaden the work is the limitation of China's data. To align all the time series, we could only collect the data from 1985, and the data is yearly instead of quarterly. For the interesting time series such as investment and public expenditure, we would also want to include them into the model to enrich the study, but these data from China are not available. Due to the above reason, in next chapter, we will continue the study switching to macroeconomic data of France.

Chapter 6

Modelling and Optimal Control of MIMO System - France Macroeconomic Model Case

In this chapter, we will introduce an optimal control on a Multiple-Input and Multiple-Output (MIMO) macroeconomic system. With french macroeconomic quarterly data from 1980Q1 to 2018Q4, MIMO linear autoregression models will be estimated and transferred to a state-space model, we implement the optimal control: Linear–Quadratic Regulator (LQR) on the estimated model to lead it reach the desired growth equilibrium. The results show that the control system can reach to the desired steady state, and by properly setting the parameters of LQR we can adjust the converging speed. If we impose constraints on one of the input signals, compensation effect shows on all other inputs and the result is therefore slightly worse than the case when there are no constraints. Perturbations on outputs are also studied, results show that our system can quickly recover from the disturbance.

6.1 Introduction

Control theory has a long history of implementation into economic domain. [28] summarizes the development of stochastic control theory in macroeconomic policy analysis into three periods. The first is pre-1970 when the major ideas of policy analysis and of optimization were formed [32]. The second is the early and middle 1970s when formal stochastic control theory was rapidly developed for and applied to the study of macroeconomic policy [27, 45]. The third period, beginning in the late 1970s, was stimulated by the introduction of the idea of rational expectations in economic analysis [18].

Recent works focus more on applications [62] proposes a general class of PID-based monetary policy rules, the feedback rules let model use control signal (e.g. central bank's policy interest rate) responds to movements in a small number of macroeconomic factors, such as the current amount of labor market slack and the deviation of the rate of inflation from its target. Under an optimal control monetary policy [20], the current and expected future path of the policy is instead typically calculated with a procedure that minimizes a cost function subject to certain constraints. For fiscal policy [90] and resource allocation [29, 66] problems, they follow the same ideas of optimal control, differences are the cost function and the constraints. To estimate the asset holdings of a portfolio, [47] uses algorithms applied to nonlinear dynamic systems to estimate the state with a discrete-time observer.

In this chapter, we will follow the optimal control idea mentioned in chapter. 4, we will apply it on the french macroeconomic model. Our objective is to maintain a constant economy growth rate (output) according to the available resources (input). We will first modelize this system as Multiple-Input and Multiple-Output (MIMO) system. After data pre-processing to make the time series stationary, the orders and parameters of MIMO system are estimated and validated before transforming it into a

state space model. The model we estimate in this chapter is autoregressive model with external inputs. The outputs of model are: GDP (Gross Domestic Production), EXP (Exportation), IMP (Importation). The inputs of model are: HC (Household Consumption), GFCF (Gross Fixed Capital Formation) and PE (public expenditure). To satisfy our objective, an optimal control solution: Linear–Quadratic regulator (LQR) is designed and implemented. By adjusting parameters of LQR, we can control the converging speed of output to steady state. A further simulation result shows that if we put constraints on the level of one of input signals, we can observe the compensation effect from other inputs, but even there is no limitation on the amplitude of inputs variables, the level of output and its converging speed to steady state is still slightly worse than the case without constraints. Perturbations on outputs are also studied, which simulate a scenario of economic crisis. The experiments are also developed with and without constraints on input signals. Results show that our system can quickly recover from the disturbance, and constraints on input signals delay the recovery.

6.2 System Identification

In this section, the process of estimating the economic model is given. After the data preparation to make original data stationary, model order is estimated for a MIMO model. The parameters of model are then estimated and validated.

6.2.1 Preparation of data

All the data are obtained from INSEE (Institut National de la Statistique et des Études Économiques). After discussions with experts on the issue, we decide to study 6 time series: GDP (Gross Domestic Production), EXP (Exportation), IMP (Importation), HC (Household Consumption), GFCF (Gross Fixed Capital Formation) and PE (public expenditure). All the data are quarterly, ranged from 1980Q1 to 2018Q4[11]. Original data are presented on the values of current price[12], we deflate it by France GDP deflator (base year: 2014) obtained from World Bank, deflated time series are showed in Fig. 6.1.

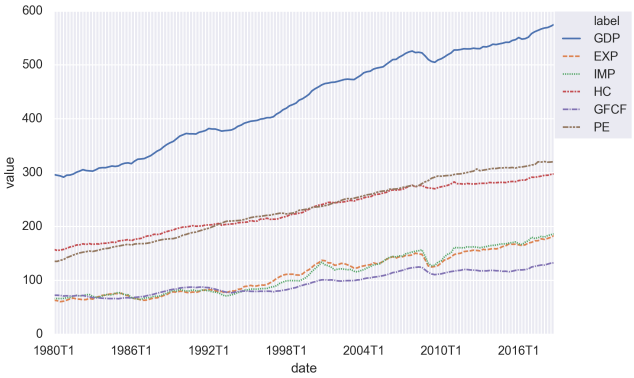


Figure 6.1 – Original data (Billion Euro).

In economic, when we deal with original time series, we prefer to use natural logarithm to better linearize them [46]. As we will use these time series to do linear regression, all the series must be stationary. ADF (Augmented Dickey–Fuller) test [25] is commonly used unit root test to examine

[11] Where Q1, Q4 denote first and fourth quarter of the year

[12] Current Prices measures GDP/ inflation/asset prices using the actual prices we notice in the economy. Current prices make no adjustment for inflation.

Table 6.1 – ADF test for unit root

Variables	ADF Level	ADF First Difference
LGDP	-1.10548(1)	-7.66072(0)***
LEXP	-3.28665(2)	-6.29252(3)***
LIMP	-2.87331(4)	-6.20372(3)***
LHC	-1.45805(3)	-5.59492(2)***
LGFCF	-3.60257(2)	-4.01801(1)***
LPE	-1.92627(1)	-5.69958(1)***

Null hypotheses: Variable contains an unit root
Notes: (1) ***, ** and * denote significance at the 1%, 5%, and 10% levels, respectively.
(2) Figures in parentheses are the number of lags (delays) used.

the stationarity of time series. We implement the natural logarithm of our original time series and their first difference. Results are showed in Tab. 6.1. LGDP, LEXP, LIMP, LHC, LGFCF and LPE denote the natural logarithm of GDP, EXP, IMP, HC, GFCF and PE. From the table we can observe that test for natural logarithm of original data cannot reject the null hypothesis that variable contains an unit root so the original time series are not stationary. But the test result for the first difference of natural logarithm of all the series all reject the null hypothesis at 99% (i.e., the $***$ following the values as indicator), so they are stationary. We will only use the first difference of natural logarithm of GDP, EXP, IMP, HC, GFCF and PE in later analysis (hereinafter referred to as DLGDP, DLEXP, DLIMP, DLHC, DLGFCF and DLPE). In order to clearly define our economic model, we first define two vectors as follows:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \text{DLGDP} \\ \text{DLEXP} \\ \text{DLIMP} \end{bmatrix} \quad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \text{DLHC} \\ \text{DLGFCF} \\ \text{DLPE} \end{bmatrix} \tag{6.1}$$

where y is the output (endogenous variables) of our model, u is the input (exogenous variables) of our model. The selections of y and u are according to their economic attributes. Exogenous variables are the factors we could manipulate in economic system (e.g., increasing public expenditure). Endogenous variables are the consequences of the economic system which we could only observe but not directly interfere.

6.2.2 Selection of model order

Consider now a m-input-p-output system represented by a canonical input-output representation [57, 34], for i = 1,2, ... p:

$$\begin{aligned} y_i(k) = &\sum_{j=1}^{p} \sum_{q=1}^{n_{ij}} a_{ijq} y_j(k+q-n_i-1) \\ &+ \sum_{j=1}^{m} \sum_{q=1}^{n_i} b_{ijq} u_j(k+q-n_j-1) + e_i(k) \end{aligned} \tag{6.2}$$

where $y_i(k)$ denotes the value of output y_i at time point k, a_{ijq} and b_{ijq} are the coefficients of $y_i(k+q+n_i-1)$ and $u_i(k+q-n_j-1)$, p and m are the numbers of outputs and inputs, $e_i(k)$ is the white

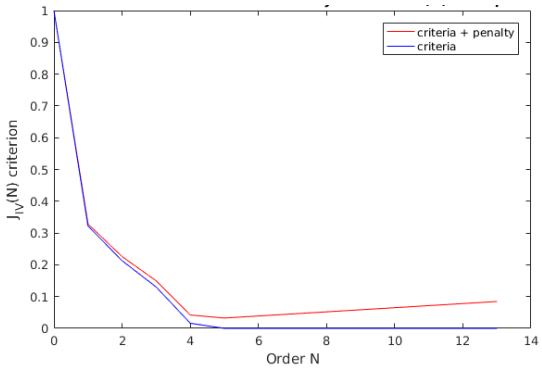
noise at time point k, n_i are the observability indices and n_{ij} are given by:

$$n_{ij} = \min\{n_i, n_j\}, \text{ if } i \leq j \tag{6.3}$$

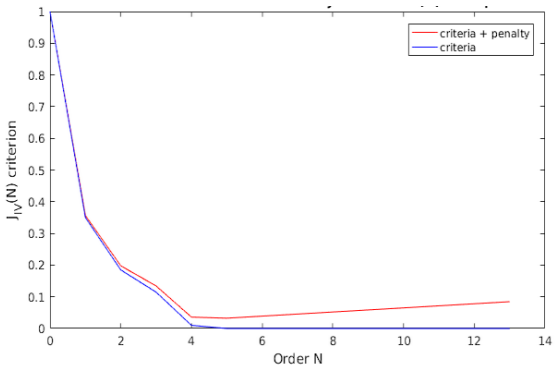
and

$$n_{ij} = \min\{n_i + 1, n_j\}, \text{ if } i > j \tag{6.4}$$

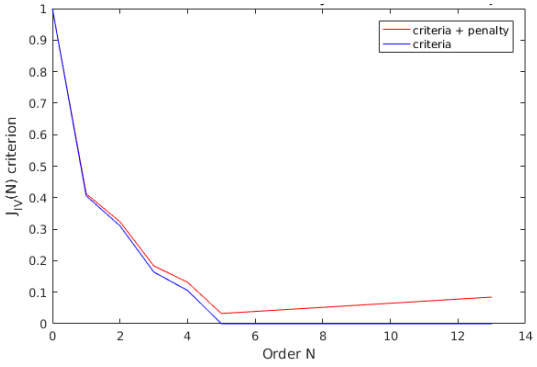
We apply this method for each output by using the technique of "instrumental variables" in [80], and implementing a criterion which penalizes the model complexity (ses [80, 35]). Fig. 6.2 shows the order selection process for y_1, y_2 and y_3. *Criteria* calculates the sum of estimation errors, as order increases, *criteria* decreases to 0. One of the objectives of system identification is to estimate models of reduced order, so [80] adds a term that penalizes the model complexity. [80] also provides Matlab scripts to implement these estimations.



(a) Result of y_1 (DLGDP)



(b) Result of y_2 (DLEXP)



(c) Result of y_3 (DLIMP)

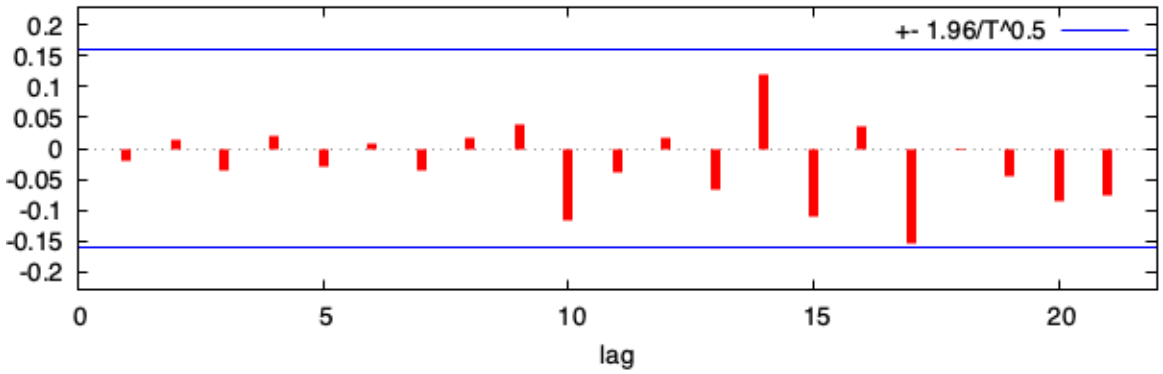
Figure 6.2 – Order Selection

As it can be seen from Fig. 6.2, the estimated model order for equation of y_1, y_2, y_3 are 5, 4, 5. One should notice that these estimated order are not definitive, we still need to pass the validation process to decide the order.

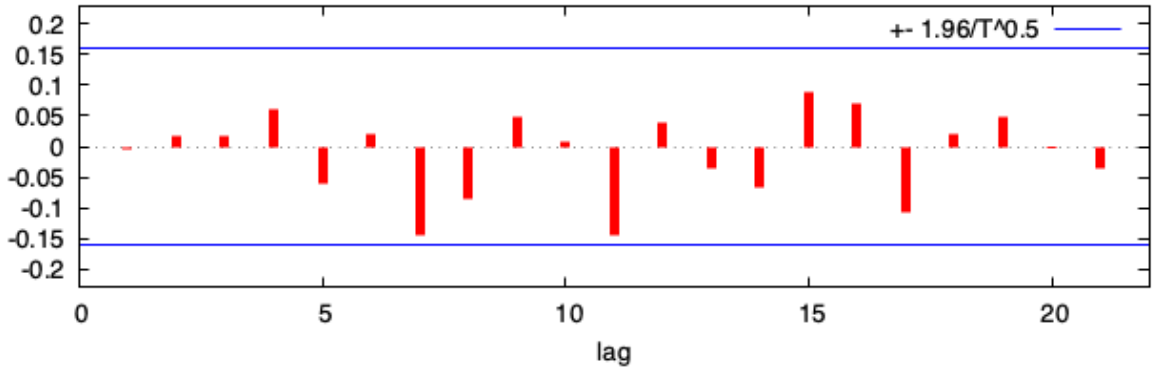
6.2.3 Estimation and Validation of parameters

After finding the model order, Least Squares method is used to estimate the parameters of the 3 equations for y1, y2 and y3. After each estimation, a whiteness test (autocorrelation test) will be applied to make sure the residuals from the estimated equation are white noise, which means the

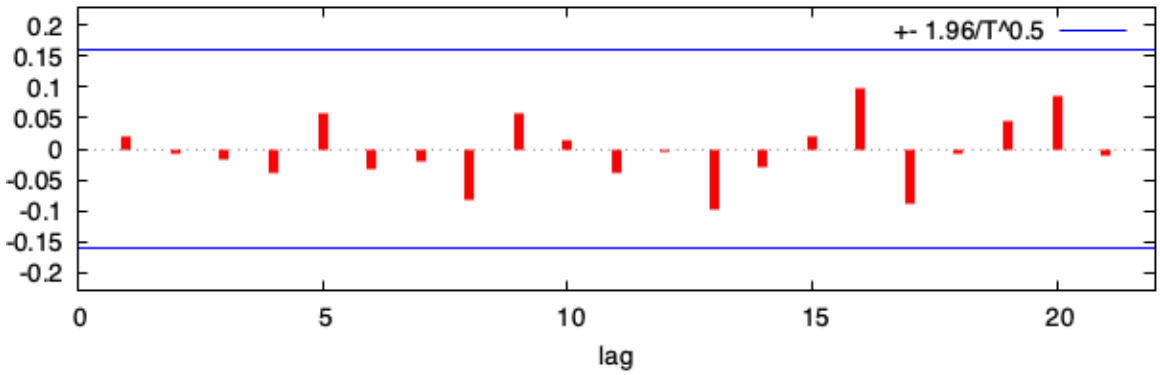
estimated model have already extracted all the knowledge from training data. To determine whether a time series is white noise, this problem is well studied in both economic [5] and automatic [80] domain. The algorithms of calculation vary a little bit through these studies, but the goal remains the same: testing if values are mutually uncorrelated. In this paper, we choose to use the default implementation of autocorrelation test function in Gretl [8]. Fig. 6.3 shows the final autocorrelation test for estimation residuals of y_1, y_2 and y_3. It states that if all the autocorrelation values should be in the range of limit $\pm\frac{1.96}{T^{0.5}}$ where T is the total number of data point in the dataset. In our case, we have 156 data points (From 1980Q1 to 2018Q4), so the limit is ± 0.157 here.



(a) y_1 Residuals



(b) y_2 Residuals



(c) y_3 Residuals

Figure 6.3 – Autocorrelation Test for Estimation Residuals

Once we find a valid model, we should continue to reduce the unimportant regressor (i.e. variable) whose coefficient is much smaller than others (comparison using the absolute value). Our objective

is to find the minimal model, which means a model with lowest order, least variables, but still valid. Every time we delete one regressor, we will re-estimate the parameters, re-do the whiteness test on residuals, in order to make sure the new model is still valid. We continue to do that until none of the regressor can be removed from the equation. We finally find the minimal model as follows:

$$\begin{aligned} y_1(k) = {} & a_{115}y_1(k-1) + a_{114}y_1(k-2) \\ & + a_{113}y_1(k-3) + a_{112}y_1(k-4) + a_{111}y_1(k-5) \\ & + a_{124}y_2(k-2) + a_{123}y_2(k-3) + a_{122}y_2(k-4) \\ & + a_{135}y_3(k-1) + a_{134}y_3(k-2) + a_{133}y_3(k-3) \\ & + b_{111}u_1(k-1) + b_{131}u_3(k-1) + e_1(k) \end{aligned} \tag{6.5}$$

$$\begin{aligned} y_2(k) = {} & a_{214}y_1(k-1) + a_{221}y_2(k-4) \\ & + a_{222}y_2(k-3) + a_{223}y_2(k-2) + a_{224}y_2(k-1) \\ & + a_{234}y_3(k-1) + b_{211}u_1(k-1) + b_{221}u_2(k-1) \\ & + e_2(k) \end{aligned} \tag{6.6}$$

$$\begin{aligned} y_3(k) = {} & a_{314}y_1(k-1) + a_{313}y_1(k-2) \\ & + a_{324}y_2(k-1) + a_{323}y_2(k-2) + a_{334}y_3(k-1) \\ & + a_{333}y_3(k-2) + a_{332}y_3(k-3) + a_{331}y_3(k-4) \\ & + b_{321}u_2(k-1) + e_3(k) \end{aligned} \tag{6.7}$$

The final orders of reduced model are 5,4,4 for y_1, y_2 and y_3. Concrete parameter values will be given in Sec. 6.2.4.

6.2.4 Transfer to Discrete-time Linear State-Space Model

We first define our state vector X according to Eq. (6.5) (6.6) and (6.7).

$$\begin{aligned} \mathrm{X(k)} = [& y_1(k-1)\ y_1(k-2)\ y_1(k-3)\ y_1(k-4)\ y_1(k-5) \\ & y_2(k-1)\ y_2(k-2)\ y_2(k-3)\ y_2(k-4) \\ & y_3(k-1)\ y_3(k-2)\ y_3(k-3)\ y_3(k-4)]^T \end{aligned} \tag{6.8}$$

Eq. (6.5) (6.6) (6.7) are written as a discrete-time state-space model in the following form:

$$\begin{aligned} \mathrm{X(k+1)} &= A \cdot \mathrm{X(k)} + B \cdot \mathrm{u(k)} \\ \mathrm{Y(k)} &= C \cdot \mathrm{X(k)} + D \cdot \mathrm{u(k)} \end{aligned} \tag{6.9}$$

where discrete time point $k \in \mathbb{Z}^+$, $A \in \mathbb{R}^{13\times13}$; $B \in \mathbb{R}^{13\times3}$; $C \in \mathbb{R}^{1\times13}$; $D \in \mathbb{R}^{1\times3}$. The values of A, B, C, D are showed in (6.10).

Eigenvalues of matrix A are checked, all its eigenvalues are within unit circle, which means the open-loop model is stable. Controllability and observability of the system are also tested, using the expressions of controllability and observability matrices showed in (4.37) and (4.38). If the rank of *observability* and *controllability* are equal to the number of states, we call the system controllable and observable, which is the case for our model.

$$A = \begin{bmatrix} 0.397 & 0.06 & -0.126 & 0.193 & 0.066 & 0.037 & 0.05 & 0 & 0 & 0 & 0 & -0.29 & -0.038 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.264 & 0.072 & -1.055 & 0.958 & 0 & 0.225 & 0.331 & 0.008 & -0.223 & 0.208 & -0.185 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1.0538 & 0.97 & -0.6 & 0 & 0 & 0.356 & 0.399 & 0 & 0 & 0.05 & -0.315 & 0.035 & -0.218 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.177 & 0 & 0.118 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.342 & 0 & 0.364 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.869 & 0.765 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} D = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{6.10}$$

6.3 Optimal Control Laws

The theory of optimal control is concerned with operating a dynamic system at minimum cost, one way of doing it is by using Linear–Quadratic Regulator (LQR). The introduction of LQR is already provided in Sec. 4.2.2, we will not repeat here again. In this section, we design our own reference signal for the system, and according to the character of our desired output, we develop our own system with modified LQR.

6.3.1 Reference Input

As we will focus on controlling GDP, then we need to implement our designed reference to let output reach the desirable value. Recall that y_1 in 6.1 is the first difference of natural logarithm of GDP, illustrated in Eq. (6.11).

$$y_1(k) = ln(\text{GDP(k)}) - ln(\text{GDP(k-1)}) = ln(\frac{\text{GDP(k)}}{\text{GDP(k-1)}}) \tag{6.11}$$

If we want our GDP to have constant p percent of increasing after each time point (i.e. $\frac{\text{GDP(k)}}{\text{GDP(k-1)}} = 1 + \frac{p}{100}$), then the quarterly GDP increasing ratio p can be interpreted as in (6.12):

$$p = (e^{y_1(k)} - 1) \times 100 \tag{6.12}$$

where e is the base of the natural logarithm. As we want a constant increasing ratio, we know that to keep p constant, y_1 needs to remain constant too.

6.3.2 Control system

Imagine when our system reaches steady state, our state and input vectors will also remain stable. We can define these optimal steady state tuple as (X_r , u_r) and desired output is defined as Y_r. When our system reaches steady state, X_r, Y_r and u_r satisfy following relations according to Eq. (6.9):

$$\begin{aligned} X_r(k) &= A \cdot X_r(k) + B \cdot u_r(k) \\ Y_r(k) &= C \cdot X_r(k) + D \cdot u_r(k) \end{aligned} \tag{6.13}$$

We see from Fig. 4.4 that, different from general control method (e.g. PID), the $Reference$ in LQR does not directly apply feedback on output, but react on feedback control law $-K \cdot X$. If there is no $Reference$, the steady state of output will be 0. To let the difference between $X(k)$ and $X_r(k)$ minimize to 0, we make a change on Eq. (6.9).

$$\begin{aligned} X(k) - X_r(k) &= A \cdot X(k) + B \cdot u(k) - X_r(k) \\ &= A \cdot X(k) + B \cdot u(k) - A \cdot X_r(k) - B \cdot u_r(k) \\ &= A \cdot (X(k) - X_r(k)) + B \cdot (u(k) - u_r(k)) \end{aligned} \tag{6.14}$$

if we define $\Delta X(k) = X(k) - X_r(k)$ and $\Delta u(k) = u(k) - u_r(k)$, then we have a new linear system:

$$\Delta X(k) = A \cdot (\Delta X(k)) + B \cdot (\Delta u(k)) \tag{6.15}$$

re-write the cost function of LQR (4.39) as:

$$\begin{aligned} J_r = \sum_{k=0}^{N-1} &((\Delta X(k))^T Q(\Delta X(k)) + (\Delta u(k))^T R(\Delta u(k)) \\ &+2(\Delta X(k))^T N(\Delta u(k))) + (\Delta X(N))^T Q(\Delta X(N)) \end{aligned} \tag{6.16}$$

feedback control law that minimizes the J_r can be written as:

$$\Delta u(k) = -K \cdot \Delta X(k) \tag{6.17}$$

and $K = (R + BP^T B)^T (B^T PA + N^T)$ is independent from state and input vectors, so the K in (4.40) and (6.17) does not change. The control law (6.17) can also be written as:

$$u(k) = -K \cdot (X(k) - X_r(k)) + u_r(k) \tag{6.18}$$

According to Eq. (6.18), the newly designed LQR control system is showed in Fig. 6.4. As we can not directly apply designed output into the feedback, we will need a pre-processing function to transfer the desired output Y_r to the desired state vector X_r and input vector u_r when the system reaches the steady state. One thing to notice that there will not be only one pair of X_r and u_r to satisfy the pre-processing function condition, it will be a range for both value, we will let experts to choose the values which make more sense in real world.

Recall the cost function of LQR: $J = X(k)^T QX(k) + \sum_{k=0}^{N-1} (X(k)^T QX(k) + u(k)^T Ru(k) + 2X(k)^T Nu(k))$. We first set N to 0. And in many cases, it is not the states X which are to be minimized, but the output variable Y. In this case, we set the weight matrix $Q = C^T Q' C$, since $Y(k) = C \cdot X(k)$, and the auxiliary matrix Q' weights the plant output [132].

When $R >> C^T Q' C$, the cost function is dominated by the control effort u, and so the controller minimizes the control action itself, this control strategy is used when the control signal is constrained. When $R << C^T Q' C$, the cost function is dominated by the output Y, and there is less penalty for using large u.

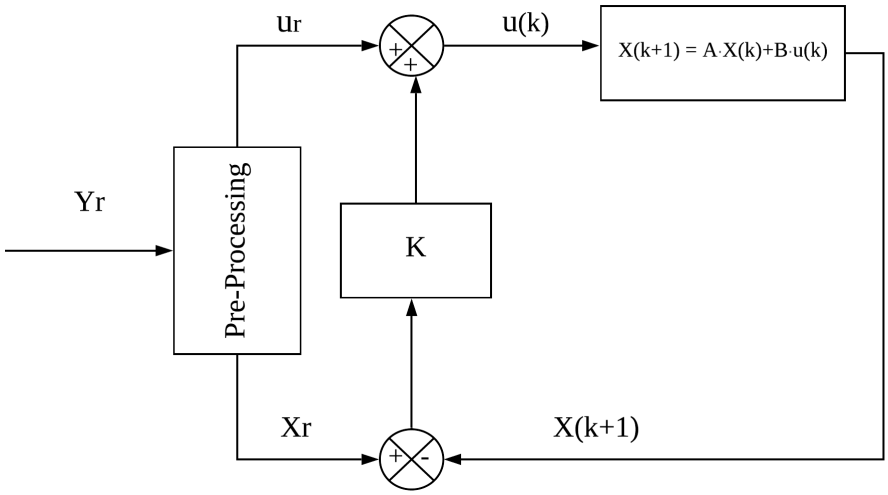


Figure 6.4 – LQR Control system.

6.4 Control Laws Evaluation

6.4.1 Experiment Setting

Reference Signals Setting According to the reality, a yearly GDP growth ratio of 3.2% is interesting to study[13]. To reach this level, the quarterly GDP increasing ratio p is around 0.8% (i.e. $(1.008)^4 \approx 1.032$). From (6.12) we know,

$$y_{1r} = ln(1 + \frac{p}{100}) = ln(1.008) = 0.007968 \approx 0.008$$

so $y_{1r} = 0.008$, in the same calculation process, we set $y_{2r} = 0.0175$ and $y_{3r} = 0.0086$, and y_{1r}, y_{1r} and y_{1r} compose the vector Y_r.

Recall the relations between Y_r, X_r and u_r in (6.13), in our experiment, since the output is identical to the first entry of the state vector, the matrix D will be 0 in our case. These equations can be re-written as:

$$\begin{aligned} X_r(k) &= (I - A)^{-1} \cdot B \cdot u_r(k) \\ Y_r(k) &= C \cdot X_r(k) \end{aligned} \tag{6.19}$$

where I is the identity matrix. As we explained in the end of Sec. 6.3.2, there is not only one pair of X_r and u_r that satisfy (6.19). After our selection, one reasonable pair of X_r and u_r is:

$$\begin{aligned} X_r(k) = [&0.008,0.008,0.008,0.008,0.008,0.0173,0.0173, \\ &0.0173,0.0173,0.0086,0.0086,0.0086,0.0086]^T \end{aligned}$$

$$u_r(k) = [0.024, 0.003, 0.003]^T$$

[13] Actually, the recent 10 years (2010-2019) average GDP growth ratio of France is 1.38%, but if we look back 25 years ago, the highest GDP growth ratio are showing during 1998-2001, which the average ratio is around 3.2%. Therefore we want to study what measurements should be implemented to sustain this growth ratio.

LQR Parameters Setting Knowing about the constraint of u, diagonal weights [100] of Q and R are used.

$$Q = \begin{bmatrix} q_1 & & \\ & \ddots & \\ & & q_{n_q} \end{bmatrix} R = \rho \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_{n_r} \end{bmatrix} \tag{6.20}$$

where $n_q = rank(A) = 13$, $n_r = rank(B) = 3$. For the sake of simplicity, we will let $q_i = 1$ for all $i \in [1,13]$, and $r_j = 1$ for all $j \in [1,3]$, we will use ρ to adjust the input/state balance. We choose three ρ ($\rho \in [1,10,100]$) to implement the experiments to compare the converging speech and observe the input signal range.

6.4.2 Experimental Evaluation

We implement the system of Fig. 6.4 in Simulink. The initial state vector of state-space model is set by the real french data from 1980Q1 to 1981Q1.

$$X_0 = [0.001,0.012,-0.008,-0.004,-0.004,0.036,$$
$$0.032,0.013,-0.03,0.007,0.017,0.006,-0.002]^T$$

6.4.2.1 Variation of ρ without Constraints on Input Signals

Our simulation goes through 50 time point (i.e. quarters). By setting ρ to 1, 10 and 100, the traces of output (i.e. y_1) is illustrated in Fig. 6.5. A more detailed result is showed in Tab. 6.2. For current experiment, we ignore the result of "1 with constraint" for now. From the results, we can conclude that when ρ is smaller, we can converge faster to steady state. From Fig. 6.5, we can see a trough at 4^{th} quarter for all curves, the lower this trough, the lower the GDP increasing ratio in this quarter.

Table 6.2 – Summary of Output

ρ	**time to steady state (unit in quarter)**	**output range**
1	18	(0.001, 0.008)
10	32	(-0.001, 0.008)
100	37	(-0.002, 0.008)
1 with constraint	20	(0.001, 0.008)

But the benefits of $\rho = 1$ comes with cost. Fig. 6.7, 6.8 and 6.9 shows the input signals evolution during the control. One thing to note is that the curves of $\rho = 1$, *1 with perturbation* and *1 with perturbation and constraint* are overlapped during the quarter from 0 to 25. And the curves of $\rho = 1$, 10, 100 and *1 with constraint* are overlapped during the quarter from 25 to 50. The signal ranges are showed in Tab. 6.3. From the figures and table, we can see that when $\rho = 1$, the input signal ranges are much larger than other two comparisons, and the maximum values are also always higher.

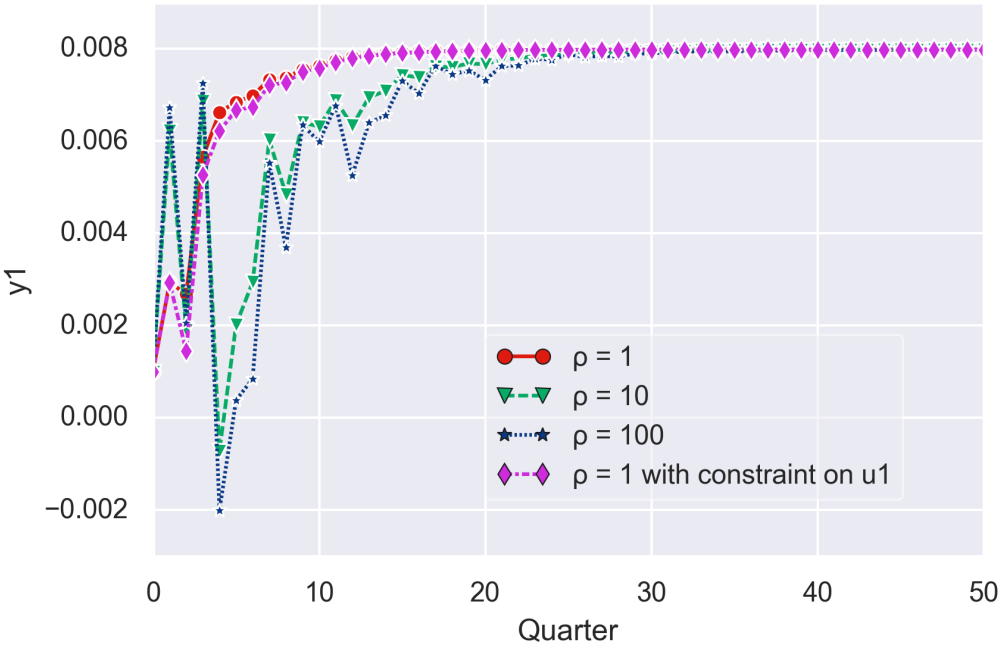


Figure 6.5 – Output $y1$ under different ρ values.

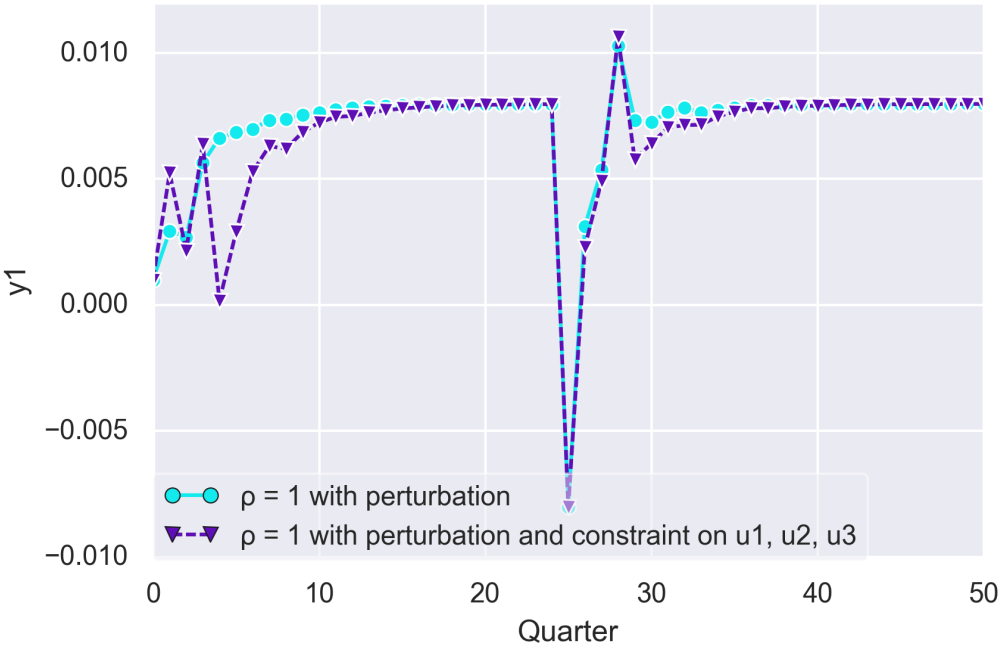


Figure 6.6 – Output $y1$ under different ρ values.

6.4.2.2 Constraints on Input Signals

In our first experiments, we do not impose constraints on input signals, but in reality, there are some levels that input signals cannot reach. Therefore in this experiment, we fix $\rho = 1$ and set a maximum limit 0.03 on signal u_1: $u_1 \leq \mathbf{0.3}$ (assuming we know the upper range of signal u_1 under $\rho = 1$ without constraint experiment is higher than 0.03). Input signals results of "$\rho = 1$ with constraint on u_1" are

showed in Fig. 6.7, 6.8 and 6.9 and Tab. 6.3. Comparing to only $\rho = 1$ result, we can see for signal u_1, the minimum value of the range is still 0.009, but the maximum value of the range becomes 0.03 (which is the limit) instead of 0.038. For u_2 and u_3, the maximum value of new curves becomes higher, being used to compensate the insufficiency of u_1.

From Fig. 6.5, 6.7, 6.8 and 6.9, we can observe that even u_2 and u_3 react to compensate limitation of u_1, at 1^{st} and 4^{th} quarter, new results are slightly lower than the result without constraint. As for the converging speed, from Tab. 6.2, we can also notice that experiment with constraint converges slower than the experiment without constraint. But still better than the results of $\rho = 10$ or 100.

6.4.2.3 Perturbation on Output Signals

In this experiment, we still keep $\rho = 1$, and we add perturbation on output signals to simulate economic crisis, to see how the system will react. From the reference signals setting in Sec. 6.4.1, we know that when the system is stable, y_1, y_2, y_3 should equal to 0.008, 0.0173 and 0.0086. And u_1, u_2, u_3 should equal 0.024, 0.003 and 0.003. From Fig. 6.6, 6.7, 6.8 and 6.9, we can see that at 24th quarter, the system has converged to a stable state. **Then we add a negative perturbation pulse signal -0.16 on y_1, y_2 and y_3 at 25th quarter.** The $\rho = 1$ *with perturbation and constraint on* u_i curves are the scenario where we not only implement the perturbation, but also implement constraints on all the input signals, for u_1 limited between 0 and 0.03, u_2, u_3 are limited between 0 and 0.008: $u_1 \in$ **[0, 0.3]**, $u_2 \leq$ **[0, 0.008]**, $u_3 \leq$ **[0, 0.008]**. $\rho = 1$ *with perturbation* experiment implements the perturbation, but has no constraints on input signals.

Fig. 6.6 shows that after about 20 quarters after 25th quarter, the two systems totally recover from the perturbation, apparently the curve without constraint recover faster than the other. From Fig. 6.7 6.8 and 6.9, we can observe that the amplitude of all inputs for the experiment of $\rho = 1$ *with perturbation* are higher than $\rho = 1$ *with perturbation and constraint on* u_i. The input signal ranges showed in Tab. 6.3 also confirm that. One interesting point in Fig. 6.8 and 6.9 reveal that if we do not impose constraint on input signals, inputs u_2 and u_3 can be negative, recall that u_2 represents first different of logarithm of Gross Fixed Capital Formation (also called investment). A negative signal means instead of investing during the crisis, we should sell our assets. As u_3 represents first different of logarithm of Public Expenditure, negative means we need to reduce government spending. Nevertheless, all these conclusions, although correct from the engineering point of view, need to be coordinated with expert's advice.

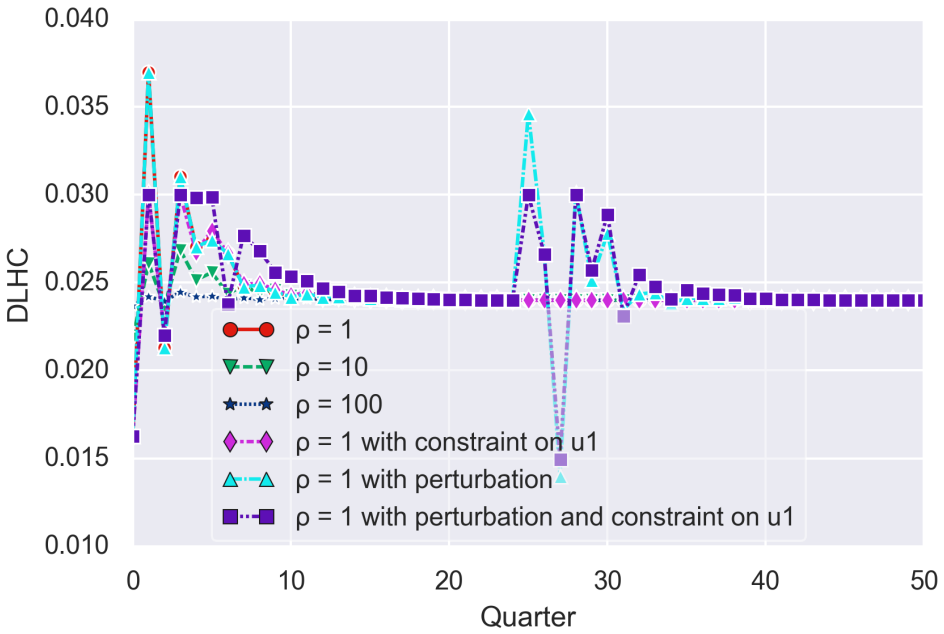


Figure 6.7 – Trace of u_1 (DLHC) under different ρ values.

Table 6.3 – Summary of Input

ρ	**input signal**	**input signal range**
1	u_1	(0.0163, 0.0375)
10	u_1	(0.0218, 0.0274)
100	u_1	(0.0237, 0.0245)
1 with constraint	u_1	(0.0163, 0.03)
1 with perturbation	u_1	(0.0146, 0.0375)
1 with perturbation and constraint	u_1	(0.0153, 0.03)
1	u_2	(-0.0095, 0.003)
10	u_2	(-0.0047, 0.0034)
100	u_2	(0.0017, 0.0032)
1 with constraint	u_2	(-0.0095, 0.0031)
1 with perturbation	u_2	(-0.0095, 0.0087)
1 with perturbation and constraint	u_2	(0.0, 0.0085)
1	u_3	(-0.0196, 0.0068)
10	u_3	(-0.0065, 0.0051)
100	u_3	(0.0014, 0.0034)
1 with constraint	u_3	(-0.0196, 0.008)
1 with perturbation	u_3	(-0.0196, 0.0103)
1 with perturbation and constraint	u_3	(0.0, 0.0099)

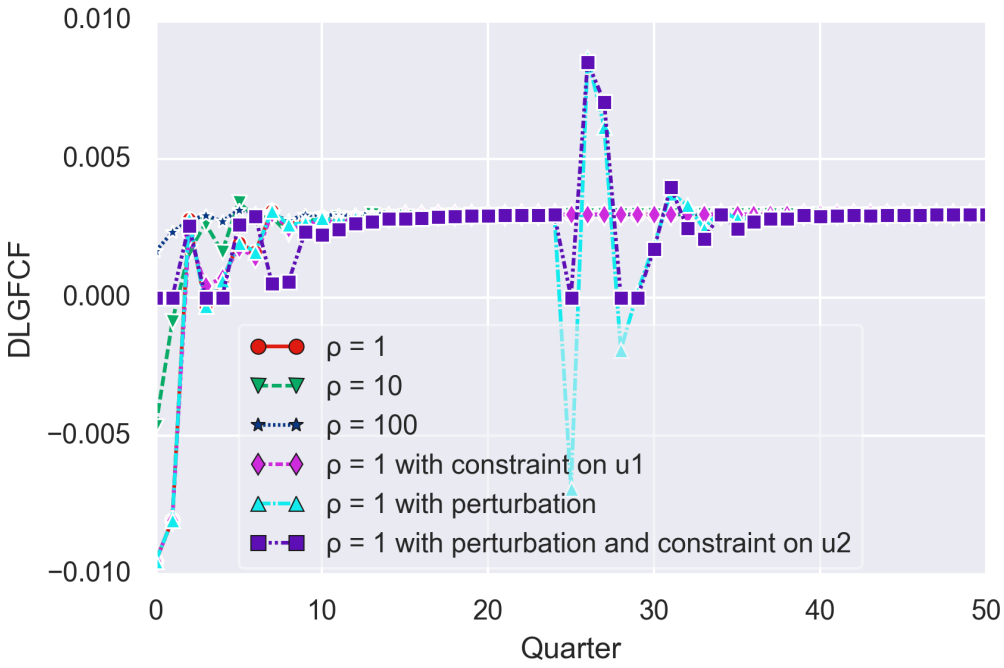


Figure 6.8 – Trace of u_2 (DLGFCF) under different ρ values.

6.5 Conclusion

Applying control theory in economic problem has been successfully studied in many cases, resource allocation is one of the well established problems in this area, it demands to dynamically choose available resources with constraints over time to maximize or minimize an objective function.

In this chapter, to apply the above theory, French macroeconomic quarterly data from 1980Q1 to 2018Q4 are used, it consists of 6 variables: GDP (Gross Domestic Production), EXP (Exportation), IMP (Importation), HC (Household Consumption), GFCF (Gross Fixed Captial Formation) and PE (public expenditure). Three canonical input-output models are estimated, which first difference of natural logarithm of GDP, EXP and IMP are endogenous variables (outputs), and first difference of natural logarithm of HC, GFCF and PE are exogenous variables (inputs). We first estimate the order of each equation, then the parameters. Once the residuals of models pass the whiteness test, we transfer the models to a state space model.

After estimating the model, optimal control solution: (Linear–Quadratic regulator) LQR is designed for our problem which is to maintain a constant GDP increasing ratio at certain level. The algorithm is developed on Simulink. The experiments with or without constraints on input signals are both implemented: 1) The results without constraints on input signals show that, a lower parameter ρ in LQR will lead to a faster convergence to steady state; 2) For experiment with constraints on inputs, furthermore, we impose a limitation on the maximum value that first difference of natural logarithm of HC can reach. The result shows that when one input cannot reach the level it used to reach, other inputs will compensate this absence. This action will force other input signals to reach a even higher level, and the final result of output is still slightly worse than the experiment without constraint as we expected.

Perturbations on outputs are also studied, the experiments are also developed with and without constraints on input signals. Results show that our system can quickly recover from the disturbance. And constraints on input signals can delay the recovery.

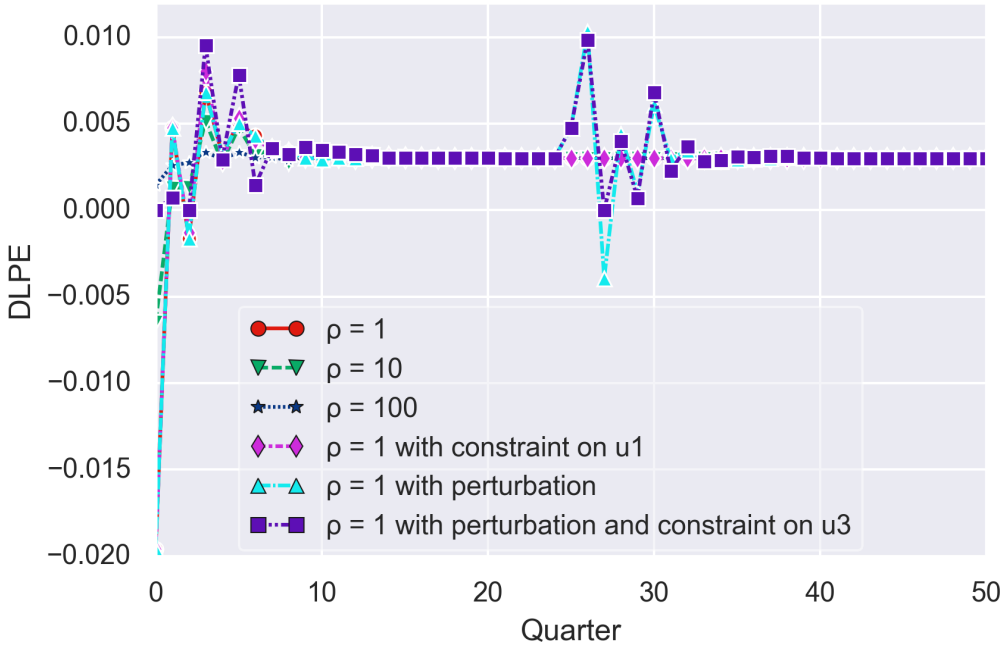


Figure 6.9 – Trace of u_3 (DLPE) under different ρ values.

The control structure designed in this chapter has a good applicability and extensibility in economic. This work can be further extended with more variables, the constraints are also not limited to only the range of input signals, it can impose the constraints on inputs and outputs subject to a specific model of the dynamics of the macroeconomy.

Chapter 7

Conclusions on Economic Model Control

In this part, two studies are performed, one is based on China's macroeconomic data, another is based on the macroeconomic data of France.

The first study is carried out only with economic methods, it first use ADF test to find the order of integration of all time series, then use different criterion (e.g. AIC) to estimate the order of VAR model. It performed the Johansen cointegration test to find the long-term relation between the time series, and finally estimated the VECM model and showed the pairwise and multivariable Granger Causlity results. The main contribution of this study is on the conclusion it draws. Many studies which based on the China data before 1980s or 1990s, all show that there are clear two-way Granger Causality between GDP and exportation. As we extend the time series with recent years data, and include new variables in the study. Our results show that even exportation is still a big part of China GDP, the two-way Granger Causality does not exist any more, but domestic consumption clearly shows the two-way Granger Causality with GDP. Consumption will thus play an important role in the China's GDP growth in the next years.

The second study comprises the system identification part and optimal control part. Even we changed the data used in this study, the system identification step is similar as in first study, but we introduced a new technique to estimate the model order, and we also added a step of model validation during system identification. To perform the optimal control, LQR is chosen and designed. Except regular experiements, we also included constraints on inputs, perturbations on outputs. Results show that our system can always converge to the desired level, it can also quickly recover from the disturbance. And constraints on input signals can delay the recovery.

From the above studies, we can see that there is no doubt that we can implement the control on economic models. And our experiments can have very good political implications. For instance in our second study, we introduce a perturbation on outputs (GDP, Importation and Exportation), it simulates a sudden crisis, and we can observe and control the inputs to make the outputs re-converge to its stable state. During this Covid-19 period, which makes this experiment even more meaningful.

Part III

Control Theory for On-line Training of Neural Networks

When we implement system identification, we observe that all the methods are model-driven approaches, which means that we need to pre-define the structure of the economic model before estimating the parameters. Data-Driven is a new way of thinking, enabled by machine learning. The main idea is to find an algorithm that can spot connections and correlations that you may not even suspect. Deep neural networks (DNNs) is part of a broader family of machine learning methods, we will address the training process of DNNs in this part.

This part tackles the issue of dynamically adapting learning rate in order to train a neural network. Learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. When the learning rate is too large, gradient descent can inadvertently increase rather than decrease the training error. When the learning rate is too small, training is not only slower, but may become permanently stuck with a high training error. Comparing to the state of the art algorithm which are mainly focusing on time-based learning rate or adaptive gradient methods, our algorithms are performance-based. The algorithm we introduce can adjust its learning rate based on the decreasing of loss value. After introducing event-based control into the algorithm, the algorithm not even outperforms all the state of the art learning rate algorithm under on-line learning scenario, it can also largely reduce the training time to result in the same accuracy.

As we bring in feedback control system into the training of neural network, Chapter. 8 introduces the plant - Convolutional Neural Network, the control signal - learning rate and the error - loss value. It also presents the setting of on-line learning, the performance metrics using to compare between the algorithms, and the state of the art learning rate algorithms. Chapter. 9 presents the algorithm of Exponential / Proportional-Derivative (E/PD) Control. It is a performance-based learning rate algorithm which periodically update the learning rate based on the changes of loss value. In Chapter. 10, continuing on the E/PD Control, we incorporate two event-based control: (1) Event-Based Learning Rate and (2) Event-Based Learning Epochs into E/PD. Event-Based Learning Rate is implemented to prevent sudden drop of the learning rate when the model is approaching the optimum; Event-Based Learning Epochs decides, based on the learning speed, when to switch to the next data batch. Eventually, Chapter. 11 concludes on the contributions of those works regarding learning rate control, and gives perspectives for future works.

Chapter 8

Neural Network and Learning Rate: Background and Related Works

The learning rate is probably one of the most important hyperparameter to take into account when training a machine learning model. Each time the model weight is updated, learning rate controls how much the model changes based on the estimated prediction error. Choosing optimal values of learning rate is challenging, because a value that is too small may cause a long training process, while a value that is too large may result in updating weights too fast and make the training process unstable. No matter the learning rate is too big or too small, it makes model converge to sub-optimal values of the weights.

Our objective is to use control theory to dynamically adjust learning rate during the training process, in order to make the controlled system converge faster. Our experiments are based on on-line learning scenario, and compared with a bunch of state of the art learning rate optimizers. Therefore, in the following part, we first introduce the model we would control, which is a Convolutional Neural Network as well as its performance metrics. Second we give details about the on-line learning setting. In the end, state of the art learning rate algorithm will be presented.

8.1 Convolutional Neural Network

Convolutional Neural Network (CNN) [77] is a kind of deep neural network made to process high-dimensional input data (such as images, website analysis for advertising, finance analysis), it takes spatial structure of data into account, hierarchically learning the features from low to high-level patterns. CNN is the state-of-the-art learning mechanism for image classification and due to its performance it has become a default method of choice to solve this kind of problem among many other techniques such as support vector machines or multilayer perceptron.

A CNN (depicted for example in Fig. 8.1) consists of an input and an output layer, as well as multiple hidden layers, which typically consist of a series of convolutional layers. The activation function is commonly a ReLU layer (rectified linear unit applied to extract linear and non-linear relations in the data) while the final convolution often involves back-propagation in order to more accurately weight the result. Each neuron takes the input values coming from the previous layer and computes an intermediate output value by applying a specific function determined by a vector of weights and, eventually, a bias. The objective of the learning phase is to make iterative adjustments to these biases and weights to better fit the data. These weights in the CNN are usually updated using Stochastic Gradient Descent (SGD) techniques, which are very similar to the algorithm of steepest descent classically used in control [80].

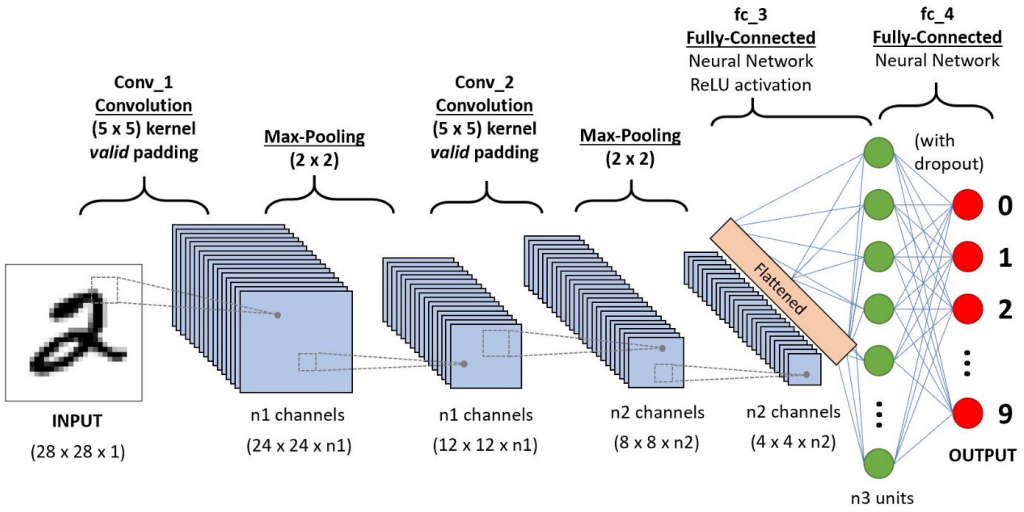


Figure 8.1 – Structure of Convolution Neural Network.

The equation below shows how SGD updates the weights of the CNN:

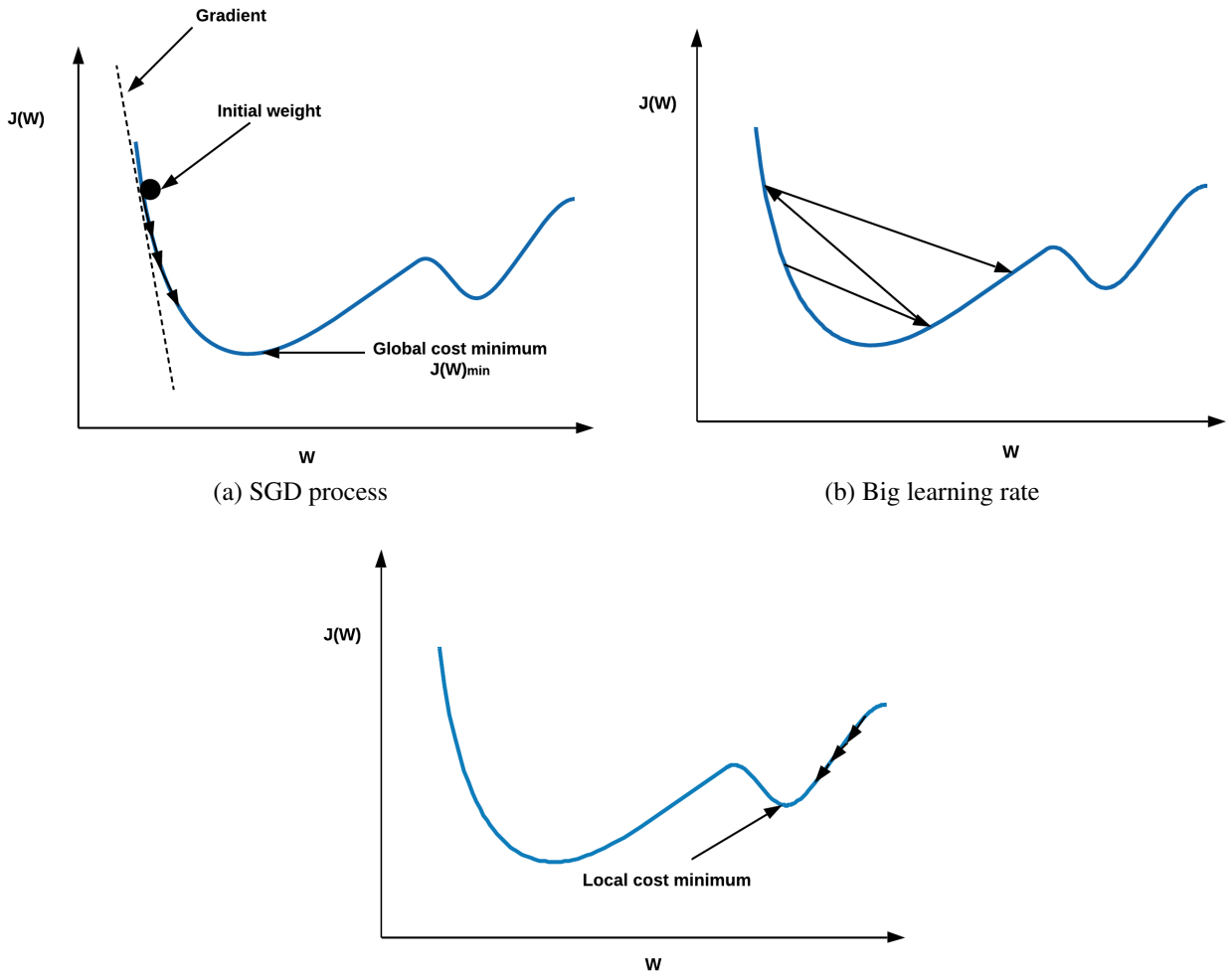
$$W_i = W_{i-1} - \eta \frac{\partial J}{\partial W}$$

where vector W_i represents the weights vector computed at i^{th} discrete time instant, η is positive and denotes the learning rate, J is the loss function (or cost function) which can be seen as an error. As we are always trying to minimize the loss function, we suppose that there exists an optimal solution of parameters W_i^*. To update the parameters from learning, back-propagation calculates gradient for each layer, we use the product of the gradient and a learning rate to update the parameters. Fig. 8.2a sketches the process of the SGD finds the global minimum (also called optimum). Training a CNN is solving a global optimization problem with random initial parameters and a varying learning rate. If we are far from optimum, we should have a large learning rate to quickly approaching it and when we are close to the optimum, we want to have a small step each time in order to not miss it. Gradient indicates the direction, while learning rate scales the step. If we constantly keep a big learning rate, we will fall in the the situation of Fig. 8.2b, where we may never reach to the optimum. If we constantly keep a small learning rate, we risk to stop at a sub-optimal situation as showed in Fig. 8.2c.

8.2 On-line Learning

In computer science, on-line is a method of machine learning in which data becomes available in a sequential order and is used to update the prediction.In opposition to that entire batch learning techniques which generate the best predictor by learning on the entire training data set at once. On-line learning is a common technique used in areas of machine learning where: (1) it is computationally infeasible to train over the entire dataset due to the capacity of memory of computer; (2) whole dataset is not totally available before training process start; (3) data distribution changes over time.

To emulate on-line learning scenario, we assume that there is no variation of data distribution between different data batches. We consider a dataset $\mathscr{T}$ with a total number of training instances T, each one belonging to a class $c : \mathbb{Z}^+ \rightarrow [1,C]$. The whole dataset is composed of B subsets (i.e. batches), T_i is the i^{th} batch where $i : \mathbb{Z}^+ \rightarrow [1,B]$. Each batch equally contains S data instances and will be used to train the model for N epochs (i.e. N times). At the reception of a new batch, the learning rate algorithm is reset with initial values. Classical simulated on-line learning scenario is illustrated in Fig. 8.3.



(a) SGD process

(b) Big learning rate

(c) Small learning rate

Figure 8.2 – SGD Process and Influence of Learning Rate.

8.3 Performance metrics

There exists many metrics to evaluate the performance of a CNN model (see for example [85]), we used two of the most classical: classification accuracy as one of the most used classification metric, and loss value.

For evaluating, machine learning researchers typically prepare a testing dataset which will not be used during the training process. At the end of each training phase (called from now on epoch), the testing dataset is used to evaluate the model by measuring the classification accuracy and the loss value. Accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}} \tag{8.1}$$

The loss L is defined as the difference between the predicted value by the model and the true value. The most common definition of L used for classification problems is cross-entropy [118]:

$$L = -\frac{1}{V}\sum_{p=1}^{V}\sum_{q=1}^{C} y_{p,q}\log(\hat{y}_{p,q}) + (1-y_{p,q})\log(1-\hat{y}_{p,q}) \tag{8.2}$$

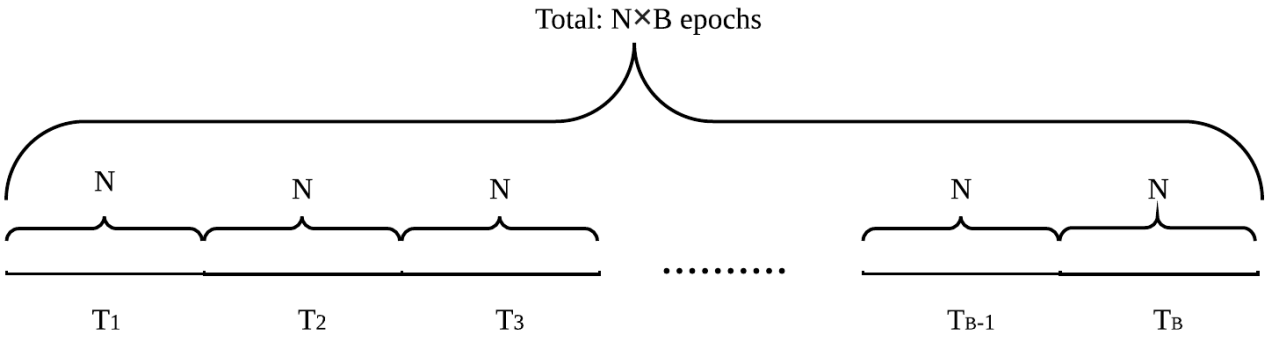


Figure 8.3 – On-line Learning Scenario.

where V is the size of testing dataset and C is the total number of classes and also the length of the prediction vector which is a probability vector (sum of all the bits is 1). $\hat{y}_{p,q}$ denotes the q^{th} bit value of prediction vector for data sample p while $y_{p,q}$ is the ground truth, indicating if data p belongs to class q ($y_{p,q} = 1$) or not ($y_{p,q} = 0$).

In general, loss and accuracy are two negative correlated metrics, and we could expect to have a higher accuracy if we have a lower loss. Nevertheless we could also see the case where we have the same accuracy and different loss values since the distance between prediction and ground truth (i.e. loss) is a continuous variable while accuracy is a true or false value for a particular sample. For instance, consider a binary (0, 1) classification, where we set a threshold at 0.5 which means prediction between 0.5 and 1 will be predicted as class 1, between 0 and 0.5 will be predicted as class 0. For a test data whose true class is 1, if output from Model A is 0.51, and output from Model B is 0.99, then the predicted class from model A and B are all class 1. Obviously, 0.99 is closer than 0.51 to 1, so Model B has a lower loss. In this case, we conclude that model B is more stable than model A even though they have the same accuracy.

Final loss and final accuracy reveal the performance of the final model that we use. Nevertheless, stability metrics are also important, if the accuracy curve experiences a big variance near the end of training process, even though we could have a good final result, we could not assure that we always get this result. Thus in our evaluation, we include standard deviation of the accuracy during the last 10% training epochs [95]. Convergence speed of accuracy is another metrics to evaluate the performance, as we focus on on-line learning scenario, so the interval between two batch data can be short. With a limited time, a faster accuracy convergence could lead to a better model performance comparing to other algorithms. Therefore, we will report the first epoch when the experiment reaches its 95% final accuracy as a metric, for the same final accuracy the smaller this epoch the faster the convergence speed. To evaluate convergence speed, another metric could also be the first epoch when the experiment reaches a fixed accuracy. The disadvantage of reporting the first epoch of reaching its own 95% final accuracy is that if an experiment's final accuracy is very low, then the first epoch to reach its own 95% final accuracy will be very small. The difficulty of reporting the first epoch of reaching a fixed accuracy is to choose this certain threshold of accuracy. For some experiments, the algorithm may not reach this accuracy.

8.4 Related Work of Learning Rate Optimizer

8.4.1 Time-Based Learning Rate Strategy

Time-Based Decay is one kind of time-based strategy. The theory behind is simple: in the beginning, as our initial weights are far from optimum point, we should use large learning rate to quickly approach the optimum. As training process goes on, we will get closer to optimum, then learning rate should be smaller, so that we does not skip over the optimum. In this learning rate strategy, η (learning rate) decreases following a predefined decay function. The only differences between these strategies is that for some of them, the learning rate decreases slowly in the beginning and faster in the end. And for others are the opposite. However, some paper suggests that adding some variations during the decay can improve the performance. For instance [2] introduced a time decreasing law with small sine oscillations.

The monotonous decrease of the learning rate is a well-established use that has rarely been questioned. However, Smith [123] presents promising results with a cyclical learning rate law of triangular shape, where two boundaries are defined and η cyclically varies between them.Indeed, we advocate that a brief increase in the learning rate could enable both to reach faster the global minimum and avoid being blocked in a local one.

The time-based learning rate algorithms can be summarized as above. The disadvantage of these algorithms is that the learning rate path is fixed before training, it cannot be adjusted when necessary. The experiments with time-based learning rate are illustrated in Sec. 9.4.4.

8.4.2 Adaptive Learning Rate Strategy

Before introducing adaptive learning rate algorithms, we need first introduce two concepts: **Momentum** [108] and **Nesterov accelerated gradient** [102]. The Momentum method is a method to accelerate learning using SGD in the relevant direction showed in Fig. 8.4. In particular SGD suffers in the following scenarios: (1) Error surface has high curvature, (2) Small but consistent gradients and (3) Noisy gradients. It does this by adding a fraction γ of the update vector of the past time step

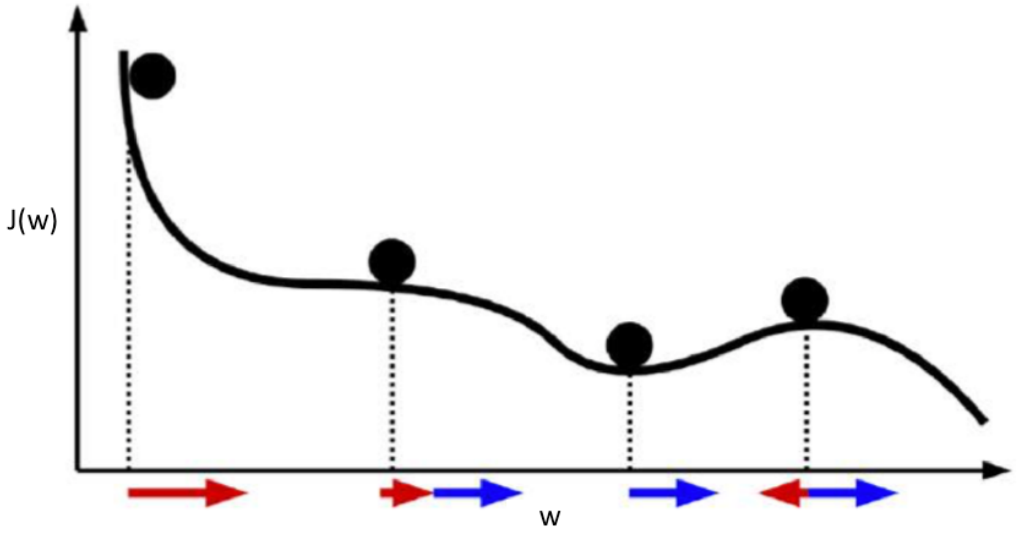


Figure 8.4 – Momentum: red arrow represents the direction of gradient, blue arrow represents the direction of momentum. (Source: A. Zhang's presentation on SASPS [141])

to the current update vector:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \frac{\partial J}{\partial W} \\ W_{t+1} &= W_t - v_t \end{aligned} \tag{8.3}$$

The momentum term γ is usually set to 0.9 [43] or a similar value, and η is the learning rate that user needs to choose.

Nesterov accelerated gradient (NAG) is a way to give our momentum term this kind of prescience. We know that we will use our momentum term γv_{t-1} to move the parameters W. Computing $W - \gamma v_{t-1}$ thus gives us an approximation of the next position of the parameters (the gradient is missing for the full update), a rough idea where our parameters are going to be. We can now effectively look ahead by calculating the gradient not with respect to our current parameters but with respect to the approximate future position of our parameters:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \frac{\partial J}{\partial W}(W_t - \gamma v_{t-1}) \\ W_{t+1} &= W_t - v_t \end{aligned} \tag{8.4}$$

We choose to set the momentum term γ to a value of 0.9. While Momentum first computes the current gradient (small blue vector in Fig. 8.5) and then takes a big jump in the direction of the updated accumulated gradient (big blue vector), NAG first makes a big jump in the direction of the previous accumulated gradient (brown vector), measures the gradient and then makes a correction (red vector), which results in the complete NAG update (green vector). This anticipatory update prevents us from going too fast and results in increased responsiveness [119], which has significantly increased the performance of recurrent neural network (RNN) on a number of tasks [13].

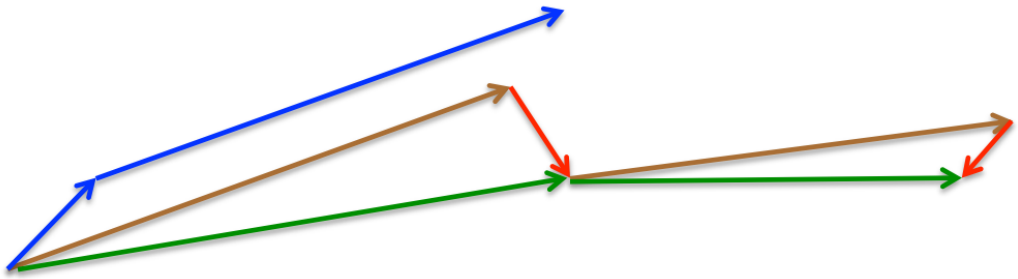


Figure 8.5 – Nesterov Update Vector. (Source: G. Hinton's Coursera lecture 6c [131])

- **RMSProp** is an unpublished adaptive learning rate method proposed by Geoffrey Hinton in Lecture 6e of his Coursera Class. It keep the moving average of the squared gradients for each weight. And then divide the gradient by square root the mean square. This is why it is called RMSProp (Root Mean Square Propagation).

$$\begin{aligned} E[g^2]_t &= \beta E[g^2]_{t-1} + (1-\beta) g_t^2 \\ W_{t+1} &= W_t - \frac{\eta}{\sqrt{E[g^2]_t}} g_t \end{aligned} \tag{8.5}$$

RMSProp update rules are showed in eq. (8.5), where $E[g^2]_t$ is the moving average of squared gradients, g_t is the gradient at time t (i.e. $\frac{\partial J}{\partial W}$), η is the learning rate, β is the moving average parameter.

- **Adam** [72] is an adaptive learning rate optimization algorithm that has been designed specifically for training deep neural networks. It can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum, but with bias correction terms for the first and second moments.

  it compute the decaying averages of past and past squared gradients m_t and v_t respectively as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{8.6}$$

  where g_t is the gradient at time t, and β_1 and β_2 are newly introduced hyper-parameters of the algorithm. The original paper proposes default values of 0.9 and 0.999 respectively, which are usually never changed in practice. The vectors of moving averages are initialized with zeros at the first iteration. The authors of Adam observe that they are biased towards zero, especially during the initial time steps, so they counteract these biases by computing bias-corrected first and second moment estimates:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{8.7}$$

  where β_1^t and β_2^t denote the value beta to the power of t. The only thing left to do is to use those moving averages to scale learning rate individually for each parameter. The way it is done in Adam is very simple, to perform weight update we do the following:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{8.8}$$

  where the authors propose 10^{-8} for ε, and η is the parameter of learning rate that user can choose.

- **Nadam** [33] (Nesterov-accelerated Adaptive Moment Estimation) combines Adam and Nesterov accelerated gradient (NAG). As we have seen before, Adam can be viewed as a combination of RMSprop and momentum: RMSprop contributes the exponentially decaying average of past squared gradients v_t while momentum accounts for the exponentially decaying average of past gradients m_t. Author of [33] propose the modifications of original NAG, and the final Nadam update rule:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} (\beta_1 \hat{m}_1 + \frac{(1+\beta_1) g_t}{(1-\beta_1^t)}) \tag{8.9}$$

  where the definitions for β_1, $\hat{m}_t$ and $\hat{v}_t$ are the same as in 8.7.

- **AMSGrad** [111]: As adaptive learning rate methods have become the norm in training neural networks, Wilson et al. [138] suggested that adaptive gradient methods do not generalize as well as SGD. These methods tend to perform well in the initial portion of training but are outperformed by SGD at later stages of training. Reddi et al. [111] formalize this issue and

pinpoint the exponential moving average of past squared gradients as a reason for the poor generalization behaviour of adaptive learning rate methods.

In settings where Adam converges to a suboptimal solution, it has been observed that some minibatches provide large and informative gradients, but as these minibatches only occur rarely, exponential averaging diminishes their influence, which leads to poor convergence. The authors provide an example for a simple convex optimization problem where the same behaviour can be observed for Adam.

To fix this behaviour, the authors propose a new algorithm, AMSGrad that uses the maximum of past squared gradients v_t rather than the exponential average to update the parameters. Instead of using v_t (or its bias-corrected version $\hat{v}_t$) directly, we now employ the previous v_{t-1} if it is larger than the current one:

$$\hat{v}_t = max(\hat{v}_t, \hat{v_{t-1}}) \tag{8.10}$$

For simplicity, the authors also remove the debiasing step that we have seen in Adam, they use the same update rule as in eq. (8.6) to update m_t and $\hat{v}_t$, and use eq. (8.10) to update $\hat{v}_t$, then the AMSGrad weights updat rule is:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} m_t \tag{8.11}$$

- **AdaBound** [89] addressed the same problem mentioned in AMSGrad, as researches suggested adaptive gradient methods do not generalize as well as SGD. It proposes new variants of ADAM and AMSGRAD, which avoids potential negative effects on generalization of excessively large and small gradients. They employ dynamic bounds on learning rates in these adaptive methods, where the lower and upper bound are initialized as zero and infinity respectively, and they both smoothly converge to a constant final step size. The new variants can be regarded as adaptive methods at the beginning of training, and they gradually and smoothly transform to SGD (or with momentum) as time step increases.

Above adaptive gradient state of the art algorithms are tested against our algorithms in Sec. 10.3.4.

Chapter 9

Exponential / Proportional-Derivative Control of Learning Rate

After reviewing the background and related works of neural network and learning rate optimizer, in this chapter, we advocate using a control-based approach to adapt the learning rate in order to reach a high network accuracy in a short amount of time. The principle is to switch from time decreasing rules to a performance-based rule to be able to increase the learning rate when necessary. P (Proportional) and PD (Proportional-Derivative) control strategies are initially developed. Then we present E/PD-Control, a hybrid strategy for setting the learning rate that combines both a time-based rule, with a first initial phase of an exponential growth of the learning rate, followed with a PD controller triggered by the network loss function. The initial E (Exponential) phase additionally allows the PD to be tuned on-line, thus getting rid of the need of an off-line profiling phase to adapt to a new dataset or network architecture. The E/PD-Control is evaluated on two classical state of the art datasets (CIFAR-10 [76] and Fashion-MNIST [139]), which are labelled image datasets commonly used to train computer vision algorithms [63]. Our control shows higher accuracy, faster rising time, lower final loss and more stable results than the state of the art techniques. Robustness regarding the initial value of the learning rate is also illustrated.

In the remaining of this chapter, we first present the problem statement in a control theory formulation and illustrate two state of the art learning rate strategies (Sec. 9.2). The control law is presented in Sec. 9.3 and its performance evaluation is given in Sec. 9.4.

9.1 Introduction

In chapter Sec. 8.4 of Chapter 8, we can see that AMSGrad and AdaBound are all aimed to solve the problem that adaptive gradient methods do not generalize as well as SGD, which makes these methods to perform well in the inital portion of training but are outperformed by SGD at later stages of training. Therefore, in this chapter, we propose a new solution Exponential / Proportional-Derivative Control, which is totally based on SGD, but which dynamically adjusts its learning rate based on the performance of algorithm. Here we consider the continual learning scenario introduced in Sec. 8.2 of Chapter. 8 where data comes dynamically in batches (not all data is initially available), previous data batches being discarded at the arrival of a new one. This type of scenario is very common in our everyday life if we think about sequential collection of a video flow or daily crowdsourcing [99][83].

In the gradient-based algorithms, there are two factors that influence the reach of the gradient global minimum: the network's weights initialization and the learning rate policy. The weights initialization is often dealt with by setting them all null or generated from a uniform distribution [50]. The learning rate parameter is used to weight the impact of a new epoch on the previously learned

model. The learning rate controls the speed to approach the minimum. A large learning rate will accelerate the converging speed but at the risk of diverging [12]. A small learning rate will slowly approach the minimum with less tendency to skip over it, but may fall into a local minimum.

The objective is thus to set the learning rate strategy in order to learn from the data as fast as possible to reach the maximum CNN's predictions accuracy. The dynamic data collection scenario raises the challenge of a learning rate scheduling able to deal with the combination of epoch learning (take the most out of the currently available data) with batch learning (being able to include new data without forgetting the previous ones).

A learning rate strategy is defined by its initial value and its evolution law. The tuning of both is a significant challenge for the deep learning community [67]. According to Bengio [12], a learning rate of 0.01 typically works as a default value for standard multi-layer neural networks. He also recommends a classic strategy to find a more suitable value for a given architecture and dataset. Its principle is to try several values on a subset of the dataset, compare the best validation accuracy for a fixed training time and the lowest training time to reach a given validation accuracy [123].

Learning rate evolution laws are usually of two kinds: (1) time-based or (2) adaptive. We have explained them in Chapter. 8. In this chapter, we mainly focus on comparing with the time-based algorithms. The issue with the state of the art time-based learning rate laws regarding our continual learning scenario is that they do not take into account the dynamic of data coming in. They are predefined functions that do not adapt to the performances of the CNN training. Even by re-initializing the learning rate rule at each batch, it will not take into account the precision improvement through time that results from the memory of the previous batches.

In this chapter, we advocate switching to a performance-based adaptation, in order to improve the learning efficiency.

9.2 Background & Motivation

We consider a CNN as being our plant, and the data used in training is seen as a disturbance, see Fig. 9.1.

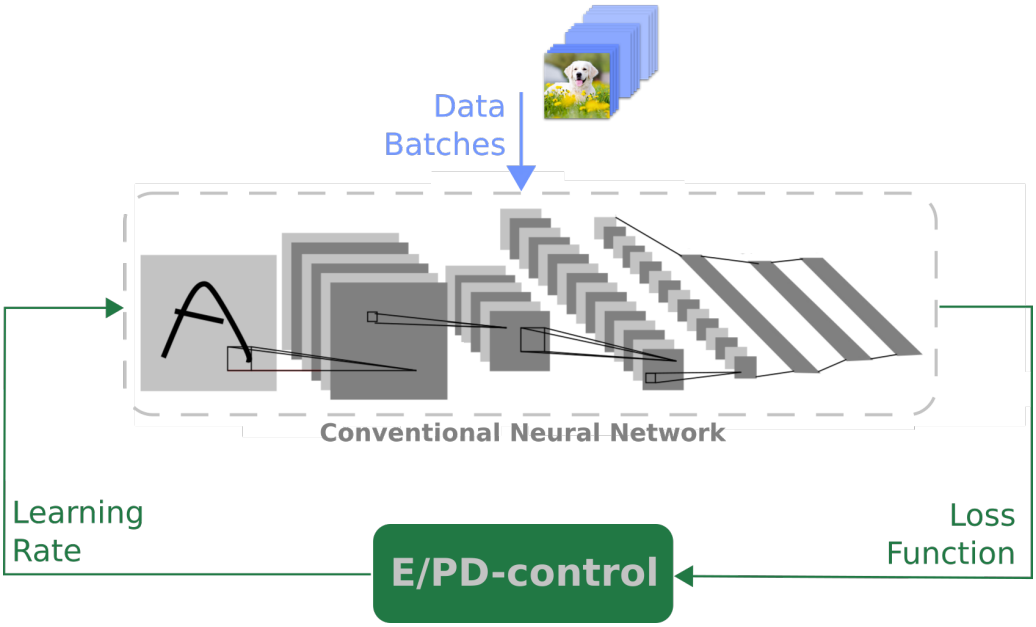
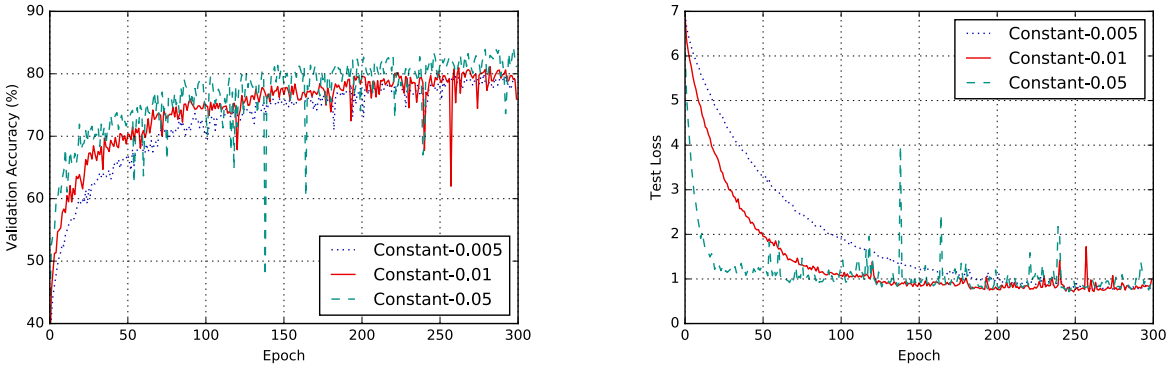


Figure 9.1 – CNN control schema.

The learning rate η is our control signal. To illustrate that the learning rate is a control signal for our continual scenario, we study the impact of different constant learning rate on the variation of the accuracy and loss functions over epochs. Fig. 9.2 is the application of three constant learning rates $\eta \in \{0.005, 0.01, 0.05\}$ (corresponding to Bengio's recommendation [12], larger and smaller) for training on CIFAR-10 dataset, with new data batches arriving every 60 epochs. The accuracy and loss signals differ according to the learning rate (see Fig. 9.2): with $\eta = 0.05$, the accuracy improves the fastest and the loss also quickly converges to its lower limit. However, the noise of the curve at the last epochs is also higher than with the two other scenarios because a large learning rate oscillates around the minimum. When $\eta = 0.005$, the accuracy increase is slower but the loss function varies more smoothly, and the loss value rarely rises. Since the learning rate is able to influence our performance indicators, it is suitable as a control signal. From Fig. 9.2 we can also see that the loss function variation is coherent with the choice of learning rate, therefore we infer the loss is suitable as a measurable signal.



(a) Validation accuracy

(b) Loss function

Figure 9.2 – Impact of different constant learning rates on accuracy and loss (CIFAR-10).

The experiments shown in Fig. 9.2 illustrate the advantages and drawbacks of large and small learning rates. A natural thought is to combine their benefits through learning rate scheduling: an initial phase with a large learning rate to quickly converge to a high-level accuracy, then a smaller value to smoothly approach the minimum and avoid the bumps on validation accuracy and loss. In the state of the art, there are some commonly used learning rate strategies that vary the learning rate through time: (i) Keras-Time-Based-decay and (ii) Exponential-Sine-Wave-decay.

Keras-Time-Based-decay is a commonly and widely used learning rate strategy in Keras[26], which is a famous python deep learning library. The learning rate is computed as follow:

$$\eta(k) = \frac{\eta(k-1)}{1+\delta k} \tag{9.1}$$

where k is the number of epochs since the arrival of the last batch, $k \in \{1, \ldots, E\}$. δ is a hyperparameter enabling to tune the steepness of the time decay. We set $\eta(0) = 0.01$ and $\delta = 0.001$ as suggested in [12].

The second common schedule is the exponential decay, it has been successfully used in neural network training. A good implementation is exponential decay sine wave learning rate schedule [2]. The original schedule is implemented to an off-line setting, so to adapt this learning rate schedule into our continual setting, we need to adjust their strategy to allow the learning rate decays to around 0 at

the ending epochs of each batch. We will refer to this strategy as Exponential-Sine-Wave-decay. The adapted version is calculated as follow:

$$\eta(k) = \eta(0)e^{\frac{-\alpha k}{E}}(\gamma sin(\frac{\beta k}{2\pi}) + e^{\frac{-\alpha k}{E}} + 0.5) \tag{9.2}$$

where k shares the same definition as in eq. (9.1); E is the training epochs per batch; α, β and γ are three hyperparameters. In order to have a same behavior as in [2] during our shorter E, we set $\alpha = 3$, $\beta = 6$ and $\gamma = 0.4$. The constant 0.5 in the equation is important, it makes sure that $\eta(k)$ is strictly positive.

9.3 Performance-based Learning Rate Laws

In all the related work strategies, the learning rate is decreasing with time, in a predefined manner. The only differences between these strategies is that for some the learning rate decreases slowly in the beginning and faster in the end and for others is the opposite. We therefore introduce our control strategy where the learning rate is automatically computed based on the loss function (see Fig 9.1). Nevertheless, according to the definition of the loss function, the absolute loss value in itself does not give us much information since different size of training dataset can change the absolute loss value. Therefore, we normalize the value of $loss(k)$ by $loss(k = 0)$, where $loss(k)$ represents the loss value at k^{th} epoch since the arrival of the last batch.

Subsequently, we try three different control laws for computing the learning rate η: Proportional-Control (P-Control), Proportional derivative-Control (PD-Control) and a Mixed Exponential PD-Control (E/PD-Control).

P-Control In this case the learning rate depends proportionally on the loss value as follows:

$$\eta(k) = K_P \frac{loss(k)}{loss(0)} \tag{9.3}$$

In general the value of $\frac{loss(k)}{loss(0)}$ varies between 0 and 1. Indeed, as the loss function decreases thanks to the Stochastic Gradient Descent, we know that it is approaching the minimum of the loss function and therefore the learning rate should be decreased in order not to skip it. The choice of K_P is important for the speed of convergence. We first make it equal to the same value as the empirical starting learning rate from [26]: $\eta(0) = K_P = 0.01$. The reason is that we believe $loss(k)$ should gradually decrease. Therefore according to eq. (9.3), $\eta(k)$ will decrease with time, at a changing range between $\eta(0)$ and 0. Making sure that learning rate converges over time is important for machine learning model, which ensure the learning process will not diverge. Experiments in Sec. 9.4.5 will show the effects with different K_P on final model's performance.

PD-Control On one side, the hypothesis behind P-Control is that the loss is always decreasing: as we are getting closer to a minimum, the learning rate should slow down to better approach it. On the other side if the loss has decreased during last epoch we consider being in the good direction to find the minimum so we should reward last learning epoch by increasing the learning rate. This can be seen as adding an derivative action to our controller. We express our PD-Control as follows:

$$\eta(k) = K_P \frac{loss(k)}{loss(0)} - K_D \frac{loss(k) - loss(k-1)}{loss(0)} \tag{9.4}$$

where $K_P = 0.01$, and the derivative coefficient K_D is empirically chosen at 5 times $\eta(0)$. As we choose $\eta(0) = 0.01$, then $K_D = 0.05$. As $-(loss(k) - loss(k-1))$ could also be negative, the derivative part will introduce oscillations to the learning rate. In order to avoid that $\eta(k)$ becomes negative, the PD-Control is turned to a P-Control if $\eta(k)$ in PD-Control gets a negative value. Indeed, the P-Control will always return a positive value for the learning rate, as the loss function is positive by definition.

E/PD-Control This third control law aims at accelerating the convergence speed by exponentially increasing the learning rate at the beginning of learning a new batch, since the data is new there are more information to learn. We present a two phase algorithm to control the learning rate: (i) an initial Exponential growth followed by (ii) a PD-Control. During the exponential growth period, the learning rate is increased each time step by a factor 2 to quickly reach the minimum. This phase is stopped when the loss starts increasing, and the learning rate is afterwards ruled by the PD-Control law. The PD parameters are initialized with the last value of the learning rate before loss growth. The E/PD-Control behavior during one batch is summarized in Algorithm. 1.

Algorithm 1 E/PD-Control

1: $\eta(0) = 0.01$
2: $k = 1$
3: **while** $loss(k) \leq loss(k-1)$ **do**
4: $\quad \eta(k) = 2\eta(k-1) = 2^k \eta(0)$
5: $\quad k = k+1$
6: **end while**
7: $\eta(k) = \eta(k-1)/2$
8: $K_P = \eta(k-1)/2$
9: $K_D = 5 \times \eta(0)$
10: **for** $i \in \{k+1, \ldots, E\}$ **do**
11: $\quad \eta(i) = K_P \frac{loss(i)}{loss(0)} - K_D \frac{loss(i) - loss(i-1)}{loss(0)}$
12: $\quad$ **if** $\eta(i) < 0$ **then**
13: $\quad\quad \eta(i) = K_P \frac{loss(i)}{loss(0)}$
14: $\quad$ **end if**
15: **end for**

from Algorithm. 1, one thing we need to make sure is that learning rate should always be positive. As the steps ($1^{st} - 7^{th}$) are the exponential increase phase, the positive learning rate is guaranteed. In order to make the calculated $\eta(i)$ to be also positive, we add the branch at step 12, if $\eta(i) < 0$, we will turn the PD control to P control. But instead of using the condition at step 12, we can also add some limitations on K_P and K_D to make sure the calculated $\eta(i)$ at step 11 is always positive, that will avoid our algorithm to turn to P control any more.

To ensure the calculated $\eta(i)$ be strictly positive, it means:

$$K_P \frac{loss(i)}{loss(0)} - K_D \frac{loss(i) - loss(i-1)}{loss(0)} > 0 \tag{9.5}$$

this condition can be rewritten as:

$$K_P > K_D (1 - \frac{loss(i-1)}{loss(i)}) \tag{9.6}$$

according to our calculation, by satisfying the following condition:

$$K_P > K_D > 0 \tag{9.7}$$

we can guarantee $\eta(i)$ is strictly positive. That is because: if $loss(i-1) > loss(i)$, $(1 - \frac{loss(i-1)}{loss(i)})$ is negative (we know that loss is non-negative if the loss function is cross-entropy or mean squared error (MSE) which are the most popular loss functions used in classification problem. Another notice is that loss is rarely 0. Even the prediction accuracy can be 100%, the loss can hardly be 0. In reality, that is almost impossible, so we do not discuss the situation where loss = 0), then Condition (9.6) holds; If $loss(i-1) = loss(i)$, $1 - \frac{loss(i-1)}{loss(i)} = 0$, Condition. (9.6) is still satisfied; Finally if $loss(i-1) < loss(i)$, $(1 - \frac{loss(i-1)}{loss(i)}) \in (0, 1)$. Condition. (9.6) validates too. Therefore, if we do not use the branch at step 12, and if we can make sure the condition. (9.7) always holds, then $\eta(0)$ can also be guaranteed to be positive.

9.4 Control Laws Evaluation

In this section, the P, PI and E/PI-Control laws are evaluated in comparison with the state of the art. The convergence of the E/PI-Control law is highlighted, and its robustness with regard to its initial configuration is presented.

9.4.1 Experimental setup

The controllers are evaluated on two complex state of the art datasets: CIFAR-10 [76] and Fashion-MNIST [139], as we did not find datasets with same characteristics and complexity containing economic data. The CNN and scenario configurations for the two datasets are detailed in Tab. 9.1. As the images in CIFAR-10 have colors and are larger than the ones of Fashion-MNIST, a more complex CNN setting with more layers and parameters is used. Meanwhile, as there are more informations to extract from CIFAR-10, the number of epochs per batch is larger, allowing the accuracy curve to converge. All the values of hyperparameters of Eq. (9.1) and (9.2) we showed in Sec.9.2 are tuned for CIFAR-10, as Fashion-MNIST has shorter epochs per batch, we will change δ to 0.01 of Eq. (9.1), and set $\alpha = 2$, $\beta = 18$ of Eq. (9.2) for Fashion-MNIST experiment learning rate schedule. This pre-set is done in order to be as least dataset depended as possible.

To eliminate the influence of the CNN's weights starting point to the final accuracy, we initialize the weights of each layer of CNN by Xavier uniform initializer [50], all the results will be averaged on 3 time experiment results. All the experiments are executed on Google Cloud ML-Engine with a P100 GPU. All code is implemented with Keras library [26].

For performance evaluation, we measure several indicators on the accuracy and loss signals, the first one being their final value. To quantify the accuracy's converging speed, we will report for each experiment the epoch at which they reached 95% of their final accuracy. To compare the influence of each learning rate strategy on stability of validation accuracy, we will also compare the standard deviation of the accuracy curve on the last 10% epochs of each experiment.

9.4.2 Convergence analysis

Stability of the presented algorithms needs to be proved to ensure that the loss will not diverge, and ideally converge to 0. The stability of the algorithm relies on the SGD: as the direction of the gradient

Table 9.1 – CNN configuration

Use case	CIFAR-10	Fashion-MNIST
#data instances to train	50,000	60,000
#data instances to test T	10,000	10,000
#classes C	10	10
image size	32×32	28×28
data batch size B	10000	10000
#trainng epochs per batch E	60	20
#CNN layers	28	15
#CNN parameters	1,641,858	422,538

is by construction always set to decrease the loss. The only case when the loss might increase is because the learning rate is too large, we skipped the minimum. The stability of the P and PD-Controllers is ensured via a proper parametrization of K_P and K_D. The E/PD-Control law allows the learning rate to exponentially grow as long as loss decreases, however the learning rate is switched to a PD law as soon as the loss starts to increase. The reset of the learning rate to the previously best value ($7^{th} - 9^{th}$ line of Algorithm 1) enables to properly initialize the PD.

9.4.3 P, PD and E/PD-Control Performances Validation

The three control laws presented in Sec. 9.3 are evaluated on CIFAR-10. Results are reported in Fig. 9.3 through the accuracy 9.3a and loss functions 9.3b, and the corresponding control signals are illustrated in Fig. 9.4, For the P and PD in 9.4a and for the E/PD law in 9.4b. There are few differences between P and PD control performances, while the E/PD-Control is significantly faster (61 epochs rising time compared to 130 for the P and PD), converges to a higher accuracy (+7%) and lower loss (-37%) and the standard deviation of the accuracy at the end of the experiment is three times lower. We see from the first epochs that the E-phase enables to properly tune the initial value of the PD, which then significantly increases the validation accuracy. The P and PD-Control learning rate signal (Figure 9.4a) illustrates that a reset of the learning rate at the arrival of a new batch is not necessary beneficial if the value is not carefully chosen, as for the E/PD-Control.

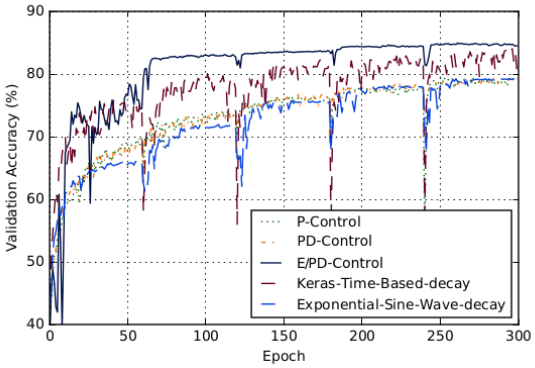
The loss function with the PD-Control declines a little bit faster than with the P-Control at beginning, and is not presenting a large peak around epoch 240. Those advantages made us opt for the PD-Control to combine with the initial E-phase.

9.4.4 Comparison with state of the art

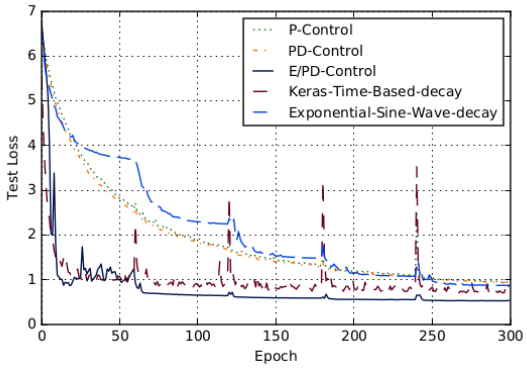
Comparison of the state of the art learning rate strategies to our E/PD-Controller is provided for CIFAR-10 (Fig. 9.3 and 9.4) and for Fashion-MNIST (see Fig. 9.5 for the accuracy and loss and Fig. 9.6 for the learning rate evolution through epochs).

E/PD-Control provides the best results for all the indicators for CIFAR-10. It converges faster and has a smallest standard deviation of last 10% epochs among all the strategies, it reaches at a higher final validation accuracy (+3%) and a lower loss. Keras-Time-Based-decay has a closer final accuracy and loss to E/PD-Control. But the deviation of its accuracy curve is bigger than E/PD-Control, especially at the beginning of learning a new batch.

Regarding Fashion-MNIST dataset, results are similarly in favor of the E/PD-Controller, even if the differences are smaller. As this dataset is easier than CIFAR-10, all strategies reached a high

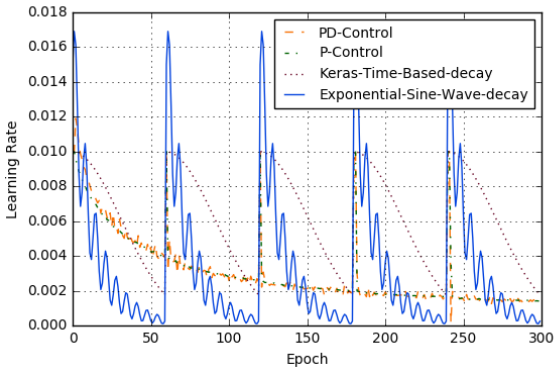


(a) Validation accuracy

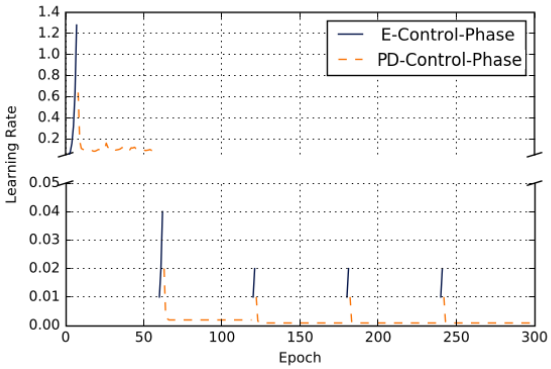


(b) Loss function

Figure 9.3 – Performances of state of the art, P, PD and E/PD-Control (CIFAR-10).



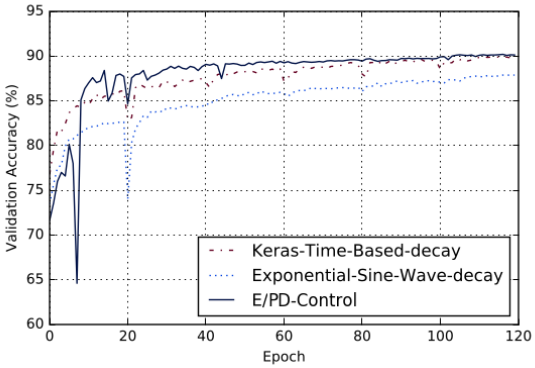
(a) P and PD-Control



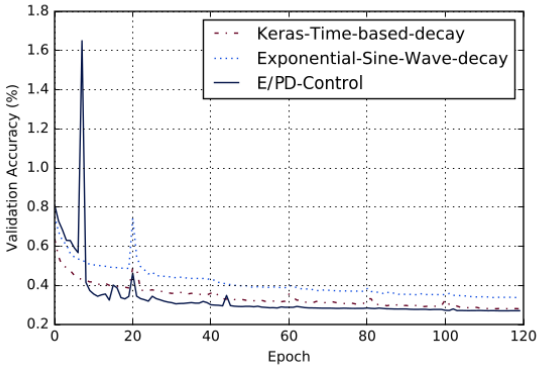
(b) E/PD-Control

Figure 9.4 – Control signal of state of the art, P, PD and E/PD-Control (CIFAR-10).

validation accuracy and lower loss, the standard deviation of accuracy of last 10% epochs is also very small.

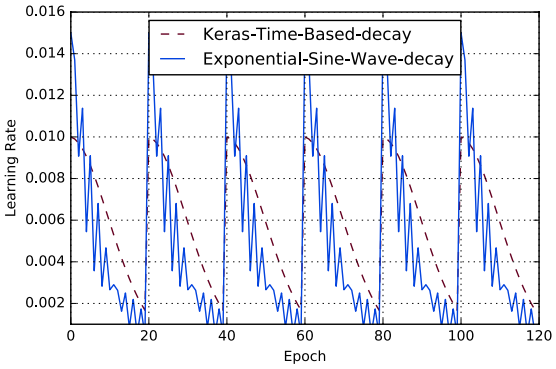


(a) Validation accuracy

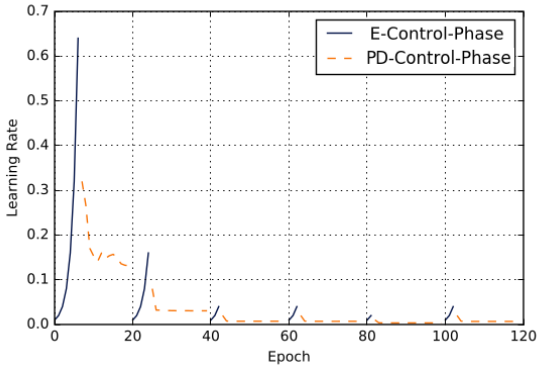


(b) Loss function

Figure 9.5 – Performances of state of the art and E/PD-Control on Fashion-MNIST.



(a) State of the art strategies



(b) E/PD-Control

Figure 9.6 – Control signal for the state of the art and E/PD-Control on Fashion-MNIST.

9.4.5 Robustness to initial value of the learning rate

The E/PD-Control is now compared with the best strategy from the state of the art (Keras-Time-Based-decay) when the initial learning rate varies. Results are showed in Table 9.2 for CIFAR-10 and Table 9.3 for Fashion-MNIST.

Among all the experiments on CIFAR-10, there are only one case for which Keras-Time-Based-decay law has a better indicator (final validation accuracy at initial learning rate 0.05). The difference is very small, and standard deviation of the indicator itself is large. Moreover, if we check the accuracy's standard deviation during the last 10% epochs, Keras-Time-Based-decay still has a strong oscillation, which makes the model unpredictable. E/PD-Control also shows the advantage on converging time, it makes the model converge faster and rarely affected by the initial values.

Tab. 9.3 shows the robustness results on Fashion-MNIST. E/PD-Control still shows a fast converging speed, reaching 95% of final accuracy just using 7 to 10 epochs. The performances for the final loss and accuracy final standard deviation are similar for the two strategies. The E/PD-Control's final accuracy performances among all the experiments is more stable than with Keras-Time-Based-decay, which again, shows that E/PD-Control is more robust to the initial learning rate variations.

Table 9.2 – Robustness experiments with varying initial learning rate on CIFAR-10. Mean value (and standard deviation) over 3 runs. The best results are highlighted in **bold**.

Algorithm	Initial learning rate	Final loss	Final validation accuracy (%)	Final accuracy standard deviation	First epoch to reach 95% accuracy
Keras	0.001	0.849(0.023)	79.075(0.485)	0.371(0.053)	161.333(24.495)/300
E/PD-Control	0.001	**0.648(0.015)**	**82.035(0.465)**	**0.057(0.013)**	**61.333(0.471)/300**
Keras	0.002	0.745(0.026)	80.180(0.445)	0.415(0.049)	118.333(9.78)/300
E/PD-Control	0.002	**0.586(0.006)**	**83.150(0.225)**	**0.077(0.017)**	**62(0)/300**
Keras	0.05	0.727(0.006)	**85.640(0.523)**	1.630(0.085)	103.333(2.859)/300
E/PD-Control	0.05	**0.555(0.005)**	85.060(0.090)	**0.117(0.001)**	**61.333(0.471)/300**
Keras	0.1	0.829(0.180)	84.433(2.82)	1.609(0.432)	77.333(32.785)/300
E/PD-Control	0.1	**0.578(0.013)**	**85.075(0.585)**	**0.345(0.16)**	**65(0.816)/300**

Table 9.3 – Robustness experiments with varying initial learning rate on Fashion-MNIST. Mean value (and standard deviation) over 3 runs. The best results are highlighted in **bold**.

Algorithm	Initial learning rate	Final loss	Final validation accuracy (%)	Final accuracy standard deviation	First epoch to reach 95% accuracy
Keras	0.001	0.413(0)	85.055(0.035)	0.054(0)	37(0.816)/120
E/PD-Control	0.001	**0.334(0.002)**	**87.955(0.105)**	**0.023(0.008)**	**10.667(1.247)/120**
Keras	0.002	0.360(0.001)	86.850(0.005)	0.066(0.002)	25.667(1.247)/120
E/PD-Control	0.002	**0.350(0.008)**	**87.415(0.103)**	**0.057(0.011)**	**8.333(0.471)/120**
Keras	0.05	0.282(0.026)	89.785(0.920)	0.145(0.012)	16.667(6.532)/120
E/PD-Control	0.05	**0.263(0.006)**	**90.425(0.200)**	**0.094(0.013)**	**9.333(1.700)/120**
Keras	0.1	0.265(0.016)	90.400(0.674)	0.133(0.010)	9(3.265)/120
E/PD-Control	0.1	**0.249(0.003)**	**91.340(0.140)**	**0.114(0.015)**	**7(0)/120**

9.5 Conclusion

When performing image classification tasks with neural networks, often comes the issue of on-line training, from sequential batches of data. Iterative training of CNNs is driven by the learning rate - how much to update the network weights with the new data - which value is usually ruled by a time decreasing function. This chapter presents a control approach to the challenge of on-line training

of CNNs, that decides the learning rate value based on the expected learning need (i.e. the CNN loss function) instead of being time-based. E/PD-Control is a strategy that combines a phase of exponential growth of the control signal (i.e. learning rate) with a PD controller, which parameters are automatically adapted based on the E-phase.

Stability of the control strategy is provided, and evaluation highlights that E/PD-Control achieves a higher accuracy level in a shorter time than the state of the art solutions. Robustness of the approach is illustrated by its performances on two different datasets, and enforced by a sensitivity analysis regarding its initialization.

This work could be further extended by the addition of a triggering mechanism to smartly adapt the number of epochs needed at each batch processing. Moreover, we want to investigate the performances of the E/PD-Control in the scenario when new classes appear in some batches.

Chapter 10

Event-Based Control for Continual Training of Neural Networks

In this chapter, we continue to address the problem we mentioned in previous chapters where we still consider the continual learning scenario as in Sec. 8.2 in Chapter. 8, but we propose two Event-Based control loops to adjust the learning rate of the E (Exponential)/PD (Proportional Derivative)-Control which in presented in Chapter. 9. In order to improve performance, the first Event-Based control loop would be implemented to prevent sudden drop of the learning rate when the model is approaching the optimum. The second Event-Based control loop will decide, based on the learning speed, when to switch to the next data batch.

Results show the Event-Based E/PD is better than the original algorithm (higher final accuracy, lower final loss value), and the Double-Event-Based E/PD can accelerate the training process, save up to 67% training time compared to state-of-the-art algorithms and even result in better performance.

The chapter is organised as follows: after a brief introduction of the problem in Sec. 10.1. The main contribution, i.e., the two event-based mechanisms, is described in Sec. 10.2. Sec.10.3 contains the experimental setup, results and analysis on Google Cloud GPUs. The chapter ends with a conclusion and perspectives for further work in Sec.10.4.

10.1 Introduction

On continuing to address the problem we mentioned in Chapter. 8, we first recall here the behaviour of the E/PD controller. Up to our knowledge, E (Exponential)/PD (Proportional Derivative) control [144] is the first adaptive learning rate algorithm which uses control theory to dynamically adapt the learning rate during the learning process. It uses only current gradient as in SGD, but its learning rate η is dynamically calculated based on the loss value. During the E phase, that corresponds to the beginning of the training when the loss value is continuously decreasing, η (learning rate) is increased each time step by a factor of two. Once the loss stops decreasing, the PD phase takes over and, considering CNN as a dynamic system, computes the control input (i.e. η) based on the CNN's output (i.e. the loss value).

The E/PD is mainly compared with time-based learning rate algorithm in [144]. Actually E/PD is also time-based, in the sense of a periodic computation of the control law regardless its utility. In this chapter, we propose two event-based control strategies to reduce the time CNN spends learning "inefficiently" from data, as well as an extensive evaluation. Moreover, while using event-based mechanisms we should expect for a reduction in the use of resources [7, 36], without degrading performances [88] and with stability and robustness guarantees [93]. Numerous Event-Based control strategies in the literature are focusing on stability and performance guarantees. Most event-based PID

controllers are based on level-crossing triggering of some measuring error (see for instance [6, 36]) or more generally rely on an event-function based on Lyapunov functions (see for instance [135, 93]).

The two introduced Event-Based control algorithms are: (i) Event-Based Learning Rate control, which will be implemented to prevent sudden drop of the learning rate when the model is approaching the optimum; (ii) Event-Based Learning Epochs control, which will decide based on the learning speed when to switch to the next data batch. First Event-Based control is implemented during the PD phase of E/PD, it prevent the update of the learning rate if loss value decreases. In the original continual learning setting, we set in advance a fixed number of training epochs for each data batch, but results with E/PD control clearly show that the performance improvements are very limited towards the end of training epochs for each batch data. We should therefore stop the "inefficient" learning and pass to the next data batch. The second Event-Based control proposed is that it inspects the record of loss values, and decides when to finish the learning for current data batch.

Our algorithm is evaluated on two classical machine learning image datasets CIFAR-10 and CIFAR-100 [76]. Here we replace the dataset Fashion-MNIST that we used in the last chapter by dataset CIFAR-100, the reason is Fashion-MNIST dataset has only 10 classes grayscale images, while CIFAR-100 has 100 classes of color images which makes it more complex. The results are compared with four best state-of-the-art algorithms: Adam, Nadam, AMSGrad and AdaBound. Our results show that the E/PD combined with the two introduced Event-Based control not only outperforms original E/PD but also converges faster than any other state-of-the-art counterpart, which means that we could either stop the training process much earlier (and still obtain same performance) or obtain much better performance while using same training time.

10.2 Event-Based Control Laws

In [144], an E/PD control of the learning rate is proposed consisting of an increasing exponential phase followed by a PD phase. However, if an increase of the performance can be achieved on both the loss and the accuracy, the learning rate is progressively decreased by the E/PD control in the PD phase, even though a larger value of learning rate would be more efficient in term of performance. Since event-based PID have shown to be more efficient in terms of convergence [36], we propose here to implement an event-based E/PD controller to control the learning rate. [144] also shows that significant improvements in terms of accuracy and loss only occurred at the first epochs of training each data batch, so after this stage there is no limited interest into continuing the learning on further epochs. Therefore, we propose a second event-based control to adapt the data batch loading process.

10.2.1 Event-Based Learning Rate

The E/PD Control algorithm from [144] is schematically presented in Fig. 9.1 and detailed in Algorithm. 1, where we suggest to look at a CNN training as a dynamical system with the learning rate as controlled input and the loss as measurable output. Initial weights of the CNN are chosen either randomly (e.g. uniform distribution or normal distribution) or by algorithms (e.g. Xavier initialization [51]). The initial learning rate $\eta(0)$ is fixed. E/PD contains two phases: a phase where the learning rate is multiplied by some factor at each step (it doubles in our case) while the loss value is continuously decreasing and a PD phase where the learning rate is computed by a PD-control law once the loss starts to increase. The E/PD learning rate strategy is defined as:

$$\eta(k+1) = 2\eta(k) \tag{10.1}$$

as long as $L(k) < L(k-1)$ (E phase) and

$$\eta(k+1) = K_P \frac{L(k)}{L(0)} - K_D \frac{L(k) - L(k-1)}{L(0)} \tag{10.2}$$

from the first instant $k = k^*$ when $L(k^*) > L(k^* - 1)$ to the end of learning process for the data batch (i.e. the PD phase). For the sake of simplicity the loss values are normalized with respect to the initial epoch loss value $L(0)$. K_P and K_D are the proportional and derivative gain.

On top of the PD phase we consider the following event-based mechanism where instead of letting the PD-Control compute the rate each time (which might be lowering the learning rate), we propose to update the learning rate only if the loss value increases during the PD-Control phase.

Let us define the event function $e_1 : \mathbb{R}^+ \to \{0,1\}$ by:

$$e_{1_k} = \begin{cases} 1 & \text{if } L(k) - L(k-1) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10.3}$$

The proposed PD event-triggered control output $\eta(k+1)$ at time $k+1$ is then:

$$\eta(k+1) = \begin{cases} K_P \dfrac{L(k)}{L(0)} - K_D \dfrac{L(k) - L(k-1)}{L(0)} & \text{if } e_{1_k} = 1 \\ \eta(k) & \text{otherwise} \end{cases} \tag{10.4}$$

where $\eta(k+1)$ is the calculated learning rate for epoch $k+1$, and $L(k)$ is the corresponding loss for epoch k.

Note that the convergence of CNN training is ensured by E/PD, and the convergence analysis of E/PD is discussed in Chapter 9.4.2. The proposed event-based control does not introduce any instability because if $e_1 = 0$, it means the loss is decreasing, thus the model is converging, and if $e_1 = 1$, the learning rate strategy returns to E/PD.

10.2.2 Event-Based Learning Epochs

Controller Design As observed in [144], significant improvement in the learning only occurs at the beginning when loading a new batch, the accuracy and loss value evolve slowly afterwards. This motivates the use of an event-based strategy on the loss value record.

Consider a maximum of N possible training epochs within each batch, the user being totally free in choosing this value. Let X_k vector contain the latest m epochs and Y_k vector contain the m latest corresponding normalized loss values:

$$X_k = \begin{bmatrix} k-m & \cdots & k-2 & k-1 & k \end{bmatrix}$$
$$Y_k = \begin{bmatrix} \frac{L(k-m)}{L(0)} & \cdots & \frac{L(k-2)}{L(0)} & \frac{L(k-1)}{L(0)} & \frac{L(k)}{L(0)} \end{bmatrix}$$

where $k \in [1, N-1]$. One can use least squares estimation to fit a regression line with X_k and Y_k:

$$Y_k = \alpha_k X_k + \beta_k \tag{10.5}$$

The purpose of this is that if the training process goes well the loss value should always decrease, therefore α_k should always be negative. Even with the presence of loss variations during the training, as long as the decreasing trend doesn't change, α_k should still be negative. Nevertheless, in the moment the loss trend becomes flat or even is increasing, α_k will become 0 or positive.

We define the event mechanism by the event function $e_2 : \mathbb{R}^+ \to \{0,1\}$ by:

$$e_{2_k} = \begin{cases} \text{call new batch} & \text{if } \alpha_k > \alpha_{thld} \text{ or } k = N \\ \text{remain on same batch} & \text{if } \alpha_k \leq \alpha_{thld} \text{ and } k < N \end{cases} \tag{10.6}$$

which enables to switch to new data batch when the learning speed is too low, i.e. the training is not efficient anymore.

The threshold α_{thld} can be adjusted in order to control the efficiency of learning. This threshold should never be positive as an increasing curve of the loss value is not desirable. With enough computing resources and no time constraints, the threshold can be set close to 0, and the training will continue even though it makes very small improvements. Nevertheless, for continual learning the time interval between two data batches can be short compared to the training time and we could encounter the scenario when before we finish the current training epochs the next data batch is already available. In this case, cutting off some useless training can be very useful. Therefore α_{thld} should also be chosen depending on the frequency of batch arrival. The choice of m is based on the constraints imposed by the CNN (or the application using CNN). A large value of m would imply a long time of inactivity as the controller would react only after m epochs (consecutive tests). A small value of m would imply that the algorithm is very sensitive to each epoch thus if $m = 0$ the event based algorithm becomes a time based one.

Continual Learning Scenario Recall the continual learning scenario defined in Sec. 8.2,the difference for this Event-Based Learning Epochs controller is that the training epochs for each batch could be varied but no larger than N. So here we could cyclically learn the data batches until it reaches the total epochs limit. Nevertheless we can see that the total training epochs are the same for both scenario for all the experiments of the same dataset. The continual learning arrangement for Event-Based Learning Epochs is illustrated in Fig. 10.1.

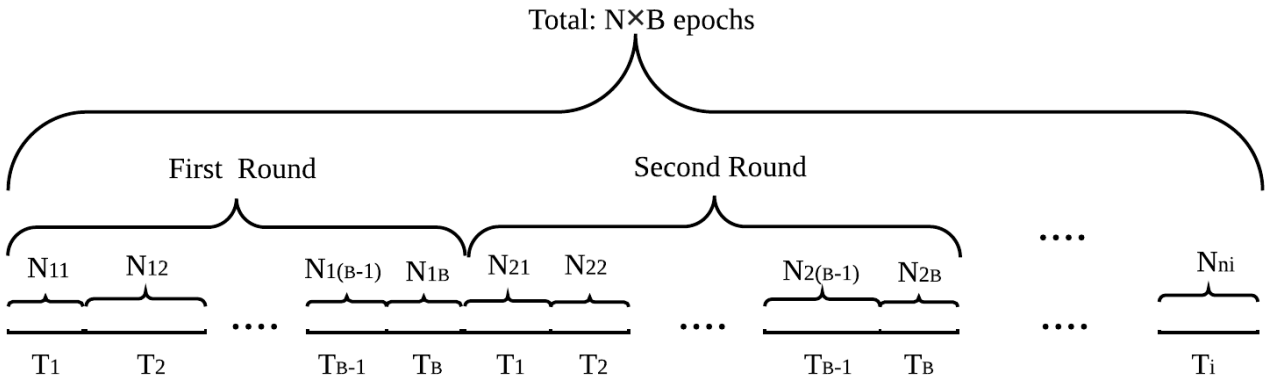


Figure 10.1 – Event-Based Learning Epochs Continual Learning Scenario. B is the number of data batches, N is the maximum training epochs per batch.

10.3 Experimental Evaluation

10.3.1 Experimental Setup

The experiments are implemented on two state of the art machine learning datasets: 1) CIFAR-10 (a natural image data set with 10 categories) and 2) CIFAR-100 (a natural image data set with 100 categories) [76] with 3 different initial learning rate. The characteristics of the two data-sets are given

in Tab. 10.1. As the CIFAR-100 dataset has more classes, we use a deeper CNN: ResNet [63] than the one used for CIFAR-10 VGG [122]. Due to the computational resource limitation, for ResNet with CIFAR-100, we train 30 epochs per data batch instead of 60 for CIFAR-10.

Table 10.1 – Experiments configuration

Use case	**CIFAR-10**	**CIFAR-100**
#data instances to train T	50,000	50,000
#data instances to test V	10,000	10,000
#classes C	10	100
data batch size S	10000	10000
total batches B	5	5
#trainng epochs per batch N	60	30
mini-batch size in one epoch	128	128
#CNN layers	28	15
#CNN parameters	1,641,858	422,538

All the experiments are implemented with Keras [26] and are carried out on Google Cloud Compute-Engine using 8 virtual CPU with 30 GB memory and one P100 GPU. Each experiment is repeated 5 times. The parameters α_{thld} and m are selected through a process of cross validation on a subset of CIFAR-10. As a small value for m leads to high sensitivity and a large m slows down the detection of the situation, we predefined a reasonable list of choice $m \in [4;5;6;7;8]$. Due to similar consideration of sensibility, we also predefined a list for the learning rate threshold $\alpha_{thld} \in [-0.1;-0.01;-0.001;-0.0001]$. Each possible pair from these two lists is tested, a good compromise between reactivity and noise sensitivity was found for $m = 4$ and $\alpha_{thld} = -0.001$.

10.3.2 Evaluation Metrics

The final loss and final validation accuracy (hereinafter referred to as FVA) reveal the performance of the final model. Nevertheless, stability metrics are also important: if accuracy curve experiences a big variance near the end of training process, even we could have a good final result, we could not assure that we always get this result. Thus, in our evaluation, we include standard deviation of the accuracy of the last 10% training epochs [95] (hereinafter referred to as FASD (Final Accuracy Standard Deviation)). Convergence speed of accuracy is another metric to evaluate the performance, as we focus on online learning scenario, the interval between two batch data can be short. With a limited time, a faster accuracy convergence could lead to a better model performance comparing to other algorithms. Therefore, we report the first epoch when the experiment reaches the 95% of best final accuracy among all the experiments. All the metrics above are the same used in Chapter. 9.

10.3.3 Evaluation of Event-Based E/PD

Event-Based E/PD (hereinafter referred to as EB E/PD) refers to the E/PD control combined with Event-Based Learning Rate control introduced in Sec. 10.2.1. To clearly show the effect of proposed algorithm, we implement the on-line training experiments with E/PD and EB E/PD on CIFAR-10 for different initial learning rate. From Fig. 10.2 we can first see the comparison between EB E/PD and original E/PD. For the first 60 epochs, we can see that EB E/PD is more stable than E/PD, then their curves are quite overlapped. The averaged comparison results are showed in Tab. 10.2. As a

conclusion EB E/PD performs better than E/PD in almost all metrics for all initial learning rate group. Even though EB E/PD has a higher FASD under 0.01 and 0.05 initial learning rate, the minimum value of FVA(±FASD) range of EB E/PD is higher than the maximum value of the range of E/PD. For final accuracy standard deviation, comparing to their final accuracy, these standard deviations are very small.

Table 10.2 – Experiments with varying initial learning rate $\eta(0)$ on CIFAR-10. Mean value over 5 runs are reported.

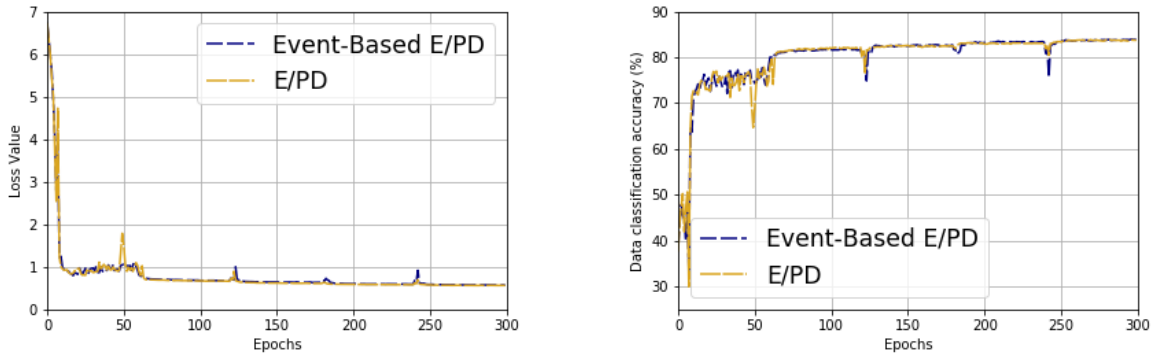
Algorithm	$\eta(0)$	**Final loss**	**FVA**[1] **(±** **FASD**[2]**) (%)**	**1st epoch to** 81.66%[3]
E/PD	0.002	0.58	83.17(±0.08)	124/300
EB E/PD	0.002	**0.56**	**83.81(±0.03)**	**93/300**
E/PD	0.01	0.55	84.35(±0.07)	88/300
EB E/PD	0.01	**0.54**	**84.91(±0.10)**	**75/300**
E/PD	0.05	0.56	85.06(±0.12)	73/300
EB E/PD	0.05	**0.50**	**85.96(±0.26)**	**63/300**

1. FVA: Final Validation Accuracy
2. FASD: Final Accuracy Standard Deviation
3. 81.66%: 85.96%(best final accuracy among all the experiments)×95%

For the sake of visibility, we zoom into the 60th to 90th training epochs from our two experiment runs and show the evolution of the loss value and learning rate in Fig. 10.3. According to the learning rate curve, we know that E phase ends at 62th epoch for E/PD-Control curve, and at 64th epoch for EB E/PD. E/PD-Control curve clearly shows the problem we mentioned above, we can observe that from 62th epoch, the loss of E/PD is continuously decreasing until 70th epoch, and its learning rate is also decreasing during this period. If the learning rate could stay constant during these 9 epochs, its loss would decrease sharply and that would improve the convergence speed. In contrast, EB E/PD keeps the learning rate when the loss continuously decreases which helps to accelerate the convergence. We can also notice that with the drop of the loss, each time when we update the learning rate for EB E/PD, its trend is also decreasing which will guarantee the stability of EB E/PD near the optimum.

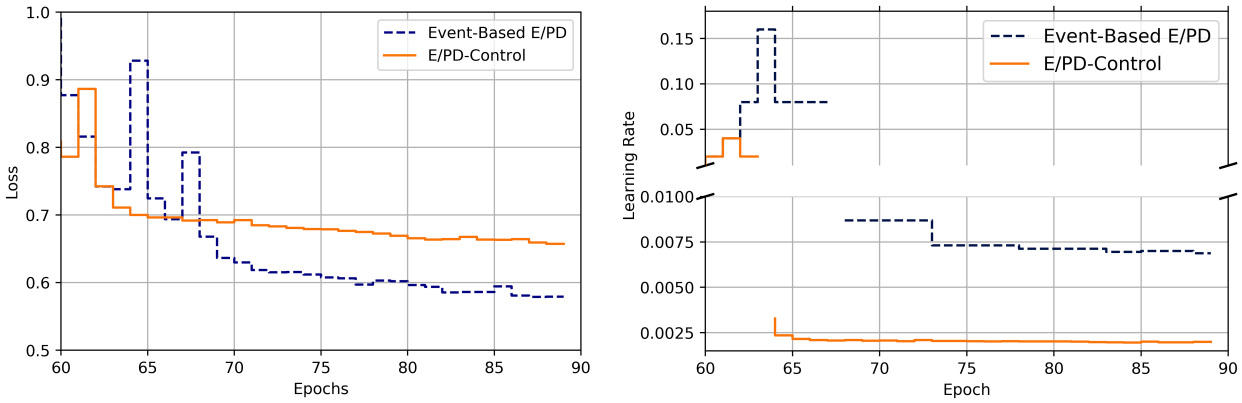
10.3.4 Evaluation of Double-Event-Based E/PD

Double-Event-Based E/PD-Control (hereinafter referred to as D-EB E/PD) refers to the E/PD control combined with Event-Based Learning Rate control (Sec. 10.2.1) and Event-Based Learning Epochs control (Sec. 10.2.2). To ensure the need of the Event-Based Learning Rate control, we implemented E/PD with only Event-Based Learning Epochs control (called E/PD Threshold); results showed that Double Event-Based E/PD always has a better performance in Final loss and FVA. The results are showed in Tab. 10.3. The results are as expected, Double Event-Based E/PD always has a better performance in Final loss and FVA. As the learning epochs for CIFAR-100 are limited, the difference between Double Event-Based E/PD and E/PD Threshold are bigger than the one on CIFAR-10. One thing to notice is the FASD of CIFAR-100 under 0.05 initial learning rate. Double Event-Based E/PD control is slightly higher than E/PD Threshold, which can be explained as the Event-Based Learning Rate forces E/PD to maintain a bigger learning rate than it should be at a right time, that could accelerate the convergence, but will slightly introduce some oscillations when e_{1_k} turns from 0 to 1.



(a) Loss value (b) Accuracy

Figure 10.2 – E/PD and EB E/PD performance comparison on CIFAR-10 with $\eta(0) = 0.01$ initial learning rate. Compact view of the results in Tab. 10.4.



(a) Loss value (b) Learning rate

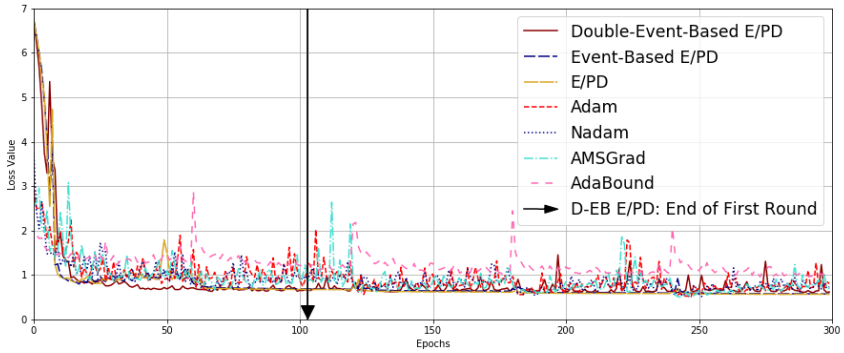
Figure 10.3 – Performances of E/PD and EB E/PD on CIFAR-10

Table 10.3 – Experiments with varying initial learning rate $\eta(0)$ on CIFAR-10 and CIFAR-100. Mean value over 3 runs are reported

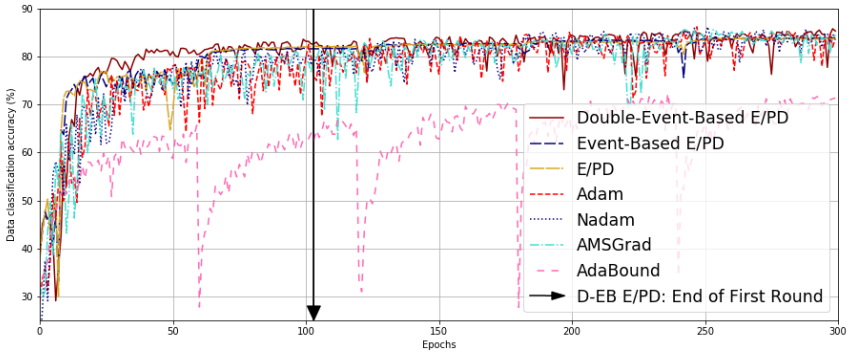
Dataset	**Algorithm**	$\eta(0)$	**Final loss**	**FVA (±FASD) (%)**
CIFAR10	Double-EB E/PD	0.002	**0.46**	**84.63**(±0.52)
CIFAR10	E/PD Threshold	0.002	0.66	83.43(±2.93)
CIFAR10	Double-EB E/PD	0.01	**0.60**	**84.15**(±1.18)
CIFAR10	E/PD Threshold	0.01	0.64	84.11(±1.57)
CIFAR10	Double-EB E/PD	0.05	**0.59**	**84.05**(±3.02)
CIFAR10	E/PD Threshold	0.05	0.64	83.31(±3.36)
CIFAR100	Double-EB E/PD	0.002	**2.53**	**45.84**(±1.80)
CIFAR100	E/PD Threshold	0.002	2.55	45.32(±1.85)
CIFAR100	Double-EB E/PD	0.01	**2.40**	**49.11**(±3.44)
CIFAR100	E/PD Threshold	0.01	2.47	45.54(±4.79)
CIFAR100	Double-EB E/PD	0.05	**2.37**	**49.91**(±10.04)
CIFAR100	E/PD Threshold	0.05	2.45	45.86(±9.74)

* FVA: Final Validation Accuracy
* FASD: Final Accuracy Standard Deviation

D-EB E/PD-Control has been tested on CIFAR-10 and CIFAR-100 and compared with 4 best state-of-the-art adaptive optimization algorithms: Adam, Nadam, AMSGrad and AdaBound. For these 4 learning rate strategies, except varying initial learning rate, all the other parameters remain as default as they mentioned in their paper or coded in Keras. As we adopt Event-Based Learning Epochs control into D-EB E/PD, the training epochs for each data batch is not fixed, we may also iterate each data batch several times. Therefore, we will not only report the results at the end of whole training process, but also the results after first round training (i.e. the training process iterates, for the first time, all the data batches, as explained above in Fig. 10.1).



(a) Loss value



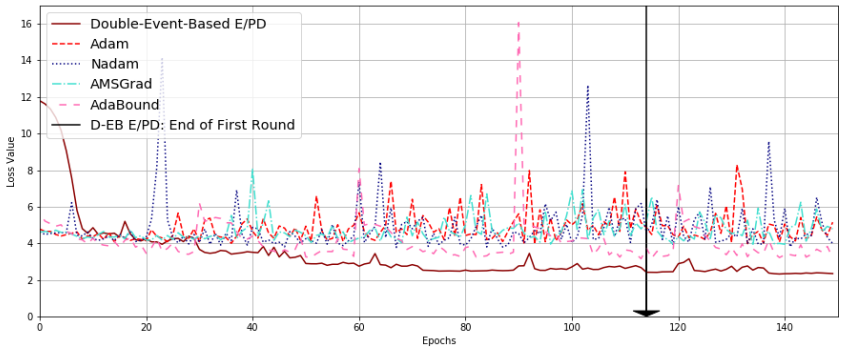
(b) Accuracy

Figure 10.4 – Performance comparison on CIFAR-10 with $\eta(0) = 0.01$ initial learning rate. Compact view of the results in Tab. 10.4.

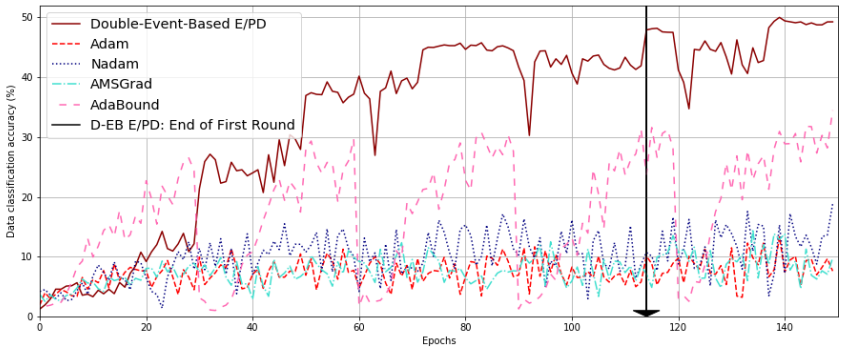
Experimental results on CIFAR-10 are showed in Fig. 10.4, all the curves are generated with the same initial learning rate 0.01. Between 25th and 60th epoch, D-EB E/PD largely outperforms all the counterparts. For the 4 state-of-the-art algorithms, they follow the rule of 60 training epochs per batch, and for our D-EB E/PD algorithm, its learning epochs per batch are dynamic. The vertical line with arrow at 104th epoch indicates that our D-EB E/PD algorithm has finished its first round learning of the whole 5 batches after this epoch. Fig. 10.4 clearly shows that at this epoch, our algorithm is reaching a lower loss value and a higher accuracy comparing to other four curves. There are two reasons that we can achieve this performance: (i) EB E/PD converges very fast, (ii) during these epochs, our D-EB E/PD algorithm have been trained with later batches data, while other 4 algorithms are still working on the first batch data. We can thus conclude that diversity of training data helps to reach better performance.

Results on CIFAR-10 are reported in Table. 10.4, for all algorithms under different initial learning rates. D-EB E/PD reaches a higher final accuracy and lower final loss no matter $\eta(0)$. Even though D-EB E/PD has a higher FASD than AdaBound with $\eta(0) = 0.01$ and $\eta(0) = 0.05$, the FVA($\pm$FASD) range of D-EB E/PD is always higher than the range of AdaBound. Additionally it only takes about 32 to 38 epochs to reach 95% best accuracy in any group. All the indicators are very stable across different groups for D-EB E/PD. One can also note that all 4 state-of-the-art algorithmsl perform very bad with $\eta(0) = 0.05$, as they cannot even reach the 95% best accuracy.

CIFAR-100 results are showed in Fig. 10.5 and reported in Tab. 10.5. According to the FVA, we know that all the algorithms did not totally converge in the end of training process, but that does not influence our conclusion of analysis. D-EB E/PD outperforms other algorithms in almost all the metrics. As the algorithms are not totally converged, the trend of accuracy curve is still increasing, therefore, the higher the initial learning rate, the faster the 1st epoch to reach 95% best accuracy.



(a) Loss value



(b) Accuracy

Figure 10.5 – Performance comparison on CIFAR-100 with $\eta(0) = 0.01$ initial learning rate. Compact view of the results in Tab. 10.5.

Tab. 10.6 shows the results of D-EB E/PD at the end of learning after the first round. It is very important to notice that all the final loss after first round learning in this table is lower than all the state-of-the-art algorithms at the end of their whole training process. Except in CIFAR-100 for $\eta(0) = 0.002$, all the FVA after first round exceed the 95% best accuracy in Tab. 10.4 and Tab. 10.5, respectively. As the learning process on CIFAR-100 is not totally converged (trend can be observed in Fig. 10.5), we can notice that the ending epoch of their first round is near the end of whole training process,

our event-based control did not cut off many epochs. For CIFAR-10, event-based control helps to massively cut off around 62% to 67% training epochs meanwhile guarantee a very good result.

Table 10.4 – Double-Event-Based E/PD algorithm experiments with varying initial learning rate $\eta(0)$ on CIFAR-10. Mean value over 5 runs are reported.

Algorithm	$\eta(0)$	**Final loss**	**FVA** $\pm$**FASD**(%)	**1st epoch to** 80.94%[1]
D-EB E/PD	0.002	**0.58**	**84.50(**$\pm$**0.59)**	**38/300**
Adam	0.002	0.73	84.14($\pm$1.34)	64/300
Nadam	0.002	0.71	83.29($\pm$1.11)	66/300
AMSGrad	0.002	0.67	84.21($\pm$1.65)	65/300
AdaBound	0.002	0.81	84.31($\pm$0.96)	75/300
D-EB E/PD	0.01	**0.61**	**84.83(**$\pm$**1.29)**	**37/300**
Adam	0.01	0.79	83.98($\pm$1.58)	64/300
Nadam	0.01	0.75	84.15($\pm$1.29)	65/300
AMSGrad	0.01	0.65	84.21($\pm$1.50)	72/300
AdaBound	0.01	0.84	79.22($\pm$1.21)	-
D-EB E/PD	0.05	**0.60**	**85.20(**$\pm$**3.14)**	**32/300**
Adam	0.05	5.98	48.93($\pm$14.06)	-
Nadam	0.05	7.74	42.27($\pm$13.95)	-
AMSGrad	0.05	2.69	59.74($\pm$12.43)	-
AdaBound	0.05	1.03	71.49($\pm$1.65)	-

* FVA: Final Validation Accuracy
* FASD: Final Accuracy Standard Deviation
1. 80.94%: 85.20%(best final accuracy among all the experiments)$\times$95%

Table 10.5 – Double-Event-Based E/PD algorithm experiments with varying initial learning rate $\eta(0)$ on CIFAR-100. Mean value over 5 runs are reported

Algorithm	$\eta(0)$	**Final loss**	**FVA (**$\pm$**FASD) (%)**	**1st epoch to** 46.56%[1]
D-EB E/PD	0.002	**2.59**	**45.69(**$\pm$**1.94)**	-
Adam	0.002	3.40	31.29($\pm$3.23)	-
Nadam	0.002	3.18	35.66($\pm$3.35)	-
AMSGrad	0.002	3.13	35.38($\pm$4.02)	-
AdaBound	0.002	3.29	39.87($\pm$4.42)	-
D-EB E/PD	0.01	**2.41**	**48.14(**$\pm$**3.34)**	**111/150**
Adam	0.01	4.94	8.11($\pm$2.04)	-
Nadam	0.01	4.55	9.70($\pm$2.32)	-
AMSGrad	0.01	4.79	8.16($\pm$0.50)	-
AdaBound	0.01	3.51	30.98($\pm$3.08)	-
D-EB E/PD	0.05	**2.38**	**49.01(**$\pm$**10.52)**	**100/150**
Adam	0.05	4.72	2.64($\pm$0.58)	-
Nadam	0.05	4.74	1.88($\pm$0.79)	-
AMSGrad	0.05	4.68	1.98($\pm$0.56)	-
AdaBound	0.05	3.69	19.03($\pm$2.42)	-

* FVA: Final Validation Accuracy
* FASD: Final Accuracy Standard Deviation
1. 46.56%: 49.01%(best final accuracy among all the experiments)$\times$95%

Table 10.6 – Double Event-Based E/PD experiments on CIFAR-10 and CIFAR-100 in the End of First Round. Mean value over 5 runs are reported.

Dataset	$\eta(0)$	**EE of FR**[1]	**FL after FR**[2]	**FVA after FR**[3] **(%)**
CIFAR10	0.002	99/300	0.60	82.47
CIFAR10	0.01	104/300	0.62	82.36
CIFAR10	0.05	113/300	0.62	82.75
CIFAR100	0.002	148/150	2.61	44.98
CIFAR100	0.01	148/150	2.44	48.04
CIFAR100	0.05	146/150	2.41	48.95

1. EE of FR: End Epoch of First Round
2. FL after FR: Final loss after First Round
3. FVA after FR: Final Validation Accuracy after First Round

10.3.5 Trade-offs and limitations

The addition of event-based mechanisms improves the performance in terms of final accuracy and loss, however at the cost of two sacrifices:

- (i) Event-Based Learning Epochs accelerate the speed of learning each data batch, however, if we are not allowed to keep in cache any data batch locally, i.e. only allowed to learn each data batch once, the performance of Double Event-Based E/PD after first round is slightly worse than the performance after all the training epochs.

- (ii) Double Event-Based E/PD will cyclically learn all data batches, and it will need to load and unload data batch more times than classical on-line learning setting. Loading (unloading) data into (from) memory needs time. These are extra costs for Double Event-Based E/PD, however negligible compared to the computing intensity of CNNs.

Regarding the limitation of the presented D-EB E/PD, we identified one potential case for which our algorithm will fail: if the training data contains mislabeled data. These data will lead the model to converge to a wrong optimum, and as the algorithm minimizes faster the loss function, it will be over-fitting to the noisy data faster than other algorithms. However, this fail is caused by poor data selection, and is not specific to our algorithm.

In order to continue this work in the following part of the manuscript, we will try some mechanisms for dealing with mislabelled data during the training phase of a CNN.

10.4 Conclusion and Future Work

Due to the limitation of computing resource or short interval time between two data batches, convergence speed of the loss and accuracy becomes especially important for on-line learning. E/PD control is a powerful learning rate algorithm when training neural network on an on-line learning scenario. Based on E/PD, this paper proposes two algorithms: (i) Event-Based Learning Rate algorithm and (ii) Event-Based Learning Epochs algorithm.

The new algorithm firstly introduces an Event-Based control on PD phase of E/PD, in order to prevent the learning rate to decrease too much. Second Event-Based control is implemented to inspect the record of the loss value. If the loss record has the tendency to increase, showing little learning efficiency, we will drop the rest learning epochs for current data batch.

Results show that Double-Event-Based E/PD can massively cut off training epochs, and even results in a lower loss value. For instance with CIFAR-10 dataset, it could save up to 67% training epochs.

As the Event-Based Learning Epochs control is independent from learning rate algorithm and dataset, this work could be further extended by implementing this control with language, image and numeric datasets on time-based decay SGD, Adam, Nadam, AMSGrad and AdaBound learning rate algorithms, to prove that by simply adding this event-based control, all the learning rate algorithms on any dataset can improve their performance on on-line learning scenario.

Chapter 11

Conclusion on Learning Rate Control

This ending chapter of Part. III highlights this thesis contributions with regards to the dynamic control of Learning Rate in a learning algorithm.

Image classification is the task of classifying an image into a class category. It is the most well-known computer vision task. Convolutional Neural Network (CNN) has become the most used method for image classification tasks. CNN's usage is not limited for computer vision area, the last proposed algorithm Temporal Convolutional Network (TCN) [82] is the new trend for time series predictions, and the time series study is also a rapidly evolving field in econometrics. During the training of CNN, the learning rate and the gradient are two key factors to tune for influencing the convergence speed of the model. Usual learning rate strategies are time-based i.e. monotonous decay over time. But recent state-of-the-art techniques focus on adaptive gradient algorithms i.e. Adam and its versions. The disadvantage of time-based algorithms is that the learning rate path is fixed before training, it cannot be adjusted when necessary. The shortcomings of adaptive gradient algorithms are that adaptive gradient methods do not generalize as well as SGD, these methods tend to perform well in the initial portion of training but are outperformed by SGD at later stages of training. As we consider a continual learning scenario, neither the time-based learning rate algorithms nor the adaptive gradients methods considers the variation between the data batches. For instance the face recognition task, the distribution of race and sex of one data batch can be largely different from another data batch. Apparently time-based algorithms can not response to these problems, and as Adam, Nadam and AMSGrad all use the decayding averages of past gradients, therefore the calculated gradient can not show the strong reflection from the new data batch.

To cope with the above challenges, we advocate switching to a performance-based adaptation, in order to improve the learning efficiency. We present E (Exponential)/PD (Proportional derivative)-Control, a learning rate strategy that combines a feedback PD controller based on the CNN loss function, with an exponential control signal to smartly boost the learning and adapt the PD parameters. We compare the E/PD results with time-based algorithms, E/PD can not only result in a higher final accuracy, but also a more stable learning curve.

To continue improving E/PD under continual learning scenario, Event-Based Learning Rate algorithm is introduced on the PD phase, to prevent the learning rate to decrease during this period. Results show the Event-Based E/PD performs better than the E/PD in all the metrics.

Observing that the Event-Based E/PD sharply increased the converging speed for each data batch, we conclude that the fixed batch size does not seem appropriate any more as we should cut off the inefficient training when there is no obvious improvement. Under such circumstances, an Event-Based Learning Epochs algorithm is also proposed to inspect the record of the loss value. If the loss record has the tendency to increase, showing little learning efficiency, we suggest to drop the rest learning epochs for the current data batch. Results show that incorporate two event-based control

with E/PD (Double-Event-Based E/PD) can massively cut off training epochs and even result in a lower loss value. For instance with CIFAR-10 dataset, it could save up to 67% training epochs.

Though Double-Event-Based E/PD can reach to higher final accuracy, it does not come without a cost, as we allow the algorithm to decide when to pass to next batch, it will therefore load (unload) more frequently into (from) memory than the case of fixed training epochs. However if the data batch size is big, that will also add training time.

Regarding the perspectives, we see a potential capacity to incorporate Event-Based Learning Rate and E/PD into adaptive gradient methods. From Sec. 8.4, we can see that Adam and its extensions all focus on modifying the calculation of gradients, its step size η is either fixed or time-based decay. If the merged algorithm can leverage the advantages from two parts, it may outperform either of them.

As the Event-Based Learning Epochs control is independent from learning rate algorithm and dataset, it could be further implemented with language, image and numeric datasets on time-based decay SGD, Adam, Nadam, AMSGrad and AdaBound learning rate algorithms, to prove that by simply adding this event-based control, all the learning rate algorithms on any dataset can improve their performance on continual learning scenario.

Recall the limitations we mentioned in Chapter 10.3.5, once the training data contains the mislabelled data, our proposed learning rate algorithm will actually accelerate the over-fitting on the wrong data. To address this problem, we introduce further studies in the next part.

Part IV

On-line Learning from Highly Unreliable Data: Anomaly Detection

The third area of contribution of this thesis is on learning models from dirty label data. When we do system identification (in control engineering) or machine learning (in computer science), the common assumption is that the data source is clean, i.e., features and labels are correctly set. However, data collected from the wild can be unreliable due to careless annotations (e.g. user uploaded image with random hashtags on Facebook or Instagram) or malicious data transformation (e.g. cyber attack of the auto-labelling system). Therefore, in this part, we propose new algorithms to train models from highly unreliable data. To show the utility of our algorithms, we applied the developed algorithms on three use cases in anomaly detection area.

Chapter. 12 introduces the background of noisy data learning and anomaly detection. It presents the three use cases that we have used to test our algorithms. The continual learning setting is also detailed in this chapter. Finally, three state of the art algorithms which we will compare with are explained.

Chapter. 13 presents our first algorithm: RAD, which is a two-layer on-line data selection framework for robust anomaly detection (RAD). The first layer is to filter out the suspicious data, and the second layer detects the anomaly patterns from the remaining data. The motivations behind this study are firstly illustrated, then experiments on IoT and Cluster use cases are implemented. Results show that our algorithm can resist the influence of noisy label data on training process. Limitations of the RAD algorithm are discussed in the end.

Chapter. 14 responds the highlighted limitations in the end of Chapter. 13. We extend RAD with additional features of conflicting opinions of classifiers, repetitively cleaning, and oracle knowledge, namely RAD Voting and RAD Active Learning. They massively improve the RAD on IoT and Cluster use cases. Additionally, we extend RAD Active Learning as RAD Slim to deal with image dataset, and evaluate the algorithm on a face recognition use case. RAD Active Learning and RAD Slim need to consult the uncertain data to an expert/oracle, but in reality we could not have unlimited consultations to the expert, so we proposed two variations RAD Active Learning Limited and RAD Slim Limited, which impose limit on the number of consultation per batch. Due to this limitation, we propose also two additional strategies: Highest Disagreement method and Highest Loss method to rank the uncertainty of data in RAD Active Learning Limited and RAD Slim Limited.

Chapter. 15 summarizes the contributions in perspective to the noisy label data learning. The limitations are drawn and ideas for future works are proposed.

Chapter 12

Noisy Data Learning and Anomaly Detection: Background and Related Works

Classification algorithms have been widely adopted to detect anomalies for various systems, under the common assumption that the data source is clean, i.e., features and labels are correctly set. However, data collected from the wild can be unreliable due to careless annotations or malicious data transformation for incorrect anomaly detection, It is always challenging to learn from noisy labels, since these labels are systematically corrupted. As a negative effect, noisy labels inevitably degenerate the accuracy of classifiers. This negative effect becomes more prominent for continual learning (which is still the experiment setting we use in this Part) scenario and deep neural network, since influence of noisy data can be left in the models.

In this chapter, we firstly introduce the notion of noisy data learning, and the core challenges in Sec. 12.1. Then we present three datasets which we will use in the following chapters to evaluate our proposed algorithms againt state of the art algorithms.

12.1 Introduction of Anomaly Detection and Noisy Data Learning

Machine learning has been extensively used for failure detection [107, 106, 114, 21], attack prediction [1, 9, 3, 75, 71, 147], and face recognition [128, 120, 137]. Considering noisy data in classification algorithms is also a problem that has been explored in the machine learning community as discussed in [44, 14, 101]. In general, when we talk about noisy data, it could be two kind of noise on data: (1) the features captured can be incorrect or (2) the labels collected can be corrupted. All over the context of this thesis, we specifically refer to the latter situation. Fig. 12.1 shows an example of that latter case. Standard machine learning algorithms typically assume clean labels and overlook the risk of noisy labels. Recent studies point out the increasing dirty data attacks that can maliciously alter the anomaly labels to mislead the machine learning models [70, 41, 64]. To handle such noisy labels, recent approaches fall in three main categories:

- One direction focuses on training only on selected samples, which leverages the sample-selection bias [65] to overcome the label noise issue. The representative works are MentorNet [68], Decoupling [92] and Co-teaching [59].

- Second direction develops regularization methods, including explicit and implicit regularizations. They implement the regularization bias to overcome the label noise issue. Explicit regularization is added to the objective function, such as manifold regularization [11] and virtual

Figure 12.1 – Noisy Label Data: Incorrectly Label a Tiger as Cat

adversarial training [98] . Implicit regularization is designed for training algorithms, such as temporal ensembling [79] and mean teacher [130]. Nevertheless, both approaches introduce a permanent regularization bias, and the learned classier barely reaches the optimal performance [31].

- The last direction estimates the noise transition matrix without introducing sample-selection bias and regularization bias. As an approximation of corruption of the real world, noisy labels are theoretically flipped out of the ground-truth labels by an unknown noise transition matrix. In this way, the accuracy of classifiers can be improved by estimating this matrix accurately. The previous methods for estimating the noise transition matrix can be roughly summarized into two solutions:

  – (i) One solution is to estimate the transition matrix before training the model, and subsequently use this transition matrix to correct loss function [104].

  – (ii) Another is to estimate the transition matrix during the training process. For instance, adding a constrained linear "noise" layer on top of the softmax layer which adapts the softmax output to match the noise distribution [126], or adds a nonlinear softmax [52].

  The problem for this last solution is that data is finite, training epochs are finite, there exists the uncertainty how long will the algorithm take to well estimate the transition matrix.

12.2 Anomaly Detection Datasets

In Part. IV, all our experiments will be examined on following three datasets. They represent three types of anomaly detection jobs: (1) failure detections, (2) attacks predictions and (3) face recognitions. First two datasets are tabular data, the third one is image data, the reason we choose these three datasets is because under these three situations, mislabelled data can cause serious problems. We did not use an economic dataset since we did not find one complying with our use case. Noisy label can also hurt the analysis result of economic problem, but since most of the economic dataset (especially

macroeconomic dataset) are published by government or international organisations, the possibility that the data containing noisy label is relative low.

Google Cluster Tasks Failure Dataset The cluster task traces comprise data instances each corresponding to a task with 27 features capturing information related to static and dynamic system states, e.g. the task start/end times, the task resource utilisations, the hosting machine, etc. Each class is labeled based on its scheduling state. A detailed description of the features and labels can be found in [112]. In particular, we are interested in the four possible termination classes: *finish*, *fail*, *evict*, or *kill*. We filter out other classes. The resulting class distribution is dominated by successful tasks (*finish*) 77.8%, followed by *kill* 22.0%, *fail* 0.2%, and *evict* <0.1%. Similar to [117], we aim to predict the task outcome to reduce the resource waste and improve the overall scheduling and system performance, e.g., in case of lack of resources and need to kill a task, help choosing the task with the least probability to succeed.

IoT Thermostat Device Attacks Dataset The IoT dataset comprises data instances collected from thermostat device describing 23 network packet-level statistics recursively computed over five different time scales totalling to 115 features. This traffic statistics are collected during normal operation, labeled as benign, or under one of ten different malicious attacks stemming from devices infected by either the *BASHLITE* or *Mirai* malware. Malicious traffic covers mainly scanning for vulnerable devices and various flooding attacks. More details are provided in [94]. We aim to apply RAD to build a noise-resistant model to categorize the attacks for post fact analysis, e.g., for threat assessment.

Celebrities Face Recognition Dataset The FaceScrub [103] dataset is used for face recognition. Original FaceScrub contains more than 100,000 face images of 530 people, with about 200 images per person. Male and Female images are almost equal. We use a subset of 12K FaceScrub images to fit the limits of our compute resources. The 12K images cover the 100 people which have the highest number of images, 55 males and 45 females. FaceScrub images were retrieved from the Internet and are taken under real-world situations (uncontrolled conditions). We resize all images to 64*64 pixels. While name is the only annotation we use. The face recognition system has been widely used in security equipment.

Cluster and IoT datasets will be used in Chapter. 13 and Chapter. 14. Face recognition dataset will only be used in Chapter. 14.

12.3 Continual Learning

In this part of thesis, we are still in the continual learning scenario designed in Sec. 8.2. First of all, as we are training the models with noisy data, we will not directly use the whole data batch, the data will pass a selection process to steam the noisy data out. Secondly, it also depends on the datasets we use.

Continual Learning for IoT and Cluster Dataset The main difference comparing to the continual learning setting in. 8.2 is that for IoT and Cluster datasets, we will not only use the data from current data batch, but also from all the previous data batches. Imagine $\mathscr{D}^*_i$ is the selected data from i^{th} data batch $\mathscr{D}_i$, then to train the models, we will use $\mathscr{D}^*_1 \dots \mathscr{D}^*_i$.

Continual Learning for FaceScrub Dataset For FaceScrub dataset, the continual learning design is the same as in Sec. 8.2. The reason that we do not use whole accumulated data to the model is

due to the computation overhead. Because of the complex characteristics of image data, if we use all accumulated data, that will increase a lot of computation to train the convolutional neural network. Facescrub data will thus use only $\mathscr{D}^*{}_i$ at batch i, where $\mathscr{D}^*{}_i$ is the selected data from i^{th} data batch $\mathscr{D}_i$.

12.4 Related Works

We select three state of the art algorithms as the comparisons to our proposed algorithms. They are: (i) IDS (Intrusion Detection System) [1], (ii) Forward [104] and (iii) Co-Teaching [59].

- **IDS (Intrusion Detection System)** is a machine-learning based pre-trained model, it is designed to detect the existence of flooding DoS attacks with a high accuracy (precision) and detection rate (recall). As the IDS will not update during the anomaly detection process, the success of IDS are highly related to the choice of model and the size of available clean dataset. As we are working on continual learning setting, in order to have fair comparison with our algorithm, we need to adapt IDS to the on-line learning scenario. We use IDS as a protection shell, and we train a separate classification model. During the continual learning, every time when there is a new data batch, we use IDS to filter out noisy data, and select the "clean" data. The separate classification model will only use the "clean" data to update itself. The selection process details will give in Chapter. 13.

- **Forward** estimates the noise transition matrix T before training the model, and subsequently use this transition matrix for loss correction. The process is showed in Algorithm. 2. The symbol $l^{\rightarrow}$ means the $forward$ correction as it multiplies the network predictions by $\hat{T}$.

Algorithm 2 Robust two-stage training

Input: the noisy training set S, any loss l
If T is unknown:
  Train a network $h(x)$ on S with l
  Obtain an unlabeled sample X'
  Estimate $\hat{T}$ by Eq. (12.1) and (12.2) on X'
Train the network $h(x)$ on S with $l^{\rightarrow}$
Output: return $h(\cdot)$

$\hat{p}(y|x)$ is the softmax output, it can be interpreted as a vector approximating the class-conditional probabilities. In supervised c-class classification, one has feature space $\mathscr{X} \in \mathbb{R}^d$ and label space $\mathscr{Y} = e^i : i \in [c]$, where e^i denotes the ith standard canonical vector in $\mathbb{R}^c$ by, i.e. $e^i \in 1,0^c$, $1^T \cdot e^t = 1$.

$$\vec{x}^i = argmax_{x \in X'} \hat{p}(\hat{y} = e^i|x) \tag{12.1}$$

$$\hat{T}_{ij} = \hat{p}(\hat{y} = e^j|\vec{x}^i) \tag{12.2}$$

To summarize, Eq. (12.1) and Eq. (12.2) try to find a 'perfect example' for each class, and the prediction probability vector of this example will be used as the probability distribution vector for this class, implementation details can be found in this git repository [14]. the problem of this

[14] https://github.com/giorgiop/loss-correction

method is that it depends on an **already trained** model. In the ideal case, the classifier should be powerful enough to only make mistakes due to label noise, but in reality, if we have already such a classifier, it would not be needed to train another model with noisy label data. Therefore in general, the estimation of $\hat{T}$ is not that accurate, which is a big disadvantage of forward loss correction.

To speed up the model convergence for Forward, we implement the E (Exponential)/PD (Proportional-Derivative)-Control [143] and Event-Based Control Learning rate [145] as learning rate schedule based on SGD (stochastic gradient descent) optimizer.

- **Co-Teaching** is a deep learning paradigm for combating with noisy labels. Namely, we train two deep neural networks simultaneously, and let them teach each other given every mini-batch: firstly, each network feeds forward all data and selects some data of possibly clean labels; secondly, two networks communicate with each other what data in this mini-batch should be used for training; finally, each network back propagates the data selected by its peer network and updates itself. The schema of data flow is showed in Fig. 12.2.

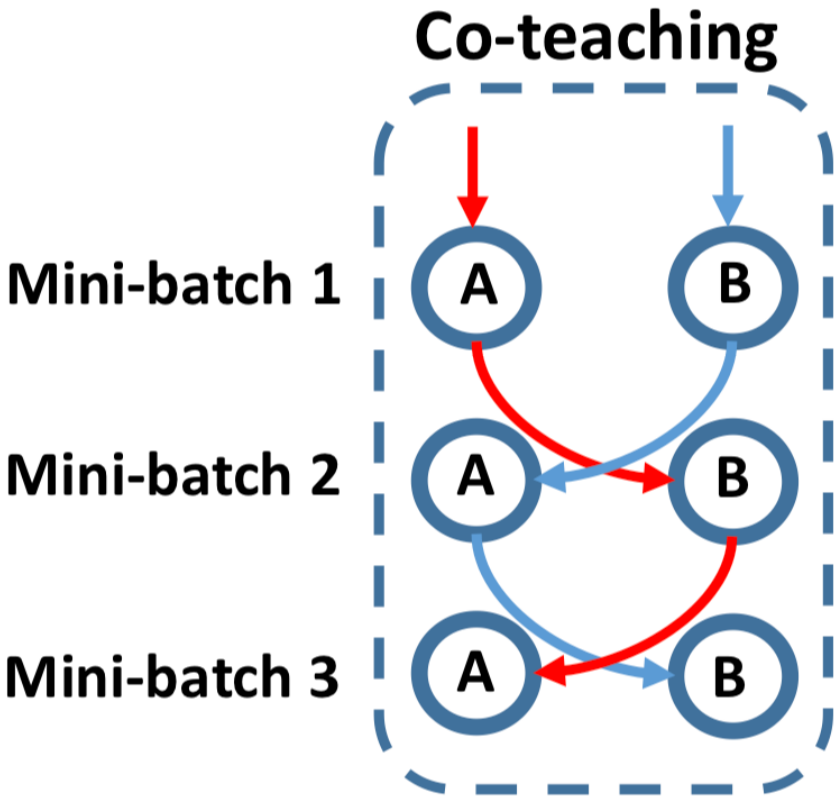


Figure 12.2 – Co-teaching maintains two networks (A & B) simultaneously. In each mini-batch data, each network samples its small-loss instances as the useful knowledge, and teaches such useful instances to its peer network for the further training. (Source: [59])

IDS, Forward and Co-Teaching will be used as comparisons in Sec. 13.4 and Sec. 14.4.

Chapter 13

Robust Anomaly Detection on Unreliable Data

This chapter presents a two-layer learning framework for robust anomaly detection (RAD) in the presence of unreliable anomaly labels. The first layer of quality model filters the suspicious data, where the second layer of classification model detects the anomaly types. We specifically focus on two use cases, (i) detecting 10 classes of IoT attacks (Sec.12.2) and (ii) predicting 4 classes of task failures of big data jobs (Sec.12.2). We continuously learn from the incoming data streams and cleanse the data. So as to adapt to the increasing learning capacity from the larger accumulated data set (Sec. 12.3). Moreover, we propose an *ensemble prediction* strategy to reconcile the prediction outcomes of two models, namely label quality model and anomaly classification model.

Sec. 13.2 describes the motivating case studies that we consider. Sec. 13.3 presents the proposed RAD framework and Sec. 13.4 details the results of its experimental evaluation and the limitations of RAD. Finally, Sec. 13.5 draws our conclusions and the lessons learned.

13.1 Introduction

Anomaly detection is one of the core operations for enforcing dependability and performance in modern distributed systems [140]. Anomalies can take various forms including erroneous data produced by a corrupted IoT device or the failure of a job executed in a datacenter [16, 15].

Dealing with this issue has often been done in recent art by relying on machine learning-based classification algorithms over system logs [42, 49]. These systems often rely on a learning dataset from which the classifier learns to distinguish between data corresponding to a correct execution of the system from data corresponding to an abnormal execution of the latter (i.e., anomaly detection).

In this context, a rising concern when applying classification algorithms is the accessibility to a reliable ground truth for anomalies [24]. Typically, anomaly data is manually annotated by human experts and hence the generation of anomaly labels is subject to quality variation, so-called noisy labels. For instance, annotating service failure types for data centers is done by operators.

However, standard machine learning algorithms typically assume clean labels and overlook the risk of noisy labels. Moreover, recent studies point out the increasing dirty data attacks that can maliciously alter the anomaly labels to mislead the machine learning models [70, 41, 64]. As a result, anomaly detection algorithms need to capture not only anomalies that are entangled with system dynamics but also the unreliable nature of anomaly labels.

Indeed, a strong anomaly classification model can be learned by incorporating a larger amount of datasets, however learning from data with noisy labels can significantly degrade the classification accuracy, even for deep neural networks, at a non-negligible computation source [134]. Such a con-

cern leads us to ask the following question: how to build an anomaly detection framework that can robustly differentiate the true and noisy anomalies and efficiently learn the anomaly classification models from a succinct amount of clean data. The immediate challenge of capturing the dynamics of data quality lies at the fact that label qualities are not directly observable but only via anomaly classification outcomes that in turn is coupled with the noise level of data labels.

In this chapter, we develop a Robust Anomaly Detector (RAD), a generic framework that continuously learns the anomaly classification model from streams of event logs that are subject to label noises. To such an end, RAD is composed of two layers of learning models, i.e., data label model and anomaly classifier. The label model mainly aims at differentiating the label quality, i.e., noisy v.s. true labels, for each batch of new data and only "clean" data points are fed in the anomaly classifier. The major job of anomaly classifier is to predict the coming-out event, that can be in multiple classes of (non)anomalies, depending on the specific anomaly use case. Both the prediction from label model and anomaly classifier contribute to the final decision of anomaly detection, using ensemble prediction technique. The specific choices of label models and anomaly classifier include standard machine learning models, e.g., random forest, Adaboost, and discriminant analysis, and deep neural networks.

To demonstrate the effectiveness of RAD, we consider two use cases, i.e., detecting 10 classes of IoT attacks [94], and predicting four types of task failures for big data processing cluster [112, 116] from open datasets. Our preliminary results show that RAD can effectively and continuously cleanse the data, i.e., selecting data streams with clean labels, and result better anomaly detection accuracy per additional data stream included, compared to classifiers without continuous data cleansing. Specifically, under 30% noise, RAD achieves up to 99.01% and 85.46% accuracy for detecting IoT device attacks and predicting cluster task failures respectively.

13.2 Motivating case studies

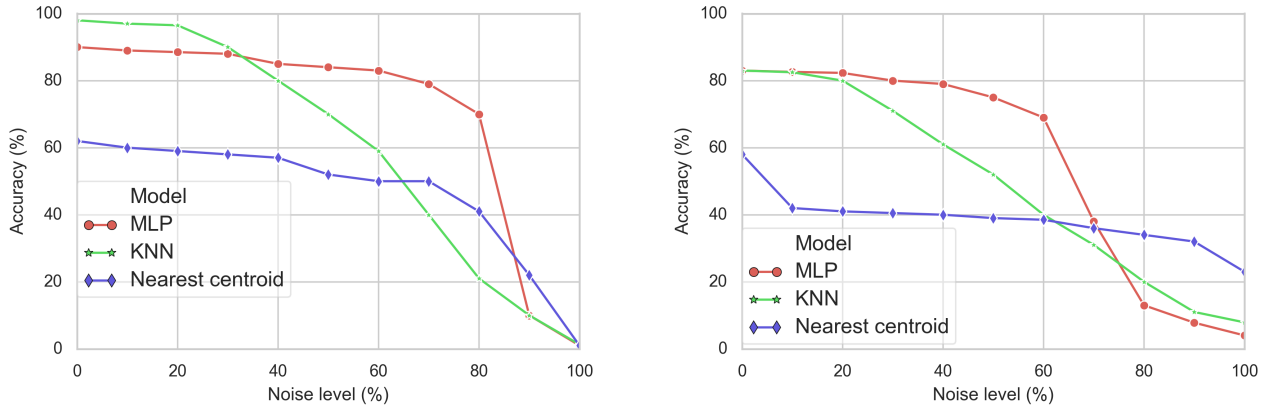
To qualitatively demonstrate the impact of noisy data on anomaly detection, we use two case studies.

- Detecting **IoT device attacks** from inspecting network traffic data collected from commercial IoT devices [94]. This dataset contains nine types of IoT devices which are subject to 10 types of attacks. Specifically, we focus on the Ecobee thermostat device that may be infected by Mirai malware and BASHLITE malware. Here we focus on the scenario of detecting and differentiating between 10 attacks. It is important to detect those attacks with high accuracies against all load conditions and data qualities.

- Predicting **task execution failures** for big data jobs running at a Google cluster [112, 117]. This trace contains a month-long jobs execution record from Google clusters. Each job contains multiple tasks, which can be terminated into four different states: *finish*, *fail*, *evict*, or *kill*. The last three states are considered as anomaly states. To minimise the computational resource waste due to anomaly states, it is imperative to predict the final execution state of task upon their arrivals.

The details about data definition, and statistics, e.g., number of feature and number of data points, can be found in Sec. 13.4.1. To recognize anomalies/faces in each use case, related studies have applied different machine learning classification algorithms, from simple ones, e.g., k-nearest neighbour (KNN), to complex ones, e.g., deep neural networks (DNN), under scenarios with different levels of label noise. Here, we evaluate how the detection accuracy changes relative to different levels of noises. We focus on off-line scenarios where we split the data in a training set affected by label noise and a clean evaluation set.

Classification models are learned from 14,000 training records and evaluated on a clean testing dataset of 6,000 records. We specifically apply KNN, nearest centroid and multilayer perceptron (MLP) (a.k.a feed-forward deep neural networks) on both the IoT device attacks and the cluster task failures. Fig. 13.1a and Fig. 13.1b summarize the accuracy results.



(a) Use case of IoT thermostat device attacks

(b) Use case of Cluster task failures

Figure 13.1 – Impact of noisy data on anomaly classification

One can see that noisy labels clearly deteriorate the detection results for both IoT attacks and task failures, across all three classification algorithms. For standard classifiers, like KNN and nearest centroid, the detection accuracy decays faster than MLP that is more robust to the noisy labels. Such an observation holds for both use cases. In IoT attacks, MLP can even achieve a similar accuracy as the case of no label noises, when 50% of label classes are altered.

Above two experiments clearly show that under the presence of noisy label data, all the models are corrupted. The stronger the noise, the worse the model's accuracy. These cases motivate us to design the RAD framework and its extension. To resist the influence of noisy label data on learning process.

13.3 Design Principles of RAD Framework

In this section, we first introduce the general structure of RAD and its extended features with respect to data selection and model prediction – ensemble prediction. All the symbols used to explain the designs are summarized in Tab. 13.1.

13.3.1 System Model

We consider a dataset that consists of several data instances. Each data instance has f features. Each data instance belongs to a class k, where $k \in \mathscr{K} = \{1, \ldots, K\}$. Data instances are pre-labeled dataset $\mathscr{D}$ with labels Y used for training. Furthermore, a labeled data instance is either correctly labeled (i.e., clean data instance), or incorrectly labeled (i.e., noisy data instance). We use the indicator variable $\hat{q}$ to indicate clean $\hat{q} = 1$ and dirty $\hat{q} = 0$ labels. Wrong labels can stem from several reasons ranging from subjectivity, and data-entry errors, to malicious error injection. The quality of a dataset $\mathscr{D}$ is measured as the percent of clean labeled data instances, denoted here as $\tilde{Q}$.

Data instances arrive at the learning system continuously over time in batches. $\mathscr{D}_i$ denotes the batch of labeled data arriving at time t_i and having labels Y_i. In general we denote the time win-

Table 13.1 – Symbol description

Symbol	Description
$\mathfrak{L}$	label quality predictor
$\mathfrak{C}$	anomaly detection classifier
$\mathscr{D}_i$	ith training data batch
$\mathscr{D}_i^*$	ith cleansed data batch from $\mathfrak{L}$
$\mathscr{P}_i$	ith test data batch
$\hat{Y}_i$	prediction of ith test data batch from $\mathfrak{C}$
$\tilde{Q}_i$	percent of clean labeled data of ith batch
$\mathscr{U}_i$	"unclean" data of ith batch determined by $\mathfrak{L}$
$\mathscr{U}_i^*$	ith cleansed data batch from $\mathfrak{C}$
S_i	"unclean" data of ith batch determined by $\mathfrak{C}$
S_i^*	data with true label from Expert of ith batch
$\hat{p}$	indicator of prediction, 1 for clean, 0 for dirty
$\hat{q}$	indicator of prediction, 1 for clean, 0 for dirty

dow with the subscript i. We assume that a small initial batch of data instances $\mathscr{D}_0$ has only clean labels, that is $\tilde{Q}_0 = 100\%$. Subsequent batches, include varying proportions of noisy labels, i.e $0 < \tilde{Q}_i < 100\%, i > 0$. For simplicity we consider arriving batches of equal size, $\forall \mathscr{D}_i, |\mathscr{D}_i| = N$, but not necessarily at regular times.

A classification request consists of a batch of non-labeled data instances $\mathscr{P}_i$ for which the classifier predicts the class k of each data instance. At each batch arrival, the classification output $\hat{Y}_i$ is thus an array of the predicted classes for each non-labeled data instance.

13.3.2 Design Overview of RAD

We propose the RAD learning framework. Its objective is threefold:

(1) Accurately learn models from noisy data.

(2) Continuously update the learned models based on new incoming data.

(3) Propose a general approach that caters to different machine learning algorithms and different application use cases.

RAD is composed of two key steps: training data selection and prediction techniques. Training data selection part focus on how to filter out suspiciously noisy data and solicit "clean" data to train the classification models subsequently. The prediction part combines different prediction models.

Fig. 13.2 describes the overall architecture of RAD training data selection. it consists of two main components: a label quality model $\mathfrak{L}$ mainly aims at discerning clean labels from dirty labels and a classifier model $\mathfrak{C}$ targets the specific classification task at hand. Both models will be used to do ensemble predictions as described in Sec. 13.3.2.

RAD follows a generic approach since the proposed classification framework can be used with any supervised machine learning algorithm, such as SVM, KNN, random forest, nearest centroid, DNN, etc. Moreover, RAD can be applied to a large spectrum of different applications where noisy data are collected and must be cleansed before used to train the classification model. Examples are the failure detection, attack diagnosis and face recognition illustrated in Sec. 13.4.

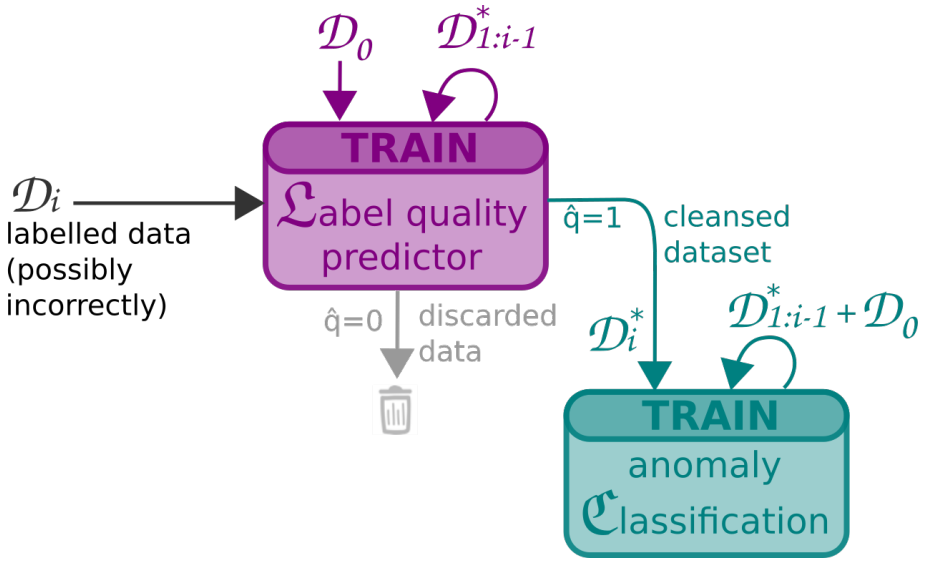


Figure 13.2 – RAD training data selection framework. Each block is a machine learning algorithm. Data used to train is represented by colored arrows from the top. The flowchart is iterated at every batch arrival with new labelled and unlabelled data coming in (black arrows on the left). The labelled training data for $\mathfrak{C}$ is cleansed based on the label quality predicted by $\mathfrak{L}$.

Data Selection Scheme The first component of RAD aims to select clean data instances from $\mathscr{D}$ through the quality model. The objective of the label quality model is to select the most representative data instances to train a strong classifier model. It solicits data instances with clean labels, avoiding the pitfall that the classifier overfits the noise. RAD uses supervised-learning algorithms to continuously train the label quality model from accumulated predicted clean data instances, to build a strong classifier.

We term the following selection procedure as "basic," that is the default data selection scheme of RAD and requires no addition history data lookup nor involvement of human experts. $\mathfrak{L}_{i-1}$ is the label quality model that is trained with data instances received up to time $t_i - 1$, that is $\mathscr{D}_0 \ldots \mathscr{D}_{i-1}$. Upon the arrival of a new batch of data instances $\mathscr{D}_i$ at time t_i, we use the currently learned label quality model $\mathfrak{L}_{i-1}$ to predict the label quality $\hat{q}$ for each data instance in $\mathscr{D}_i$ by comparing the given k and predicted class $\hat{k}^{\mathfrak{L}_i}$. If they coincide, we consider the label as clean $q = 1$, otherwise as dirty $q = 0$. Then we build $\mathscr{D}^*{}_i$ as the subset of data instances from $\mathscr{D}_i$ with $q = 1$ and discard the instances with $q = 0$. This data flow is summarized in Algorithm. 3.

Algorithm 3 RAD

Input: Data batch $\mathscr{D}_i$ and its given label Y_i, label quality model $\mathfrak{L}_{i-1}$, anomaly classification model $\mathfrak{C}_{i-1}$

Output: $\mathfrak{L}_i$, $\mathfrak{C}_i$

1: Predict $\mathscr{D}_i$ by $\mathfrak{L}_{i-1}$, get prediction label $Y_i^{\mathfrak{L}}$.
2: Compare Y_i and $Y_i^{\mathfrak{L}}$, extract the data where predictions in Y_i and $Y_i^{\mathfrak{L}}$ are identical, combine them as $\mathscr{D}_i^*$.
3: $\mathfrak{L}_{i-1}$ sends $\mathscr{D}_i^*$ to $\mathfrak{C}_{i-1}$ (if two models are not in the same location), two model catch $\mathscr{D}_i^*$ locally. Discard other data.
4: $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$ use all accumulated $\mathscr{D}_t^*$ t ∈ [0,i] to train the model, get $\mathfrak{L}_i$ and $\mathfrak{C}_i$
5: **return** $\mathfrak{L}_i$, $\mathfrak{C}_i$

Generic Approach to Handle Dynamic Data The second component of RAD is the dynamic data classifier $\mathfrak{C}$, whose data input has dynamic noise ratios. $\mathfrak{C}_i$ is trained on all the predicted clean data instances $\mathscr{D}^*$ received until time t_i, that is $\mathscr{D}^*{}_0 \ldots \mathscr{D}^*{}_i$. We assume that $\mathscr{D}_0$ contains only clean data instances to kickstart the framework and use the label quality model $\mathfrak{L}_0 \ldots \mathfrak{L}_{i-1}$ to cleanse $\mathscr{D}_1 \ldots \mathscr{D}_i$ and produce $\mathscr{D}^*{}_1 \ldots \mathscr{D}^*{}_i$. Thus, the RAD framework uses the batch-by-batch updated data label quality model to enrich the training data of the classification model.

Prediction Techniques Fig. 13.3 shows the structure of ensemble prediction, which combines the prediction outcomes of both quality and classification models. The merging decision leverages the confidences from the output probability vectors and the test accuracy of two models from last training epoch. We provide the details in Algorithm. 4.

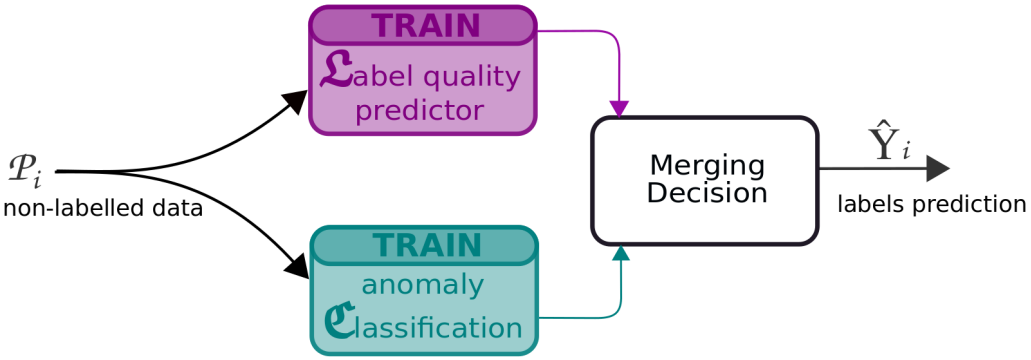


Figure 13.3 – Ensemble Prediction.

Algorithm 4 Ensemble Prediction

Input: Data $\mathscr{P}_i$, label quality model $\mathfrak{L}_i$, anomaly classification model $\mathfrak{C}_i$, testing accuracy of $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$: $acc^{\mathfrak{L}}_{i-1}$, $acc^{\mathfrak{C}}_{i-1}$. Conv(): convert probability vector to category (class). Max(): return maximum value in a vector.

Output: Predicted label $\hat{Y}_i$

Predict $\mathscr{P}_i$ by $\mathfrak{L}_i$ and $\mathfrak{C}_i$, get prediction lists $Y_i^{\mathfrak{L}\mathscr{P}}$ and $Y_i^{\mathfrak{C}\mathscr{P}}$. They are in the same length (length of $\mathscr{P}_i$ ($|\mathscr{P}_i|$)). Each element of prediction list is a probability vector on length of number of classes, sum of all bits is 1.
Initialize an empty list $\hat{Y}_i$ of length $|\mathscr{P}_i|$
for $k \in \{1,2,\ldots,|\mathscr{P}_i|\}$ **do**
    if Conv($Y_i^{\mathfrak{L}\mathscr{P}}$[k]) == Conv($Y_i^{\mathfrak{C}\mathscr{P}}$[k]) **then**
        $\hat{Y}_i$[k] $\leftarrow$ Conv($Y_i^{\mathfrak{C}\mathscr{P}}$[k])
    end if
    if Conv($Y_i^{\mathfrak{L}\mathscr{P}}$[k]) != Conv($Y_i^{\mathfrak{C}\mathscr{P}}$[k]) **then**
        if $acc^{\mathfrak{C}}_{i-1}\times$ Max($Y_i^{\mathfrak{C}\mathscr{P}}$[k])>$acc^{\mathfrak{L}}_{i-1}\times$ Max($Y_i^{\mathfrak{L}\mathscr{P}}$[k]) **then**
            $\hat{Y}_i$[k] $\leftarrow$ Conv($Y_i^{\mathfrak{C}\mathscr{P}}$[k])
        else
            $\hat{Y}_i$[k] $\leftarrow$ Conv($Y_i^{\mathfrak{L}\mathscr{P}}$[k])
        end if
    end if
end for
return $\hat{Y}_i$

13.4 Experimental Evaluation

In this section, we implement RAD, on IoT and Cluster datasets. Evolution of learning accuracy under 30% and 40% noise level are reported for all three frameworks. For RAD, impact of noise level on final accuracy is discussed in Sec. 13.4.4.

13.4.1 Use Cases and Datasets

In order to demonstrate the general applicability of the proposed RAD framework for anomaly detection, we consider the following two use cases: (i) Cluster task failures, and (ii) IoT botnet attacks.In our experiments, we use real data collected in cluster and IoT platforms. These two datasets detailed in Sec. 12.2

The main dataset characteristics are summarized in Tab. 13.2.

Table 13.2 – Dataset description

Use case	**Cluster task failures**	**IoT device attacks**
#trainig data	60,000	33,000
#test data	6,000	6,000
#classes K	4	11
#features f	27	115
data batch size	600	300
$\lvert\mathscr{D}_0\rvert$	6,000	6,000

13.4.2 Experimental Setup

RAD is developed in Python using scikit-learn [105]. The main performance evaluation metric is accuracy. All results are averaged on 3 times experiments.

Noise. We inject noise into the two datasets by exchanging the true label of data instances with a random one except itself. The label noise is symmetric, i.e., following the noise completely at random model [44] where a label is picked with equal probability from all classes except the true one. The noise level $\tilde{Y}$ represents the percentage of data instances with noisy labels. We assume that all data is affected by label noise, except the $\mathscr{D}_0$ and testing data.

Continual learning. We start with an initial data batch of 6000 data instances for the Cluster task failures and the IoT devices dataset. Then, data instances arrive continuously in batches of 600 (Cluster) and 300 (IoT) data instances. To kick-start the label and classification models in RAD we assume first batch contains only clean data, and subsequent data batches are affected by noise. We select 6000 clean data instances as the test dataset for both use case. Test dataset will be used at the end of each epoch to evaluate the accuracy of the trained classification models. We show the evolution of the model accuracy over data batch arrivals until the performance of RAD converges.

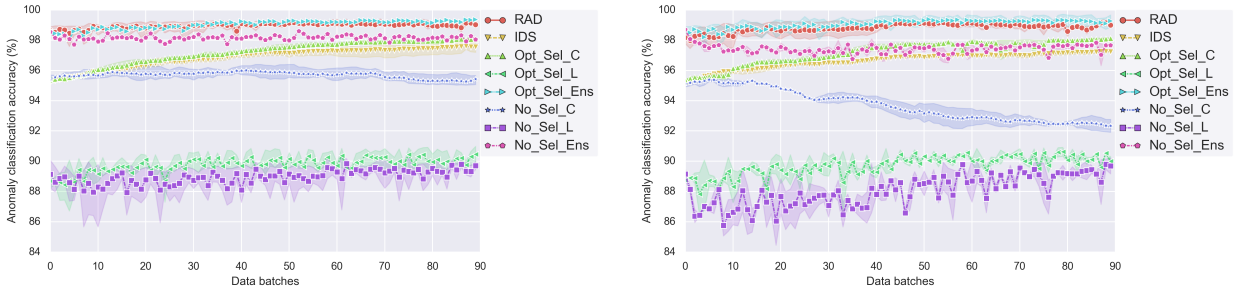
Label model. We use a multilayer perceptron to mainly assess the quality of each label, it will also be used to join the ensemble prediction. For IoT and Cluster dataset, the neural network consists of two layers with 28 neurons each. The precision and robustness of the label model are critical to filter out the noisy labels and provide a clean training set to the classification model. We considered different models, neural networks provided the best results in terms of accuracy and stability over

time. Adaboost gave excellent accuracy when training from the initial data with ground truth, but it is too sensitive to noise in the labels. Random forest is also known to be robust against label noise [44], however its accuracy was below the neural network one.

Classification model. We use KNN to jointly do the ensemble prediction with label model.Higher values can increase the resilience of the algorithm to residual noise, but also induce extra computational cost. The current choice stems from good results in preliminary experiments.

Baselines. The proposed RAD is compared against following baseline data selection schemes: (1) *No-Sel*, where all data instances of arriving batches are used for training the classification model; and, (2) *Opt-Sel* which emulates an omniscient agent who can perfectly distinguish between clean and noisy labels, and only use clean data to train the models; (3) *IDS*: intrusion detection system from [1]. The main idea and structure of *IDS* are similar to the proposed RAD. The differences are: i) IDS only trains label quality model with $\mathscr{D}_0$ once without continuous updated; and, ii) IDS only uses classification model for predictions, instead of combining prediction results of quality and classification models. In the following text, the model name ends with '_C' means the prediction obtained from the anomaly classification model, with '_L' means the prediction obtained from label quality model, with '_Ens' means the prediction obtained from both anomaly classification and label quality model specified in Algorithm4.

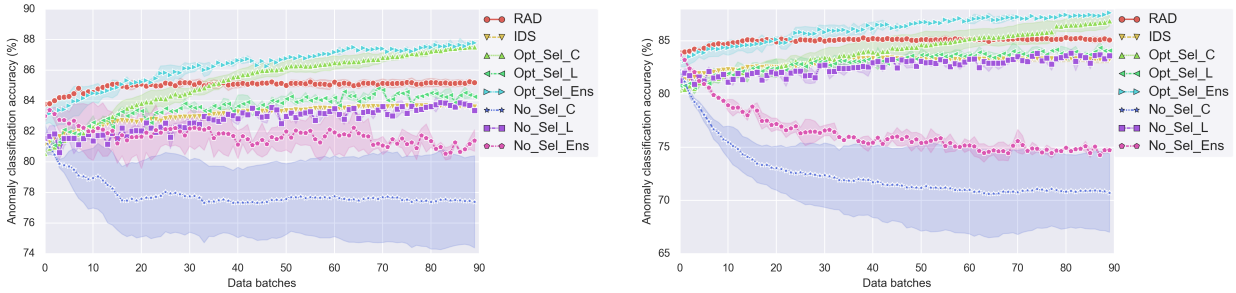
13.4.3 Handling Dynamic Data



(a) With data noise level of 30%

(b) With data noise level of 40%

Figure 13.4 – Evolution of learning over time – Use case of IoT thermostat device attacks. Opt_Sel and No_Sel stand for optimal data selection and no filtering, respectively. _C, _L, and _Ens denote the model or strategy chosen for prediction.

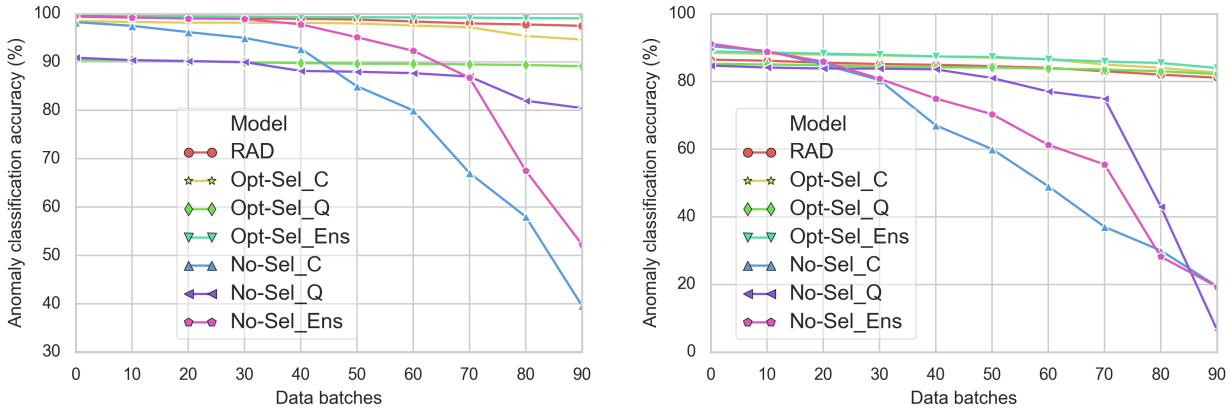


(a) With data noise level of 30%

(b) With data noise level of 40%

Figure 13.5 – Evolution of learning over time – Use case of Cluster task failures

Fig. 13.4 and 13.5 show the evolution of the mean and variance of the classification accuracy achieved by RAD on the thermostat and task failure datasets, respectively. Each figure moreover



(a) IoT thermostat device attacks

(b) Cluster task failures

Figure 13.6 – Impact of data noises on RAD accuracy

presents results under two levels of label noise: 30% and 40%. We compare RAD against no selection (*No-Sel*), optimal selection (*Opt-Sel*) and IDS. One can notice that learning from all data instances without cleansing (i.e., *No-Sel* curves) gives consistently lower accuracy in all cases. For the task failure dataset, the accuracy even oscillates and diverges. The performance when using RAD is better. First because the accuracy does not diverge. Second because it always consistently increase until it saturates. The end accuracies are around 99% and 85.5% for the IoT attack and cluster tasks datasets, respectively. For the first dataset, the accuracy of RAD follows closely the ensemble prediction accuracy of *Opt-Sel*. As for the second dataset, RAD follows ensemble prediction of *Opt-Sel* at first but then saturates after 30 data batch arrivals. Note that RAD gives also more stable results as shown by shorter variance bars which in magnitude are in line with the ones obtained by an ideal data cleansing. For *No-Sel* the bars are significantly larger.

We note that ensemble prediction can greatly enhance the learning outcomes in the presence of noisy data, compared to conducting prediction with the label quality model or classification model. Such an observation holds for different data selection schemes discussed in the subsequent section. Due to the space limit, we skip the presentation of those results.

In summary: (i) continual learning is advantageous compared to using only the initial dataset; however, (ii) continual learning exposes us to possible classification accuracy degradation stemming from noisy labels if proper data selection is lacking, (iii) RAD improves the classification accuracy compared to taking all labels, (iv) the data selection of RAD is good, and close to being optimal in some cases, and (v) ensemble prediction can greatly enhance the robustness against noisy data.

13.4.4 Evaluation of Noise Robustness of RAD

Next we investigate the impact of different noise levels on the RAD performance in terms of classification accuracy.

Fig. 13.6a and 13.6b present the classification accuracy for various levels of noise, ranging from 0% (all data are clean) up to 90% for our two main reference datasets: IoT thermostat device attacks and Cluster task failures. All the experiment setting remains the same as before, only noise level of training data batches varies. Once again, the RAD performance is compared to learning from all data (*No-Sel*) and an omniscient data cleanser (*Opt-Sel*).

As illustrated in Sec. 13.2, for *No-Sel* the noisier the data are, the worse the classification accuracy, with ensemble prediction, dropping to 20% and 52% for the Cluster and IoT datasets, respectively. A

decreasing trend can also be found for RAD and *Opt-Sel*, however the drops are significantly smaller: at most 5%. As there is by definition no noise in *Opt-Sel* case, the decrease in classification accuracy is only due to the reduction of the overall amount of clean data to learn from. Since the data cleansing of RAD is not perfect, the accuracy reduction is caused by noise pollution and overall clean data reduction. Nevertheless, the impact is small and any huge accuracy pitfall is avoided which results in RAD's performance being close to *Opt-Sel*. We can conclude that RAD can limit the impact of the amount of noise across a wide range of noise levels.

13.4.5 Analysis of All Datasets

Summary results are reported in Tab. 13.3. We can see that results of RAD are always better than IDS and *No-Sel*. For IoT dataset with 40% noise, RAD is even better than any single model of *Opt-Sel*. But there are still improvement room between RAD and *Opt-Sel_Ens*.

Table 13.3 – Evaluation of the all algorithms for Cluster task failures datasets and IoT device attacks datasets on 30% and 40% noise level. All the results are averaged on 3 runs.

Algorithm	Cluster(30)	IoT(30)	Cluster(40)	IoT(40)
Opt-Sel_C	87.68	98.08	87.16	98.06
Opt-Sel_L	84.37	90.81	84.18	89.70
Opt-Sel_Ens	87.88	99.35	87.60	99.25
No-Sel_C	77.40	95.47	71.02	92.27
No-Sel_L	83.54	89.95	83.35	89.57
No-Sel_Ens	81.53	98.06	74.92	97.51
RAD	85.46	99.01	85.03	98.95
IDS	83.63	97.83	83.31	97.23

1. *_C: Using label predictor
2. *_L: Using classification model
3. *_Ens: Using ensemble prediction

The resilience to high levels of noise might be even more important than the benefits of continual learning. Under such levels, the classification accuracy without data cleansing diverges for all datasets. Even if it is rare to have noise levels of 90% or above, they might still happen for short periods of time in case of attacks to the auto-labelling system via flooding of malicious labels. Hence this property can be crucial for the dependability of the auto-labelling system.

13.4.6 Limitation of RAD Framework

Though RAD works well for datasets of Cluster task failures and IoT device attacks, we can still see the potential limitations of this framework: (1) the assumption of availability of a small fraction of clean data which may not be possible; (2) if data is coming at high rates, training two models simultaneously instead of one can slow down the system; (3) as anomaly classifier receives only the data selected by label model, there is a risk that the classifier model overfits to label model. To address these issues we devised two extensions presented in Chapter 14.

13.5 Concluding Remarks

While machine learning classification algorithms are widely applied to detect anomalies, the commonly employed assumption of clean anomaly labels often does not hold for the data collected in the wild due to careless annotation and malicious dirty label pollution. The noisy labels can significantly degrade the accuracy of anomaly detection with an increasing amount of data and are challenging to tackle due to the lack of ground truth of label quality. In this chapter, we present a framework for robust anomaly detection, RAD, which can continuously learn the system dynamics and anomaly behaviours from streams of arriving data after filtering out suspicious noisy data.

RAD is a general framework which consists of a sequence of a quality model and a classification model, where the former mainly captures the label dynamics and the latter majorly focus on detection anomaly. By incorporating ensemble prediction into RAD, we demonstrate the effectiveness of RAD on two uses cases, i.e., detecting IoT device attacks, and predicting task failure at Google clusters. RAD can robustly improve the detection accuracy against different levels of label noises, reaching up to 85.56% and 99.01% accuracy under 30% noise for predicting task failure and detecting IoT device attacks, respectively, whereas learning directly from all the data streams without filtering degrades the detection accuracy. We also observe the limitations showed in Sec. 13.4.6, in next chapter, we will provide the improved versions of RAD to response these shortcomings.

Chapter 14

Extension of RAD framework for On-line Anomaly Detection for Noisy Data

In Chapter. 13. we see RAD's ability to robustly improve the detection accuracy against different levels of label noises. We also observe several limitations of this frameworks, as anomaly classifier receives only the data selected by label model, thus there is a risk that the classifier model overfits to label model. We extend RAD with additional features of conflicting opinions of classifiers, repetitively cleaning, and oracle knowledge, namely RAD Voting and RAD Active Learning. These extensions are examined with the same use cases as RAD did in Chapter. 13: (i) detecting 10 classes of IoT attacks and (ii) predicting 4 classes of task failures of big data jobs. To show the broad applicability of our framework, we propose another extension of RAD: RAD Slim, which is specifically designed to deal with image data. Therefore, we introduce a new use case: (iii) recognising 100 celebrities faces. Along with RAD Slim, we also introduce two other state of the art algorithms to compare: (1) Forward loss correction and (2) Co-Teaching, which are introduced in Sec. 12.4.

14.1 introduction

The purpose of this chapter is to extend Robust Anomaly Detector (RAD) [142], a generic framework that continuously learns an anomaly classification model from streams of event logs or images that are subject to label noise. The original design of RAD is composed of two layers of learning models, i.e., a data label model and an anomaly classifier. The label model aims at differentiating the label quality, i.e., noisy v.s. true labels, for each batch of new data and only "clean" data points are fed in the anomaly classifier. The anomaly classifier predicts the event outcomes that can be in multiple classes of (non)anomalies, depending on the specific anomaly use case. In this extension, we derive three alternatives of RAD, namely, voting, active learning and slim, which use additional information, e.g., opinions of conflicting classifiers and queries of oracles. We iteratively update the prediction of historical windows such that the weak prediction can be continuously improved the latest model.

To demonstrate the effectiveness of RAD, we consider three use cases, i.e., detecting 10 classes of attacks on IoT devices [94], predicting four types of task failures for big data processing cluster [112, 116] and recognising the 100 most abundant celebrity faces [103] from open datasets.

Our results show that RAD Voting can improve RAD. If we implement RAD Active Learning on cluster dataset with the same noise level, the final accuracy could reach to 90.51%. For face image dataset, final accuracy of RAD Slim under 30% noise achieves to 76.51% (comparing to 40.01% of no selection on dataset). Furthermore, our study also shows that RAD Voting is as stable as RAD even when the noise is very strong. And if we do not have many clean data at beginning to pre-train

the model, RAD Active Learning and RAD Active Learning Limited could still perform very well from a very bad starting model.

14.2 Motivating case studies

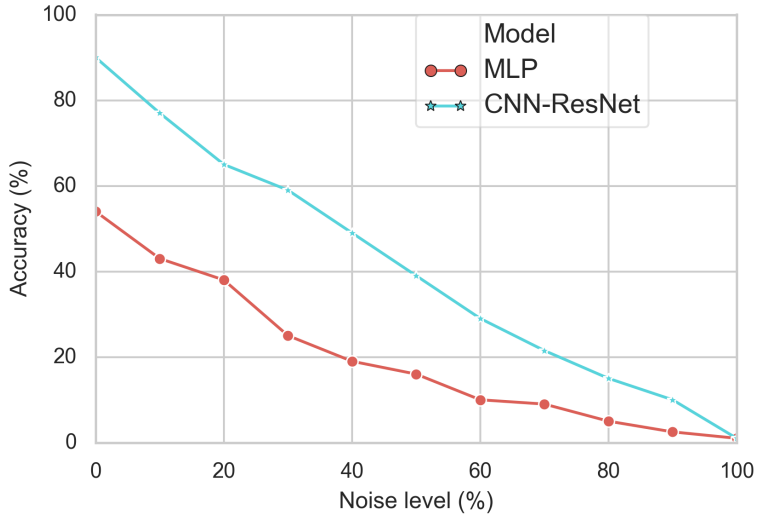


Figure 14.1 – Impact of noisy data on anomaly classification: Use case of Face Recognition

IoT device attacks and Cluster task failures datasets are already introduced in the last chapter. In this chapter, we introduce a new dataset: face recognition.

- FaceScrub dataset [103] is a collection of photos of celebrities roughly half female and half male. The task is to recognize faces by matching each photo to the identity of the celebrity shown on it. Here we focus on the face recognition of the 100 celebrities with the highest number of photos in the dataset totalling to 12K images. Face recognition is widely used in biometric identification systems for security applications, e.g., access control, which makes the robustness of such systems critical. Furthermore, this image dataset is studied also because we want to show the broad applicability of our proposed framework.

The details about data definition, and statistics, e.g., number of feature and number of data points, can be found in Sec. 14.4.1. To recognize anomalies/faces in each use case, related studies have applied different machine learning classification algorithms, from simple ones, e.g., k-nearest neighbour (KNN), to complex ones, e.g., deep neural networks (DNN), under scenarios with different levels of label noise. Here, we evaluate how the detection accuracy changes relative to different levels of noises. We focus on off-line scenarios where we split the data in a training set affected by label noise and a clean evaluation set.

For face recognition we use a small subset of our complete dataset (which contains 100 celebrities), it contains 2,639 images from 20 celebrities with varying degrees of label noise as the training set and 665 clean images as the testing set. Due to the particularity of image data, we use MLP and a specific CNN (Convolutional Neural Network) - ResNet (Residual neural Network) [63] (56 layers) as classification models. Fig. 14.1 shows the accuracy results under the different label noise levels. One can see, similar to the previous use cases, that label noise strongly affects the performance of both classifiers, although, the effect here is approximately linear. Moreover, ResNet performs better than MLP for this dataset under any noise level.

Above experiment clearly show that under the presence of noisy label data, deep neural network indeed corrupts. The stronger the noise, the worse the model's accuracy. That case motivates us to construct a new algorithm which can better resist the influence of noisy label data on training process.

14.3 Design of RAD Extensions

In this section, we will introduce the design of the extensions of RAD. All the symbols used to explain the designs have same definitions as in Tab. 13.1. To make this manuscrit easy to follow, we recall the symbol table as follows:

Symbol description

Symbol	Description
$\mathfrak{L}$	label quality predictor
$\mathfrak{C}$	anomaly detection classifier
$\mathscr{D}_i$	ith training data batch
$\mathscr{D}_i^*$	ith cleansed data batch from $\mathfrak{L}$
$\mathscr{P}_i$	ith test data batch
$\hat{Y}_i$	prediction of ith test data batch from $\mathfrak{C}$
$\tilde{Q}_i$	percent of clean labeled data of ith batch
$\mathscr{U}_i$	"unclean" data of ith batch determined by $\mathfrak{L}$
$\mathscr{U}_i^*$	ith cleansed data batch from $\mathfrak{C}$
S_i	"unclean" data of ith batch determined by $\mathfrak{C}$
S_i^*	data with true label from Expert of ith batch
$\hat{p}$	indicator of prediction, 1 for clean, 0 for dirty
$\hat{q}$	indicator of prediction, 1 for clean, 0 for dirty

14.3.1 Overview of Design

RAD, same as its extensions, are composed of two key steps: training data selection and prediction techniques, as shown in Fig.14.2. Training data selection part focus on how to filter out suspiciously noisy data and solicit "clean" data to train the classification models subsequently. It has four options: basic, voting, active and slim. The prediction part combines different prediction models. The available options are "ensemble", which combines the prediction outcomes of quality and classification models, and "slim", which has only one model to filer and classify anomaly images. The specific combinations are following: (i) Basic, RAD Voting and RAD Active Learning are followed by the ensemble prediction, (ii) RAD Slim is followed by the Slim prediction, which only uses one model to save the computation resources.

14.3.2 Data Selection Schemes

We will introduce the schemes of three extensions of RAD, namely RAD Voting, RAD Active Learning, and RAD Slim. We explain their specific pitfalls and opportunities.

Algorithm 5 RAD Voting and RAD Active Learning

Input: Data batch $\mathscr{D}_i$ and its given label Y_i, label quality model $\mathfrak{L}_{i-1}$, anomaly classification model $\mathfrak{C}_{i-1}$

Output: $\mathfrak{L}_i$, $\mathfrak{C}_i$

Predict $\mathscr{D}_i$ by $\mathfrak{L}_{i-1}$, get prediction label $Y_i^{\mathfrak{L}}$.
Compare Y_i and $Y_i^{\mathfrak{L}}$, extract the data where predictions in Y_i and $Y_i^{\mathfrak{L}}$ are identical, combine them as $\mathscr{D}_i^*$.
$\mathfrak{L}_{i-1}$ sends $\mathscr{D}_i^*$ to $\mathfrak{C}_{i-1}$ (if two models are not in the same location), two model catch $\mathscr{D}_i^*$ locally. Discard other data.
if Algorithm is **RAD Voting** or **Active Learning then**
    Compare Y_i and $Y_i^{\mathfrak{L}}$, extract the data where predictions in Y_i and $Y_i^{\mathfrak{L}}$ are different, combine them as $\mathscr{U}_i$. Send $\mathscr{U}_i$, the corresponded given label Y_i^U and the corresponded prediction by $\mathfrak{L}_{i-1}$: $Y_i^{\mathfrak{L}^U}$ to $\mathfrak{C}_{i-1}$.
    Predict $\mathscr{U}_i$ by $\mathfrak{C}_{i-1}$, get prediction label $Y_i^{\mathfrak{C}^U}$.
    if Algorithmis **RAD Voting then**
        Select $\mathscr{U}_i^*$ from $\mathscr{U}_i$ where: (1) labels in Y_i^U and $Y_i^{\mathfrak{C}^U}$ are identical. Or 2) If condition (1) does not hold, but predictions in $Y_i^{\mathfrak{C}^U}$ and $Y_i^{\mathfrak{L}^U}$ are identical, we change given labels of these data as the prediction in $Y_i^{\mathfrak{C}^U}$, and include them in $\mathscr{U}_i^*$. Send $\mathscr{U}_i^*$ to $\mathfrak{L}_{i-1}$, two model catch $\mathscr{U}_i^*$ locally.
        Calculate S_i: difference between $\mathscr{U}_i$ and $\mathscr{U}_i^*$.
        Add S_i as element i to inactive data list L_{inac}.
        Select t_1, t_2 from [0,i-1] where $L_{inac}[t_1]$, $L_{inac}[t_2]$ are top two elements who have the largest number of data in L_{inac}.
        Repeat state 1-10 with input ($\mathscr{D}_{t_1}$, Y_{t_1}, $\mathfrak{L}_{i-1}$, $\mathfrak{C}_{i-1}$) and ($\mathscr{D}_{t_2}$, Y_{t_2}, $\mathfrak{L}_{i-1}$, $\mathfrak{C}_{i-1}$). Use them to update $\mathscr{D}_{t_1}^*$, $\mathscr{D}_{t_2}^*$, $\mathscr{U}_{t_1}^*$, $\mathscr{U}_{t_2}^*$, $\mathscr{S}_{t_1}$ and $\mathscr{S}_{t_2}$.
        $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$ use all accumulated $\mathscr{D}_t^*$, $\mathscr{U}_t^*$ t $\in$ [0,i] to train the model, get $\mathfrak{L}_i$ and $\mathfrak{C}_i$
        return $\mathfrak{L}_i$, $\mathfrak{C}_i$
    end if
    if Algorithm is **RAD Active Learning then**
        Select $\mathscr{U}_i^*$ from $\mathscr{U}_i$ where labels in Y_i^U and $Y_i^{\mathfrak{C}^U}$ are identical. Send $\mathscr{U}_i^*$ to $\mathfrak{L}_{i-1}$, two model catch $\mathscr{U}_i^*$ locally.
        Calculate S_i: difference between $\mathscr{U}_i$ and $\mathscr{U}_i^*$.
        Send S_i to Expert, Expert will return S_i with its true labels to $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$, we call these data S_i^*.
        $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$ use all accumulated $\mathscr{D}_t^*$, $\mathscr{U}_t^*$ and S_t^* t $\in$ [0,i] to train the model, get $\mathfrak{L}_i$ and $\mathfrak{C}_i$
        return $\mathfrak{L}_i$, $\mathfrak{C}_i$
    end if
end if

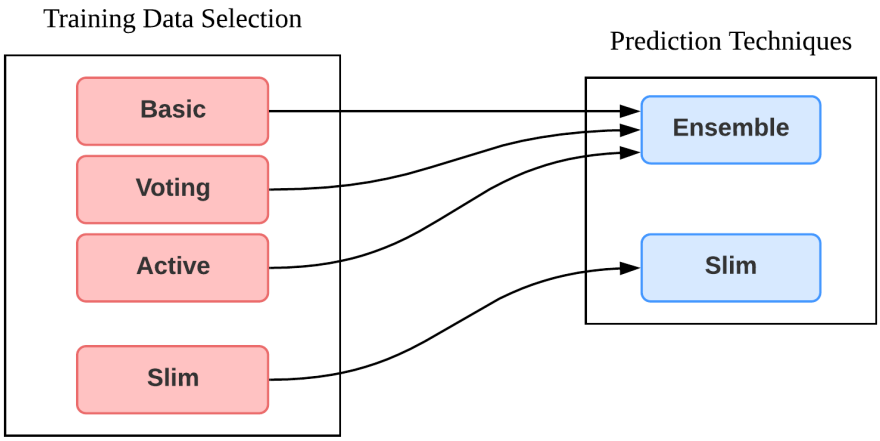


Figure 14.2 – Structures of RAD and its extensions: four choices of data selection and two choices of prediction technique.

14.3.2.1 RAD Voting

The baseline RAD algorithm separates distinctive goals for the two models. However this approach biases the results towards the label quality model $\mathfrak{L}$ and we want the classifier model $\mathfrak{C}$ to also play a role in selecting clean data instances. We do this via the voting extension shown in Fig. 14.3.

Comparing to the base RAD, predicted dirty labels having $\hat{q} = 0$ are not discarded by $\mathfrak{L}$ but passed to $\mathfrak{C}$ as uncertain data $\mathscr{U}$. Then the classifier $\mathfrak{C}$ is used to further cleanse the uncertain data to produce $\mathscr{U}^*$. For each data instance in $\mathscr{U}$ we predict its class $\hat{k}^{\mathfrak{C}}$ using $\mathfrak{C}$ and looking for agreement with the given class k and the class $\hat{k}^{\mathfrak{L}}$ predicted by $\mathfrak{L}$. We add data instances to $\mathscr{U}^*$ if either $\hat{k}^{\mathfrak{C}}$ equals k, or if $\hat{k}^{\mathfrak{C}}$ equals $\hat{k}^{\mathfrak{L}}$. In the latter we replace the given class by the predicted class.

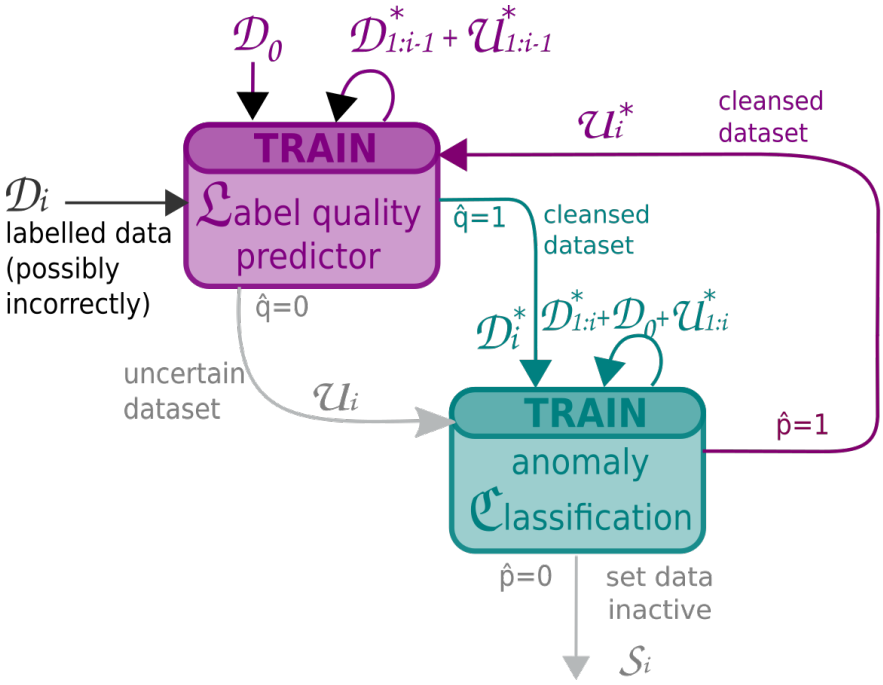


Figure 14.3 – RAD - Voting.

Batches of data instances not added to $\mathscr{U}_i^*$ at time t_i are not immediately discarded but kept in a batch S_i of inactive data. The idea is that since the accuracy of the classifier improves over time

(see Sec. 13.4.3), we can use the new classifier to re-evaluate old batches of inactive data and further increase the training data. We maintain a list of the batches of inactive data S_i After we finished training a new classifier we select the two biggest batches and re-process them via the voting system as before (see Algorithm. 5.

14.3.2.2 RAD Active Learning

In RAD Voting we use the two models $\mathfrak{C}$ and $\mathfrak{L}$ to correct labels and increase the overall amount of data used for training aiming for improved framework accuracy, however not all data is used. To increase further the amount of training data we resort to active learning, i.e., we ask an expert for the true class of the data instances we are least certain about.

Fig. 14.4 shows the structure of RAD Active Learning. The difference with the structure of RAD Voting is that in RAD Active Learning, we do not use the predictions from two models to correct the labels and we do not send the most uncertain data instances to the inactive list L_{inac} but to an oracle to ask for the true label. In RAD Active Learning, potentially every data instance will be used to train $\mathscr{L}$ and $\mathfrak{C}$ and there is no inactive data anymore. In reality, consulting an oracle for every single uncertain data instance might be too expensive. Hence we also consider RAD Active Learning Limited which additionally imposes a configurable limit N_{lim} on the number of queries to be asked to the oracle at each batch arrival. When the number of uncertain data instances exceeds this limit, we rank those instances in the decreasing order of their euclidean distance between the corresponding prediction probability vectors of $\mathfrak{C}$ and $\mathfrak{L}$. Only the first N_{lim} data instances are chosen for the expert query. We call it **Highest Disagreement Method**. The idea behind is that we want to ask the expert to check data instances with the lowest confidence in their prediction.

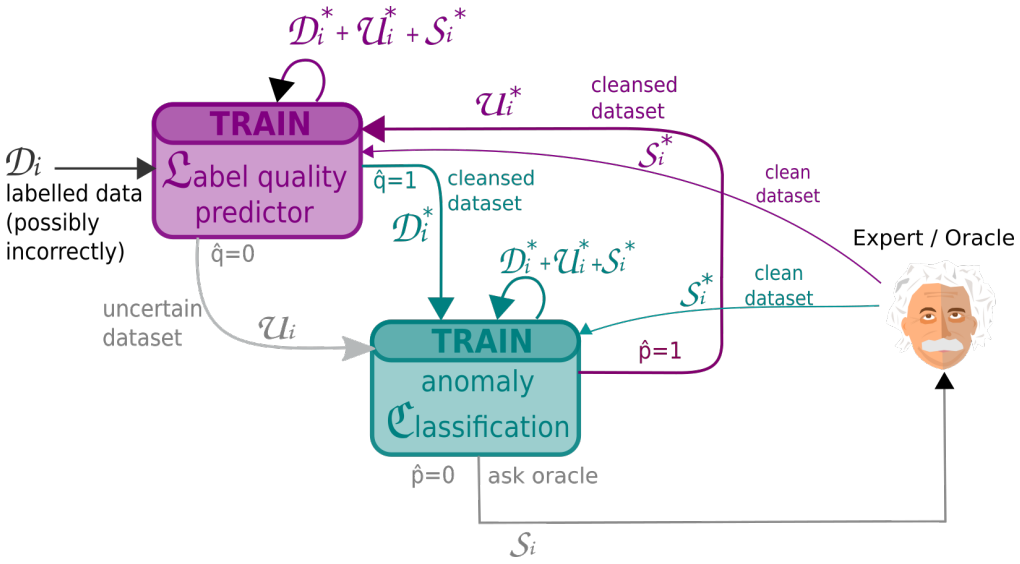


Figure 14.4 – RAD - Active Learning.

14.3.2.3 RAD Slim

The RAD framework requires two models. and depending on the complexity of the models used the cost of training might be excessive. Especially in scenarios relying on complex deep neural networks, such as Convolutional Neural Networks (CNNs) for image classification, it might be too expensive and time consuming to train two models. To reduce the computational costs we propose a slimmed

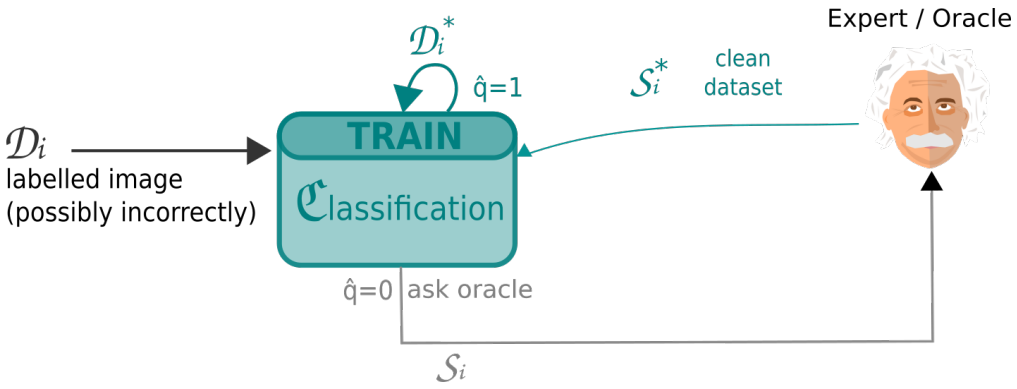


Figure 14.5 – RAD Slim.

version of RAD Active Learning named RAD Slim. The idea is to partially delegate the role of the label quality $\mathfrak{L}$ model to the oracle.

In RAD Slim new data batches arrive directly at the $\mathfrak{C}$ model, see Fig. 14.5. For each data instance we compare the given label k and to the predicted label $\hat{k^{\mathfrak{C}}}$. If they are the same we add it to $\mathscr{D}^*$, if they differ we ask the oracle for the true label and add it to $\mathscr{S}^*$. To train the model $\mathfrak{C}_{i-1}$, we use only current $\mathscr{D}_i^*$ plus S_i^*, not all the accumulated cleansed data as before. Considering computational cost, one pair of $\mathscr{D}_i^*$ and S_i^* will be used to train the model for 60 epochs. Similar to RAD Active Learning, we also impose a query limit here, termed RAD Slim Limited. We query experts for those data instances with high uncertain model prediction outcomes. To rank the uncertainty, we resort to the cross-entropy (loss) between given label and prediction probability vector made by $\mathfrak{C}$. Specifically, we rank all the data of S_i in decreasing order of their cross-entropy, and select corresponding data of first N_{lim}. We call this **Highest Loss Method**.

14.4 Experimental Evaluation

In this section, we implement RAD, RAD Voting and RAD Active Learning on IoT and Cluster datasets. Evolution of learning accuracy under 30% and 40% noise level are reported for all three frameworks. For RAD Voting, analysis on percentage of active and active-truth data changing over time is carried out in Sec. 14.4.3. RAD Active Learning and its small update RAD Active Learning Limited are explained in section. The Impact of size of initial data batch $\mathscr{D}_0$ on above frameworks are studied in Sec. 14.4.5. To demonstrate the applicability of the framework to image dataset, RAD Slim and RAD Slim Limited are studied in Sec. 14.4.6.

14.4.1 Use Cases and Datasets

In order to demonstrate the general applicability of the proposed frameworks for anomaly detection, we consider the following three use cases: (i) Cluster task failures , (ii) IoT botnet attacks and (iii) Face recognition. The characteristic of first two use cases have been introduced in Sec. 13.4.1. In this section, we focus on the third one.

The FaceScrub [103] dataset is used for face recognition. Original FaceScrub contains more than 100,000 face images of 530 people, with about 200 images per person. Male and Female images are almost equal. We use a subset of 12K FaceScrub images to fit the limits of our compute resources. The 12K images cover the 100 people which have the highest number of images, 55 males and 45 females. FaceScrub images were retrieved from the Internet and are taken under real-world situations (uncontrolled conditions). We resize all images to 64*64 pixels. Name is the only annotation we use.

The face recognition system has been widely used in security equipment. We apply RAD Slimmed to FaceScrub dataset to show that our framework can help to build also robust face recognition models.

The main dataset characteristics are summarized in Tab. 14.1.

Table 14.1 – Dataset description

Use case	Cluster task failures	IoT device attacks	FaceScrub
#trainig data	60,000	33,000	12,000
#test data	6,000	6,000	3,000
#classes K	4	11	100
#features f	27	115	64*64
data batch size	600	300	2400
$\lvert\mathscr{D}_0\rvert$	6,000	6,000	2400

14.4.2 Experimental Setup

RAD is developed in Python using scikit-learn [105]. The main performance evaluation metric is accuracy. All results are averaged on 3 times experiments.

Noise. To compare with RAD, we stick to the same constant symmetric noise model as introduced in Sec. 13.4. We still assume that all data is affected by label noise, except the $\mathscr{D}_0$ and testing data.

Continual learning. To have a fair comparison with RAD, the continual learning setting for RAD Voting and RAD Active Learning Limited remains the same as in Sec. 13.4 for IoT and Cluster datasets. For RAD Slim with FaceScrub dataset, due to the computational overhead, we will only use current data batch to train the model. We have 2400 clean label images to kickstart the training, then the subsequent batches contains 2400 images per batch. Test set has 3000 clean label images, we evaluate the model after each training epoch.

Slimmed framework model. For the face recognition task we use RAD Slim. In this case we use a 110 layers ResNet [63] as classification model. ResNet is a type of CNN architectures which introduces residual functions to alleviate the vanishing gradient problem in training deep neural networks improving the classification performance. Also to make model converge.

Baselines. The proposed RAD Voting and RAD Active Learning are compared against following baseline data selection schemes: *Full-Clean* baseline which simulates perfectly recovered labels, i.e., all wrong labels have been correctly identified and recovered by an oracle. This represents the ideal solution which provides all clean data in each data batch. In the following text, the model name ends with '_C' means the prediction obtained from the anomaly classification model, with '_L' means the prediction obtained from label quality model, with '_Ens' means the prediction obtained from both anomaly classification and label quality model specified in Algorithm4.

To compare with RAD Slim on image dataset, we introduce two state-of-the-art approaches: 1) Forward [104] which estimates the noise transition matrix before training the model, and subsequently use this transition matrix for loss correction; and 2) Co-Teaching [59] which trains two deep neural networks simultaneously to let them teach each other. For Forward experiment, we use the same network architecture, i.e., 110 layers ResnNet, as the one for RAD Slim. As Co-Teaching trains two models, we use two 56 layers ResNet for Co-Teaching. To speed up the model convergence for RAD Slim, RAD Slim Limited, and Forward, we implement the E (Exponential)/PD (Proportional-Derivative)-Control [143] (Chapter. 9) and Event-Based Control Learning rate [145] (Chapter. 10) as learning rate schedule based on SGD (stochastic gradient descent) optimizer. Co-Teaching has its own learning rate scheduler.

Table 14.2 – Final accuracy for all algorithms for Cluster task failures datasets and IoT device attacks datasets on 30% and 40% noise level. All the results are averaged on 3 runs. Full-Clean means that no label noise is injected. Opt-Sel means use only clean label data out of all data (mixed with clean and dirty labels). No-Sel means use directly all data.

Algorithm	Cluster(30)	IoT(30)	Cluster(40)	IoT(40)
Full-Clean_C	89.35	98.28	89.35	98.28
Full-Clean_L	85.17	90.87	85.17	90.87
Full-Clean_Ens	91.08	99.83	91.08	99.83
Opt-Sel_C	87.68	98.08	87.16	98.06
Opt-Sel_L	84.37	90.81	84.18	89.70
Opt-Sel_Ens	87.88	99.35	87.60	99.25
No-Sel_C	77.40	95.47	71.02	92.27
No-Sel_L	83.54	89.95	83.35	89.57
No-Sel_Ens	81.53	98.06	74.92	97.51
RAD	85.46	99.01	85.03	98.95
IDS	83.63	97.83	83.31	97.23
RAD Voting	86.01	99.21	85.73	99.07
RAD-AL[1]	90.84	99.72	90.77	99.58
RAD-AL-L[2]	89.57	99.66	-	-
PSO[3]	87.83	98.85	-	-

1. RAD-AL: RAD Active Learning
2. RAD-AL-L: RAD Active Learning Limited
3. PSO: Pre-Select Oracle

14.4.3 RAD Voting and History Extension

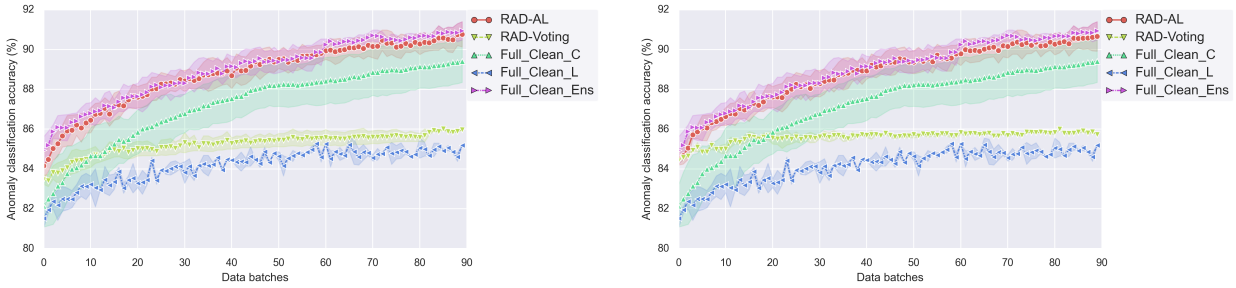
In the first extension we let both the label and classifier models vote on the label quality and include the possibility to recover instances from history to be evaluated as the models performance improves over time.

We evaluate the accuracy of RAD Voting over time and different noise levels in Fig. 14.6 and Fig. 14.7 for the IoT thermostat and Cluster task failures, respectively. For IoT dataset, RAD Voting is better than any single model of *Full-Clean*. And for Cluster dataset, RAD Voting does not saturate as RAD in Fig. 13.5. Tab. 14.2 summarize and compare the RAD Voting performance with others. We can see that RAD Voting performance is always better than RAD. This is because we correct labels in RAD Voting algorithm, which thus increases the number of training instances than RAD

(a) Iot data with noise level of 30%

(b) Iot data with noise level of 40%

Figure 14.6 – Evolution of learning over time – Use case of IoT thermostat device attacks with RAD Voting and RAD Active Learning (RAD-AL). Full_clean means that no label noise is injected.



(a) Cluster data with noise level of 30%

(b) Cluster data with noise level of 40%

Figure 14.7 – Evolution of learning over time – Use case of Cluster task failures with RAD Voting and RAD Active Learning

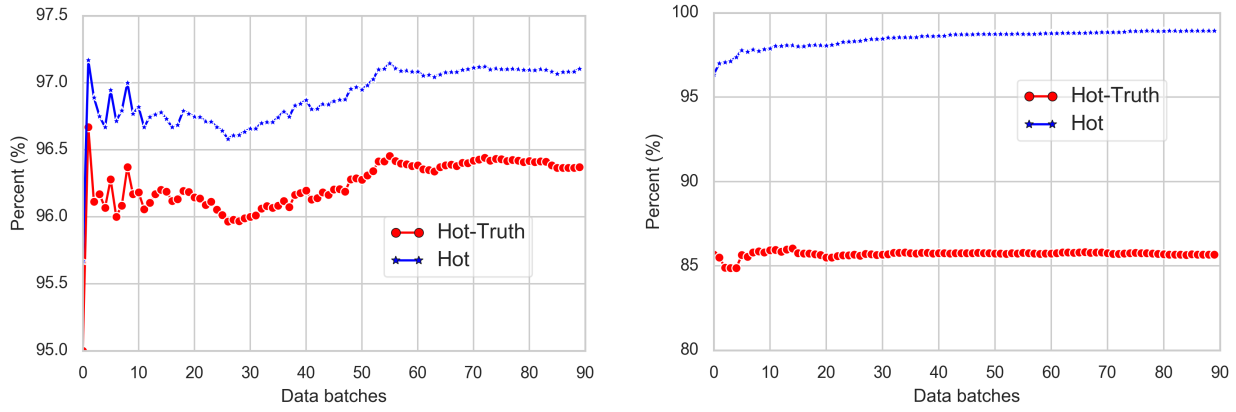
To better understand the different performance between the two data sets let us define A (called Hot) as the percent of data used for training till time t_i:

$$A = \frac{\sum_{k=1}^{i}(|\mathscr{D}_k^*| + |\mathscr{U}_k^*|)}{\sum_{k=1}^{i}|\mathscr{D}_k|}.$$

Knowing the number of true clean labels used per batch C_i^T we further define A^{HT} (called Hot-Truth) as the percent of true clean active data.

$$A^{HT} = \frac{\sum_{k=1}^{i} C_k^T}{\sum_{k=1}^{i}(|\mathscr{D}_k^*| + |\mathscr{U}_k^*|)}$$

In both formulas, we exclude the initial clean batch $\mathscr{D}_0$. Intuitively, A tells how much of the incoming data we use for training, and A^{HT} shows how clean the used training data is.



(a) Iot data with 30% noise

(b) Cluster data with 30% noise

Figure 14.8 – RAD Voting: percentage of hot data and its ground truth.

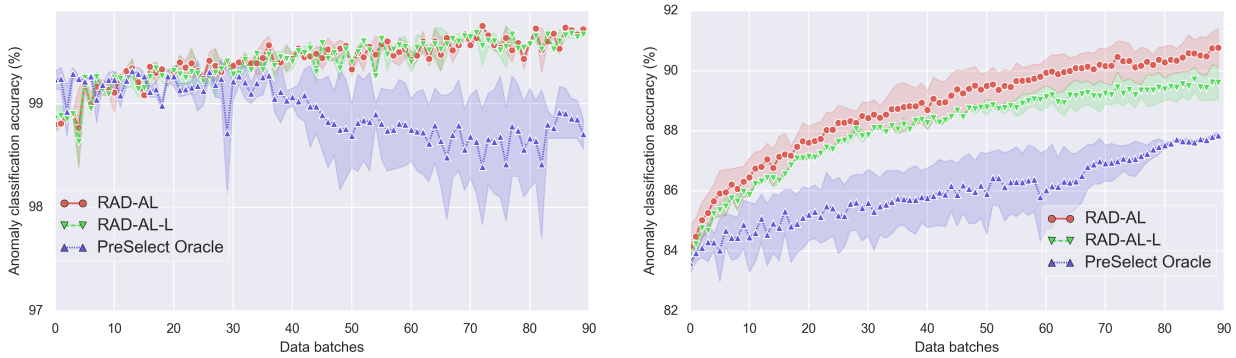
Fig. 14.8(a) and (b) plot over time these two metrics for the IoT and task failures datasets, respectively. As seen in Fig. 14.8(a), for the IoT dataset both A and A^{HT} improve over time. This means that the active data percentage both the amount of active data, i.e. A, and the quality of active data A^{HT} improve over time. On the contrary, looking at Fig. 14.8(b), for the Cluster dataset A^{HT} does not improve over time even if A increases. We impute this to the fact that both $\mathfrak{C}$ and $\mathscr{L}$ predict the same wrong class and this class is used to replace the original label of the data instance.

14.4.4 RAD Active Learning

RAD Active Learning extends the RAD with the ability of asking an oracle to provide the true label for the data instances where the two models do not agree. First we consider RAD Active Learning with no limits on the number of oracle requests followed by RAD Active Learning Limited which limits the number of oracle interactions.

Fig. 14.6 and Fig. 14.7 show the performance of RAD Active Learning (RAD-AL) for the IoT and Cluster datasets under 30% and 40% noise, respectively. The figures compare RAD Active Learning to RAD Voting, *No-Sel* and *Full-Clean*. We can see that RAD Active Learning is always better than RAD Voting and almost as good as *Full-Clean_Ens* across the two different datasets and different noise levels. From the results in Tab. 14.2, we can observe that the result of RAD Active Learning is extremely close to *Full-Clean_Ens* who are the best in every column. That shows that our training data selection in RAD Active Learning is very accurate, almost all the noisy data are filtered out for consultations to expert.

In reality consulting every single uncertain data instance with expert might be too expensive or impossible. Hence, we consider RAD Active Learning Limited which limits the capacity of consultation with experts. Here we set the number of limit on 20% of batch size per batch, the reason is because the experiment is with 30% noise data, the 20% consultation of batch size can reduce the total consultation number comparing to no limitation. To illustrate the power of our training data selection process, we introduce a new comparison for RAD Active Learning Limited: Pre-Select Oracle. Pre-Select Oracle has the same number of consultation to oracles as RAD Active Learning Limited, but the data instances here are randomly selected before the training. Fig. 14.9 shows the results for IoT and Cluster datasets, one can notice that curve of RAD Active Learning Limited (RAD-AL-L) increases along with RAD Active Learning and largely outperforms Pre-Select Oracle. The accuracy difference here is due to the uncertainty ranking used in the highest disagreement method. From the result in Tab. 14.2, we can see that after imposing a expert query limit of 20%, RAD Active Learning Limited reaches similar accuracy as RAD Active Learning, and higher accuracy than RAD and RAD Voting.



(a) Iot data with noise level of 30%

(b) Cluster data with noise level of 30%

Figure 14.9 – Comparison of RAD Active Learning Limited (RAD-AL-L) and Pre-Select Oracle, showing the power of selection.

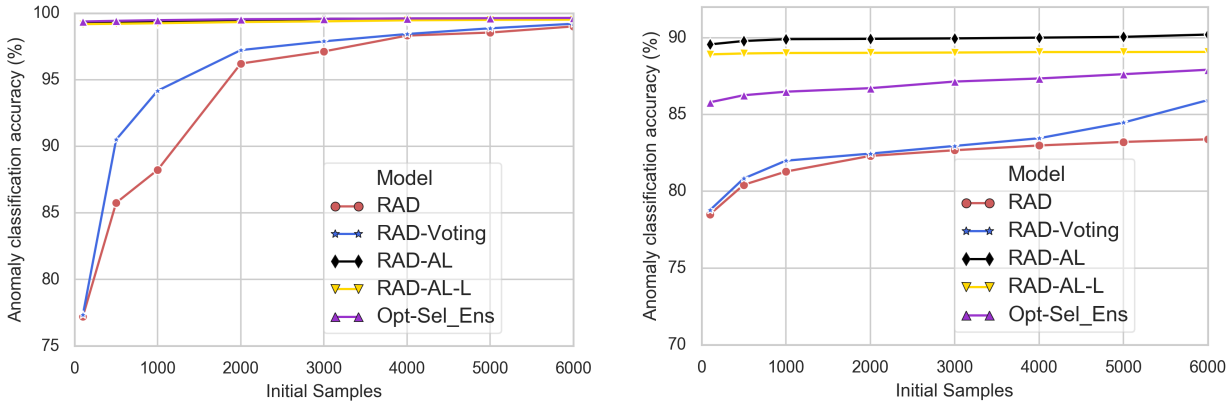
14.4.5 Impact of Initialization

Here we study the impact on the RAD and its extensions of the size of the initial dataset $\mathscr{D}_0$. We vary the number of initial clean data instances from 100 to 6000, and measure the classification accuracy

after 90 data batch arrivals. Her we consider the *Opt-Sel* baseline since the No-Sel baseline is meant for the framework configuration, not its performance evaluation.

In Fig. 14.10a and Fig. 14.10b show the results for the IoT and Cluster datasets, respectively. *Opt-Sel_Ens* seems to be independent from the number of initial data instances ($|D_0|$) in Fig. 14.10a. This is due to the fact that after 90 batch arrivals the amount of training data is sufficient for the accuracy to converge. But in Fig. 14.10b, we can see $|D_0|$ has influence on *Opt-Sel_Ens*, the model is yet to converge at the end of learning, but this influence is clearly smaller than that for RAD and RAD Voting. For these two models the size of $\mathscr{D}_0$ does matter: the larger the better. At $|\mathscr{D}_0| = 2000$ their performances are similar to *Opt-Sel_Ens* (less than 5% difference) for IoT dataset, and at $|\mathscr{D}_0| = 6000$ they almost overlap. RAD Voting outperforms RAD in both datasets under all sizes of D_0. This is because RAD Voting can correct data labels and thus increase the number of training instances. Finally, RAD Active Learning and RAD Active Learning Limited (20% limit) do not depend on the size of $\mathscr{D}_0$, since they can ask the oracle for the label of uncertain data instances.

This justifies our earlier choice of D_0 having 6000 data instances as it enables to achieve the best accuracy. However, all proposed frameworks could also perform well with only half of the data instances in D_0.



(a) IoT thermostat device attacks

(b) Cluster task failures

Figure 14.10 – Impact of size of initial data batch $\mathscr{D}_0$ on RAD accuracy with 30% noise level

14.4.6 RAD Slim on Image Data

Here we evaluate the RAD framework on the challenging case of noisy image classification. Specifically, we apply RAD Slim and RAD Slim Limited (20% limit of batch size per batch) to train a classifier that encounters on-line noisy image benchmarks. Fig. 14.11 shows the accuracy results across the batch arrivals. We can observe that RAD Slim is close to the *Full-Clean* baseline and largely outperforms other baselines. A summary is presented in Tab. 14.3. Fig. 14.12 shows the comparison between RAD Slim, RAD Slim Limited and Pre-Select Oracle (same design as in Sec. 14.4.4). One can see that RAD Slim Limited performs much better than Pre-Select Oracle while two experiments have the same number of expert queries.

Another observation is that, under 30% noise level, the accuracy difference (10.33%) between RAD Slim and RAD Slim Limited is much bigger than the resulting difference between RAD Active Learning and RAD Active Learning Limited for dataset IoT (0.06%) and Cluster (1.27%). One reason is that, to find out noisy data, the Highest Disagreement Method based on the predictions of two models in RAD Active Learning Limited is more efficient than the Highest Loss Method based on the

prediction of only one model in RAD Slim Limited. Even if the consultation numbers are the same, all data points do not contribute equally for model training process, certain data are more critical than others [60].

To further display the effectiveness of RAD Slim on different type of attacks, we design a series of unbalanced noisy data batches. For the original facescrub dataset, the image ratio of male and female is about 55%:45%. For $\mathscr{D}_0$ of the unbalanced data batches, image ratio of male and female is 90%:10%, where for the following noisy data batch, the ratio of male and female is 45%:55%. Fig. 14.13 shows the results, RAD Slim performs definitely better than no selection, and very close to full clean scenario, which shows that RAD Active Learning can not only defend the model from different levels of noise, but also can resist other type of attacks.

Another thing we could notice is that all curves suffer a periodic up-down pattern. This is because for image dataset, each time a new batch comes, we only use this new batch data as training dataset. As different batches provide different subviews of the data the empirical distribution can be different as well as the calculated optimum, but the overall model remains the same. So for the first epoch of a new batch, we will generate a gradient which is based on new data but applied on an old model. This influences the accuracy of the model. Moreover, when retraining on each new data batch we reset the learning rate which causes a bump in the learning rate. Therefore, even if all batches follow the same distribution, the system could temporarily wander off from the previous optimum.

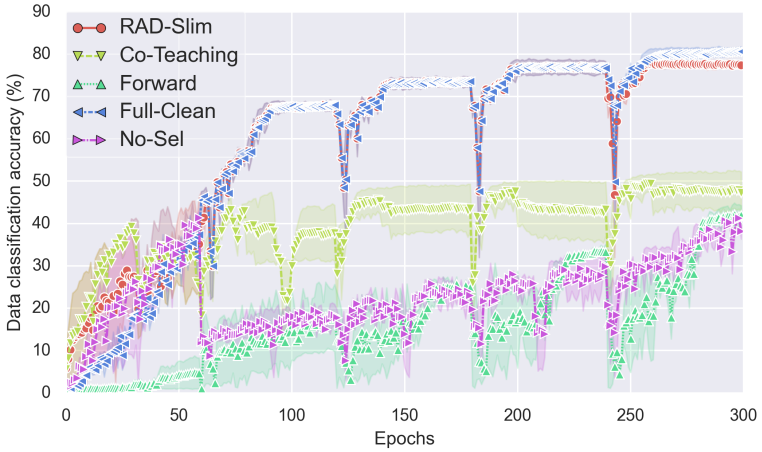


Figure 14.11 – FaceScrub with noise level of 30%, comparison with state of the art algorithm

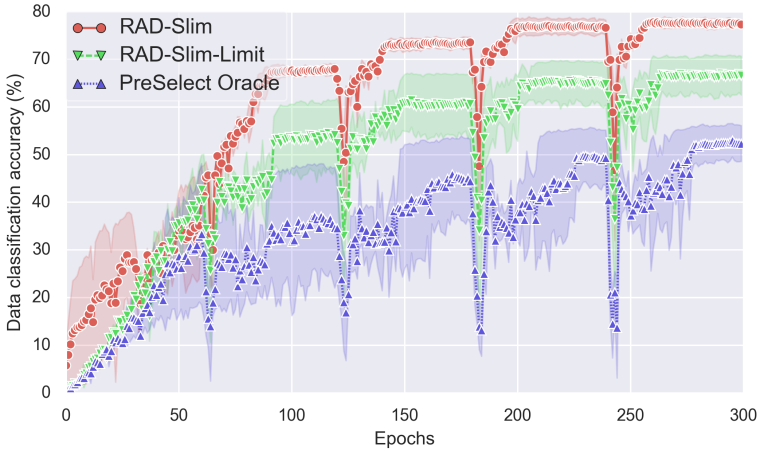


Figure 14.12 – RAD Slim Limited on FaceScrub with 30% noise

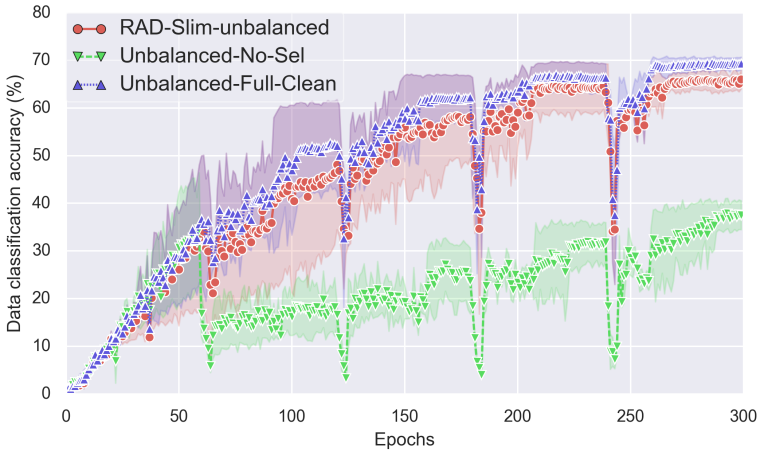


Figure 14.13 – Unbalanced FaceScrub with 30% noise

Table 14.3 – Final accuracy of the different algorithms on FaceScrub dataset with 30% noise, all the results are averaged on 3 runs.

Algorithm	Accuracy
Full-Clean	81.72
No-Sel	38.89
Forward	41.71
Co-Teaching	47.39
RAD Slim	77.51
RAD Slim Limited	67.18
Pre-Select Oracle	52.12
RAD Slim-Unbalanced	66.12
Unbalanced-No-Sel	37.11
Unbalanced-Full-Clean	69.95

14.5 Concluding Remarks

RAD is a general framework that consists of label quality predictor and classification model, where the former is mainly used to capture the label dynamics while the latter focuses on increasing the diversity of prediction. But their predictions all contribute to the final decision on detecting anomaly by the design of ensemble prediction. To adapt to the on-line nature of anomaly detection, we extend RAD with additional features of conflicting opinions of classifiers, repetitively cleaning, and oracle knowledge, corresponding to RAD Voting, RAD Active Learning, and RAD Active Learning Limited. We demonstrate the effectiveness of RAD and its extensions on three uses cases: detecting IoT device attacks, predicting task failure at Google clusters and recognising celebrity faces using FaceScrub. The evaluation results on the three use cases show remarkable accuracy values that are close to the case without anomalies in the input. In a nut shell, we prove RAD is a general robust learning framework that can be applied on different classification models and enhance their robustness against noisy inputs during the on-line training.

Chapter 15

Conclusions on Noisy Data Learning

While classification algorithms are widely applied to detect anomalies, the commonly employed assumption of clean anomaly labels often does not hold for the data collected in the wild due to careless annotation and malicious dirty label pollution. The noisy labels can significantly degrade the accuracy of anomaly detection with an increasing amount of data and are challenging to tackle due to the lack of ground truth of label quality.

Therefore, we introduce an on-line framework for robust anomaly detection: RAD, which can continuously train the system to detect abnormal behaviours from streams of arriving data after filtering out suspicious noisy ones.

RAD is a general framework that composes of label quality predictor and classification model, where the former is mainly used to capture the label dynamics while the latter focuses on increasing the diversity of prediction.

We extend RAD to RAD Voting, RAD Active Learning, and RAD Active Learning Limited to fit in the on-line nature of anomaly detection. RAD and its extensions are evaluated on three uses cases, i.e., detecting IoT device attacks, predicting task failure at Google clusters and recognising celebrity faces using FaceScrub. Results show remarkable accuracy that are close to the case without encountering anomaly input. To summarize, RAD is a general robust learning framework that can be applied on different classification models and enhance their robustness against noisy inputs during the on-line training.

However, since the data for training are selected on the fly rather than selected in the beginning, it is hard to characterize these sample-selection biases, and then it is hard to give any theoretical guarantee on the consistency of learning.

To continue improving the RAD and its extensions, one possible direction is to incorporate uncertainty ranking method into RAD Voting. RAD Voting sets directly some data as inactive, and we can thus introduce the uncertainty ranking method to RAD Voting as we proposed for RAD Active Learning Limited and RAD Slim Limited, which could re-activate more data to join the training without help of expert. For the uncertainty ranking method, such as Highest Loss Method introduced for RAD Slim Limited, we can observe that, under 30% noise and 20% consultation limit, the result difference (10.33%) between RAD Slim and RAD Slim Limited is bigger than the difference between RAD Active Learning and RAD Active Learning Limited for dataset IoT (0.06%) and Cluster (1.27%). One possible reason could be due to the uncertainty ranking method as Highest Disagreement Method for RAD Active Learning Limited is calculated based on the predictions of two models. And it is more efficient than Highest Loss Method for RAD Slim Limited, which is calculated based on the prediction of only one model. Another possible reason could be that Highest Loss Method is not good enough, further studies needed to explore this.

Part V

Conclusions and Perspectives

This thesis conducted several studies concerning three domains: system identification, control engineering and machine learning. All the issues addressed in this work can be grouped in three parts: 1) system identification and optimal control on macroeconomic models, 2) performance-based feedback control and event-based control on learning rate algorithm and 3) training machine learning model with noisy label data.

The first part explores the usage of system identification on China and France macroeconomic model. For the China use case, we conduct the analysis using methods from econometrics. We not only estimate the model, but also identify the Granger causality between variables. The contribution of this work is more on the economic side, it reveals the trend of China's economic growth transition: from export-oriented to consumption-oriented. For the France case, system identification is also performed. The final model is represented as state space equations instead of auto-regressive equations which are used in economic. We designed an optimal control schema via LQR, the objective of this control is to maintain a constant growth rate of one of the outputs (i.e. GDP). The control engineering can impose constraints on inputs and perturbations on outputs, which can emulate real world situations such as the 2008 financial crisis. When a perturbation is injected on outputs (e.g. GDP, Export and Import), one can observe the recovery trajectory of outputs and the reactions of inputs (e.g. Interest rate, Household consumption and investment). We believe that this tool can be really helpful for economist to study the evolution of an economic system. This control algorithm is coded in Simulink it is more user-friendly, it can extend its inputs and outputs with more variables, and easy to adopt to any other economic data.

The second part investigated the possibility to incorporate control theory into learning algorithms. Due to the limitation on capacity of computational resources and availability of data, continual learning are indispensable for many use cases. As the data comes in batches, first problem raised is that the time interval between two data batches can be short, therefore we need to learn as fast as possible. Another uncertainty is that data distribution between different data batches can vary a lot, so we need to make sure the model continuously improves. Moreover, converging speed and stability are two key indicators to evaluate the machine learning algorithms in a continual learning scenario. Learning rate and gradient are two main factors to control the converging speed. State of the art learning rate algorithms can be grouped into two categories: 1) time-based and 2) adaptive gradient. The theory behind the time-based ones is that at the beginning of the training, since the model is far from its optimum, we should use larger learning rate. But when the model is close to the optimum, we should use smaller learning rate to avoid skipping it. The difference between the existing time-based algorithms is only that some of them decrease faster (slower) than others, and also that some of them also add oscillations during the descent process. The disadvantage of time-based learning rate is that the trajectory of learning rate is prefixed, it can not be adjust itself based on the performance of the model. The adaptive gradient is the recent state of the art, it considers not only the current gradient, but also implement a moving window on historical gradients. The result shows that it indeed accelerates the learning process, however there are also studies show that it has two shortcoming: i) adaptive gradient algorithm tends to converge faster at the beginning of the training, but the final model has worse performance comparing to the algorithm based on stochastic gradient descent (SGD), ii) some of the adaptive gradient methods are shown not being able to converge to optimal solution in certain cases.

Therefore, we propose our first performance-based learning rate algorithm: E (Exponential)/PD (Proportional Derivative) control. The learning rate is calculate based on the current loss value and historical loss values, and we evaluate the trend of learning process to decide the current learning rate. Since our algorithm is also based on SGD, the stability is assured [61]. The experiments are based on convolutional neural network (CNN) with CIFAR10 and Fashion-MNIST datasets, the results show that not only our algorithm outperforms the comparisons in final accuracy, final loss and converging speed, but also the result curve of accuracy and loss are also extremely stable near the end of training.

Although the E/PD performs well in continual learning scenario, there is still the room for improvement. One observation from E/PD experiments is that when the loss value continuously decreases, our learning rate decreases too. This property is good to ensure the stability, but it sacrifices the converging speed. Because the continuously decreases mean the mode is approaching the optimum, we should not reduce the learning rate to slower this steps. This observation motivates us to propose our event-based learning rate algorithm based on E/PD, it prevent the sudden drop of learning rate during the PD phase of E/PD control. New results show that it improves the final accuracy, final loss value and converging speed of the model.

Another observation from E/PD experiment is that the learning curve converges so fast with E/PD control that only after a few training epochs, the model accuracy becomes stable for one data batch. Since we fix a training epoch number for each data batch, there are actually many wasted training epochs on which do not improve the model performance. This observation motivates us to think: If the learning rate can be adjusted by performance, why not also the training epochs? Therefore, the event-based training epochs algorithm is proposed, it monitors the learning process, once it detects that there is no improvement over last epochs, stops training for current data batch and turns to welcome next data batch. The results show that event-based training epochs can massively reduce the training epochs while maintain a similar performance of the model. This means it can enormously help the training if the time interval between data batches is limited.

The most important is that Event-based training epochs algorithm is independent to dataset and machine learning algorithms, therefore all the algorithms implemented on continual learning scenario can use event-based training epochs method. Future studies need to focus on comparisons between the machine learning algorithm with and without integrating event-based training epochs, as it can potentially show that this method could improve all algorithms under continual learning scenario.

The third part focus on the dirty label data learning problem. Even though is a hot topic in computer science community in recent years, it is actually a fundamental problem for both system identification and machine learning. Since the collected data is bigger and bigger it becomes much more difficult to ensure the data quality. Check each data point is a time-consuming work, but wrongly labelled data influences the results of a training model. The Robust Anomaly Detector (RAD) is a general two-layer framework we proposed to solve the dirty label data learning problem for anomaly detection use case. During the training data selection period, this first layer is primarily used to filter out the suspicious data, while this second layer mainly detects the anomaly patterns from the remaining data. With our design, predictions from both layers contribute to the final anomaly detection decision. The learning scenario is still a continual learning scenario, the difference in this work is that we will use all the accumulated data batches to update models. Experiments are implemented on two dataset: 1) IoT device attack detections 2) Google Cluster task failure predictions. Results show that RAD can definitely improve the accuracy of prediction over time, while the model can diverge if no intervention is conducted. The experiment with varying noise level shows that RAD can resist strong noise on labels.

Nevertheless, one can see an obvious shortcoming in that we train two models in RAD, but we use only first layer to select training data, the second layer only used to increase the diversities for ensemble prediction. To leverage the prediction from second layer, we proposed RAD Voting, which takes the predictions from both layers in consideration to select training data. As from the experiments with RAD we can observe that the two models are all improving over time, we let RAD Voting not only select the training data from current data batch, but also re-do the selection on historical data batch to constantly improve the data quality. Results show that RAD Voting inherits the stability from RAD, it can also resist high noise level on labels and improves the prediction accuracy comparing to RAD.

Another extension of RAD is RAD Active Learning, it introduces an expert in the framework

which we can query for the true label of data. We extract the conflicts of predictions from two models to construct a "disagreement zone" on data, and we only query this part of data to the expert. Results show that RAD Active Learning can achieve the final accuracy as if there is no noise in the labels, even if we are under 30% and 40% noise level. In reality, to send all the uncertain data to an expert can be expensive or even computationally impossible, therefore we proposed the RAD Active Learning Limited which imposes a limitation on the number of consultations send to the expert. Highest Disagreement method is also proposed to rank the uncertainty of data. The experiment result with 20% limit of batch size under 30% noise shows that the final accuracy of RAD Active Learning Limited is just slightly degraded than RAD Active Learning, which means the highest disagreement method really filter out the most uncertain data, and they are the critical one to construct models.

To implement the experiment on image dataset, due to the limitation of computational resources as spontaneously training two CNN can be too expensive and time-consuming, therefore we propose a variant of RAD Active Learning: RAD Slim, which contains only one layer, and totally delegate the usage of second layer to the expert. We also propose the framework RAD Slim Limited as a counterpart to RAD Active Learning Limited. Highest Loss method is proposed to rank the uncertainty in RAD Slim Limited. To accelerate the training process, E/PD control is also introduced in this experiment. Results show that RAD Slim largely outperforms the state of the art comparisons on the face recognition dataset, its result is very close to the case without any noise in the labels.

Obviously, Highest Loss Method still has room for improvement in order to minimize the result difference between RAD Slim and RAD Slim Limited. Since the original data label can be a dirty one, the calculated loss value can be biased. So one possible direction to continue this research is to study the distribution of the logits before last softmax layer in CNN.

In a bird's-eye view to summarize this thesis, the above three parts are all around one theme: System Identification & Machine Learning. First part studies the applications of system identification on economic data, and incorporate the estimated model into the control system. Second part touches the core part of system identification, using control theory to improve the training efficiency of machine learning model. Third part deals with the vulnerability of machine learning model regarding to mislabelled training data. All these studies show the significant benefits from conducting theoretical research to solving real world problems in a multidisciplinary team. And all work shows the possibility for further development.

Part VI

Résumé en français

Cette thèse se concentre sur l'utilisation des méthodes d'apprentissage automatique et de la théorie du contrôle pour fournir de nouvelles méthodes d'analyse aux problématiques économiques. D'abord il faut savoir que l'identification des systèmes et l'apprentissage automatique sont deux concepts similaires mais utilisés indépendamment dans les communautés automaticiennes et informaticiennes. L'identification des systèmes utilise des méthodes variées pour construire des modèles mathématiques à partir de données mesurées ou estimées. Ces modèles sont par la suite utilisées pour contrôler l'évolution future du procédé ou pour faire du diagnostic et de la maintenance. Les algorithmes d'apprentissage automatique construisent un modèle mathématique à partir des échantillons de données, appelés «données d'apprentissage» (qui sont propres ou non), afin de faire des prédictions ou des décisions sans être explicitement programmé pour le faire. La précision de la prédiction, la vitesse de convergence et la stabilité sont des facteurs clés pour évaluer le processus de l'apprentissage, en particulier dans le scénario d'apprentissage en ligne. Nous remarquons cependant que ces propriétés sont déjà été bien étudiées dans la théorie du contrôle. Le but de cette thèse est de mettre en œuvre des recherches interdisciplinaires entre ces domaines.

Les études menées dans cette thèse peuvent être divisées en trois parties/thèmes : 1) Identification du système et contrôle optimal sur les données macroéconomiques ; 2) Utilisation de la théorie du contrôle pour améliorer l'apprentissage en ligne du réseau neuronal profond ; 3) Apprentissage automatique à partir de données non fiables. Le travail de recherche a débuté par le premier thème puis il s'est élargi progressivement avec la volonté de création d'un cadre robuste pour faire face aux données d'apprentissage bruitées. Nos travaux étant axés sur la méthodologie, peuvent être appliquées aux différents types de données (économiques ou pas).

15.1 Identification du système et contrôle optimal avec des données macroéconomiques

Il existe de nombreux concepts similaires utilisés à la fois en économétrie et dans l'automatique pour réaliser l'identification des systèmes tels que: l'utilisation des modèles Exogenous Vector Autoregressive (VARX) pour représenter le système, l'utilisation du critère d'information Akaike (AIC) pour estimer l'ordre du modèle et l'utilisation du moindre carré ordinaire (MCO) pour l'estimation des paramètres. Dans cette partie notre but est d'appliquer ces techniques d'indentification dans deux études différentes. Pour montrer l'interopérabilité des méthodes, dans la première étude nous allons utiliser seulement les méthodes couramment utilisées dans le domaine de l'économétrie et dans la deuxième étude nous allons utiliser un mélange des méthodes pour optimiser les résultats. Même si la plupart des étapes sont les mêmes il existe encore des différences (par exemple sur la manière d'identifier l'ordre des modèles). Dans la seconde étude nous réalisons également un contrôle optimal de type Linear Quadratic Regulator (LQR) sur le modèle estimé en introduisant des contraintes sur les entrées, des perturbations sur les sorties. Ceci a pour but de montrer la robustesse de notre loi de contrôle.

- Nous effectuons d'abord une analyse de tendance de la transition économique de la Chine avec les méthodes d'identification des systèmes largement utilisées dans le domaine économique. Cette étude est réalisée en utilisant les données macroéconomiques de la Chine : les exportations annuelles, la consommation des ménages et le Produit Intérieur Brut (PIB) de la Chine de 1985 à 2014 qui sont obtenus auprès de la Banque mondiale. L'exportation se compose de biens et de services. Les données annuelles sur les entrées d'Investissement Direct à l'Etranger (IDE) pour la même période proviennent de diverses sources, notamment le Fonds Monétaire International (FMI) de 1985 à 1989 et UN Comtrade de 1990 à 2014. Toutes les séries sont déflatées par le déflateur du PIB (2010 = 100), les données du déflateur du PIB est également

obtenu auprès de la Banque mondiale. Toutes les variables sont exprimées en logarithme pour linéariser les séries temporelles car leurs courbes originales sont plus similaires à la distribution exponentielle.

Dans un premier temps, afin d'utiliser le régresseur des moindres carrés et d'estimer un modèle vectoriel à correction d'erreur (VECM), nous allons tester la stationnarité de chaque variable et leur première différence par le test de racine unitaire Augmented Dickey-Fuller (ADF). Le résultat montre que l'on ne peut pas rejeter l'hypothèse nulle pour les variables de niveau, mais les valeurs de leur première différence sont stationnaires. Avant d'estimer le modèle VEC, un test de cointégration est également mis en œuvre. Une équation de cointégration est estimée et elle montre que le PIB chinois dépend fortement des exportations, même si la consommation et les IDE y contribuent également.

Ensuite, nous utilisons l'équation de cointégration et la première différence de toutes les séries temporelles pour estimer un modèle VEC. Avec le modèle VEC, nous pourrions implémenter un test de Wald pour identifier la causalité de Granger entre les séries temporelles. Les résultats montrent qu'il n'y a pas de causalité de Granger bidirectionnelle entre la croissance (c'est-à-dire la première différence) du PIB et la croissance des exportations, ce que de nombreuses études sur le stade précoce (entre 1978 et 1990) de la politique de la porte ouverte de la Chine ont montré. De plus, comme nous prenons en compte la consommation, ce qui n'a pas été fait dans les études précédentes, les résultats montrent qu'il existe une causalité de Granger bidirectionnelle entre la croissance de la consommation et la croissance du PIB de la Chine. Cette étude révèle la tendance de la transition de la croissance économique de la Chine: de l'exportation à la consommation.

- Au fur et à mesure de l'avancement nous avons eu la volonté d'introduire plus de variables avec des données plus riches (par exemple des données trimestrielles et non annuelles), mais certaines données chinoises (par exemple l'investissement total ou les dépenses publiques) sont très limitées. Ces raisons nous conduisent à mener un travail en utilisant les données macroéconomiques de France. Toutes les données sont obtenues auprès de l'Institut National de la Statistique et des Études Économiques (INSEE). Après des discussions avec des experts sur la question, nous décidons d'étudier 6 séries temporelles: Production Intérieure Brute (PIB), exportation (EXP), importation (IMP), consommation des ménages (HC), formation brute de capital fixe (FBCF) et public dépense (PE). Dans notre analyse ultérieure du modèle Multi-Input Multi- Output (MIMO), GDP, EXP et IMP seront utilisés comme sorties, HC, FBCF et PE seront utilisés comme entrées. Toutes les données sont trimestrielles, allant du premier trimestre de 1980 au quatrième trimestre de 2018. Les données originales sont présentées sur les valeurs du prix courant, nous le dégonflons par le déflateur du PIB de la France (année de base: 2014) obtenu auprès de la Banque mondiale. Comme nous l'avons montré dans l'étude du modèle macroéconomique de la Chine, nous allons également implémenter le logarithme sur les séries chronologiques françaises. Pour construire un modèle vectoriel autorégressif (VAR), des tests de racine unitaire sont également effectués, le résultat montre que la première différence de logarithme de la série chronologique d'origine est stationnaire. Par conséquent, toutes les variables utilisées dans les études ultérieures seront transférées en tant que telles. Pour estimer un modèle VAR, nous estimons d'abord l'ordre de chaque équation. Cette méthode considérera non seulement les critères (c'est-à-dire les résidus du modèle estimé sur les données de validation), mais également une pénalité (c'est-à-dire le nombre de commandes). Nous obtenons une première estimation de l'ordre cependant après avoir examiné les paramètres des équations nous réduisons encore l'ordre de certains équations pour arriver à l'ordre final: 5 pour l'équation du PIB, 4 pour celle de l'exportation et 4 pour celle de l'importation. Après avoir estimé les co-

efficients du modèle VAR, nous le mettons dans une représentation de type espace d'états car cette représentation facilite la conception et l'implémentation de l'algorithme de contrôle de type Régulateur Linéaire-Quadratique (LQR) que nous allons utiliser. Comme Matlab fournit des fonctions prédéfinies pour calculer les paramètres de la loi de commande LQR nous allons utiliser ce logiciel.

La théorie du contrôle optimal concerne l'exploitation d'un système dynamique en minimisant une fonction de coût. Le cas où la dynamique du système est décrite par un ensemble d'équations différentielles linéaires et le coût est décrit par une fonction quadratique est appelé un problème de type LQ. L'un des principaux résultats de la théorie est que la solution est fournie par le régulateur linéaire- quadratique (LQR).

Nous utilisons LQR pour concevoir le système de contrôle, l'objectif de ce système est d'amener d'une façon optimale les sorties aux valeurs désignées tout en respectant certaines contraintes. Dans le critère quadratique d'optimisation que nous utilisons il y a deux matrices de paramètres Q et R que nous faisons varier pour donner plus de priorité au contrôle ou aux sorties. Nous introduisons également des contraintes sur les entrées ainsi que des perturbations sur les sorties pour simuler une crise économique. L'intégralité des lois de commande sont implémentées avec Matlab et Simulink. Les résultats montrent que la variation de la matrice R peut contrôler la vitesse de convergence du système à contrôler. Les tests avec des contraintes sur les entrées et des perturbations sur les sorties montrent que notre algorithme peut en effet simuler des scénarios proches de la réalité, ainsi ce système peut nous aide à étudier les différentes voies de reprise après une crise économique.

15.2 Utilisation de la théorie du contrôle pour améliorer l'apprentissage en ligne du réseau neuronal profond

Cette partie aborde la question de l'adaptation dynamique du taux d'apprentissage (learning rate) afin d'entrainer un réseau de neurones. Le taux d'apprentissage est un hyperparamètre qui contrôle dans quelle mesure le modèle doit être modifié en réponse à l'erreur estimée chaque fois que les poids du modèle sont mis à jour. Lorsque le taux d'apprentissage est trop élevé, la descente de gradient peut augmenter plutôt que réduire l'erreur d'apprentissage. Lorsque le taux d'apprentissage est trop faible, l'apprentissage est non seulement plus lent, mais peut rester définitivement bloqué dans un minimum local au lieu d'aller vers le minimum global. Par rapport aux nouveaux algorithmes qui se concentrent principalement sur le taux d'apprentissage basé sur le temps ou les méthodes de gradient adaptatif, nous proposons un algorithme basé sur la performance. Avant de présenter notre algorithme, nous introduisons d'abord le scénario d'apprentissage en ligne: les données sont livrées par batch et pour chaque batch de données, nous utilisons les données avec les mêmes epoch d'apprentissage pour entraîner le modèle. Il faut remarquer que nous utilisons que le nouveau batch de données pour entraîner notre modèle, l'ancien batch de données sera jeté à l'arrivée d'un nouveau batch de données.

- E (Exponential) / PD (Proportional Derivative) Control est notre algorithme de taux d'apprentissage proposé, qui vise à accélérer la vitesse de convergence en augmentant de façon exponentielle le taux d'apprentissage au début de l'apprentissage à partir d'un nouveau batch, car les données sont nouvelles et donc elles ont plus d'informations pour apprendre. Nous présentons un algorithme en deux phases pour contrôler le taux d'apprentissage: (i) une croissance exponentielle initiale suivie de (ii) un contrôle de celle-ci à partir d'une régulateur de type proportionnel dérivé (PD). Pendant la période de croissance exponentielle, le taux d'apprentissage est double augmenté à chaque pas de temps dans le but d'atteindre rapidement le voisinage du minimum

global. Cette phase est arrêtée lorsque le coût (c.à.d. l'erreur entre la classe réelle et celle prédite par le réseau) commence à augmenter, et l'évolution du taux d'apprentissage est ensuite régi par la loi de commande PD. Les paramètres PD sont initialisés avec la dernière valeur du taux d'apprentissage avant la croissance des coûts. Après avoir mis en œuvre les expériences sur CIFAR-10 et Fashion-MNIST, E/ PD est comparé à deux algorithmes de pointe Keras-Time-Based-decay et Exponential-Sine-Wave-decay. Les résultats montrent que l'E/PD surpasse les deux non seulement en termes de précision finale et de coût final, mais aussi que la vitesse de convergence et la variabilité de ces courbes de coût et de précision sont également meilleures

- Après analyse des résultats de l'expérience E / PD, nous avons pensé qu'il y avait encore de la place pour améliorer. Par exemple, nous remarquons que pendant la phase ou le taux d'apprentissage est calculé par la loi PD, alors que le coût diminue, le taux d'apprentissage est parfois mis à jour et donc diminue également. Si le coût diminue ceci signifie que nous sommes dans la bonne direction, donc abaisser le taux d'apprentissage à ce moment-là n'est pas bon pour la vitesse de convergence. Une autre amélioration est la suivante: pour chaque batch de données, la plupart de l'amélioration des performances ne se produit que pendant les premières epoch d'apprentissage. Pour résoudre ce problème, nous proposons deux méthodes basées sur le contrôle événementiel. Le premier algorithme, appelé calcul évènementiel du taux d'apprentissage, interrompt la mise à jour du taux d'apprentissage pendant la phase contrôlée par le PD tant que la fonction cout continue de diminuer. Le deuxième algorithme, appelle calcul évènementiel du taux d'apprentissage basé sur les epoch, contient une approximation locale linéaire de la fonction de cout avec m derniers valeurs. Si la pente de la courbe est supérieure à un seuil prédéfini ceci signifie que la tendance de diminution de la fonction de coût ne diminue plus et que donc nous pouvons passer à l'apprentissage à partir du prochain batch. Les résultats montrent que l'E/PD avec les deux couches évènementielles (c'est-à-dire une pour le taux d'apprentissage et l'autre pour les epoch) peut massivement réduire le temps d'apprentissage, avec parfois une valeur de coût plus faible et une plus grande précision. Par exemple, avec l'ensemble de données CIFAR-10, il pourrait économiser jusqu'à 67% des epoch d'apprentissage.

15.3 Apprentissage automatique à partir de données non fiables

L'apprentissage automatique a été largement adopté pour résoudre des problèmes tels que la classification d'images ou la reconnaissance vocale, sous l'hypothèse courante que la source de données est propre, c'est-à-dire que les caractéristiques et les étiquettes sont correctement définies. Cependant, les données collectées à partir des dispositifs automatiques ou à partir des réseaux sociales, peuvent ne pas être fiables en raison d'annotations imprudentes ou des transformations malveillantes des données (dans le but d'une détection incorrecte des anomalies par exemples. Dans cette partie nous nous concentrons uniquement sur le problème d'étiquetage faux et pas sur des erreurs sur les caractéristiques de la donnée. Il est bon à savoir qu'il est difficile d'apprendre à partir des étiquettes bruitées car ces étiquettes risquent d'être systématiquement corrompues. En tant qu'effet négatif, les étiquettes bruitées dégénèrent inévitablement la précision des classificateurs. Cet effet négatif devient plus important dans le cas d'un scénario d'apprentissage en ligne ou sur le modèle d'un réseau neuronal profond.

Dans cette partie nous proposons plusieurs algorithmes :

- **RAD:** Pour minimiser l'influence des données faussement étiquetés (noisy label en anglais), nous proposons le cadre Robust Anomaly Detector (RAD). RAD est composé de deux couches

de modèles d'apprentissage, c'est-à-dire un modèle qui juge la qualité des étiquettes suivi par un autre modèle de classificateur d'anomalies. Le premier vise principalement à différencier entre une étiquette vraie et une fausse. Pour chaque batch de nouvelles données, le modèle de qualité d'étiquette filtre les données et autorise uniquement les données «propres» à alimenter le classificateur d'anomalies. Les données "sales" sont supprimées. Le travail principal du classificateur d'anomalies est de prédire l'événement de sortie. Le processus ci-dessus décrit la phase d'apprentissage de RAD. Dans la phase de prédiction, les prédictions du modèle de qualité d'étiquette et du classificateur d'anomalies contribuent ensemble à la décision finale de classification. Cependant cette méthode pose des problèmes car toutes les données d'apprentissage pour le modèle de classification sont filtrées par le prédicteur de qualité et donc le modèle de classification sera de plus en plus similaire au celui du prédicteur de qualité. Si donc le prédicteur de qualité fait des erreurs, le modèle de classification en apprendra sur des données erronées ce qui n'est pas ce que nous voulons.

- **RAD Voting:** Par conséquent, nous mettons à jour RAD ce qui nous donne une nouvelle version: RAD Voting. Par rapport au RAD de base, nous n'utilisons pas seulement le modèle de qualité d'étiquette pour effectuer le filtrage mais nous utilisons également un classificateur d'anomalies pour rejoindre ce processus. Lorsqu'un nouveau batch de données arrive, le modèle de qualité d'étiquette effectuera d'abord le filtrage. Donne ensuite les données incertaines au classificateur d'anomalies. Le classificateur d'anomalies effectuera le même filtrage que le modèle de qualité d'étiquette, sélectionne également certaines données «propres» et certaines données incertaines. Pour la partie incertaine, si les opinions de deux modèles sont identiques, nous utiliserons la prédiction de deux modèles pour remplacer l'étiquette de la donnée. Les donnes incertaines ne sont pas jetées mais nous les ajouterons ensuite à une liste de données inactive parce que nous savons que lorsque nous continuons l'apprentissage des modèles, leur précision augmentera et nous pourrons consulter à nouveau ces données inactives.

- **RAD Active Learning:** Dans les deux cas précédents, nous supposons que nous ne sommes pas en mesure d'utiliser un opérateur humain pour vérifier la vraie valeur d'une étiquette. Dans cette partie nous supposons que nous sommes en mesure d'envoyer les données incertaines à un expert humain qui peut donner la véritable étiquette des données. Nous l'appelons RAD Active learning. Dans ce cas, pour toutes les données incertaines filtrées par les deux modèles, nous ne remplaçons pas leurs étiquettes mais nous les envoyons à un expert humain qui retournera les vraies valeurs des étiquettes pour ces données incertaines. Bien sûr, faire appel à l'opérateur humain a un coût supplémentaire. Nous ne pouvons donc pas envoyer toutes les données incertaines à l'opérateur humain, parfois nous devons limiter le nombre de données que nous pouvons envoyer à l'expert. Nous l'avons appelé RAD Active learning Limited. Dans ce cas, nous proposons un algorithme pour classer l'incertitude dans la liste des données incertaines, appelé Highest Uncertainty Method.

  Mais bien sûr, l'opérateur humain a un coût supplémentaire, nous ne pouvons donc pas envoyer toutes les données incertaines à l'opérateur humain, parfois nous devons limiter le nombre de données que nous pouvons envoyer à l'expert. Nous l'avons appelé **RAD Active learning Limited**. Dans ce cas, nous proposons un algorithme pour classer l'incertitude dans la liste des données incertaines, appelé *Highest Uncertainty Method.*

Pour évaluer les algorithmes présentées ci-dessus nous utilisons trois bases de données classiquement utilisées en machine learning: la classification des attaques IoT, la détection des pannes de cluster et la reconnaissance faciale. Les résultats montrent que RAD et ses extensions peuvent améliorer les performances du modèle d'apprentissage automatique en présence de données avec des étiquettes

fausses mais aussi dans le cas où il n'y a aucun bruit dans les étiquettes. Highest Uncertainty Method pour l'algorithme RAD Active Learning Limited est très utile lorsque le temps et les ressources de calcul sont limités.

Bibliography

[1] M. Agarwal, D. Pasumarthi, S. Biswas, and S. Nandi. Machine learning approach for detection of flooding dos attacks in 802.11 networks and attacker localization. *International Journal Machine Learning & Cybernetics*, 7(6):1035–1051, 2016.

[2] W. An, H. Wang, Y. Zhang, and Q. Dai. Exponential decay sine wave learning rate for fast deep neural network training. In *IEEE Visual Communications and Image Processing*, pages 1–4, Dec 2017.

[3] M. Anbar, R. Abdullah, B. N. Al-Tamimi, and A. Hussain. A machine learning approach to detect router advertisement flooding attacks in next-generation ipv6 networks. *Cognitive Computation*, 10(2):201–214, 2018.

[4] B. Anderson and J. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Books on Engineering. Dover Publications, 2007.

[5] C. Araujo, J.-F. Brun, and J.-L. Combes. *Econométrie*. Amphi Économie. Bréal, 2004.

[6] K. E. Årzén. A simple event-based PID controller. In *Preprints of the 14th World Congress of IFAC*, Beijing, China, 1999.

[7] K. J. Aström. Event based control. In A. Astolfi and L. Marconi, editors, *Analysis and Design of Nonlinear Control Systems*, pages 127–147. Springer Berlin Heidelberg, 2008.

[8] G. Baiocchi and W. Distaso. GRETL: Econometric software for the GNU generation. *Journal of Applied Econometrics*, 18(1):105–110, 2003.

[9] S. Banescu, C. S. Collberg, and A. Pretschner. Predicting the resilience of obfuscated code against symbolic execution attacks via machine learning. In *26th USENIX Security Symposium*, pages 661–678, Vancouver, BC, Canada, 2017.

[10] B. B. V. A. (BBVA). Overview china consumptions trends - china digital banking report. `https://www.bbvaresearch.com/wp-content/uploads/2017/03/2017-China-Consumption-Trends-1.pdf`, 2017.

[11] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, Dec. 2006.

[12] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer Berlin Heidelberg, 2012.

[13] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in Optimizing Recurrent Networks. *arXiv e-prints*, page arXiv:1212.0901, Dec. 2012.

[14] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112, Taoyuan, Taiwan, 2011.

[15] R. Birke, G. Ioana, L. Y. Chen, D. Wiesmann, and T. Engbersen. Failure analysis of virtual and physical machines: patterns, causes and characteristics. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 1–12, Atlanta, Georgia USA, 2014.

[16] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni. Virtualization in the private cloud: State of the practice. *IEEE Transaction Network and Service Management*, 13(3):608–621, 2016.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 1 edition, 2007.

[18] O. J. Blanchard. The production and inventory behavior of the american automobile industry. *Journal of Political Economy*, 91(3):365–400, 1983.

[19] O. J. Blanchard and S. Fischer. *Lectures on Macroeconomics*. The MIT Press, 1989.

[20] F. Brayton, T. Laubach, and D. Reifschneider. Optimal-control monetary policy in the frb/us model. *FEDS Notes*, Nov. 2014.

[21] J. R. Campos, M. Vieira, and E. Costa. Exploratory study of machine learning techniques for supporting failure prediction. In *14th European Dependable Computing Conference (EDCC) Iaşi, Romania*, pages 9–16. IEEE Computer Society, 2018.

[22] D. Cass. Optimum Growth in an Aggregative Model of Capital Accumulation1. *The Review of Economic Studies*, 32(3):233–240, 1965.

[23] S. Cerf. *Control Theory for Computing Systems: Application to big-data cloud services & location privacy protection*. Phd thesis, Université Grenoble Alpes, May 2019.

[24] S. Cerf, R. Birke, and L. Y. Chen. Duo learning for classifications with noisy labels. In *Continual Learning Workshop, in conjunction with Neural Information Processing Systems (NIPS), Montréal, Canada*, 2018.

[25] Y.-W. Cheung and K. S. Lai. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics*, 13(3):277–280, 1995.

[26] F. Chollet. *Deep Learning with Python*. Manning Publications Co., Greenwich, CT, USA, 2017.

[27] G. Chow. *Analysis and Control of Dynamic Economic Systems*. A Wiley publication in applied statistics. Wiley, 1975.

[28] G. C. Chow. *Development of Control Theory in Macroeconomics*, volume 7. Springer, Dordrecht, 1987.

[29] C. W. Clark. *Mathematical bioeconomics : the optimal management of renewable resources*. Wiley New York, 1976.

[30] P. P. CPIC (China Pacific Insurance Company). White paper on china's insurance consumers. `https://www.cpic.com.cn/c/2018-10-26/1561780.shtml`, 2018.

[31] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-oneloss. *Machine Learning - ML*, 29:103–130, 1997.

[32] J. C. R. Dow. The optimum rate of saving. *The Economic Journal*, 69(274):353–356, 1959.

[33] T. Dozat. Incorporating Nesterov momentum into Adam. In *4th International Conference on Learning Representations*, San Juan, Puerto Rico, May 2016.

[34] H. N. Duong. *Identification structurelle et paramétrique des systèmes linéaires monovariables et multivariables*. Phd thesis, Grenoble INP, 1993.

[35] H. N. Duong and I. D. Landau. On a criterion for estimating the structure of linear mimo systems. In *European Control Conference*, Groningen, Netherlands, 1993.

[36] S. Durand and N. Marchand. Further results on event-based PID controller. In *Proceedings of the European Control Conference*, Budapest, Hungary, 2009.

[37] T. U. T. H. . Edition. China foreign investment report. `http://images.mofcom.gov.cn/wzs/201810/20181009090547996.pdf`, 2018.

[38] B. Eichengreen, D. Park, and K. Shin. When Fast Growing Economies Slow Down: International Evidence and Implications for China. NBER Working Papers 16919, National Bureau of Economic Research, Inc, Mar. 2011.

[39] R. Engle and C. Granger. Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–76, 1987.

[40] Z. Eric, W. Jeffrey, and K. Siem Jan. State space modeling in macroeconomics and finance using ssfpack in s+finmetrics. 2004.

[41] Y. Fan, J. Li, and D. Zhang. A method for identifying critical elements of a cyber-physical system under data attack. *IEEE Access*, 6:16972–16984, 2018.

[42] Z. Fang, J. Tzeng, C. C. Chen, and T. Chou. A study of machine learning models in epidemic surveillance: Using the query logs of search engines. In *Pacific Asia Conference on Information Systems (PACIS), Taipei,Taiwan*, page 137. AIS eLibrary, 2010.

[43] L. Fei-Fei, R. Krishna, and D. Xu. Cs231n: Convolutional neural networks for visual recognition. `https://cs231n.github.io/neural-networks-3/`.

[44] B. Frénay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Transaction Neural Networks Learning Systems*, 25(5):845–869, 2014.

[45] B. M. Friedman. *Economic Stabilization Policy: Methods in Optimization*. North-Holland Publishing Company and American Elsevier Publishing Company, Amsterdam and New York, 1975.

[46] A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*, volume Analytical methods for social research. Cambridge University Press, New York, 2007.

[47] D. Georges and I. Girerd-Potin. A Discrete-Time State Observer Approach to Discovering Portfolio Holdings. In *20th IFAC World Congress (IFAC WC 2017)*, volume 50, pages 946 – 951, Toulouse, France, July 2017.

[48] A. Ghiassi, T. Younesian, Z. Zhao, R. Birke, V. Schiavoni, and L. Y. Chen. Robust (deep) learning framework against dirty labels and beyond. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 236–244, 2019.

[49] G. Giantamidis and S. Tripakis. Learning moore machines from input-output traces. In J. S. Fitzgerald, C. L. Heitmeyer, S. Gnesi, and A. Philippou, editors, *FM 2016: Formal Methods - 21st International Symposium*, volume 9995 of *Lecture Notes in Computer Science*, pages 291–309, Limassol, Cyprus, 2016.

[50] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 2010.

[51] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010.

[52] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[53] G. C. Goodwin, S. F. Graebe, and M. E. Salgado. *Control System Design*. Prentice Hall PTR, USA, 1st edition, 2000.

[54] P.-O. Gourinchas. Notes for Econ202a: The Ramsey-Cass-Koopmans Model, U.C. Berkeley,. `https://eml.berkeley.edu/~webfac/gourinchas/e202a_f14/Notes_Ramsey_Cass_Koopmans_pog.pdf`, 2014.

[55] C. Granger and P. Newbold. Spurious regressions in econometrics. *Journal of Econometrics*, 2(2):111–120, 1974.

[56] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

[57] R. P. Guidorzi. Invariants and canonical forms for systems structural and parametric identification. *Automatica*, 17(1):117–133, 1981.

[58] L. Guilhot. Le nouveau modèle de croissance de l'économie chinoise, un moyen pour relever le défi de la trappe à revenu intermédiaire ? In *XXXIèmes journées ATM "Le bilan des Objectifs du Millénaire pour le développement 15 ans après : réduction de la pauvreté et/ou montée des inégalités ?"*, page 16 p., Rouen, France, June 2015.

[59] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. W. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pages 8536–8546, Montréal, Canada, 2018.

[60] R. Han, C. H. Liu, S. Li, L. Y. Chen, G. Wang, J. Tang, and J. Ye. Slimml: Removing non-critical input data in large-scale iterative machine learning. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[61] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, page 1225–1234, 2016.

[62] R. Hawkins, J. Speakes, and D. Hamilton. Monetary policy and PID control. *Journal of Economic Interaction and Coordination*, 10, 2014.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 770–778, Las Vegas, USA, 2016.

[64] Y. He, G. J. Mendis, and J. Wei. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. *IEEE Transaction Smart Grid*, 8(5):2505–2516, 2017.

[65] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 601–608, Vancouver, B.C., Canada, 2006.

[66] M. Intriligator. *The applications of control theory to economics*, volume 28, pages 605–626. Springer, Berlin, Heidelberg, 2005.

[67] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[68] L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning, ICML Stockholmsmässan, Stockholm, Sweden*, pages 2309–2318, 2018.

[69] S. Johansen. Testing weak exogeneity and the order of cointegration in uk money demand data. *Journal of Policy Modeling*, pages 313–334, 1992.

[70] J. Kang, I. Joo, and D. Choi. False data injection attacks on contingency analysis: Attack strategies and impact assessment. *IEEE Access*, 6:8841–8851, 2018.

[71] D. Karagiannis and A. Argyriou. Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning. *Vehicular Communications*, 13:56–63, 2018.

[72] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[73] T. C. Koopmans. On the Concept of Optimal Economic Growth. Cowles Foundation Discussion Papers 163, Cowles Foundation for Research in Economics, Yale University, 1963.

[74] J. Koza, F. Bennett III, D. Andre, and M. Keane. *Automated Design Of Both The Topology And Sizing Of Analog Electrical Circuits Using Genetic Programming*, pages 151–170. Springer, Dordrecht, 1996.

[75] R. Kozik, M. Choras, M. Ficco, and F. Palmieri. A scalable distributed machine learning approach for attack detection in edge computing environments. *Journal of Parallel and Distributed Computing*, 119:18–26, 2018.

[76] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[77] A. Krizhevsky, I. Sutskever, and H. Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012), Lake Tahoe, Nevada USA*, pages 1097–1105. 2012.

[78] R. Kunst. State Space Models and the Kalman Filter. Seminar paper prepared for 40461 Vektorautoregressive Methoden, 2007.

[79] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR) (Poster)*, Toulon, France, 2017.

[80] I. D. Landau and G. Zito. *Digital Control Systems: Design, Identification and Implementation (Communications and Control Engineering)*. Springer-Verlag, UK, London, 2006.

[81] N. Lardy. China: Toward a consumption-driven growth path. *SSRN Electronic Journal*, 10 2006.

[82] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In G. Hua and H. Jégou, editors, *Computer Vision – European Conference on Computer Vision (ECCV) 2016 Workshops*, pages 47–54, Amsterdam, Netherlands, 2016.

[83] M. Lease. On quality control and machine learning in crowdsourcing. In *Human Computation*, volume 11, 2011.

[84] M. Lemoine and F. Pelgrin. Introduction aux modèles espace état et au filtre de Kalman. *Revue de l'OFCE*, (86):203–229, June 2003.

[85] X. Li, G. Zhang, H. H. Huang, Z. Wang, and W. Zheng. Performance analysis of Gpu-based convolutional neural networks. In *45th IEEE International Conference on Parallel Processing*, pages 67–76, Philadelphia, PA, USA, 2016.

[86] Y. Li, Z. Chen, L. Yan, and S. Luo. Research on the relationship between foreign trade and gdp growth in west china - empirical analysis based on panel causality. *International Journal of Intellectual Property Management (IJIPM)*, 1:84–93, 10 2010.

[87] X. Liu, P. Burridge, and P. Sinclair. Relationships between economic growth, foreign direct investment and trade: Evidence from china. *Applied Economics*, 34:1433–40, 02 2002.

[88] J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46:211–215, 2010.

[89] L. Luo, Y. Xiong, Y. Liu, and X. Sun. Adaptive gradient methods with dynamic bound of learning rate. In *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[90] D. Léonard and N. V. Long. *Optimal Control Theory and Static Optimization in Economics*, page 117–126. Cambridge University Press, 1992.

[91] H. Lütkepohl. *New introduction to multiple time series analysis*. Springer, Berlin, 2005.

[92] E. Malach and S. Shalev-Shwartz. Decoupling "when to update" from "how to update". In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 961–971, Red Hook, NY, USA, 2017.

[93] N. Marchand, S. Durand, and J. F. Guerrero-Castellanos. A general formula for event-based stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, (5):1332–1337, 2013.

[94] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[95] S. Minaee. 20 popular machine learning metrics. part 1: Classification & regression evaluation metrics. `https://towardsdatascience.com/@shervin.minaee`, 2019.

[96] P. R. O. C. MINISTRY OF COMMERCE(MOFCOM). China foreign investment report. `http://images.mofcom.gov.cn/wzs/201612/20161230131233768.pdf`, 2016.

[97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[98] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 04 2017.

[99] B. T. Morris and M. M. Trivedi. Learning, modeling, and classification of vehicle track patterns from live video. *IEEE Transactions on Intelligent Transportation Systems*, 9(3):425–437, Sep. 2008.

[100] R. M. Murray. Control and dynamical systems. `http://www.cds.caltech.edu/~murray/courses/cds110/wi06/lqr.pdf`, 2006.

[101] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, Lake Tahoe, Nevada, United States, 2013.

[102] Y. NESTEROV. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Doklady Akademii Nauk, USSR*, 269:543–547, 1983.

[103] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 343–347, Paris, France, 2014.

[104] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 2233–2241, Hawaii, USA, 2017.

[105] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[106] A. Pellegrini, P. di Sanzo, and D. R. Avresky. A machine learning-based framework for building application failure prediction models. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 1072–1081, Hyderabad, India, 2015.

[107] T. Pitakrat, A. van Hoorn, and L. Grunske. A comparison of machine learning algorithms for proactive hard disk drive failure detection. In P. Kruchten and S. Malek, editors, *Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems, ISARCS, Vancouver, BC, Canada*, pages 1–10, Vancouver, BC, Canada, 2013.

[108] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[109] F. P. Ramsey. A mathematical theory of saving. *The Economic Journal*, 38(152):543–559, 1928.

[110] A. Rao. Understanding risk-aversion through utility theory. `http://web.stanford.edu/class/cme241/lecture_slides/UtilityTheoryForRisk.pdf`, 2020.

[111] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, 2018.

[112] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14, 2011.

[113] M.-F. Renard. *L'économie de la Chine*. Repères. Editions La Découverte, 2018.

[114] U. Reuter, A. Sultan, and D. S. Reischl. A comparative study of machine learning approaches for modeling concrete failure surfaces. *Advances in Engineering Software*, 116:67–79, 2018.

[115] L. A. Rivera-Batiz and P. M. Romer. Economic integration and endogenous growth. *Quarterly Journal of Economics*, 106:531–555, 1990.

[116] A. Rosà, L. Y. Chen, and W. Binder. Understanding the dark side of big data clusters: An analysis beyond failures. In *IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 207–218, Rio de Janeiro, Brazil, 2015.

[117] A. Rosà, L. Y. Chen, and W. Binder. Failure analysis and prediction for big-data systems. *IEEE Transaction Services Computing*, 10(6):984–998, 2017.

[118] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. In *Methodology And Computing In Applied Probability*, page 127–190, 1999.

[119] S. Ruder. An overview of gradient descent optimization algorithms. `https://ruder.io/optimizing-gradient-descent/index.html#fn7`, 2016.

[120] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 815–823, Boston, USA, 2015.

[121] A. Seierstad and K. Sydsaeter. *Optimal Control Theory with Economic Applications*. Elsevier North-Holland, Inc., USA, 1986.

[122] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[123] L. N. Smith. Cyclical learning rates for training neural networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, Santa Rosa, California, 2017.

[124] T. Söderström and P. Stoica. *System Identification*. Prentice-Hall, Inc., USA, 1988.

[125] R. M. Solow. A Contribution to the Theory of Economic Growth. *The Quarterly Journal of Economics*, 70(1):65–94, 02 1956.

[126] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. In *3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[127] T. W. Swan. Economic growth and capital accumulation. *The Economic Record*, 32(2):334–361, November 1956.

[128] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, Columbus, OH, USA, 2014.

[129] S. Tang, E. A. Selvanathan, and S. Selvanathan. Foreign direct investment , domestic investment , and economic growth in china a time series analysis. `https://www.econstor.eu/bitstream/10419/63416/1/560603630.pdf`, 2008.

[130] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1195–1204, Red Hook, NY, USA, 2017.

[131] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[132] M. Triantafyllou. Maneuvering and control of surface and underwater vehicles. `https://ocw.mit.edu/courses/mechanical-engineering/`, 2004.

[133] W. H. Tsen. Exports, domestic demand, and economic growth in china: Granger causality analysis. *Review of Development Economics*, 14:625–639, 2010.

[134] V. N. Vagin and M. V. Fomina. Problem of knowledge discovery in noisy databases. *International Journal Machine Learning & Cybernetics*, 2(3):135–145, 2011.

[135] M. Velasco, P. Martí, and E. Bini. On Lyapunov sampling for event-driven controllers. In *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, 2009.

[136] J. Vlcek. Maximum likelihood estimation of parameters in state-space models. *IFAC Proceedings Volumes*, 34:105–109, 2001.

[137] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, Salt Lake City, UT, USA, 2018.

[138] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 4148–4158, Long Beach, CA, USA, 2017.

[139] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[140] J. Xue, R. Birke, L. Y. Chen, and E. Smirni. Spatial-temporal prediction models for active ticket managing in data centers. *IEEE Transaction Network and Service Management*, 15(1):39–52, 2018.

[141] Z. Zhang. Lecture on deep learning for spsas. `https://sites.usp.br/datascience/wp-content/uploads/sites/449/2019/08/SPSAS_Deep_Learning_Atlas_Wang.pdf`, 2019.

[142] Z. Zhao, S. Cerf, R. Birke, B. Robu, S. Bouchenak, S. B. Mokhtar, and L. Y. Chen. Robust anomaly detection on unreliable data. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA*, pages 630–637, 2019.

[143] Z. Zhao, S. Cerf, B. Robu, and N. Marchand. Feedback control for online training of neural networks. In *IEEE Conference on Control Technology and Applications*, pages 136–141, Hong Kong, SAR, China, 2019.

[144] Z. Zhao, S. Cerf, B. Robu, and N. Marchand. Feedback control for online training of neural networks. In *IEEE Conference on Control Technology and Applications*, pages 136–141, Hong Kong, China, 2019.

[145] Z. Zhao, S. Cerf, B. Robu, and N. Marchand. Event-based control for online training of neural networks. *IEEE Control Systems Letters*, 4:773–778, 2020.

[146] Z. Zhao, L. Job, L. Dugard, and B. Robu. Modelling and dynamic analysis of the domestic demand influence on the economic growth of China. In *Conference on International Development Economics*, Proceedings of the Conference on International Development Economics, Clermont-Ferrand, France, Nov. 2018.

[147] B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu, and Z. Li. Machine-learning-based online distributed denial-of-service attack detection using spark streaming. In *2018 IEEE International Conference on Communications, Kansas City, MO, USA*, pages 1–6. IEEE, 2018.

Part VII

Appendix

.1 Inter-temporal elasticity of substitution

It's a measure of responsiveness of the growth rate of consumption to the real interest rate, if the real rate rises, current consumption may decrease due to increased return on savings; but current consumption may also increase as the household decides to consume more immediately, as it is feeling richer. The effect on current consumption is the inter-temporal elasticity of substitution

.2 Discount Rate

Discount rate refers to the interest rate used in discounted cash flow analysis to determine the present value of future cash flows. For instance, if company A have a financial bill of company B, this financial bill can take $100 from bank in one month, but company A do not want to wait a month, so he can ask bank to take the money now, but bank can only give company A $98, so the discount rate here is: 100/98 - 1 = 2.04%. Given a discount Rate ρ, the value of $u(0)$ becomes $u(0)(1-\rho)^t$ after time t.

.3 Risk Aversion

A person is given the choice between two scenarios, one with a guaranteed payoff and one without. In the guaranteed scenario, the person receives $40. In the uncertain scenario, a coin (unbiased) is flipped to decide whether the person receives $100 or nothing, so the expected value is $50. Will you receive the $40 or take the chance? Before introduce the concepts, we first define the symbol in Tab. .1 which will be used to explain the notions.

Table .1 – Symbol description for Risk Averse

Symbol	Description
CE	Certainty equivalent
$E(U(W))$	Expected value of the utility (expected utility)
$E(W)$	Expected value of the uncertain payment
$U(CE)$	Utility of the certainty equivalent
$U(E(W))$	Utility of the expected value of the uncertain payment
$U(W_0)$	Utility of the minimal payment
$U(W_1)$	Utility of the maximal payment
W_0	Minimal payment
W_1	Maximal payment
RP	Risk premium

One should notice that in the following three circumstances, the case.3.1 is chosen for our Ramsey-Cass-Koopmans model.

.3.1 Risk Averse

Figure. .1 shows the relation between payment and utility for risk-averse case. By the axiom continuity of "Expected Utility Hypothesis", $E(U(W)) = (U(W_) + U(W_1))/2$. At CE, it has the same utility of E(W)(where E(U(W)) = U(CE)), here CE is the $40 for sure, and W0, W1, E(W) are the $0, $100

and $50 for gambling. For a person of “Risk Aversion” with utility function in the figure, take the $40 or take the gambling has the same utility. He/She tends to accept a guaranteed value which may be smaller than the expected value.

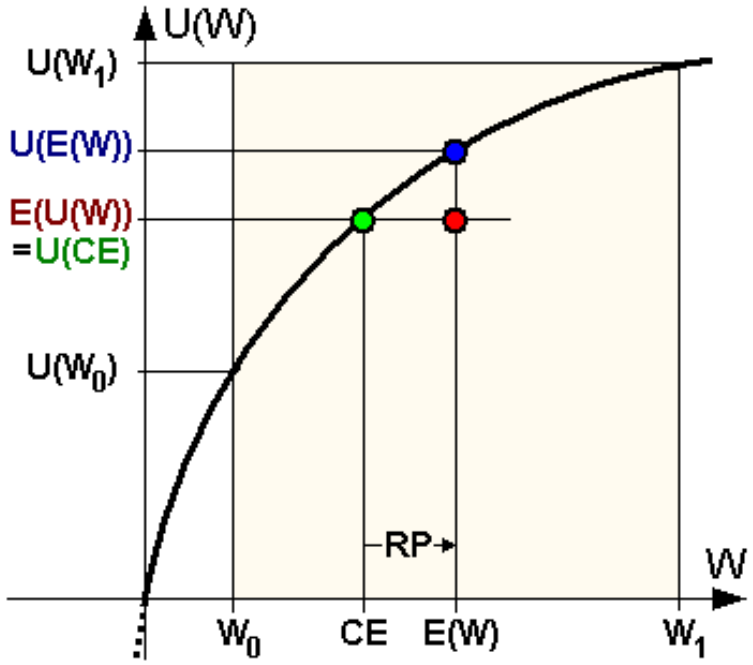


Figure .1 – Utility function of a risk-averse (risk-avoiding) individual.

.3.2 Risk Neutral

Risk neutral means the person is totally rational, if the guaranteed payment is lower than expected value, he will take the chance.

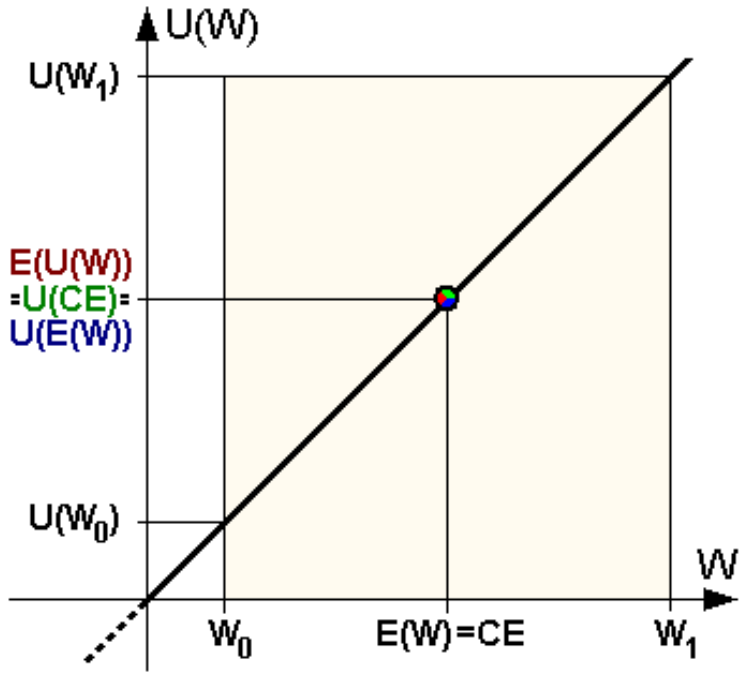


Figure .2 – Utility function of a risk-neutral individual.

.3.3 Risk Loving

Risk loving person will take the chance even the certain value is bigger than expected value, they will still take the gambling for $100!.

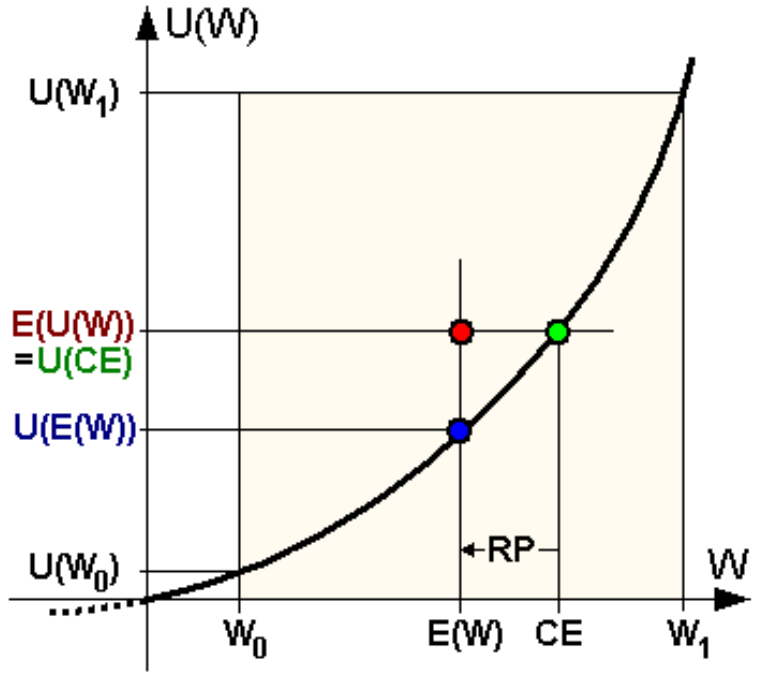


Figure .3 – Utility function of a risk-loving (risk-seeking) individual

.4 An Alternative Formulation of Household Function

$$b = B/AL$$

$$\begin{aligned}\dot{b} &= (B/AL)' \\ &= \frac{\dot{B}AL - B(AL)'}{(AL)^2}\end{aligned}$$

$$\begin{aligned}AL &= A(0)L(0)e^{g+n}t \\ (AL)' &= (g+n)AL \\ \dot{B} &= \dot{b}AL + b(g+n)AL\end{aligned}$$

.5 Inferential reasoning formula

$$u'(c) = \lambda(t)$$

$$\begin{aligned}\dot{\lambda}(t) &= \frac{\partial u'(c)}{\partial t} \\ &= \frac{\partial u'(c)\partial t}{\partial c \partial t} \\ &= u''(c)\dot{c}\end{aligned}$$

$$\frac{\dot{\lambda}(t)}{\lambda(t)} = \frac{u''(c)\dot{c}}{u'(c)}$$

From the definition in Eq. (4.11), we know that $\theta = -Cu''(C)/u'(C)$, then:

$$\begin{aligned}\frac{u''(c)}{u'(c)} &= -\frac{\theta}{c} \\ \frac{u''(c)\dot{c}}{u'(c)} &= -\frac{\theta\dot{c}}{c}\end{aligned}$$

then:

$$\frac{\dot{\lambda}(t)}{\lambda(t)} = -\theta\frac{\dot{c}(t)}{c(t)}$$