Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Curso de Graduação em Engenharia de Computação

CROSS-DOMAIN FAULT DIAGNOSIS THROUGH OPTIMAL TRANSPORT

Eduardo Fernandes Montesuma

Fortaleza – Ceará
2021

Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Curso de Graduação em Engenharia de Computação

CROSS-DOMAIN FAULT DIAGNOSIS THROUGH OPTIMAL TRANSPORT

Autor
Eduardo Fernandes Montesuma

Orientadora
Profa. Dra. Michela Mulas

Trabalho de Conclusão de Curso submetido à Coordenação do Programa de Graduação em Engenharia de Computação da Universidade Federal do Ceará como parte dos requisitos para a obtenção do grau de ***Bacharel em Engenharia de Computação****.*

Fortaleza – Ceará
2021

Eduardo Fernandes Montesuma

Cross-Domain Fault Diagnosis Through Optimal Transport

Este trabalho foi julgado adequado para a obtenção do título de Bacharel em Engenharia de Computação e aprovado em sua forma final pelo Departamento de Engenharia de Teleinformática da Universidade Federal do Ceará.

Eduardo Fernandes Montesuma

Banca Examinadora:

Profa. Dra. Michela Mulas (Orientadora)
Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará

Dr. Fred-Maurice Ngolè Mboula
CEA-List
Université Paris-Saclay

Prof. Dr. Charles Casimiro Cavalcante
Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará

Fortaleza, 8 de Abril de 2021

Dedico este trabalho à todas e todos os pesquisadores brasileiros,
que em meio a ataques frequentes, têm trabalhado incansavelmente
para a expansão deste ente abstrato chamado *conhecimento humano*.

Agradecimentos

Aos meus pais, Viviane e Gigliani, por todo o amor, carinho e atenção. Agradeço por sempre terem reconhecido a importância da educação formal, e por sempre terem me incentivado a estudar. Por isso, à vocês devo a realização deste trabalho. Sobretudo, agradeço por terem feito com que eu acredite na minha própria capacidade.

À minha namorada, Maria Eduarda, pelo amor perene, e por ser mesmo em meio ao caos e à inconstância da vida, o ponto de equilíbrio do meu dia-a-dia. Também agradeço os comentários e sugestões que ajudaram a melhorar a qualidade deste trabalho.

À minha orientadora, profa. Michela Mulas, pelo acompanhamento atento nestes últimos três anos de curso. Agradeço o direcionamento dado à minha carreira enquanto pesquisador. Também, agradeço o conhecimento transmitido em discplinas e durante a orientação dos meus trabalhos. Finalmente, agradeço a atenção dada na orientação desta monografia, sem a qual esta não teria a qualidade que teve.

Ao Dr. Fred-Maurice Ngolè Mboula, pelo acompanhamento na reta final do meu duplo diploma na França, e por ter me dado a oportunidade de começar a estudar Optimal Transport, assunto este que sou curioso desde então.

À todos os meus colegas de curso, sejam da minha turma, de turmas anteriores ou posteriores. Agradeço o prazer de poder compartilhar estes últimos 6 anos da minha vida. Agradeço também todo o apoio e encorajamento. Por fim, agradeço pelos comentários e discussões que ajudaram a tornar este trabalho melhor.

À banca examinadora deste trabalho, pelo tempo dedicado a ler esta monografia e pelos comentários pertinentes.

"There is Nothing More Practical Than A Good Theory."
Kurt Lewin on Field theory in social science (1952)

Resumo

Os sistemas de diagnóstico automático de falhas são um importante componente para a tolêrancia à falhas em sistemas de controle. No entanto, o treinamento de sistemas de diagnóstico pode ser custoso, ou mesmo perigoso dado que dados de falha precisam ser coletados quando o processo opera em condições pergiosas. Uma possível solução é treinar sistemas de diagnóstico automático em dados de simulação. Todavia, devido a erros de modelagem, os dados adquiridos podem não refletir o processo real. Esse caso é caracterizado por uma mudança na distribuição de probabilidade à qual os dados são amostrados. Esse problema é conhecido na literatura como adaptação de domínios, ou diagnóstico de falha entre domínios no nosso contexto. Portanto, esse trabalho analisa o diagnóstico de falhas entre domínios sob um ponto de vista do transporte ótimo. Nós aplicamos nossa metodologia em um estudo de caso chamado de reator perfeitamente agitado (CSTR). Nossa contribuição é tríplice: 1. nós apresentamos um estudo comparativo sobre a escolha de método para a extração de características, e sobre a escolha de algoritmos de adaptação de domínios, 2. Nós analisamos a relação entre a especificação errada de paramêtros de um modelo e a distância entre distribuições de probabilidade, e o impacto desta distância na performance de classificação 3. Nós analisamos o impacto de erros de modelagem na qualidade do plano de transporte ótimo, e a influência que este tem na performance de classificação.

Em síntese, nossos resultados mostram que a adaptação de domínios baseada em transporte ótimo é a melhor escolha para resolver a problemática da detecção de falhas entre domínios. Além disso, nós verificamos que um grau maior de erros de modelagem está correlacionado com um aumento na distância entre distribuições de probabilidade. Nesse sentido, nossos resultados mostram que quanto maior a distância, pior tende à ser a performance de classificação, confirmando resultados teóricos prévios. Finalmente, nós mostramos que erros de modelagem podem levar o plano de transporte a transferir massa entre falhas distintas, prejudicando a taxa de acerto de classificadores.

Palavras-Chaves: Diagnóstico de Falhas, Sistemas Dinâmicos, Adaptação de Domínios, Aprendizagem por Transferência, Transporte Ótimo.

Abstract

AUTOMATIC fault diagnosis systems are an important component for fault tolerance in modern control loops. Nonetheless, the training of such diagnosis systems can be costly or even dangerous since faulty data need to be collected by driving the process to dangerous conditions. A possible solution to the said problem is training an automatic diagnosis system solely on simulation data. However, due to modeling errors, the data acquired may not reflect real process data. This is characterized by a change in the probability distributions upon which data is sampled. This problem is known in the literature as domain adaptation or cross-domain fault diagnosis in our context. Thus this work analyzes the cross-domain diagnosis problem through the point of view of optimal transport. We apply our methodology in a case study concerning the continuous stirred tank reactor (CSTR) system. Our contributions are three-fold: 1. we perform a comparative study concerning feature extraction and domain adaptation algorithms, 2. we analyze the relation between wrongful model specification and the distance between source and target distributions, and its impact on classification performance 3. we analyze the impact of modeling errors in the quality of optimal transport plans, and the influence of this latter factor into classification performance.

In summary, we found that optimal transport-based domain adaptation is the best choice for solving the distributional shift problem. In addition, we further verified that an increasing degree of modeling error is correlated with an increase in the distance between source and target distributions. Furthermore, we found experimentally that the latter distance is correlated with a decrease in classification performance, confirming previous theoretical findings. Finally, the degree of modeling error can cause the transportation plan between source and target domain to transfer mass between different classes, harming classification performance.

Keywords: Fault Diagnosis, Dynamical Systems, Domain Adaptation, Transfer Learning, Optimal Transport.

Contents

List of Figures

List of Tables

List of Abbreviations

ACF Autocorrelation Function. iii–v, 23–27, 31, 40, 43, 69–75, 77–79, 83, 93

AWGN Additive White Gaussian Noise. 24

BCD Block Coordinate Descent. 55, 56

CNN Convolutional Neural Network. iv, v, 4, 19, 21, 29, 33–35, 53, 69–74, 77, 83, 84, 94, 95

CSTR Continuous Stirred Tank Reactor. iv, vii, 63–65, 68, 72, 76, 83, 121, 122

DA Domain Adaptation. 45

DANN Domain-Adversarial Neural Network. 51, 53, 74, 95

DNN Deep Neural Network. 4, 34, 53, 84

EMD Earth-Mover Distance. 39

FDD Fault Detection and Diagnosis. 1–4, 9, 17, 63, 84

FOPTD First Order plus Time Delay. v, 101–104

GFK Geodesic Flow Kernel. 51, 52, 74, 95

i.i.d independently and identically distributed. 25

IAE Integral Absolute Error. 105, 106

JDOT Joint Distribution Optimal Transport. 53, 56, 75–77, 95, 97

KKT Karush-Kuhn-Tucker. 98, 100

KLIEP Kullback-Leibler Importance Estimation Procedure. 49–51, 72, 73, 95, 106, 107, 109

KMM Kernel Mean Matching. iv, 49, 51, 72, 73, 95, 108, 109

KP Kantorovich Problem. 39, 41

LSIF Least Squares Importance Fitting. 49, 50, 108

MLP Multi-Layer Perceptron. 33, 79

MMD Maximum Mean Discrepancy. 45–48, 50–52, 58, 59, 78, 109, 121

MP Monge Problem. 38, 75

ODE Ordinary Differential Equation. 1, 9, 64, 114

OT Optimal Transport. iv, 6, 37, 60, 76, 77, 83, 84

OTDA Optimal Transport for Domain Adaptation. iv, 5, 6, 53–55, 59, 75–77, 80, 81, 84, 95

PCA Principal Components Analysis. 34, 52, 74, 95

PID Proportional Integral Derivative. iii, v, 16, 103, 104, 106, 116

PLS Partial Least Squares. 52, 74

RBF Radial Basis Function. 30, 31, 79

ReLU Rectified Linear Unit. 33

RKHS Reproducing Kernel Hilbert Space. 46, 108, 119, 120

RMSE Root Mean Squared Error. 14, 102, 103

RNN Recurrent Neural Network. 84

SLP Single-Layer Perceptron. 33

SSS Strict-Sense Stationary. 23

SVD Singular Value Decomposition. 110

SVM Support Vector Machine. iii, iv, 4, 19, 21, 29–31, 50, 52, 69–73, 79, 92, 93, 95, 97, 99, 100

t-SNE t-Stochastic Neighbor Embeddings. iv, 34, 35, 69

TCA Transfer Component Analysis. 51, 52, 74, 95, 109

TL Transfer Learner. 58

TV Total Variation. 45, 46, 105

uLSIF unconstrained Least Squares Importance Fitting. 51, 72, 73, 95, 97, 107

UMF Undesired Mass Flow. iv, 60, 61, 80, 81, 84

UNTG Unsupervised Negative Transfer Gap. 58, 80

WN White Noise. 24, 25

WSS Wide Sense Stationarity. 23

Chapter 1

Introduction

This chapter introduces the topics and structure of this thesis. In Section 1.1 we provide a brief overview of previous work in dynamical systems, fault diagnosis, optimal transport, domain adaptation, and cross-domain fault diagnosis. In Section 1.2 we presented the motivation behind our line of work. In Section 1.3 we describe the thesis objectives as well as we highlight our main contributions. Finally, Section 1.4 presents the structure of this work.

1.1 Related Work

In the context of Fault Detection and Diagnosis (FDD), the present work deals with the design of automated fault diagnosis tools. In this regard, FDD concerns the detection, and later diagnosis of faults occurring in a given system. A fault, as first defined by [Isermann, 1997], is an unpermitted deviation of at least one characteristic property of a variable from an acceptable behavior. In this regard, we will further focus on faults occurring in physical and control systems.

A physical system, or a process, is any physical phenomenon. These phenomenon are typically studied through mathematical modeling, which focus on devising a mathematical relationship between their inputs and outputs. In this context, there are least two choices for modeling this mathematical relationship, namely, input-output and state-space representations. The latter is done through Ordinary Differential Equations (ODEs), and also called a dynamical system [Strogatz, 2018].

Moreover, controls systems are one of the building blocks of modern technology. Indeed, as highlighted by [Nise, 2020], they are an integral part of rockets, space shuttles, self-guided vehicles, robots and chemical plants, to name a few. Thus, the goal of a control system is to manipulate a given process input, in order to drive its output to a desired value. Throughout this thesis a particular kind of control system will be explored, called feedback system, as in Figure 1.1.

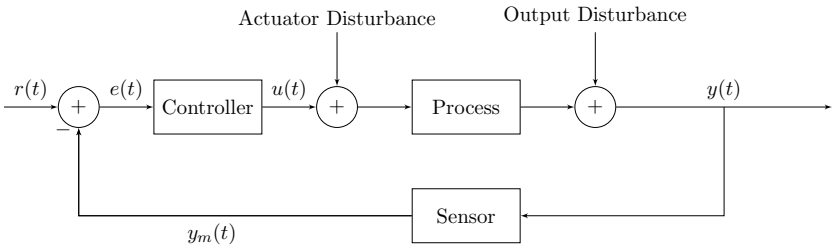


Figure 1.1: Diagram of a general control system for controlling an arbitrary dynamical system. In this setting, the controller estimates the needed action by measuring the error between the measured output, y_m, and the desired output, also known as reference r. Figure reproduced from [Nise, 2020].

With respect to Figure 1.1, faults may appear in each component. For instance, sensors may display wrongful readings, affecting the input's estimation by the controller. In addition, disturbances may affect the control system behavior, harming the performance. Note that the consequences of persistent or long enough faults can be very diverse, depending on the nature of the controlled process. For instance, the Chernobyl disaster is a well-known consequence resulting from a fault in a chemical reactor [Patton et al., 2013], endangering not only human life but also the environment. This example supports the need for FDD systems, which are an important part of fault-tolerant control systems [Noura et al., 2009].

As another example, consider a refrigerator controlling the temperature T of a vaccine batch, whose behavior is shown in Figure 1.2. If the vaccines need to be stored at a temperature no higher than 0° Celsius, one has an unpermitted deviation in the temperature, until around 240 minutes, where the control system is able to maintain the temperature at acceptable levels. In this context, if the faulty condition lasts long enough, vaccines may expire, leading to potential economic loss.

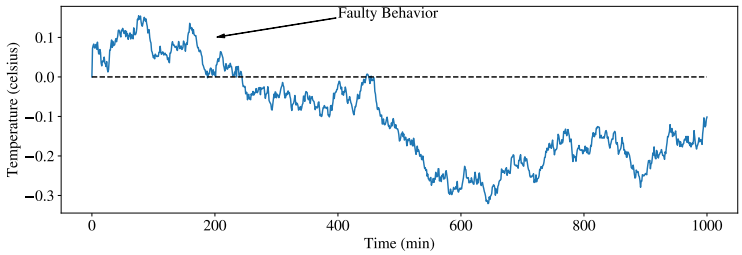


Figure 1.2: Faulty behavior of a refrigerator.

Formally, a FDD system can be understood as a supervision loop of a control system. Provided that the system's normal behavior has been defined, and consequently the faults that may occur, a human expert (e.g. an engineer) or a computer system can be used for detecting whenever a fault occurs. In the latter case, the FDD is also called an automatic FDD system, which is of particular interest for this thesis.

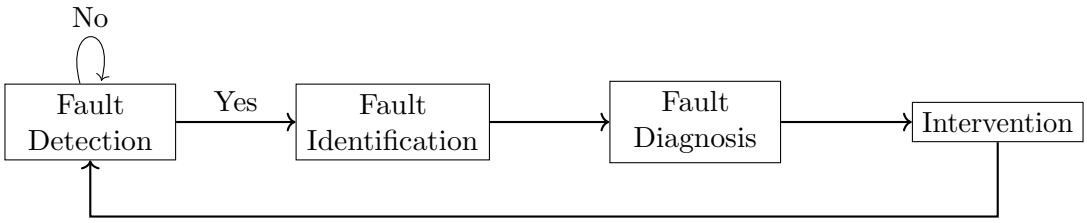


Figure 1.3: Process monitoring loop as presented in [Chiang et al., 2000].

In Figure 1.3, the fault supervision loop is presented. Following [Chiang et al., 2000], it is composed by 4 steps,

1. **Detection:** is concerned with determining if, at a given time, a fault has occurred.

2. **Identification:** is concerned with determining the variables or sub-systems more relevant for diagnosing the fault.

3. **Diagnosis:** is concerned with determining which fault, among a set of possible malfunctions, has occurred.

4. **Intervention:** is concerned with removing the fault's effect.

Even though the various steps in a FDD system are, by no means, trivial tasks, in this thesis we focus on the automation of the diagnosis task, assuming the faults have been previously detected. In addition, note that the diagnosis step is somehow equivalent to classification, as one may identify each fault type with an individual class. With this hypothesis, the design of an automated fault diagnosis algorithm is shown in Figure 1.4 as a sequential model.

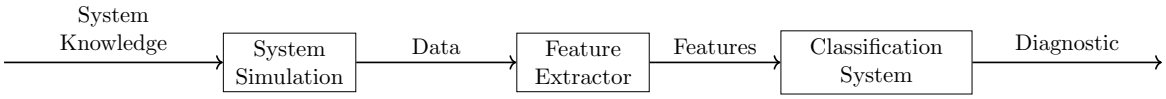


Figure 1.4: Sequential pipeline for automatic fault diagnosis proposed in this thesis. Note that this pipeline relies on a simulation model for the process for building the classification system, rather than acquiring data from sensors.

In the following discussion we detail each step of Figure 1.4. The starting point is knowledge about the system. This corresponds to the result of mathematical modeling, either through input-output or state-space representation. Such model is used by general purpose programming language or simulation environments, such as Python or Matlab's Simulink toolbox, respectively. The result from simulation is data in form of time-series, as shown in Figure 1.5.

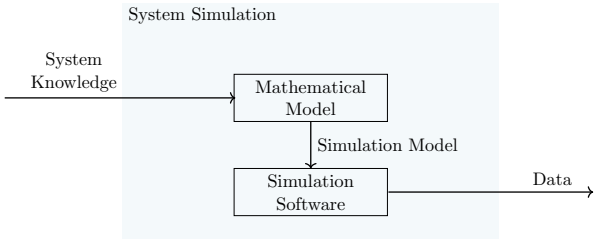


Figure 1.5: An overview of the simulation step in the diagnosis pipeline.

Note that alternatively to employing software for acquiring data, one may also use sensors on physical plants. This will be detailed later in the text. For now, let us suppose that the data was acquired through simulation. One can directly employ classification algorithms to it, or apply a feature extraction step. The application of this step is entirely optional, but often leads to higher performance. The internal logic of a feature extraction is shown in Figure 1.6.

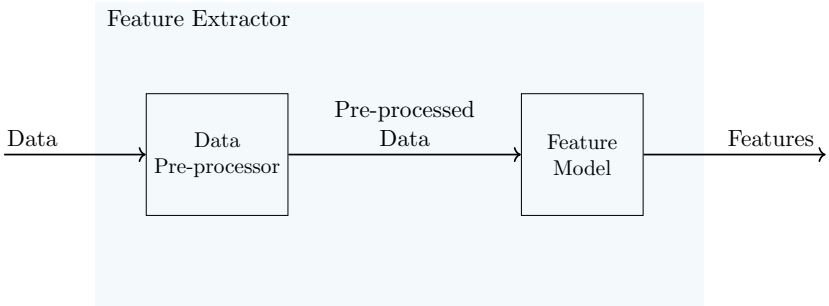


Figure 1.6: Diagram showing the flow of information in a feature extractor.

Note that building a feature model is a important part of the extraction pipeline. This is also known in the literature as feature engineering, and can either be done by a trial and error procedure, or automatically. In general, trial and error involves knowledge about the problem at hand. For instance, Gabor filtering is known to work well in image recognition tasks [Lyons et al., 1998]. In the context of fault diagnosis, other types of features have been employed, such as wavelets [Yan et al., 2014], sample statistics [Jegadeeshwaran and Sugumaran, 2015], and frequency domain statistics [Zhang et al., 2015].

The last step in Figure 1.4 is the classification algorithm. It takes features as inputs, and learns a rule for determining the type of fault. As noted by [Zheng et al., 2019], various classification algorithms have been proposed for fault diagnosis, such as Support Vector Machines (SVMs) [Zhang et al., 2015], random forests [Wang et al., 2017] and k-Nearest Neighbors [Su et al., 2014]. With the recent attention that Deep Neural Networks (DNNs) have attracted, Convolutional Neural Networks (CNNs) have also been applied [Jiang et al., 2017]. In this work we particularly focus on two algorithms, namely SVM and CNN, as they constitute the state-of-the-art in classification.

Regardless of the classifier choice, a major limitation of the conventional machine learning framework is that they suppose training and testing data coming from the same probability distribution [Ben-David et al., 2010]. In general, the study of learning problems in the previous context has been called domain adaptation [Ben-David et al., 2007]. If we further assume that the phenomenon happens in a FDD, it has been called cross-domain fault diagnosis [Zheng et al., 2019]. In practice, there are many reasons for why this assumption does not hold. Here, we present two examples:

- **Simulation vs. Real Systems:** Due modeling errors, or inherent limitations on the modeling heuristic, the mathematical model for a process may not reflect the real process for which one wants to diagnose faults. As consequence, there is a shift between the probability distribution from which simulation data is sampled, and the one from which sensor data is sampled.

- **Fault diagnosis on parameter settings:** In this case, we suppose that the mathematical model available for the process remains the same between source and target domains, but their parameters change. This roughly translates to different machines (e.g. refrigerators from different brands). Even though the mathematical rules for the system dynamic behavior are the same, one can still have different behaviors due to parametric changes. As consequence, the distributional shift scenario also occurs.

We highlight the importance of each of the aforementioned problems. The first is of particular interest, since the training of our automatic FDD system rely largely on simulation, as acquiring real data from processes involves driving it to dangerous conditions [Li et al., 2020]. Moreover, the second problem can have interesting implications in terms of data usage, as one may simply train a system for a given machine, then adapt to new using as few data as possible.

Figure 1.7 shows the consequence of having data samples from two distinct probability distributions. In particular, sensor data (in blue), acquired from the real process, differs in distribution from simulation data (in red). Notice that this may harm the performance of classifiers trained with source data only, in the desired/target domain. In addition, the problem of fault diagnosis on different machines is analogous in terms of flow of information, as one may substitute the process plant by machine 1, and system simulation by machine 2.

The aforementioned problem fits into the definition of domain adaptation, which is a sub-field of transfer learning [Pan and Yang, 2009]. In particular, it is known to appear in many areas of application, such as image and natural language processing. Especially, it happens whenever the conditions upon which data is generated changes. In practice, for the domain adaptation setting one is supposed to have labeled samples from the so-called *source domain*, and unlabeled samples in the *target domain*.

For solving this type of domain, various approaches and techniques exists. Specially, [Pan and Yang, 2009] has classified those in two categories: feature-based and instance-based transfer learning. The former category tries to find a sub-set, or a transformation of features such that a divergence between the domains probability distributions is minimized. Indeed, this is the approach taken in [Pan et al., 2010] and [Gong et al., 2012]. The latter category, on the other hand, assigns weights to the instances in the source domain, so that the source weighted

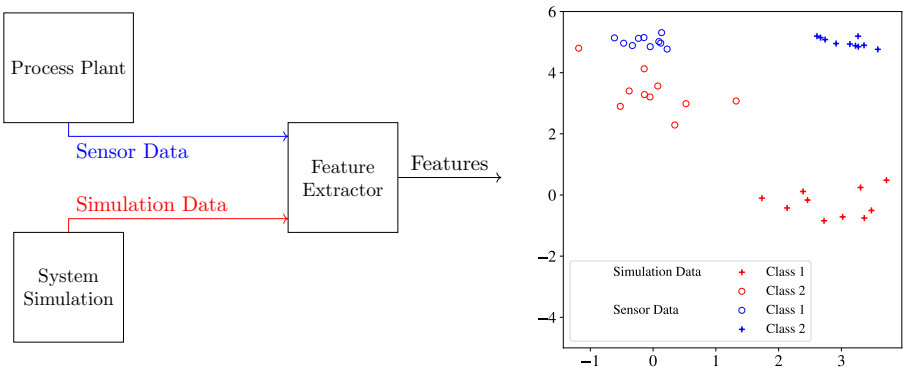


Figure 1.7: Exemplification of distributional shift occurring in automatic fault diagnosis systems. In this case, sensor data differs in distribution from simulation data.

distribution resembles the target distribution. This approach is taken, for instance, by [Gretton et al., 2009], [Sugiyama et al., 2008] and [Kanamori et al., 2009].

Despite the relevance and success of the aforementioned techniques, in the seminal paper of [Courty et al., 2016] another strategy was applied for solving domain adaptation. It consisted on using the Wasserstein distance between probability distributions to devise a mapping T, transforming source data into target data. This framework, known as Optimal Transport for Domain Adaptation (OTDA), obtained state-of-the-art performance in many domain adaptation benchmarks. Since then, the framework has been adapted for kernelized transformations [Perrot et al., 2016] and for handling more general distributional shift scenarios [Courty et al., 2017].

In addition to the practical success of domain adaptation algorithms, the theoretical background for these has been an active field of research. For instance, the theoretical description of the distributional shift problem has first been established in [Kifer et al., 2004]. Then, the problem of generalizing for different domains was formalized in [Ben-David et al., 2007], where the authors further proposed a new divergence, proving that the miss-classification rate in the target domain is bounded by miss-classification rate in source domain, plus the proposed divergence between each domain feature's probability distribution. These results were then summarized in [Ben-David et al., 2010], where the authors have set the theoretical framework for analyzing domain adaptation problems. In addition [David et al., 2010] brings a discussion on the necessary and sufficient conditions for domain adaptation.

In the context of this previous discussion, the theoretical reason for proposing a new domain adaptation algorithm revolves around three key points. First, one needs to set a distribution divergence, and prove a result similar to the results of [Ben-David et al., 2007]. Second, one needs to prove that the established divergence can be accurately estimated from finite samples. Finally, one needs to prove that the algorithm is indeed minimizing the defined divergence. In the context of the OTDA framework, this was first proven by [Redko et al., 2017].

Finally, taking into account the distributional shift that may occur in the data generation process, the pipeline presented in Figure 1.4 is substituted, including the different machines or simulator and physical plant. The new pipeline is shown in Figure 1.8.

1.2 Motivation

This work seeks to present an optimal transport-based perspective on cross-domain fault diagnosis. Moreover, the main idea behind our work is promoting diagnosis performance by leveraging knowledge from different but related machines or simulation models.

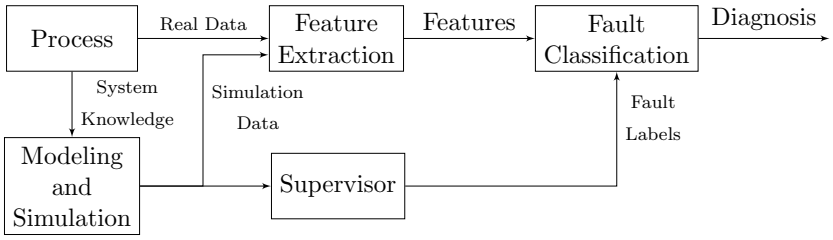


Figure 1.8: Cross-Domain fault diagnosis pipeline. In this pipeline the supervisor plays the role as the ground-truth labeling function f, giving the true label of each simulation sample. In addition, it corresponds to the unsupervised transfer learning case, since the target domain data (named real data) has no labels.

The aforementioned motivation was first presented by [Zheng et al., 2019], and is very similar to that explored by [Li et al., 2020]. Hence, this work focuses on simulation data under different parametric conditions. Therefore, the source and target data come from realizations of a dynamical systems with different parametric settings. In both approaches, the source domains are assumed to be labeled and the target domain is assumed unlabeled during learning. For purposes of performance assessment, target domain labels are available.

1.3 Objective and Contribution

This thesis proposes a discussion on the foundations of cross-domain fault diagnosis system. Especially, each step in the pipeline presented by Figure 1.4 is presented in the next chapters. In addition, we discuss the foundations of transfer learning and domain adaptation, with a special focus on optimal transport. This latter theory will serve as a support for the analysis of domain adaptation problems, as well as the bedrock of strategies for solving it. Moreover, we aim at presenting a comparative study between different strategies of transfer learning, covering instance, feature and optimal transport-based algorithms. Thus, this work presents four contributions,

C1 We present a comprehensive comparative study covering feature extraction choice and state-of-the-art domain adaptation methods.

C2 We present an experimental analysis correlating the degree upon which simulation environment changes, and the consequent change in the data probability distribution.

C3 We present an experimental analysis correlating the degree of mass flow between source and target samples having different classes, and the success of OTDA techniques.

C4 The entire code for our experiments and examples is published as an open-source repository hosted on Github[1]

We highlight that our contributions are of both practical and theoretical interest. In particular, C1 shows that Optimal Transport (OT)-based transfer learning has state-of-the-art performance, surpassing all other tested algorithms on average. C2 justifies experimentally the divergence-based strategies, which are popular in the domain adaptation literature. Moreover, C3 provides further evidence that class-based regularization, as proposed by [Courty et al., 2016], is important for adaptation. Finally, C4 ensures the reproducibility of our results.

[1] https://github.com/eddardd/CrossDomainFaultDetection

1.4 Thesis Structure

The workflow of this thesis is divided into two categories. First, each stage at the automatic fault diagnosis pipeline shown by Figure 1.4 is detailed. A toy example is used for contextualizing the concept and a more complete system is used to investigate the proposed methodology.

- Chapter 2 discusses the mathematical foundations of dynamical systems and fault diagnosis,
- Chapter 3 discusses the methods employed for extracting features from time-series data. Particularly, we employed features extracted from the autocorrelation of the studied signals. The chapter briefly introduces the theoretical framework behind classification, and it explains the classification algorithms used in our experiments.
- Chapter 4 introduces the field of transfer learning, and optimal transport. For transfer learning, we aim at defining the foundations, and answering the questions how and when to apply transfer learning. For optimal transport, we present a computational formulation for the theory, and contextualize it as a tool for analysis, and for performing transfer learning.
- Chapter 5 presents the case study for this work, namely, the continuously stirred tank reactor. It also presents our experimental results and their discussion.
- Chapter 6 reinforces our findings, as well as sets the directions for future works.

Thus, following the logical dependency of the aforementioned chapters, we propose the reading order presented in Figure 1.9. This is merely a suggestion, as the reading can also be done sequentially.

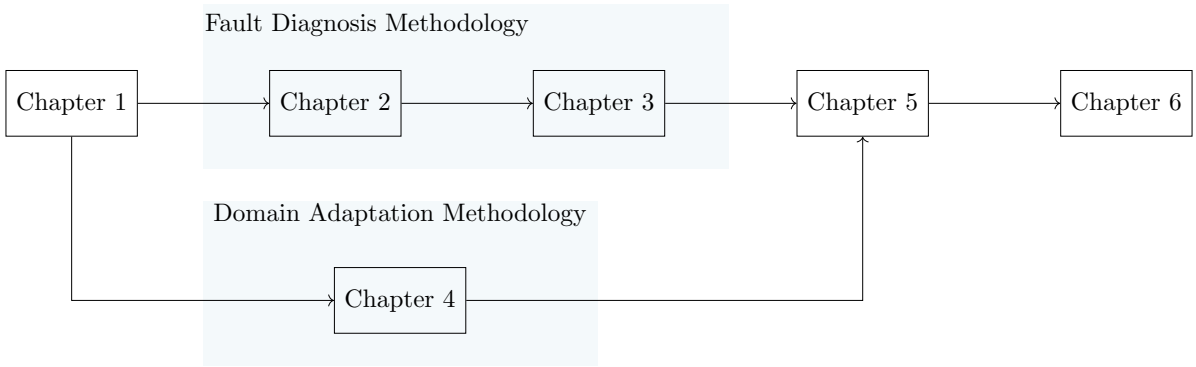


Figure 1.9: Suggested order for reading the chapters.

Chapter 2

Dynamical Systems Modeling

This chapter seeks to establish the mathematical tools for the analysis of physical processes. These tools are essential for having a deeper insight on the data used for the training of FDD systems. Thus, its purpose is twofold:

1. On one hand, with a formal mathematical analysis we may establish how faulty data is generated (Chapter 5), and have an intuition on which features are relevant for classification (Chapter 3).

2. On the other hand, it is fundamental for understanding the FDD problematic, as we introduce the concept of control system and fault diagnosis.

This chapter is organized as follows: Section 2.1 introduces the methodology for analyzing dynamical systems. Section 2.2 presents the background for linearization and stability analysis. Finally, Section 2.3 introduces the concept of control systems and fault diagnosis.

2.1 Introduction

Following [Chen, 1984], the analysis of physical processes can be conduced using empirical, or analytical methods. On one hand, when employing empirical methods data is acquired by applying various input signals to the system, and measuring its output. An example of empirical method is system identification, which is briefly outlined in Appendix B. On the other hand analytical methods use mathematical analysis for building a set of equations describing the system. These equations commonly reflect the dynamical properties of a system by means of ODEs.

In this Section, we introduce an analytical method for representing dynamical systems, the state-space representation. To illustrate this method, we use a toy example consisting on a physical process of two connected tanks as shown in Figure 2.1. The tank 1 is progressively filled with water, from an inlet flow-rate q_{in} in m^3/s. The water in tank 1 flows to the environment and tank 2 through two orifices, thus, the tank 2 is also progressively filled. In addition, it also has an orifice so that water may flow from it, to the environment.

Regardless of the method used for analysis, the role of mathematical modeling is using knowledge about a real-world process to find the desired representation analytically. In this regard, many heuristic exists. For instance, when modeling mechanical systems one may employ Newton's law, while for modeling electric systems one may apply Kirchoff's laws [Chen, 1984].

This thesis has a focus on chemical plants, which rely heavily on conservation laws. This strategy of analysis is based on the assumption that reflect the physical world, but that do not follow from other propositions. Examples are: conservation of mass and energy. Assuming that a physical quantity is conserved is equivalent to say that, as time passes, no amount of such

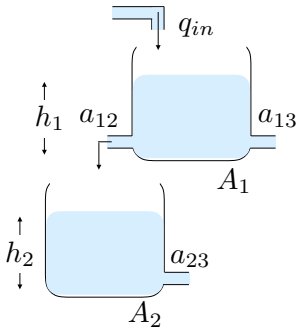


Figure 2.1: Two-Tank system, used as a toy example.

quantity is lost nor created. Therefore, the following holds:

$$\text{Accumulation} = \text{Input} - \text{Output} + \text{Generation} - \text{Consuption}. \tag{2.1}$$

where the words are related to the same physical quantity (e.g. energy). Note that while input and output refers to the exchange between the system and the environment, generation and consumption are consequence of the system's internal mechanics. To illustrate the application of first principle analysis, and of Equation 2.1 as well, we provide a toy example that will be used throughout Chapters 2 to 3. We begin by defining the notion of State-Space representation in Definition 2.1.

Definition 2.1. (State-Space Representation) Let $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{y}(t) \in \mathbb{R}^q$ be vector functions, and $\mathbf{f} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{g} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^q$ be given vector fields. A system is said to be in state-space representation if the following holds,

$$\begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)). \end{cases} \tag{2.2}$$

$\mathbf{u}(t)$, $\mathbf{x}$, and $\mathbf{y}$ are called respectively the system's input, state and output vector.

In addition to the previous definition, we may express Equation 2.2 graphically, as in Figure 2.2. This figure shows the flow of information for transforming the inputs $\mathbf{u}(t)$ into the output $\mathbf{y}(t)$. The state-vector $\mathbf{x}(t)$ is an internal variable to the system.

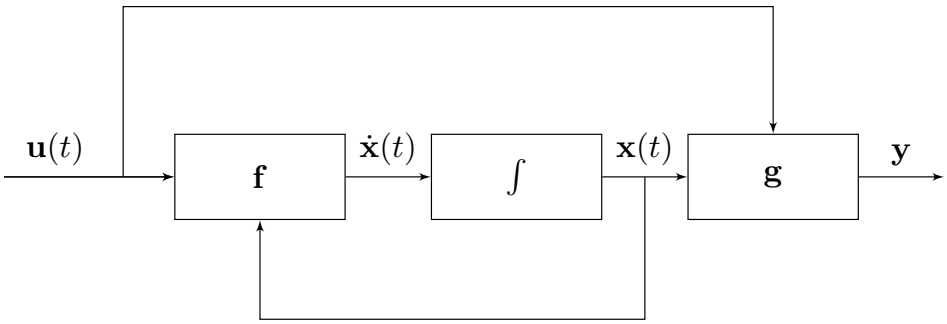


Figure 2.2: General State Space representation diagram.

In order to build a state-space representation, one needs to use an heuristic, such as first principles analysis mentioned earlier. To further exemplify this method of analysis, we apply it to the two-tank system described at the beginning of this section.

Example 2.1. (State-Space representation of Two-Tank System) Consider the Two-Tank system as shown in Figure 2.1. We consider as output the second tank height $y(t) = h_2(t)$, and the inlet flow-rate $q_{in}(t)$ as the input $u(t)$. Furthermore, we set $\mathbf{x}(t) = [h_1(t), h_2(t)]^T$ as the process state vector. Furthermore, Table 2.1 presents a summary of the variables and parameters of the system.

In addition, some constraints are assumed for the input q_{in}, and the state vector, as $q_{in}(t) \leq 1$ m^2/s, $h_1 \leq 1.5$ m, and $h_2 \leq 1.5$ m. The fluid is considered perfect (no sheer stresses, no viscosity, no heat conduction), and subject only to gravity. The tanks are filled with water (incompressible fluid), and the external pressure is constant (atmospheric pressure).

Type	Symbol	Description	Units	Value
Input	q_{in}	Water volumetric flow-rate	m^3/s	-
State Variable	h_1	Tank 1 water level	m	-
	h_2	Tank 2 water level	m	-
Output	y	Tank 2 water level	m	-
Parameter	ρ	Water density	kg/m^3	1000
	A_1	Tank 1 cross-sectional area	m^2	1.0
	A_2	Tank 2 cross-sectional area	m^2	1.0
	a_{12}	Orifice area between tanks 1 and 2	m^2	0.1
	a_{13}	Orifice area between tank 1 and the environment	m^2	0.1
	a_{23}	Orifice area between tank 2 and the environment	m^2	0.1

Table 2.1: Description of parameters, inputs and variables in the Two-Tank system.

In these conditions, considering Torricelli's law, $q_{ij} = a_{ij}\sqrt{2g\ h_i(t)}$, where $g = 9.807$ m/s^2, we may start our first principle analysis under the assumption of volume conservation. Especially, we identify the following,

$$\begin{aligned}\text{Mass accumulation in Tank 1} &= \frac{dm_1}{dt} = \rho A_1 \frac{dh_1}{dt},\\ \text{Mass input in Tank 1} &= \rho q_{in},\\ \text{Mass output in Tank 1} &= \rho q_{12} + \rho q_{13} = \rho a_{12}\sqrt{2gh_1} + \rho a_{13}\sqrt{2gh_1},\end{aligned}$$

the calculations for Tank 2 are analogous. Thus, these information may be plugged into Equation 2.1, generating an ordinary differential equation for each tank,

$$\rho A_1 \frac{dh_1}{dt} = \rho q_{in}(t) - \rho(q_{12}(t) + q_{13}(t)), \tag{2.3}$$

$$\rho A_2 \frac{dh_2}{dt} = \rho q_{12}(t) - \rho q_{23}(t). \tag{2.4}$$

Therefore, one has the following state-space representation,

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \underbrace{\begin{bmatrix}0 & \dfrac{\sqrt{2g}(a_{12}+a_{13})}{A_1}\\ \dfrac{a_{12}\sqrt{2g}}{A_2} & \dfrac{a_{23}\sqrt{2g}}{A_2}\end{bmatrix}}_{\mathbf{\Phi}}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \frac{1}{A_1}u(t), \tag{2.5}$$

that is, $\mathbf{f}(\mathbf{x}(t), u(t)) = \mathbf{\Phi}\sqrt{\mathbf{x}}(t) + (1/A_1)u(t)$. Finally, the observation model is given by $y(t) = h_2(t) = x_2(t)$. Therefore,

$$y(t) = \underbrace{\begin{bmatrix}0 & 1\end{bmatrix}}_{\mathbf{C}}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}. \tag{2.6}$$

hence, $\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{C}^T\mathbf{x}$.

In general, it is difficult to devise a closed solution that works for all pairs $\mathbf{f}$ and $\mathbf{g}$ in Definition 2.1. Even though there exists a closed-form solution for the linear case [Chen, 1984], it is common to have a system of equations that has no analytical solution. Hopefully, nonlinear state-space systems may be solved numerically, as we discuss in Appendix C. Below, we simulate the Two-Tank system for different scenarios.

Example 2.2. (Simulation of Two-Tank System) We consider two situations for the flow-rate q_{in},

1. The tanks are previously full, which implies that the initial conditions are $h_1(0) = h_2(0) = 1.5$ m. We thus want to empty the tanks, by letting $q_{in}(t) = 0$ m^3/s.
2. The tanks are empty, which implies that the initial conditions are $h_1(0) = h_2(0) = 0.0$ m. Over time, we want to fill both tanks, which implies that $q_{in}(t) = 1.0\ m^3$.

The simulation for each case is shown in Figure 2.3.

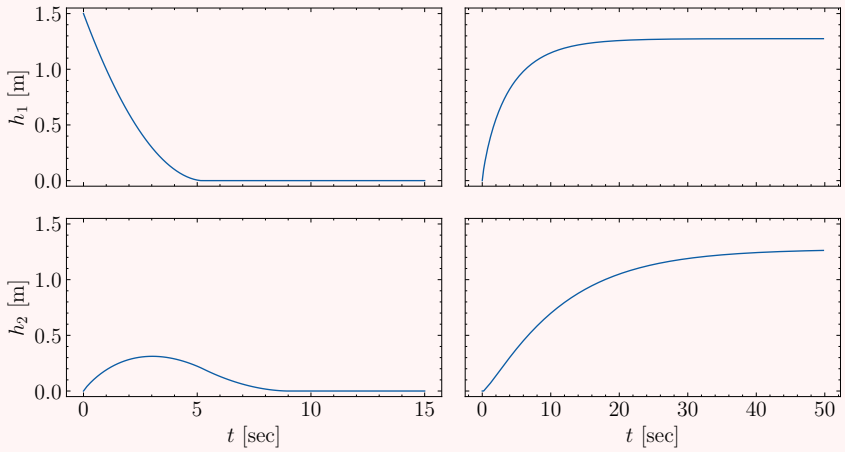


Figure 2.3: Simulation of Two-Tank system for different values of q_{in}. On the left, one has the simulation for the emptying case. On the right, one simulates the system for the filling case.

2.2 Linear Analysis and Stability

In this section we analyze how general state-space and input-output representations can be turned into linear ones, through a process known as linearization [Chen, 1984]. From the established linear representation, we then develop the concept of stability in a dynamic system. The starting point of our analysis is the concept of equilibrium points,

Definition 2.2. (Equilibrium Points) Given a dynamic system whose state-space representation is given by $\mathbf{f}$ and $\mathbf{g}$, the point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is said to be an equilibrium point of $\mathbf{f}$ (resp. $\mathbf{g}$) if, and only if $\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \mathbf{0}$.

As follows, we may develop a linear representation for a state-space represented using nonlinear equations $\mathbf{f}$ and $\mathbf{g}$ through Taylor series expansion. This is shown in the next theorem,

Theorem 2.1. *(Linear Representation of Nonlinear State-Spaces) Let a state-space be given by functions* $\mathbf{f}$ *and* $\mathbf{g}$*, with equilibrium points* $(\mathbf{x}, \mathbf{u})$*. Let* $\Delta\mathbf{x}(t) = \mathbf{x}(t) - \bar{\mathbf{x}}$ *(resp.* $\mathbf{u}$ *and* $\mathbf{y}$*) be the difference between* $\mathbf{x}(t)$ *and the equilibrium state. In a small vicinity of* $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$*, the dynamics of the perturbations are approximated by the linear state-space representation,*

$$\begin{cases} \Delta\dot{\mathbf{x}}(t) &= \mathbf{A}\Delta\mathbf{x}(t) + \mathbf{B}\Delta\mathbf{u}(t), \\ \Delta\mathbf{y}(t) &= \mathbf{C}\Delta\mathbf{x}(t) + \mathbf{D}\Delta\mathbf{u}(t) \end{cases} \tag{2.7}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$*, and* $\mathbf{D}$ *are the Jacobian matrices with the derivatives of* $\mathbf{f}$ *and* $\mathbf{g}$ *with respect to* $\mathbf{x}$ *and* $\mathbf{u}$*.*

Proof. Available in Section D.1. □

The last theorem describes the linearization of state-space representation, as well as it defines how one may calculate such linearization. In the following example we show the linearization procedure in the context of the two-tanks state-space representation,

Example 2.3. (Two-Tank State-Space Linearization) We begin the linearization of system 2.5 by finding the equilibrium points of $\mathbf{f}$, for $u(t) = q_{in}(t) = 1\ \mathrm{m}^3/\mathrm{s}$. The steady-state is found through the following system of equations,

$$\begin{cases} \dfrac{u}{A_1} - \dfrac{\sqrt{2g}(a_{12} + a_{13})}{A_1}\sqrt{x_1} &= 0 \\ \dfrac{a_{12}\sqrt{2g}}{A_2}\sqrt{x_1} - \dfrac{a_{23}\sqrt{2g}}{A_2}\sqrt{x_2} &= 0 \end{cases},$$

by manipulating the previous system, we may get x_1 and x_2, as,

$$\begin{aligned} x_1 &= \left(\frac{u}{\sqrt{2g}(a_{12} + a_{13})}\right)^2, \\ x_2 &= \left(\frac{a_{12}}{a_{23}}\right)^2 x_1. \end{aligned}$$

Using the values presented in Example 2.1, one has as equilibrium points $h_1 = h_2 = 1.274$ m, which are the values for which the differential equation solution converges to. Once we defined $\bar{\mathbf{x}} = [1.274, 1.274]$, we may linearize the system by calculating the Jacobian matrix $A_{ij} = \frac{\partial f_j}{\partial x_i}$:

$$\mathbf{A} = \begin{bmatrix} -\dfrac{\sqrt{2g}(a_{12} + a_{13})}{2A_1\sqrt{\bar{x}_1}} & 0 \\ \dfrac{a_{12}\sqrt{2g}}{2A_2\sqrt{\bar{x}_1}} & -\dfrac{a_{23}\sqrt{2g}}{2A_2\sqrt{\bar{x}_2}} \end{bmatrix} = \begin{bmatrix} -0.392 & 0 \\ 0.196 & -0.196 \end{bmatrix}$$

The linearized version of the system 2.5 is,

$$\Delta\dot{\mathbf{x}} = \begin{bmatrix} -0.392 & 0 \\ 0.196 & -0.196 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} + \tilde{u}, \tag{2.8}$$

$$\Delta y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}. \tag{2.9}$$

Note that, due the approximate nature of the linearization, the linear response $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \tilde{\mathbf{x}}$ may not match $\mathbf{x}$ precisely. To assess the gap between the nonlinear system response $\mathbf{x}$, and the linear one $\hat{\mathbf{x}}$, we use the Root Mean Squared Error (RMSE) metric, defined as,

$$\text{RMSE}(\mathbf{x}, \hat{\mathbf{x}}) = \sqrt{\frac{1}{t_2 - t_1} \int_{t_1}^{t_2} (\mathbf{x} - \hat{\mathbf{x}})^2 dt}. \tag{2.10}$$

In Figure 2.4 shows a visual comparison between the linear and nonlinear simulation. Although the linear response does approximate the nonlinear one, there is some error between the two. Particularly, for h_1 the RMSE between the two responses is 0.207, and for h_2 it is of 0.522.

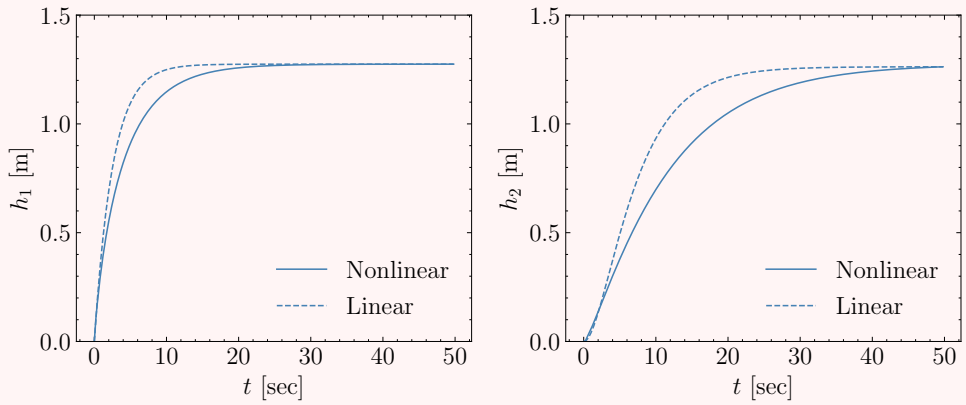


Figure 2.4: Comparison between linear and nonlinear responses. The dashed lines correspond to the linear simulation of the Two-Tank system.

In addition to the linearization, we can further classify equilibrium points according to their properties;

Definition 2.3. (Equilibrium Point Stability) Let $\mathbf{f}$ and $\mathbf{g}$ define a state-space representation, and let $\mathbf{x}(t)$ be its solution. An equilibrium point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is said to stable if for every $\delta > 0$, there exists an ϵ such that,

$$d(\mathbf{x}(0), \bar{\mathbf{x}}) < \delta \rightarrow d(\mathbf{x}(t), \bar{\mathbf{x}}) < \epsilon,$$

that is, solutions that start in a δ vicinity of $\bar{\mathbf{x}}$ stay $\epsilon-$close as the time evolves.

Thus, to study a non-linear system qualitatively, one can gain substantial insight through the analysis of stability of a given equilibrium point. For instance, if $\bar{\mathbf{x}}$ is a stable equilibrium point we expect the solutions $\mathbf{x}(t)$ of the dynamic system to remain near this state. In the context of real physical systems (e.g. the two-tank system), we can somehow understand the system's behavior without solving it. Thus, the equilibrium points can be classified as stable or not through the linearization of Equation 2.2. The criteria for stability, in this case, is that all eigenvalues of the Jacobian matrix need to have negative real part [Chen, 1984]. Thus, we highlight the practical approach for inspecting if $\bar{\mathbf{x}}$ is a stable equilibrium point,

1. Calculate the Jacobian matrix $\mathbf{A}$ for $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$,

2. Calculate the spectrum of $\mathbf{A}$, defined by its n eigenvalues $\lambda_1, \cdots, \lambda_n$,

3. If the real part $\text{Re}(\lambda_i) < 0$, $\forall i$, then $\bar{\mathbf{x}}$ is stable. It is said to be unstable otherwise.

In addition common practice is to calculate the spectrum of $\mathbf{A}$ and plot their values on the complex plane. The next example shows this line of analysis for the two-tanks system.

Example 2.4. (Analysis of Stability for the Two-Tanks Equilibrium Points) In the previous examples we have found $h_1 = h_2 = 1.274$ m and $q_{in} = 1$ m^3/s to be an equilibrium point of the two-tanks state-space representation. In addition, note that by Figure 2.4 we expect this equilibrium point to be stable. Using Equation 2.9, we may calculate the spectrum of $\mathbf{A}$, which is $\lambda_1 = -0.196$, $\lambda_2 = -0.392$. Since the eigenvalues have negative real part, the system is indeed stable at that point. In Figure 2.5 we show the these eigenvalues in the complex plane.

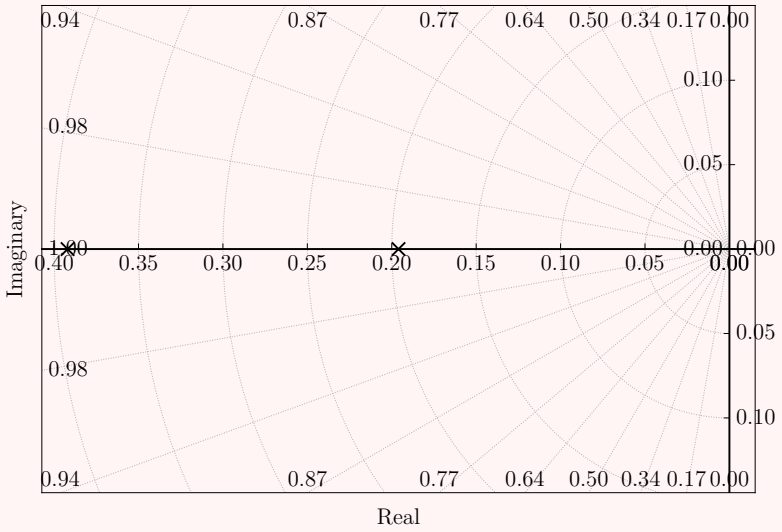


Figure 2.5: Eigenvalues of the Jacobian matrix $\mathbf{A}$ shown on the complex plane. Note that since λ_i have negative real part, the system is stable.

2.3 Fault Diagnosis on Control Systems

Control Systems

In this section we briefly introduce how dynamical systems can be automatically controlled. For simplicity, we will assume that the dynamical systems of interest have one input, and one output. Although this is a restrictive assumption, more general descriptions are out of the scope of this thesis. Using the state-space representation of a dynamical system, three functions are important for us: the input $u(t)$, the state-vector $\mathbf{x}(t)$ and the output $y(t)$. Below, we given a formal definition of a controller in terms of these inputs and outputs,

Definition 2.4. (Controller) Let $u(t)$ denote the input of a dynamical system, and $y(t)$ define its output. In addition, let $r(t)$ be a reference signal, denoting the desired output for the system at each time t. A controller is a function $C : \mathbb{R} \to \mathbb{R}$, such that for $e(t) = r(t) - y(t)$,

$$u(t) = C(e(t)).$$

The intuition behind this previous definition is that C infers the input $u(t)$ necessary to drive $y(t)$ towards $r(t)$ from the error signal $e(t) = r(t) - y_m(t)$. This reasoning is shown visually in the block diagram in Figure 2.6.

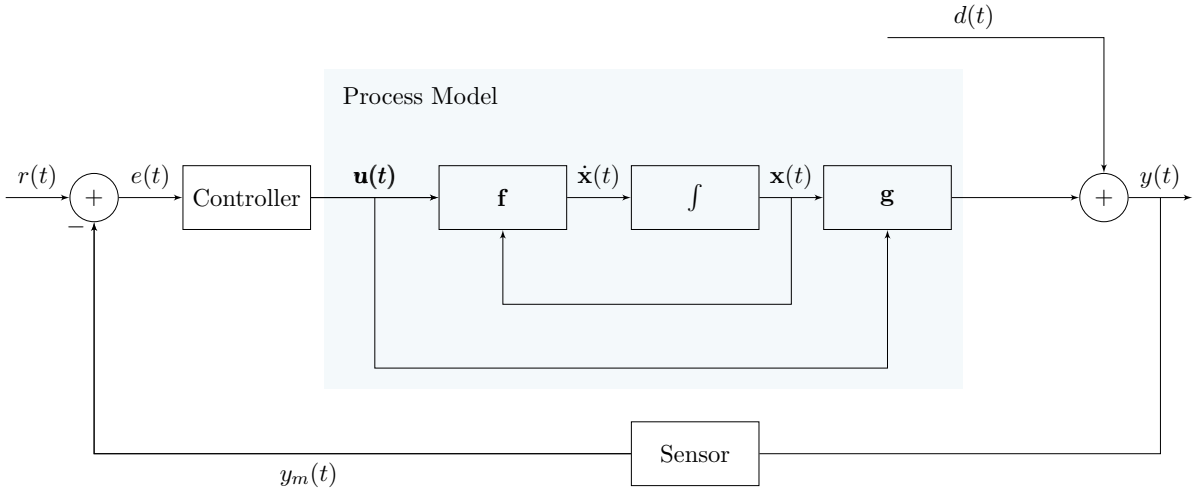


Figure 2.6: Feedback control loop for a dynamical system in state-space representation. As shown, the controller's role is estimating the necessary input for driving the system's output $y(t)$ to the reference $r(t)$. The system's output is measured through sensors, which results in $y_m(t)$, the measured output. This measurement is then compared to $r(t)$, resulting in a error signal $e(t)$.

Note that Definition 2.4 is rather general. Especially, there are various ways to devise a rule C for the inference of $u(t)$. One of the most commonly used in practice is the Proportional Integral Derivative (PID) controller, which is composed by three terms: a proportional, an integral and a derivative term, as follows,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}(t).$$

Due to its simplicity, the PID will be used as the controller of the control system's discussed throughout this thesis. Note that by adopting it, one needs to define K_p, K_i, and K_d. This is known as PID controller design. This design can either be done through a process of trial and error (known as fine tuning), or by using a predefined rule, such as the Ziegler-Nichols tuning rules [Ziegler et al., 1942]. In Appendix B we present a brief discussion of a strategy for tuning PID controllers, known as Direct Synthesis [Chen and Seborg, 2002]. In the next example we show the control system for the two-tanks process, using a PID controller.

Example 2.5. (PI Control of the Two-Tank process) We consider a PI controller, which corresponds to a PID with $K_d = 0$. Note that computationally, the derivative term is complicated because error's signal derivative is sensitive to noisy measurements. For the controller's specification, we will consider a constant reference signal $r(t) = 0.75$, for which we want $y(t) = h_2(t)$ to follow. Note that in our previous experiments, the state-vector $\mathbf{x}(t) = [h_1(t), h_2(t)]^T = [1.274, 1.274]^T$ for $q_{in}(t) = 1$ m$^3/s$, $\forall t$. Thus, in order to drive $y(t)$ towards the reference, a controller is needed.

As said previously, the controller's design is covered in Appendix B. More specifically, we cover in Example B.5 the design for the Two-Tank process. In this example we will assumed $K_p = 2.161$ and $K_i = 0.191$. In Figure 2.7 we show the simulation results.

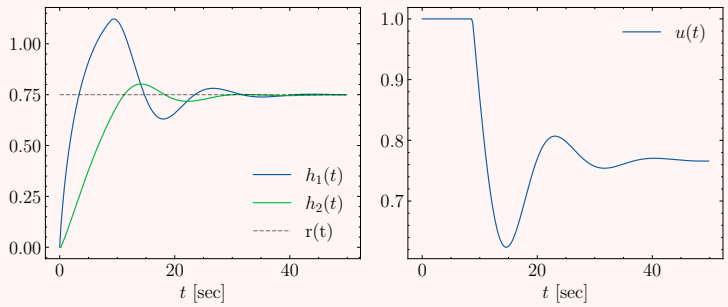


Figure 2.7: Two-Tank system controlled through a PI controller designed using the direct synthesis method.

Fault Diagnosis

In the previous sections, we have defined a theoretical methodology for analyzing control systems. Especially, we detailed two components presented in Figure 1.1, namely, the controller block, and the process block. The first of these components was discussed in the previous section, while the second was detailed throughout this chapter introduction. In this section we proceed in detailing a more realistic simulation scenario, as well as the introduction of faults in the simulation of dynamical systems.

As follows, consider a dynamical system modeled using state-space representation, that is, suppose that $\mathbf{f}$ and $\mathbf{g}$ specify the process being studied. So far, we have assumed that the mathematical model matches the process exactly. In practice, even though the real process is not stochastic, we introduce a *process noise* to reflect our uncertainty about our model [Särkkä, 2013]. In addition, a realistic assumption of the sensor block in Figures 1.1 and 2.6 is that it is subject to noise. This is called measurement noise. These two assumptions slightly change the state-space representation to,

$$\begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \nu, \\ y(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) + \eta, \end{cases} \tag{2.11}$$

for a random noises ν and η. In simulation, these noises are commonly sampled from Gaussian distributions with zero mean, but this assumption is not mandatory. As will be discussed throughout Chapter 3, a state-space model with process noise behaves randomly. The intensity of this noise (e.g. the noise variance) presents a trade-off between the original process dynamics (deterministic state-space) and a random-walk (purely random dynamics). In Figure 2.8 a comparison between behavior of the two-tanks system is shown for various intensities of process noise.

In addition, measurement noise affects the system's output, rather than the process dynamic. In real-world situations, processes are subject to both kinds of noises, thus in a simulation environment a common practice is to set σ_η and σ_ν so that samples are statistically diverse. Note that one should choose σ_ν with caution, since for high values the system behavior may not reflect the original state-space. The same warning is valid for σ_η, since for noises with high intensity the measurements of $y(t)$ may not reflect the actual process output.

The next step in considered a realistic FDD system is introducing faults in the control system. Following [Isermann, 2006], a fault $\mathbf{F}(t)$ may affect the system's inner parameters $\Theta(t)$ by $\Delta\Theta(t)$, or the state-vector $\mathbf{x}(t)$ by $\Delta\mathbf{x}(t)$. This is likely to influence the system's output $y(t)$ by $\Delta y(t)$. This discussion can be introduced in the control loop, as shown in the Figure 2.9. A direct consequence of Figure 2.9 is that the design presented throughout this section may not be

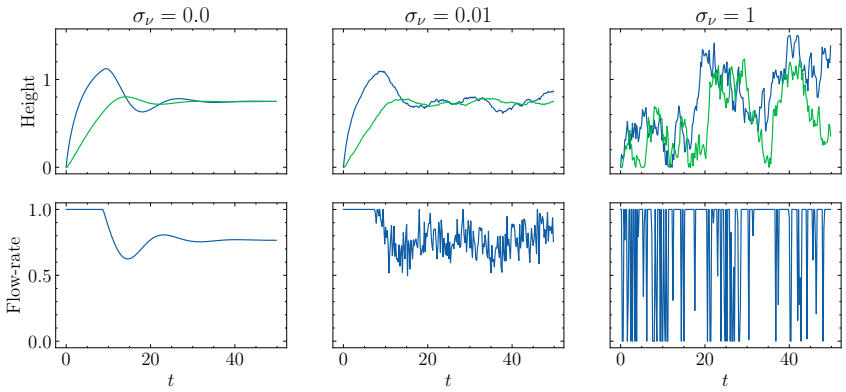


Figure 2.8: Simulation of the two-tanks system for various values of process noise variance σ_ν.

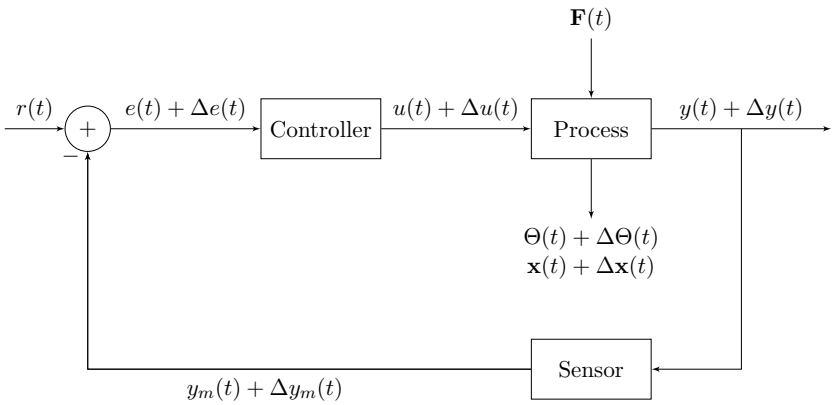


Figure 2.9: Figure adapted from [Isermann, 2006]. Diagram showing the effect of faults $\mathbf{F}(t)$ in a control system. Note that a fault in the process affects almost all variables in the control loop, except from the reference signal.

enough for creating a robust control loop. This issue is further illustrated in the next example, which shows the simulation of a faulty Two-Tank system.

Example 2.6. (Orifice Obstruction in the Two-Tank System) In this example we will consider the set of system parameters $\Delta\Theta(t) = [a_{12}(t), a_{13}, a_{23}, A_1, A_2]$, thus the only affected parameter is the area of the orifice connecting tanks 1 and 2. We will model a faulty scenario where the orifice is progressively obstructed, following the mathematical formula,

$$a_{12}(t) = a_{12}(0)e^{-0.03t},$$

this assumption allows us to express $\Delta\Theta(t)$ as,

$$\begin{aligned}\Delta\Theta(t) &= a_{12}(0) - a_{12}(0)e^{-0.03t},\\ &= a_{12}(0)(1 - e^{-0.03t}),\end{aligned}$$

that is, as time goes on, $a_{12} \to 0$, and $\Delta\Theta- \to a_{12}(0)$. This corresponds to the case where the connection between tanks 1 and 2 is completely obstructed. A comparison the system's output is shown in Figure 2.10, for the aforementioned fault scenario.

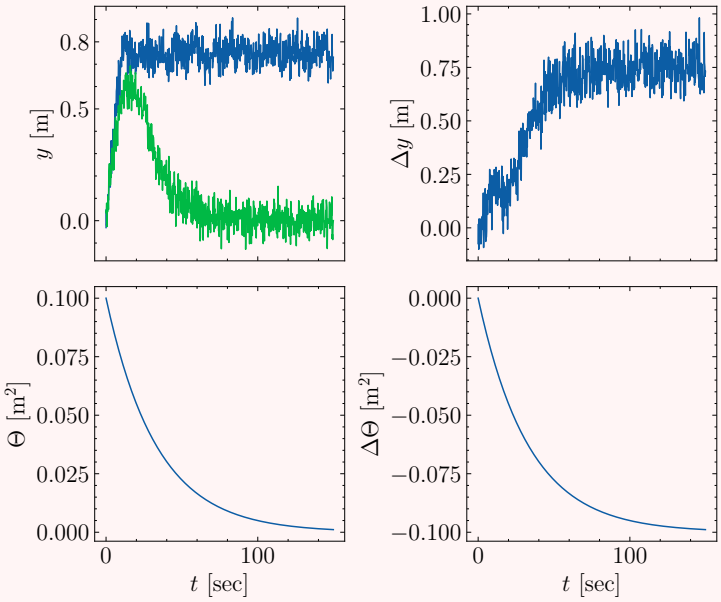


Figure 2.10: Comparison between the system's responses as $\Delta\Theta$ grows. On the upper left figure we show a comparison between the system's output for normal operation (blue) and a_{12} obstruction (green).

Example 2.6 highlights that when faults occur, we cannot expect the system to follow its expected behavior. A possible way to solve this issue is using the fault supervision loop shown in Figure 1.4. First, we need to detect the fault whenever it happens. This can be done in several ways. For instance, one may define a safe region for the process variable to be. In the two-tanks example one may consider that outside the range $y_{min} = 0.2$ m and $y_{max} = 1.2$ m, the system is faulty. In this case, $y(t) < y_{min}$ (resp. y_{max}) is considered to be dangerously low (resp. high). This is known as limit-checking [Isermann, 2006].

The identification step, as presented by [Chiang et al., 2000], resembles the feature extraction procedure that is discussed in Chapter 3, but is more general. In summary, it consists on identifying the system's parts relevant for the fault diagnosis step. For instance, in Example 2.6, the fault affects primarily the water level on tank 2. Even though it affects all components of the control system, its effects are more evident on $h_2(t)$. This information may guide the feature extraction process, which are then used in the diagnosis step.

Finally, the diagnosis task is concerned with classifying the data from the identification step in a set of possible faults. The classification step can be done through various algorithms, such as SVMs and CNNs. To further exemplify the data that will be used in the course of the next sections, we will consider two sets of faults for the Two-Tank system,

1. **Sensor Shift:** For this type of fault the sensors exhibit a trend, biasing the measurements. Let $\hat{y}$ be the sensor measurement. The sensor shift fault is modeled through the following mathematical equation,

$$\hat{y}(t) = y(t) + \delta t,$$

where δ is a parameter that dictates the trend intensity. For the two-tanks system, $\delta = \sqrt{2} \times 10^{-2}$.

2. **Orifice Obstruction:** For this type of fault, the orifices between the tanks, and between each tank and the environment are progressively obstructed. This fault affects the

parameters a_{12}, a_{13} and a_{23}, which are now dependent on time and change through the following formula,

$$a_{ij}(t) = a_{ij}(0) \times e^{-\delta t}$$

where, again, δ is a parameter corresponding to the fault intensity. In this case, $\delta = 0.03$.

Considering these two fault scenarios, Figure 2.11 shows a comparative on the measured signals for different faults, and for noises $\sigma_\nu = \sigma_\eta = 5 \times 10^{-2}$. Note that each fault is assumed to be detected previously. In addition, the sensor bias affects both $y(t)$ and $u(t)$, as the flow-rate is set by the controller. Furthermore the obstruction in the connection of tank 1 to the environment, and of tank 2 to the environment have similar effect on the system behavior as in both cases one needs less flow-rate to maintain the tank 2 water level. Finally, when a_{12} is obstructed, the water in tank 1 cannot flow to tank 2.

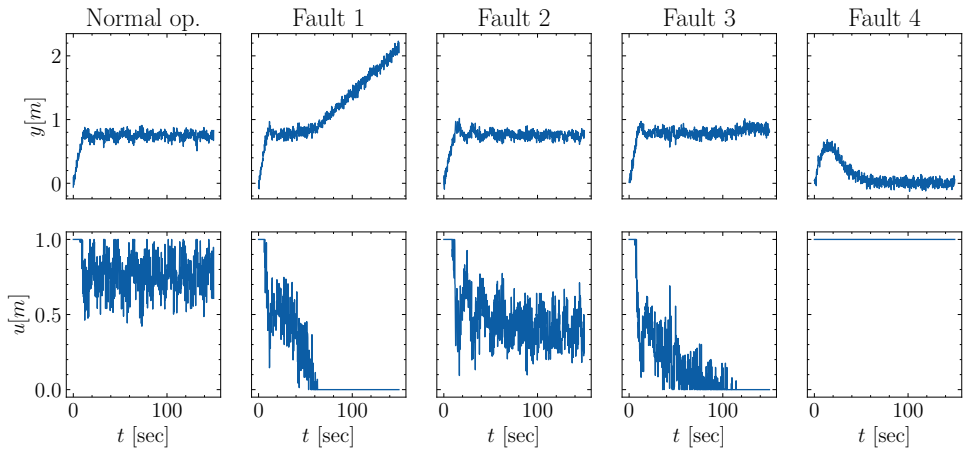


Figure 2.11: Simulation of the Two-Tank control system for different faults.

Chapter 3

Feature Extraction and Classification

This chapter is focused on describing two steps in the pipeline 1.4: feature extraction and classification. In Section 3.1, we present a signal-based feature extraction modeling based on the autocorrelation between samples in time. Moreover, in Section 3.2 we discuss two state-of-the-art algorithms used in this work: the SVM and CNN models.

3.1 Signal-Based Feature Extraction

In the following discussion, we adopt a discrete approach for analyzing time series. Thus, we assume that N samples are collected from a continuous signal $x(t)$. These values extracted from x are sampled at instants $t_i = ih$, for $i = 0, \cdots, T-1$ and time-step h. This generates a vector representation for x, corresponding to $\mathbf{x} = \{x(t_k)\}_{k=0}^{T-1}$. We say that these samples form a time-series, since they represent quantities obtained sequentially over time [Box et al., 2011]. Thus, we begin by the notion of a (discrete) stochastic process.

Definition 3.1. (Stochastic Process [Papoulis and Pillai, 2002]) A stochastic process $\mathbf{X}$ is a family of random variables $\{X_i : i = 1, \cdots, n \cdots\}$.

The theory of stochastic processes is particularly useful for the analysis of time-series since these may not always be deterministic. Indeed, this is often the case when working with empirical data. In this case as posed by [Box et al., 2011], a time series $x_0, x_1, \cdots, x_{T-1}$ of T successive observations is regarded as a sample realization of an infinite population of such time series, that could be generated by the stochastic process. Thus, we assume $x_k \sim X_k$.

Assuming N samples acquired from the time series $\mathbf{X}$, at each time-step i, the random variable X_i has its own mean and variance, given by:

$$\mu_i = \mathbb{E}[X_i] = \frac{1}{N}\sum_{i=1}^{N} x_{ki} \tag{3.1}$$

$$\sigma_i^2 = \mathbb{E}[(X_i - \mu_i)^2] = \frac{1}{N-1}\sum_{k=1}^{N}(x_{ki} - \mu_i)^2. \tag{3.2}$$

Notice that, additionally to evaluating the variance of X_i, we may also evaluate the covariance of random variables X_i and X_j. This gives the autocovariance matrix $\Sigma \in \mathbb{R}^{T\times T}$, where,

$$\Sigma_{ij} = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)] = \frac{1}{N-1}\sum_{k=1}^{N}(x_{ki} - \mu_i)(x_{kj} - \mu_j). \tag{3.3}$$

The elements of Σ can be normalized by the factor $\sigma_n \sigma_m$, defining the correlation matrix R,

$$R_{ij} = \frac{\mathbb{E}[(X_i - \mu_i)(\mathbf{X}_j - \mu_j)]}{\sigma_i \sigma_j} = \frac{\Sigma_{ij}}{\sigma_i \sigma_j}. \tag{3.4}$$

each element R_{ij} is the correlation coefficient between random variables X_i and X_j. This coefficient gives us a degree of relationship between these two random variables. Moreover, $-1 \leq R_{ij} \leq 1$. As discussed in [Casella and Berger, 2002], the correlation coefficient measures the degree of relationship of how linearly related X_i and X_j are. The next theorem formalizes this concept,

Theorem 3.1. *(Correlation Coefficient [Casella and Berger, 2002]) For any random variables X_i and X_j,*

1. *$R_{ij} \in [-1, 1]$,*
2. *$R_{ij} = 1$ if and only if there exists numbers $a \neq 0$ and b such that $P(Y = aX + b) = 1$. If $R_{ij} = 1$ (resp -1), then $a > 0$ (resp < 0).*

Proof. Available in Section D.2. □

Example 3.1. (Correlation Analysis on Two Tanks Data) We consider $N = 560$ realizations of the two tanks system. The data is collected using Gaussian process noise with $\mu_\nu = 0$ and $\sigma_\nu = 5 \times 10^{-2}$, and Gaussian measurement noise with $\mu_\eta = 0$ and $\sigma_\eta = 5 \times 10^{-2}$. This gives us a data matrix $\mathbf{Y} \in \mathbb{R}^{N \times T}$, for $N = 560$ and $T = 750$. Note that Y_{ij} corresponds the realization y_i, sampled at time-step $t_j = jh$, for $h = 0.2$s. Being so, we calculate the sample correlation between random variables X_i and X_j, assuming $y(t_i) \sim X_i$ and $y(t_j) \sim X_j$, using Equation 3.4. As follows, in Figure 3.1 we show the correlation matrix R alongside the scatter plot of $y_i(t_j) \sim X_j$, for $j = 0, 10, 50$ and 60.

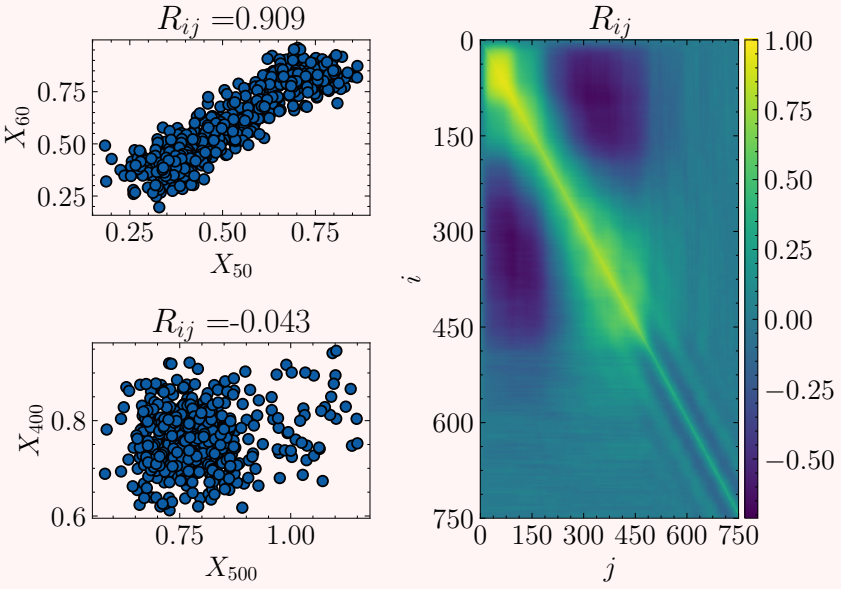


Figure 3.1: From left to right, samples extracted from Gaussian distributions corresponding to case 1, 2 and 3. As the correlation coefficient becomes closer to 1, the samples become more aligned. The sign of R_{ij} determines the line's slope sign.

Concerning Figure 3.1, we may see that random variables close together in time (e.g. X_{50} and X_{60}) are highly correlated ($|R_{ij}| \approx 1$), whereas for samples far apart in time (e.g. X_{400} and X_{500}), they are approximately uncorrelated ($|R_{ij}| \approx 0$). This can also be verified by inspecting at the heat map of R_{ij}. Note that along the diagonal one has $R_{nn} \approx 1$, which means that random variables X_i are correlated with themselves. Until the process settles $n, m < 450$, time-steps with small $\tau = j - i$ are correlated.

In addition, we will explore more specific types of processes, such as stationary and trend-stationary stochastic processes. Concerning the former, there are forms of stationarity.

Definition 3.2. (Strict-Sense and Wide-Sense Stationary Processes) A stochastic process is said to be Strict-Sense Stationary (SSS) if its statistical properties are invariant to a shift of the origin. On the other hand, a stochastic process is said to be Wide Sense Stationarity (WSS) if its mean is independent of the time-step:

$$\mu = \frac{1}{N}\sum_{k=1}^{N} x_{ki},$$

for any $i = 0, \cdots, T-1$, and its autocorrelation only depends on $\tau = j - i$,

$$\Sigma_{ij} = f(\tau).$$

Thus stationary processes represent a sub-set of stochastic processes with stronger properties. As we will see, this class of process will be useful in our analysis. For now, note that since μ does not depend on the time-step i, the variance is independent of i as well,

$$\sigma = \frac{1}{N-1}\sum_{k=1}^{N}(x_{ki} - \mu)^2. \tag{3.5}$$

In addition, the fact that Σ_{ij} only depends on $\tau = j - i$ motivates the specification of the function f used in Definition 3.2. Indeed, this allows us to define the autocovariance function:

$$K_{xx}(\tau) = \frac{1}{N-\tau}\sum_{k=0}^{N-\tau-1}(x_{ki} - \mu)(x_{k(i+\tau)} - \mu),$$

note that, since the correlation coefficient defined in Equation 3.4 is calculated in terms of the correlation, we may as well define an Autocorrelation Function (ACF),

$$R_{xx}(\tau) = \frac{1}{(N-\tau)\hat{\sigma}^2}\sum_{k=0}^{N-\tau-1}(x_{ki} - \mu)(x_{k(i+\tau)} - \mu). \tag{3.6}$$

Notice that for a lag τ, $R_{xx}(\tau)$ gives how correlated samples from time steps i and $i + \tau$ are. In the following discussion, we will give examples of various stochastic processes used in this thesis. We will finish by describing the usage of the ACF for fault diagnosis for the two-tanks system.

Definition 3.3. A stochastic process ϵ_i is called White Noise (WN) if it has zero mean, that is, $\mathbb{E}[\epsilon_i] = 0$, and if $R_{\epsilon\epsilon}(\tau) = \delta(\tau)$.

If, in addition, $\epsilon_k \sim \mathcal{N}(0, 1)$, ϵ_k is said to be a Additive White Gaussian Noise (AWGN). The AWGN process is particularly useful because it allows us to describe noisy variables, especially those coming from sensors under the action of Gaussian noise. In this case, one has,

$$x_i = a + \epsilon_i. \tag{3.7}$$

In the next two examples we characterize the randomness occurring in a dynamical system due to process and measurement noise. As follows, we start with the latter since it is simpler,

Example 3.2. (Measurement Noise) In this example we assume the sampling of two tank system near the equilibrium point. Furthermore, we assume a negligible process noise, and measurement noise $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$. We analyze the behavior of the measurement's ACF near the system's equilibrium point $\bar{\mathbf{x}}$, $\bar{\mathbf{u}}$,

$$y(t) = \mathbf{g}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) + \eta(t).$$

Now, supposing $y_i = y(t_i)$, and $\mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \bar{x}_2$, one has,

$$y_i = \bar{x}_2 + \eta_i,$$

which corresponds to measurements corrupted with AWGN noise. An example of a realization of such stochastic process is shown in Figure 3.2.

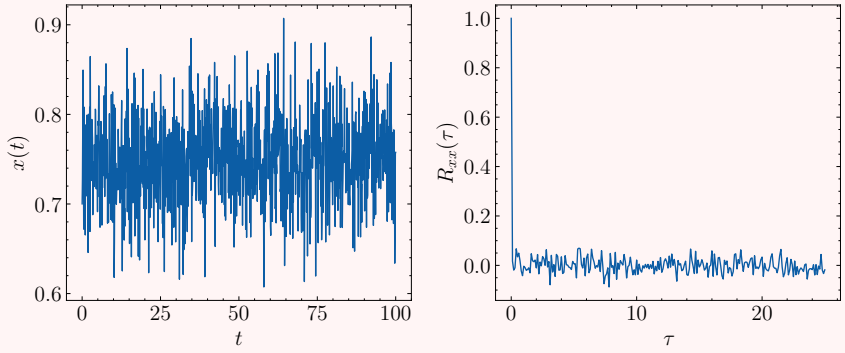


Figure 3.2: On the left, sampling from dynamical system corrupted with AWGN. On the right, the autocorrelation function of the samples.

In this case, $\mathbb{E}[\mathbf{X}_k] = \bar{x}_2$, hence one may retrieve the measurement through averaging. Even if $\epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$, one still has $R_\epsilon(\tau) = \delta(\tau)$ as the autocorrelation is normalized by σ_ϵ.

Example 3.3. (Process Noise) Besides the case of WN, suppose that we are simulating the two-tanks system near the steady-state. Moreover, let $\nu(i) = [\nu_1(i), \nu_2(i)]^T$ be the process noise vector, where $\nu_i(i) \sim \mathcal{N}(0, \sigma_i^2)$. In the following discussion we will approximate the differential

equation using the difference operator, that is, $\dot{\mathbf{x}} = \mathbf{x}_{i+1} - \mathbf{x}_k$, for a time-step i. Using this notation, we has,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}(\mathbf{x}_i, u_i) + \nu(k).$$

Notice that since we are assuming $\mathbf{x}_i \approx \bar{\mathbf{x}}$, we get,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \nu(i),$$

which is a stochastic dynamical system known as random walk. The samples $\mathbf{x}_i$ will be randomly distributed around $\bar{\mathbf{x}}$, provided that σ_i is small enough so that no $\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) \approx 0$. Additionally, provided that each sample $\nu(i)$ is independently and identically distributed (i.i.d), following a Gaussian distribution, one may expand the latter equation, getting,

$$\mathbf{x}_{i+1} = \bar{\mathbf{x}} + \tilde{\nu}(i),$$

where $\tilde{\nu}_i(i) \sim \mathcal{N}(0, k\sigma_i^2)$. Notice that since the variance of $\tilde{\nu}_i(k)$ depends on the time-step i, one might expect that for large i the system leaves the vicinity of the steady-state, and the system dynamics can no longer be neglected. An example of random walk is presented in Figure 3.3, alongside its ACF.

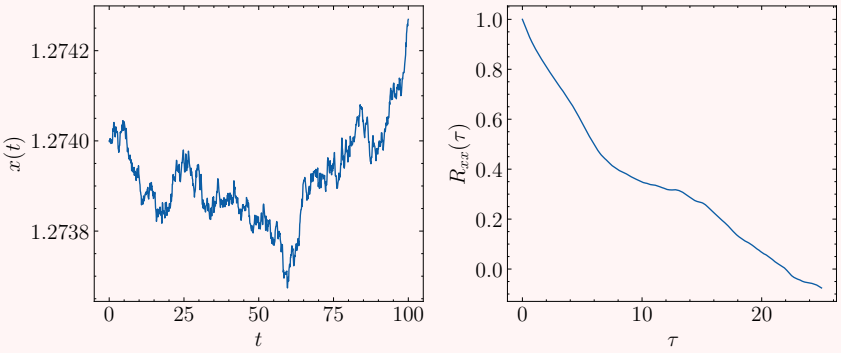


Figure 3.3: On the left, the realization of a random walk for the two-tank system in the proximity of the steady-state. On the right, its autocorrelation function. Notice that R_{hh} has a slow decay as τ increases. This indicates that samples are correlated with each other for small lags.

The previous analysis allow us to further give an intuition for the addition of process noise in the system dynamic equations, such as done in Equation 2.11: it introduces randomness in the system's behavior. Moreover, the noise variance σ_i presents a trade-off between the system's noiseless dynamics ($\sigma_i \to 0$), and a pure random-walk ($\sigma_i \to +\infty$).

In addition to the pure random walk example, we will further explore trend-stationary processes. This kind of stochastic process is characterized by a deterministic function $f(t)$ added with a WN component ϵ_t,

$$X_i = f(t_i) + \epsilon_i,$$

notice that since $\mathbb{E}[\epsilon_i] = 0$, the function $f(t_i)$ may be retrieved through time-averaging.

There are two cases that fit under the trend-stationary definition: (i) Noisy processes, such as the one presented in Equation 2.11, and (ii) Data acquired under sensor bias fault. These two cases are presented in Figure 3.4. Concerning the first type of trend-stationary process,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \nu,$$

where ν is a noise vector whose components are distributed according a Gaussian distribution. If the noise variance is small enough, the system will evolve towards the steady-state. Moreover, if the steady-state is stable, the system is expected to oscillate around it due the influence of noise. This is shown in the left part of Figure 3.4.

In the second case, the readings of the measured variable are affected by a trend given by $f(t) = h_1(t) + \delta t$, which is called sensor bias. Since the bias only affects the measurements, it has no influence on the system dynamics. The bottom part of Figure 3.4 shows an example of such case.

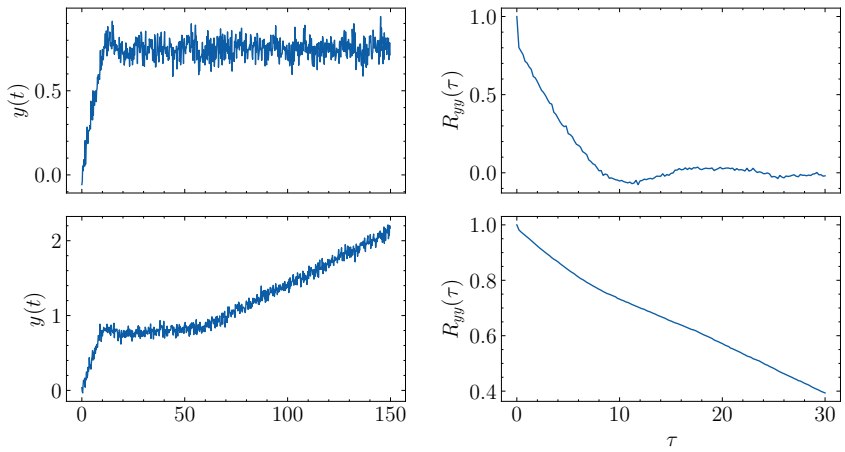


Figure 3.4: On the left, the system's output $y(t)$. On the right, the corresponding ACF. Two cases are shown: the system's normal operation (upper) and sensor bias fault (bottom).

Notice that the ACF for these two types of trend-stationary processes are quite different. In general, the correlation between samples may persist over time. Thus, for trend-stationary processes $R_{yy}(\tau)$ slowly decays towards zero as τ grows.

In the next example, we consider realizations of the system 2.11 for a fault diagnosis system, that is, for a classification problem involving possible faults. The resulting observations $y_i(t)$ and $u_i(t)$ are considered as samples from random variables due the presence of process noise ν_1, ν_2 and measurement noise η.

Example 3.4. (Autocorrelation Features) In a first moment, the raw data is acquired through simulation. For each run, a sample $S_m = \{(u(t_i), y(t_i))\}_{i=0}^{N-1} \in \mathbb{R}^{2N}$ is generated. This generates a raw data matrix $\mathbf{X} \in \mathbb{R}^{M\times 2N}$ containing the time series for the inputs and outputs. The ACF for the signal $y(t)$ (resp. $u(t)$) is calculated as:

$$R_{yy}(\ell) = \frac{1}{(N-\ell)\sigma^2} \sum_{k=0}^{N-\ell-1} (y_k - \mu_y)(y_{k+\ell} - \mu_y),$$

for lags $\ell = \{0, 1, \cdots, L-1\}$. Now, each sample S_m has an associated vector $R_m \in \mathbb{R}^L$ of autocorrelation coefficients. A comparative between the ACF in various fault scenarios is shown in Figure 3.5, for $L = 188$. To further reduce the matrix $\mathbf{R} = [R_1, R_2, \cdots, R_M] \in \mathbb{R}^{M\times L}$, we may further use the ℓ^1 norm of the ACF, defined as,

$$||R_{uu}||_1 = \sum_{\ell=0}^{L} \hat{R}_{uu}(\ell).$$

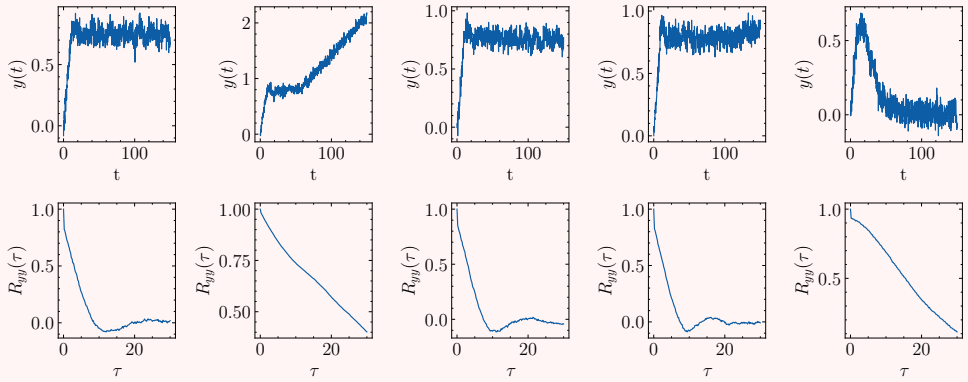


Figure 3.5: On the first row, samples of $y(t)$. On the second row, R_{yy} for various values of lag ℓ. Each row correspond to a different type of fault.

Note that, by using the ℓ_1 norm for the ACF of $u(t)$ and $y(t)$, one obtains two features for classification, $||\hat{R}_{uu}||_1$ and $||\hat{R}_{yy}||_1$, respectively. In Figure 3.6, the distribution of classes according to the two extracted features is shown. Notice that the faults are well separated from each other, indicating that these features can indeed distinguish them. In addition, other statistics may be extracted from the time-series, such as average value and variance. We maintain the two features mentioned previously since they allow us to visualize the two tanks data in $\mathbb{R}^2$.

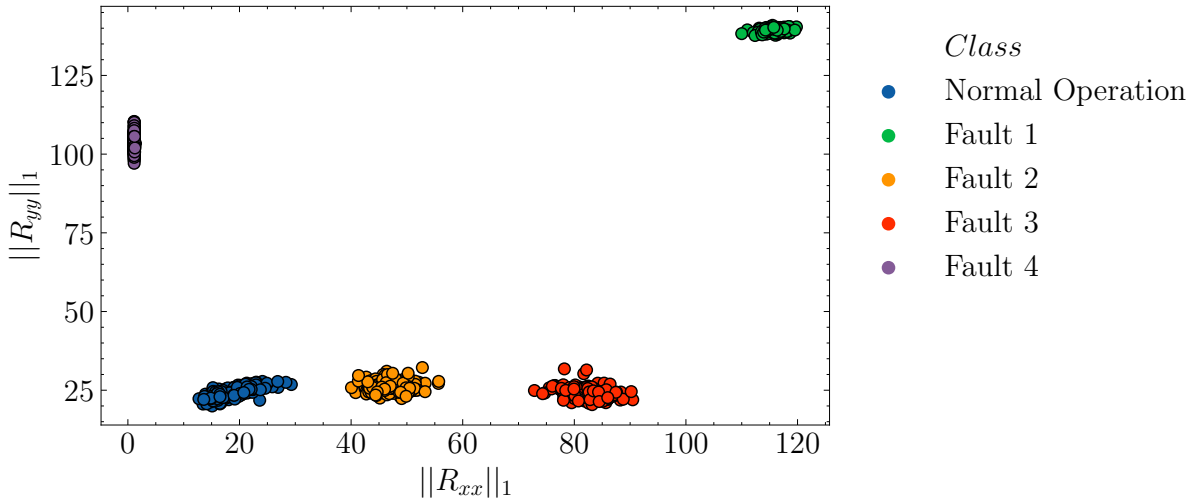


Figure 3.6: ℓ^1 norm of ACF function for input signal $u(t)$, and output signal $y(t)$.

3.2 Classification

Statistical Learning Theory

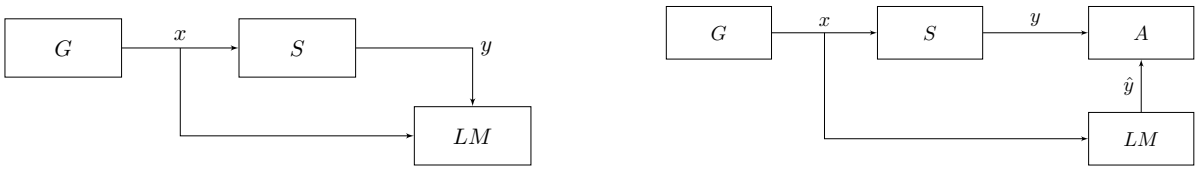
In this section we briefly discuss the building blocks of statistical learning theory. Unlike [Vapnik, 2013], we present this mathematical theory in the context of classification. The general supervised learning setting can be summarized by three components [Vapnik, 2013]:

1. A generator (G) of random vectors $x \in \mathbb{R}^d$, drawn independently from a fixed but unknown probability distribution $P(X)$,

2. A supervisor (S) who returns a label y to every input vector x, according to a conditional distribution function $P(Y|X)$,

3. A learning machine (LM) capable of implementing a set of functions $h(x;\alpha)$, $\alpha \in \Lambda$, also called hypothesis, for which Λ is a set of parameters.

4. An assessment machine (A), responsible for assessing the performance of LM based on the ground truth y, and the prediction $\hat{y}$



(1) Training phase (2) Evaluation phase

Figure 3.7: Theoretical setting for supervised learning. Figure adapted from [Vapnik, 2013]. Two steps are shown in the figure. First, on the right the training phase is shown, where LM learns from pairs (x, y) generated by G and S. Then, LM is evaluated by A, from new samples generated by G, and labeled by S.

Each of these three entities is shown in Figure 3.7. Especially, we may further define a family of functions $\mathcal{H} = \{h(\cdot;\alpha) : \mathcal{X} \to \{0,1\} : \alpha \in \Lambda\}$. This family of functions will be called hypothesis class. As follows, we may define the risk of a given hypothesis,

Definition 3.4. (Risk Minimization) Given a hypothesis class $\mathcal{H}$, a distribution P such that $\mathbf{x} \sim P$, a loss function ℓ, and the ground-truth labeling function f, the risk of a hypothesis h is defined as,

$$\mathcal{R}_P(h, f) = \mathop{\mathbb{E}}_{x \sim P}[\ell(h(x), f(x))]. \tag{3.8}$$

Moreover, under the same assumptions, the hypothesis function h^* such that

$$h^* = \mathop{argmin}_{h \in \mathcal{H}} \mathcal{R}_P(h, f), \tag{3.9}$$

is said the solution, in $\mathcal{H}$, of the learning problem posed by the distribution P and labeling function f. This framework is called risk minimization.

Moreover, we will adopt the short notation $R(h)$ for the risk functional, understanding that P and f can be inferred by the context. Following [Vapnik, 2013], three learning problems fit the risk minimization framework. They are: classification, regression and density estimation. This work particularly focuses on classification, for which [Vapnik, 2013] defines the loss function as:

$$\ell(h(x), f(x)) = \begin{cases} 0 & \text{if } h(x) = f(x) \\ 1 & \text{if } h(x) \neq f(x) \end{cases},$$

which is also called $0-1$ loss in the literature [Redko et al., 2020]. Using this definition of loss implies that Equation 3.8 represents probability of classification error of h. Notice that, in general, one does not have knowledge about $P(X)$ or $P(Y|X)$, but samples $(x_i, y_i)_{i=1}^n$ are

available. Therefore, Equation 3.8 cannot be calculated directly, making the minimization posed by Equation 3.9 intractable.

To overcome this limitation, the common practice is to substitute the true risk functional by its empirical approximation. Therefore, consider $\{x_i\}_{i=1}^n$ to be independently generated by G, and labeled by the ground-truth labeling function f, yielding $y_i = f(x_i)$. The empirical risk functional is defined as,

$$\hat{\mathcal{R}}_P(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i), \tag{3.10}$$

In the following discussion we will present two strategies for minimizing Equation 3.10: the SVM and CNN. The former is the standard state-of-the-art for classification when features have already been extracted, and are known to discriminate the classes. The latter is used when data is abundant, and assumes no features previously extracted.

Support Vector Machines

Originally, the SVM classifier is formulated as a linear classifier for binary classification problems. For purposes of illustration, supposes that indeed one has a classification task with two classes. Thus, the feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ have labels $y_i \in \{-1, +1\}$. For such problem, we may write the hypothesis function corresponding to the classifier at hand as,

$$h(\mathbf{x}; \mathbf{w}, b) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b),$$

where $\text{sign}(v) = 1$, if $v > 0$, -1 if $v < 0$, and 0 if $v = 0$. Therefore, the *decision boundary* of h, defined by $\Pi = \{\mathbf{x} \in \mathbb{R}^d : h(\mathbf{x}) = 0\}$ are characterized as hyperplanes in $\mathbb{R}^d$. The intuition behind the linear SVM algorithm is as follows: find the parameters $\mathbf{w}$ and b such that the distance between the hyperplane Π and each class is maximized. This latter distance is known as the classifier's margin, and can be defined in terms of $\mathbf{w}$ [Bishop, 2006]:

$$M(\mathbf{w}, b) = \frac{1}{||\mathbf{w}||}\underset{i=1,\cdots,N}{\text{minimize}}\ y_i h(\mathbf{x}_i).$$

Assuming that the classes are linearly separable one also constrains Π to not make classification mistakes, that is, $y_i h(\mathbf{x}_i) > 0\ \forall i$. Thus, the SVM optimization problem may be written as,

$$\begin{aligned} \underset{\mathbf{w},b}{\text{argmax}} \quad & \frac{1}{||\mathbf{w}||}\underset{i=1,\cdots,n}{\text{minimize}}\ y_i h(\mathbf{x}_i) \\ \text{subject to} \quad & y_j(\mathbf{w}^T\mathbf{x}_j + b) > 0 \end{aligned}$$

As pointed out by [Bishop, 2006], this problem is hard to solve since it involves two nested minimization problems. However, it may be rewritten in a simpler form,

$$\begin{aligned} \underset{\mathbf{w},b}{\text{argmin}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{aligned} \tag{3.11}$$

This latter equation is a quadratic program under linear constraints, and is known as the primal formulation of the linear hard-margin SVM algorithm. As follows, the previous optimization problem can be further converted into his dual, using the Lagrangian formalism. The steps are detailed in Appendix B. The dual problem for the SVM algorithm is,

$$\begin{aligned} \underset{\lambda}{\text{argmin}} \quad & \lambda^T\mathbb{1} - \frac{1}{2}\lambda^T\mathbf{K}\lambda \\ \text{subject to} \quad & \mathbf{y}^T\lambda = 0 \\ & \lambda \geq 0 \end{aligned} \tag{3.12}$$

At that point, this formulation may be generalized in several ways. First, note that early on the derivation we supposed that the classes were *linearly separable*. In practice, this is commonly not the case. To relax this rather strong assumption one needs to use the so-called slack variables [Bishop, 2006] ξ_i, so that the new optimization problem becomes,

$$\begin{aligned} \underset{\mathbf{w},b}{\text{argmax}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_j(\mathbf{w}^T\mathbf{x}_j + b) \geq 1 - \xi_j \end{aligned} \tag{3.13}$$

Note that whenever $\xi_j > 0$, we make a classification error, thus we penalize $\xi_j \neq 0$ with the term added to the objective function. The importance of this penalty is regulated by C. As in Equation 3.11, the same reasoning employed for deriving the dual problem may be used for 3.13. Therefore, one has the following optimization problem in dual form:

$$\begin{aligned} \underset{\lambda}{\text{argmin}} \quad & \lambda^T\mathbb{1} - \frac{1}{2}\lambda^T\mathbf{K}\lambda \\ \text{subject to} \quad & \mathbf{y}^T\lambda \\ & \lambda_i \in [0, C] \end{aligned} \tag{3.14}$$

Thus the relaxed SVM problem is also a quadratic problem under linear constraints, but additionally each λ_i is constrained to the interval $[0, C]$. Note that in the dual SVM formulation, $\mathbf{K} = [K_{ij}]_{i,j=1}^{N}$ is a matrix of inner products. Here we introduce the second generalization to the original SVM formulation: non-linearity. In principle, one can change K_{ij} by considering a feature mapping $\phi : \mathbb{R}^d \to \mathcal{F}$ so that,

$$K_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{F}}.$$

As follows, through the Kernel Trick [Bishop, 2006], we can devise a function $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ rather than specifying ϕ, as long as k is a positive-definite function, that is [Saitoh and Sawano, 2016], $\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{x}_i^T\mathbf{x}_j k(\mathbf{x}_i, \mathbf{x}_j) > 0$. In particular, throughout this work two types of kernel functions will be widely used: the linear, and the Radial Basis Function (RBF) kernels. These functions are defined as,

$$\text{Linear Kernel: } K_{ij} = \mathbf{x}_i^T\mathbf{x}_j, \tag{3.15}$$

$$\text{RBF Kernel: } K_{ij} = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||), \tag{3.16}$$

where γ, also known as the kernel's bandwidth, is a hyper-parameter controlling how fast K_{ij} drops to 0 with respect to the distance $||\mathbf{x}_i - \mathbf{x}_j||$. In this sense, for a threshold K_0, and considering neighbor points of $\mathbf{x}_i$ as those $\mathbf{x}_j$ such that $K_{ij} > K_0$, a lower bandwidth implies in a wider neighborhood.

Finally, there is the binary classification assumption. To generalize the SVM algorithm for handling multi-class problems a simple approach consists on converting the multi-class problem into a one-versus-rest problem. To do so, one fits K SVMs, where K corresponds to the number of classes in the problem. For each class C_i, its corresponding SVM is fit considering C_i as the positive class, and the rest C_j, $j \neq i$ as the negative class. Hence, consider that each classifier k, $k = 1, \cdots, K$ is denoted as h_k. For new predictions, the SVM decision is given by,

$$h(\mathbf{x}_i) = \underset{k=1,\cdots K}{\text{argmax}}\ h_k(\mathbf{x}_i).$$

As follows, the next examples shows the classification of faulty data acquired from the two tanks system simulation. It servers as an illustration of the SVM classifier.

Example 3.5. (SVM-based Fault Diagnosis) We consider the classification of faulty data acquired from the simulation of the two tanks system as described in Example 3.4. For each setting (faults 1 through 4 or normal operation), we sample 200 samples, resulting in a total of 1000 realizations. Note that each sample $\mathbf{x}_i = (\mathbf{u}_i, \mathbf{y}_i)$ consists on a 1500−dimensional vector. For $j < 750$, $x_{ij} = u_i(t_j)$, whereas for $j > 750$, $x_{ij} = y_i(t_{j-750})$. This is represented as a data matrix $\mathbf{X} \in \mathbb{R}^{1000\times1500}$, for which we extract the ℓ_1-norm of the ACF, as done in Example 3.4, yielding a matrix $\mathbf{R} \in \mathbb{R}^{1000\times2}$. The data is normalized, that is,

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} R_{ij},$$
$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(R_{ij} - \mu_j)^2,$$
$$R_{ij} = \frac{R_{ij} - \mu_j}{\sigma_j^2}$$

In Figure 3.8 we show a comparison for the decision boundary of a SVM for various kernel settings. Note that, while a linear kernel implies has a linear decision boundary, using a RBF kernel may lead to nonlinear boundaries. In addition, the bandwidth γ plays an important role in the classification result. Especially, note that for large values of γ, the region for which the classifier decides that its prediction belongs to a specific class shrinks considerably (e.g. compare $\gamma = 1$ and $\gamma = 10$).

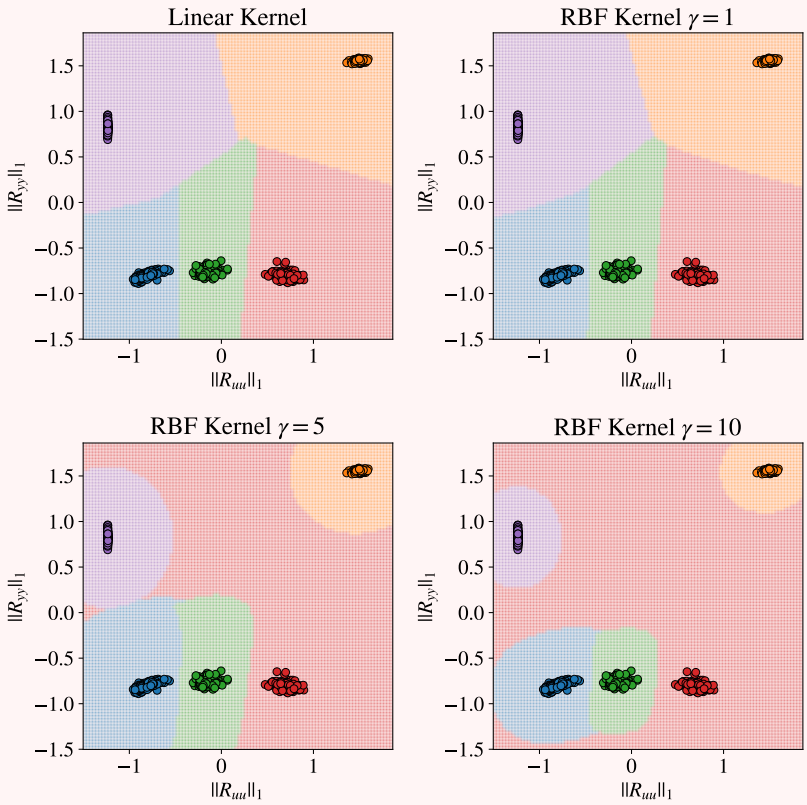


Figure 3.8: Classification of two tanks data with SVM.

Convolutional Neural Networks

A neural network is a learning machine, originally inspired by human's nervous system. In this discussion, we will adopt a bottom-up approach, the starting point for defining neural networks is by describing its building block: a neuron.

Definition 3.5. (Artificial Neural) A neuron is composed by inputs $\mathbf{x}$, weights $\mathbf{w}$, a bias b, and an activation function φ as shown in Figure 3.9.

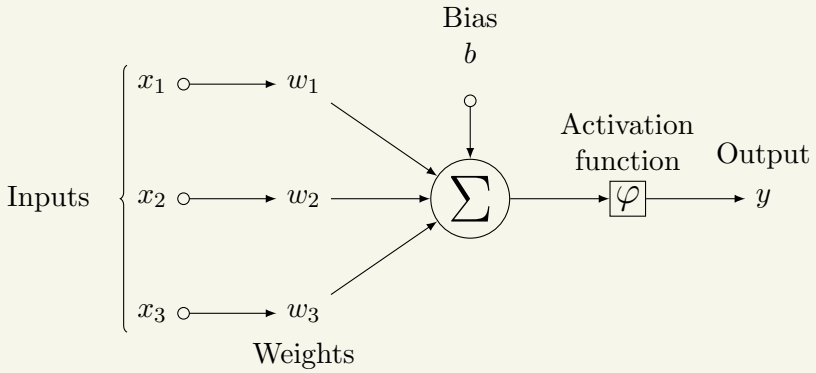


Figure 3.9: Mathematical model of an artificial neuron with three inputs.

In these terms, and following the diagram in Figure 3.9, one may express the neuron's output as,

$$y = \varphi(\mathbf{w}^T\mathbf{x} + b).$$

From the previous definition, neurons are represented through a inner product followed by a non-linearity. Going up in terms of abstraction, neurons may be further grouped into layers. These layers are connected linearly, creating a network, as suggests Figure 3.10.

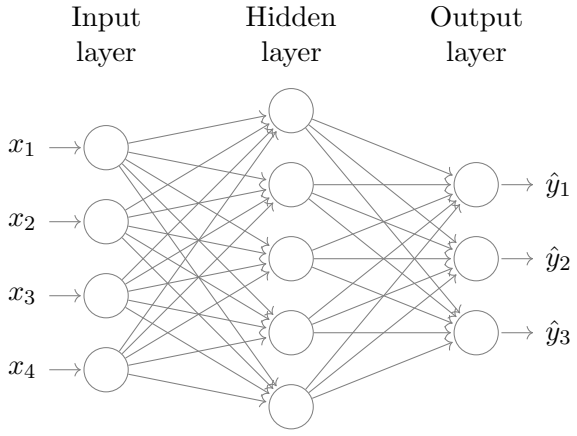


Figure 3.10: Representation of a neural network composed by various neurons.

In this case, a layer ℓ is composed by various neurons with weights $\mathbf{w}_i^\ell$. Thus, for a given layer, its operation may be represented in matrix form,

$$\hat{\mathbf{y}} = \varphi(\mathbf{W}^\ell\mathbf{x} + \mathbf{b}^\ell),$$

These networks have been called historically Perceptrons [Rosenblatt, 1958]. Here, we further make a distinction. Whenever a neural network does not have hidden layers, it is called a Single-Layer Perceptron (SLP). If, however, it has one or more, it is called Multi-Layer Perceptron (MLP). In addition, concerning the function φ, some common choices of non-linearities are the Rectified Linear Unit (ReLU) and softmax activation functions. The ReLU function is defined as,

$$\varphi(\mathbf{v}) = \max(0, \mathbf{v}),$$

and is commonly used in the inner layers of the neural network. The softmax activation, on the other hand, is defined as,

$$\varphi(\mathbf{v})_i = \frac{e^{\mathbf{v}_i}}{\sum_{j=1}^{n} e^{\mathbf{v}_j}},$$

and is used on the output layer. The motivation for using the ReLU activation is mainly due biological and efficiency reasons, as discussed by [Glorot et al., 2011]. In addition, the softmax activation has as its main advantage when used at the output layer. Indeed, by using it the output y is a probability distribution over possible classes, so that $y_i = P(y = i|X = x)$.

Moreover, note that the training of neural networks is done through unconstrained optimization. First, one defines a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \times \mathbb{R}$ which takes as input the ground-truth labels $\mathbf{y}$ and the predictions $\hat{\mathbf{y}}$ and outputs a value $\ell(\mathbf{y}, \hat{\mathbf{y}})$ that represents the cost of missing the predictions. An example is the cross-entropy cost, given by,

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i, \tag{3.17}$$

where K corresponds to the number of classes in the problem. Note that by assuming φ differentiable, one may take derivatives of ℓ with respect to each parameter in the network through the chain rule. This is known as the backpropagation algorithm [Rumelhart et al., 1986]. More details about the training of neural networks can be found in [Goodfellow et al., 2016].

In the context of this work, a CNN is simply a neural network for which one or more layers are based on the convolution operation, instead of matrix multiplication. Following [Goodfellow et al., 2016], the convolution of a vector $\mathbf{x}$ with a filter $\mathbf{w}$ is given by,

$$s_k = (\mathbf{x} * \mathbf{w})_k = \sum_{i=-\infty}^{+\infty} x_i w_{k-i}. \tag{3.18}$$

As shown in Figure 3.11, the convolution operation as presented by Equation 3.18 corresponds to rolling $\mathbf{w}$ over $\mathbf{x}$, while performing a sum and multiplication operation over the matched elements.

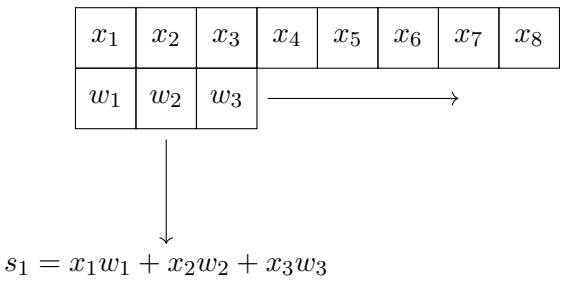


Figure 3.11: Illustration of the convolution operation between 1-dimensional arrays.

In addition to convolutional layers, another important type of component in a CNN is the pooling layer. This layer is summarizes the input signal over a local region by sub-sampling it. As pointed out by [Goodfellow et al., 2016], pooling helps to make the CNN approximately invariant to small translations in the input. Moreover, it helps in reducing the size of the representation, affecting the number of parameters in the network.

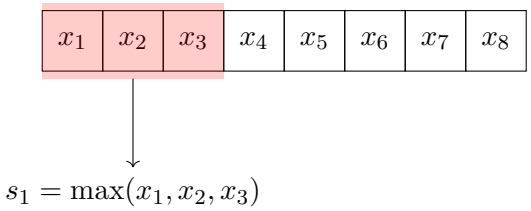


Figure 3.12: Illustration of the pooling operation in a max-pooling layer.

Furthermore, the design of a DNNs for a given task is commonly challenging. First, one needs to have enough amount of data to train the feature extractor and the classifier. This can be partially overcome by acquiring a model that has been trained in a large and diverse enough database, and then fine-tuning the last layers on a smaller dataset. Indeed, as demonstrated by [Yosinski et al., 2014], this kind of representational transfer can be done successfully in some cases.

Note that this last topic bears resemblance with the topic of this thesis. Indeed this is a case of transfer learning, but in this case the target data is assumed to be labeled so that the network can be fine-tuned. For the rest of this work this possibility is not considered, as our focus in unsupervised transfer learning.

Moreover, once the a CNN has been trained, one may retrieve the latent representation by feeding the inputs to the convolutional layers, thus generating $\mathbf{z} = g(\mathbf{x})$. As follows, to visualize the features $\mathbf{z}$, one may employ dimensionality reduction techniques, such as Principal Components Analysis (PCA) or t-Stochastic Neighbor Embeddings (t-SNE) [Van der Maaten and Hinton, 2008]. This latter technique is preferred over the former, since it provides captures the manifold structure which the data lies in, even when it is nonlinear. In the next example we discuss this visualization in the context of two tanks data.

Example 3.6. (t-SNE Visualization of CNN Features) We consider the design of a CNN for classifying raw fault diagnosis data acquired through the simulation of the two tanks system. Note that, as before, the features are originally stored in a matrix $\mathbf{X} \in \mathbb{R}^{1000\times1500}$. From this matrix, 800 samples are randomly chosen for training, while the other 200 are used for test. These samples were sampled in a stratified fashion, so that one has balanced classes in the training and test sets. As follows, we use the architecture specified in Table 3.1.

Layer	Tensor Shape	Parameters	Activation Function
Input	$(n_{batch} \times 1500 \times 1)$	-	-
Convolutional Layer 1	$(n_{batch} \times 1500 \times 32)$	32× filters of shape (6×1), bias of shape (32×1)	ReLU
Pooling Layer 1	$(n_{batch} \times 750 \times 32)$	-	-
Convolutional Layer 2	$(n_{batch} \times 750 \times 64)$	64× filters of shape (6×1), bias of shape (64×1)	ReLU
Pooling Layer 2	$(n_{batch} \times 375 \times 64)$	-	-
Convolutional Layer 3	$(n_{batch} \times 375 \times 128)$	128× filters of shape (6×1), bias of shape (128×1)	ReLU
Pooling Layer 3	$(n_{batch} \times 187 \times 128)$	-	-
Dense Layer 1	$(n_{batch} \times 1200)$	weight matrix $\mathbf{W}$ of shape $(1200, 23936)$, bias of shape (1200×1)	ReLU
Dense Layer 2	$(n_{batch} \times 200)$	weight matrix $\mathbf{W}$ of shape $(200, 1200)$, bias of shape (200×1)	ReLU
Dense Layer 3	$(n_{batch} \times 5)$	weight matrix $\mathbf{W}$ of shape $(5, 200)$, bias of shape (5×1)	Softmax

Table 3.1: CNN architecture for the classification of two-tanks data.

In addition, Figure 3.13 shows the course of training for the aforementioned network, alongside the t-SNE visualization of the extracted features. Especially, note that as the learning progresses, the CNN perfectly classifies training and test data. This is further verified by noting that the t-SNE representation shows five different clusters, corresponding to each class. This implies that, in the latent space, these classes are well separated.

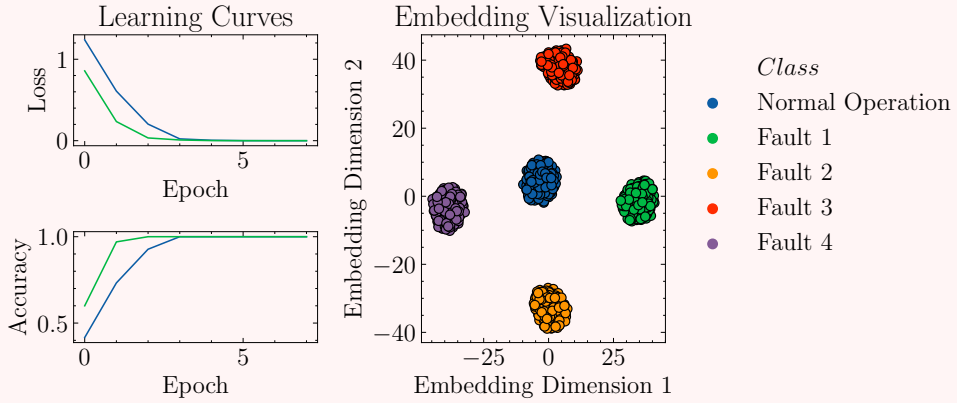


Figure 3.13: On the left, training curves for the CNN described in Table 3.1. On the right, embedding of CNN latent space generated using the t-SNE algorithm.

Chapter 4

Transfer Learning and Optimal Transport

The objective of this chapter is twofold,

1. Introducing the mathematical theory of optimal transport. This is done through sections 4.1 and 4.2, where we present the main concepts needed for our analysis.

2. Introducing the theoretical framework, and algorithms for transfer learning. This is done through sections 4.3, 4.4 and 4.5.

Briefly, in this chapter we present the methodology behind domain adaptation, which will be used to perform cross-domain fault diagnosis.

4.1 Introduction to Optimal Transport

The origins of OT theory dates the 18-th century, being founded by Gaspard Monge. Following the historical discussion presented by [Villani, 2008], Monge considered a problem which an amount of soil has to be extracted from a source (called *déblais*), and transported to a destination (called *remblais*). The question asked by Monge is the following [Villani, 2008]: to which destination should one send the material that has been extracted at a certain place, which is shown in Figure 4.1.

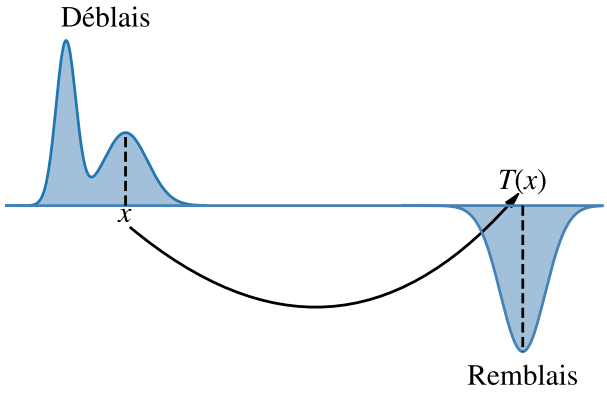


Figure 4.1: Transportation problem between *déblais* and *remblais*. Figure reproduced from [Villani, 2008].

Upon the assumption that the mass transported from the source to the target remains the same, the notion of mass and probability are analogous. Therefore, this discussion starts by the definition of empirical probability measures,

Definition 4.1. (Empirical Measure) Suppose that $S = \{\mathbf{x}_S^i\}_{i=1}^{n_S}$ is a sample of size n_S sampled from a probability measure μ. An empirical probability measure $\hat{P}_S = (\mathbf{a}, \mathbf{X}_S)$ is defined in terms of its sample weights $\mathbf{a} \in \mathbb{R}^{n_S}$ and support $\mathbf{X}_S = [\mathbf{x}_S^1, \cdots, \mathbf{x}_S^{n_S}] \in \mathbb{R}^{n_S \times d}$, through the formula,

$$\hat{P}_S(\mathbf{x}) = \sum_{i=1}^{n_S} a_i \delta(\mathbf{x} - \mathbf{x}_i),$$

where δ is the Dirac delta function.

Therefore an empirical measure is an approximation to the underlying probability distribution for which its support has been sampled. This definition is favorable to machine learning, since the underlying probability distribution μ is unknown, as discussed in Chapter 3.

As follows, let us consider two empirical measures $\hat{P}_S = (\mathbf{a}, \mathbf{X}_S)$ and $\hat{P}_T = (\mathbf{b}, \mathbf{X}_T)$. The Monge formulation of optimal transport is devising a mapping $T : \mathbb{R}^d \to \mathbb{R}^d$, such that $\hat{P}_S$ is transported into $\hat{P}_T$. Intuitively, T transports the sample $\mathbf{x}_S^i$ with mass a_i into the sample $\mathbf{x}_T^j$, with mass b_j. Following this intuition, T cannot split the mass a_i from $\mathbf{x}_S^i$, since the sample is transported into one sole location. In addition, we enforce T to be mass preserving, thus,

$$b_j = \sum_{i:T(\mathbf{x}_S^i)=\mathbf{x}_j^T} a_i,$$

that is, all mass received from $\mathbf{b}$ must come from some elements from $\mathbf{a}$. This property may be written in short as $T_{\#}\hat{P}_S = \hat{P}_T$, where $T_{\#}$ is the push-forward operator [Peyré et al., 2019]. From this discussion, it is not clear how to define T as mapping. We thus define the concept of Monge Problem (MP), which gives us the transportation at least effort from $\hat{P}_S$ to $\hat{P}_T$.

Definition 4.2. (Monge Formulation of Optimal Transport [Peyré et al., 2019]) Given two probability measures $\hat{P}_S = (\mathbf{a}, \mathbf{X}_S)$ and $\hat{P}_T = (\mathbf{b}, \mathbf{X}_T)$, and a cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, the optimal transport mapping T^* is given by

$$T^* = \underset{T_{\#}\hat{P}_S = \hat{P}_T}{\operatorname{argmin}} \sum_{i=1}^{n_S} c(\mathbf{x}_S^i, T(\mathbf{x}_S^i)), \tag{4.1}$$

we will henceforth refer to this optimization problem as MP.

A couple of technical difficulties arise from the MP. First, not being able to split mass among samples restricts the cases where Equation 4.1 has solutions. Especially, one is restricted to $n_S \leq n_T$. Second, the constraint in the minimization problem is not convex [Peyré et al., 2019].

The latter limitations motivate the use of another formulation for optimal transport, called Kantorovich formulation [Kantorovich, 2006]. Thus, instead of defining a mapping T, one treats a coupling $\gamma \in \mathbb{R}^{n_S \times n_T}$, such that γ_{ij} represents the amount of mass transported from $\mathbf{x}_S^i$ to $\mathbf{x}_T^j$. As before, we enforce γ to be mass-preserving, thus,

$$a_i = \sum_{j=1}^{n_T} \gamma_{ij},$$

$$b_j = \sum_{i=1}^{n_S} \gamma_{ij},$$

that is, the mass a_i of sample $\mathbf{x}_S^i$ is completely transported to samples $\mathbf{x}_T^j, j = 1, \cdots, n_T$, and b_j receives its mass from samples $\mathbf{x}_S^i, i = 1, \cdots, n_S$. This constraint can be written in short by defining the set $\Pi(\hat{P}_S, \hat{P}_T)$ as follows,

$$\Pi(\hat{P}_S, \hat{P}_T) = \{\gamma \in \mathbb{R}^{n_S \times n_T} : \gamma^T 1_{n_S} = \mathbf{a}, \gamma 1_{n_T} = \mathbf{b}\},$$

where the constraints have been rewritten in matrix form, with 1_n being a vector in $\mathbb{R}^n$, where every entry is equal to 1. Note that Π is defined in terms of linear constraints. This allows the definition of optimal transport in terms of a linear program,

Definition 4.3. (Kantorovich Formulation of Optimal Transport [Peyré et al., 2019]) Let $\hat{P}_S$ and $\hat{P}_T$ be two probability measures, and c be a cost functional as in Definition 4.2. Let $C_{ij} = c(\mathbf{x}_S^i, \mathbf{x}_T^j)$ be a cost matrix. The optimal transport plan γ^* is given by

$$\gamma^* = \underset{\gamma \in \Pi(\hat{P}_S, \hat{P}_T)}{\operatorname{argmin}} \sum_{i=1}^{n_S} \sum_{j=1}^{n_T} C_{ij} \gamma_{ij}, \tag{4.2}$$

henceforth we will refer to Equation 4.2 as Kantorovich Problem (KP).

Note that the objective function in the minimization problem 4.2 is linear with respect to γ_{ij}. As remarked before, the constraints are linear as well. Thus the KP is a linear program, for which there is always at least one solution. This can be verified by noting that $\gamma_{ij} = a_i b_j \in \Pi(\hat{P}_S, \hat{P}_T)$. Thus, for machine learning the Kantorovich's formulation is preferred over the Monge's.

Additionally one of the main usage of optimal transport theory for machine learning is defining a distance between probability distributions, known as Wasserstein distance. In the context here presented, it rather defines a distance between empirical measures. Note that if enough samples are available, this gives an accurate estimate of the true Wasserstein distance between the underlying measures.

Definition 4.4. (Wasserstein Distance [Peyré et al., 2019]) Let $p \in [1, \infty)$. For empirical $\hat{P}_S$ and $\hat{P}_T$ as in Definition 4.2, and $c(\mathbf{x}_S^i, \mathbf{x}_T^j) = ||\mathbf{x}_S^i - \mathbf{x}_T^j||_p$, the Wasserstein distance of order p is defined by the formula

$$W_p(\hat{P}_S, \hat{P}_T) = \left(\underset{\gamma \in \Pi(\mu, \nu)}{\text{minimize}} \sum_{i=1}^{n_S} \sum_{j=1}^{n_T} C_{ij} \gamma_{ij} \right)^{1/p}$$

Some common choices for p are 1, for which the Wasserstein distance is known as Earth-Mover Distance (EMD), and 2. Furthermore [Villani, 2008] proves that the Wasserstein distance is indeed a distance between probability distributions. Still following [Villani, 2008], the Wasserstein distance profits from several nice properties. Among them, we select the two more important for this work:

1. They encode good geometric information, if the cost c is defined in term of the underlying geometry. In particular, the Wasserstein distance has shown good results in problems such as barycenter calculation [Cuturi and Doucet, 2014] and displacement interpolation [Solomon et al., 2015].

2. They can be accurately estimated from finite samples [Bolley et al., 2007]. This result is especially important for optimal transport applications to machine learning.

In the following example we give an illustration of the optimal transport plan calculated through linear programming in the context of two tanks faulty data.

Example 4.1. (Transportation Plan between Two Tanks samples) We consider ACF ℓ^1-norm features extracted from the two-tanks system signals. Moreover, they are normalized as in Example 3.5. The points are sampled using different conditions for the tank parameter specification (the areas $A_1, A_2, a_{12}, \cdots, a_{23}$). Figure 4.2 display two visualizations of γ.

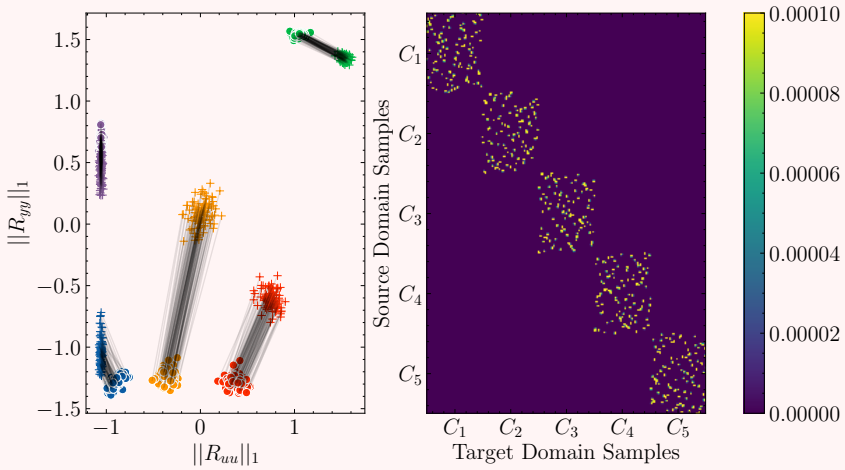


Figure 4.2: Visualizations of the transport plan matrix γ. On the left, we present a scatter plot of the data points, with links reflecting the elements γ_{ij}. Circles were sampled using parameters $a_{12} = a_{13} = a_{23} = 0.1$ and $A_1 = A_2 = 1$, while crosses were sampled using $a_{12} = a_{13} = a_{23} = 0.68$ and $A_1 = A_2 = 1.68$. On the right, a heat map displaying each value γ_{ij}.

In the left part of Figure 4.2, lower values of γ were intentionally omitted, so that only the stronger connections are displayed. Moreover we remark that γ is approximately sparse with respect to the faults, namely, if $\mathbf{x}_S^i$ and $\mathbf{x}_T^j$ have different labels, $\gamma_{ij} = 0$. This latter claim is evidenced in the right part of Figure 4.2. Hence, this example suggests that even when the process mathematical model presents differences with respect to the parameters, the transport plan is intuitive: to match distributions $\hat{P}_S$ and $\hat{P}_T$ one needs to move the mass from each class C_i in the source domain to C_j, in the target domain.

4.2 Regularized Optimal Transport

The Kantorovich Problem, stated in Definition 4.3, provides a formal description of the Optimal Transport problem in terms of Linear Programming. A major drawback from this approach is its complexity, which increases on the number of samples x_i and y_j.

As stated in [Courty et al., 2016], for samples $S = \{\mathbf{x}_S^i\}_{i=1}^{n_S}$ and $T = \{\mathbf{x}_T^j\}_{j=1}^{n_T}$, the complexity of estimating γ with linear programming is $\mathcal{O}((n_S + m_T)n_S m_T log(n_S + m_T))$, or simply, for $n = n_S = n_T$, $\mathcal{O}(n^3 log(n))$. A different approach consists on introducing a regularization term to the cost defined by Equation 4.2

$$\gamma_\lambda^* = \underset{\gamma \in \Pi(\hat{P}_S, \hat{P}_S)}{\operatorname{argmin}} \sum_{i,j} C_{ij}\gamma_{ij} + \lambda\Omega(\gamma). \tag{4.3}$$

The function $\Omega(\gamma)$ added to Equation 4.3 is called regularization term. Moreover, $\lambda \geq 0$ is called penalty, and it represents a trade-off between the original cost, and the regularization term. The most used regularization consists on using the entropy of γ. Therefore

$$\Omega(\gamma) = -\sum_{i,j} \gamma_{ij}(\log(\gamma_{ij}) - 1). \tag{4.4}$$

This term is popular since it yields an efficient method called Sinkhorn algorithm [Cuturi, 2013]. By letting $\Omega(\gamma) = -\sum_{i,j} \gamma_{ij}(log(\gamma_{ij}) - 1)$, the Kantorovich problem may be restated as

$$\gamma_\lambda^* = \underset{\gamma \in \Pi(\hat{P}_S, \hat{P}_T)}{\operatorname{argmin}} \sum_{i,j} C_{ij}\gamma_{ij} + \lambda \sum_{i,j} \gamma_{ij}(log(\gamma_{ij}) - 1). \tag{4.5}$$

When $\lambda \to 0$, the original problem defined by Equation 4.2 is restored. Otherwise, when $\lambda \to \infty$, Equation 4.5 is equivalent to

$$\gamma_\infty^* = \underset{\gamma \in \Pi(\hat{P}_S, \hat{P}_T)}{\operatorname{argmin}} \sum_{i,j} \gamma_{i,j}(log(\gamma_{i,j}) - 1) \tag{4.6}$$

The solution of this latter equation is simply $\gamma_{ij} = a_i b_j$. To prove this claim, notice that the coupling γ is a joint distribution over (X, Y). With a slight abuse of notation, let $H(\gamma) = H(X, Y)$ be the entropy of the coupling γ, $I(\gamma) = I(X; Y)$ the relative entropy of the pair (X, Y) and $H(\mu) = H(X)$ (resp. $H(\nu) = H(Y)$) be the entropy of each marginal. Hence,

$$\Omega(\gamma) = H(X, Y) = I(X; Y) - H(X) - H(Y),$$

as consequence, the minimization in Equation 4.6 corresponds to the minimization of mutual information. The minimum is attained precisely when $\gamma_{ij} = a_i b_j$, which also respects the constraints. This latter case implies that random variables X and Y are independent. Even though the solution given by the Sinkhorn algorithm is not sparse, its execution is much faster than the standard KP, which is a linear program. In the following discussion we will derive the Sinkhorn algorithm. Thus, we begin with an important property that follows from the addition of the entropic regularization term,

Proposition 4.1. *(Entropic Penalty and Kullback-Leibler Divergence [Peyré et al., 2019]) Let $\hat{P}_S = (\mathbf{a}, X)$ and $\hat{P}_T = (\mathbf{b}, Y)$ be two empirical measures. The regularized Optimal Transport Problem defined by Equation 4.5 is equivalent to,*

$$\gamma_\lambda^* = \underset{\gamma \in \Pi(\hat{P}_S, \hat{P}_T)}{argmin} \; KL(\gamma | \mathbf{K}).$$

Where KL is the Kullback-Leibler divergence, defined by

$$KL(\gamma | \mathbf{K}) = \sum_{i,j} \gamma_{ij} \log\left(\frac{\gamma_{ij}}{K_{ij}}\right) - \gamma_{ij} + K_{ij}, \tag{4.7}$$

and $K_{ij} = e^{-C_{ij}/\lambda}$ is the Gibbs Kernel.

Proof. Available in Section D.3. □

As already remarked, the fact that Ω is differentiable allows for a much faster algorithm to be derived, the Sinkhorn Algorithm. To describe it, a starting point is to notice that the optimization problem defined by Equation 4.5 has a nice closed form.

Proposition 4.2. *(Sinkhorn Algorithm Iterations [Peyré et al., 2019]) Let* $K_{ij} = e^{-C_{ij}/\lambda}$*, for* $\lambda > 0$*. The solution of Equation 4.5 has the for*

$$\gamma = diag(\mathbf{u})\mathbf{K}diag(\mathbf{v}), \tag{4.8}$$

for unknowns $\mathbf{u} \in \mathbb{R}_+^n$ *and* $\mathbf{v} \in \mathbb{R}_+^m$

Proof. Available in Section D.3. □

The consequence of this proposition is that Equation 4.5 has a closed form solution, given by Equation 4.8. Since the solution needs to suffice the constraints $\gamma \mathbf{1}_n = \mathbf{a}$ and $\gamma^T \mathbf{1}_m = \mathbf{b}$, it follows that,

$$diag(\mathbf{u})\mathbf{K}diag(\mathbf{v})\mathbf{1}_n = \mathbf{a},$$
$$diag(\mathbf{v})\mathbf{K}^T diag(\mathbf{u})\mathbf{1}_m = \mathbf{b},$$

which, in vector form, may be written as a pair of non-linear equations,

$$\mathbf{u} \odot (\mathbf{K}\mathbf{v}) = \mathbf{a},$$
$$\mathbf{v} \odot (\mathbf{K}^T\mathbf{u}) = \mathbf{b},$$

where we used $\odot$ as the element-wise product between vectors. This approach results on an iterative algorithm for finding $\mathbf{u}$ and $\mathbf{v}$

$$\mathbf{u}^{\ell+1} = \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}^{\ell}}, \tag{4.9}$$
$$\mathbf{v}^{\ell+1} = \frac{\mathbf{a}}{\mathbf{K}^{\mathbf{T}}\mathbf{u}^{\ell+1}}, \tag{4.10}$$

Algorithm 1 Computation of γ and d

Input: Cost matrix C, marginals $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, penalty λ
  $\mathbf{K}_{ij} = e^{-C_{ij}/\lambda}$
  $\mathbf{u} = \mathbf{1}_n/n$; $\mathbf{v} = \mathbf{1}_m/m$
  while not converged **do**
    $\mathbf{u} = \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}}$
    $\mathbf{v} = \frac{\mathbf{b}}{\mathbf{K}^T\mathbf{u}}$
  end while
  $\gamma = diag(\mathbf{u})\mathbf{K}diag(\mathbf{v})$
  $d = \mathbf{1}_n^T\Big(C \odot (diag(\mathbf{u})(\mathbf{K})diag(\mathbf{v}))\Big)\mathbf{1}_m$
Output: Optimal Transport plan, γ, Optimal Transport cost d.

where the division should be understood element-wise. Equations 4.9 and 4.10 define the vanilla Sinkhorn algorithm [Cuturi, 2013], which approximates the optimal transport solution for small values of λ. In Algorithm 1 we present an alternative formulation of Algorithm 1 of [Cuturi, 2013]. The interest in using Algorithm 1 is twofold, as it can either be used to approximate the optimal transport plan γ, or to approximate the optimal transport cost d. The next example shows the estimation of γ_λ^* in the context of teh two-tanks system. The motivation of this example is very similar to that of Example 4.1.

Example 4.2. (Transportation Plan estimated through Sinkhorn Algorithm) For solving Equation 4.5, we will choose $\lambda = 10^{-2}$ as it illustrates important properties of the Sinkhorn algorithm. Again, we will assume that the data was generated through the extraction of ACF ℓ^1 norm from two-tanks system simulation.

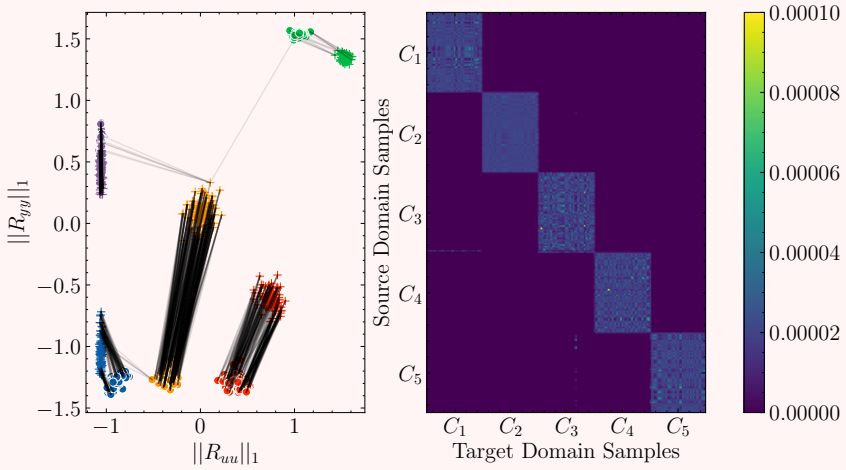


Figure 4.3: Visualizations of the transport plan matrix γ. The sampling and visualization conditions are the same of Figure 4.2.

The visualization of γ is shown in Figure 4.3. In particular, notice that the same pattern of class sparsity appears, showing that the transport plan estimated by the Sinkhorn algorithm is similar to the exact solution. However, as mentioned before, transport plans calculated by the Sinkhorn algorithm are not sparse, as can be seen in the right part of Figure 4.3. This claim can also be verified in the left part of this figure, as much more links appear between the support of the two measures.

The previous example shows an undesirable property of entropic regularization: the transport matrix becomes non-sparse. In particular, as discussed by [Blondel et al., 2018] this is problematic in fields where γ is of interest, such as domain adaptation. The issue becomes evident as the number of samples to transport grows, as sparse solutions become cheaper with respect to memory storage. To address this issue, [Blondel et al., 2018] proposes the usage of ℓ^p regularization instead of the entropic term,

$$\gamma^*_\lambda = \underset{\gamma \in \Pi(\hat{P}_S, \hat{P}_T)}{\operatorname{argmin}} \sum_{i,j} C_{ij}\gamma_{ij} + \sum_{i,j} \gamma_{ij}^2.$$

In this previous equation one must further enforce the constraint $\gamma_{ij} > 0$, since the objective function is defined for negative values of γ_{ij}, unlike the entropic regularization.

4.3 Introduction to Transfer Learning

Foundations

As formalized by [Pan and Yang, 2009], transfer learning is concerned with transferring knowledge between domains and tasks. These two are defined below,

Definition 4.5. (Domains and Tasks) A domain $\mathcal{D}$ is a pair $(\mathcal{X}, P)$, where $\mathcal{X}$ is the feature space, and $P(X)$ is the feature probability distribution. Moreover a task $\mathcal{T}$ is pair $(\mathcal{Y}, f)$, where $\mathcal{Y}$ is the target space and f is the predictive function.

In the context of binary classification, $\mathcal{Y} = \{0, 1\}$, and f can be identified as the ground-truth labeling function. Without loss of generality, one may assume $f \in \mathcal{H}$, where $\mathcal{H}$ is the space of hypothesis functions. It is important to remark that, throughout Section 3.2, an assumption has been made: there is an unique domain $\mathcal{D}$ and task $\mathcal{T}$. Whenever that is not the case, there is transfer learning. Moreover, as shown in 4.4, one may further categorize the various types of transfer learning. This is illustrated in Definition 4.6.

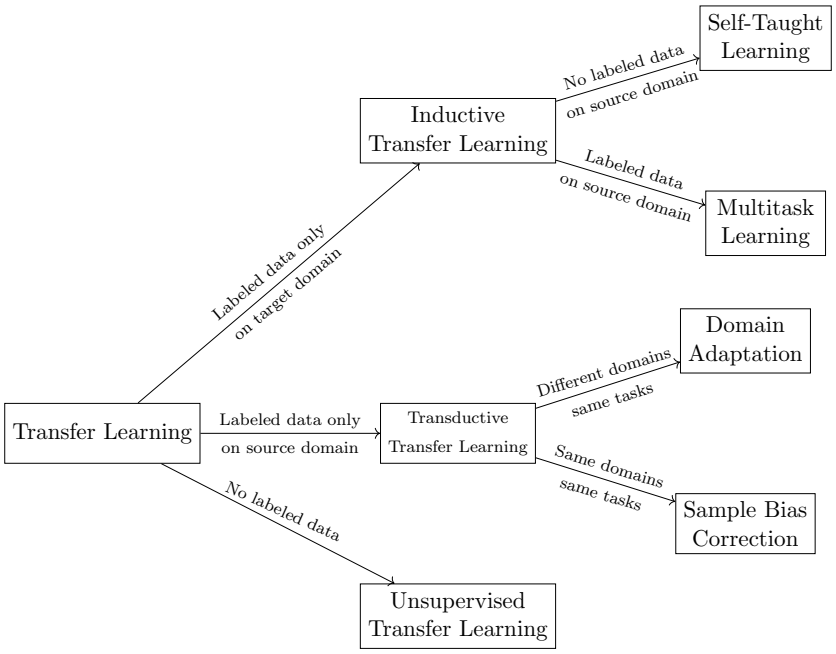


Figure 4.4: Relationship between different types of transfer learning. From left to right, the boxed presented different transfer learning settings with increasing degree of specificity. This figure was adapted from [Pan and Yang, 2009].

Definition 4.6. (Categories of Transfer Learning [Pan and Yang, 2009]) Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, *transfer learning* aims to help improving the learning of the target predictive function f_T in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$. We thus have the following categories:

Inductive Transfer Learning where $\mathcal{T}_S \neq \mathcal{T}_T$.

Transductive Transfer Learning where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$

Unsupervised Transfer Learning where $\mathcal{T}_S \neq \mathcal{T}_T$ and $\mathcal{Y}_S$ and $\mathcal{Y}_T$ are not observable.

Inductive transfer learning is named that way since it parts from the reasoning about a general case (source domain), to reasoning about a particular case (target domain). This setting for transfer learning comprises two frameworks: self-taught learning and multitask learning.

- Self-taught learning has been applied in self-taught clustering [Dai et al., 2008], where authors have proposed an algorithm for clustering small amounts of unlabeled data by using auxiliary data that not necessarily follows the same probability distribution. In addition, [Raina et al., 2007] proposes an algorithm for learning features in an unsupervised fashion, on the target domain. Note that this algorithm is very similar in idea to the approach of unsupervised pre-training of deep neural networks proposed in [Bengio et al., 2007] and further analyzed by [Erhan et al., 2010].

- Multitask learning has various applications, especially when used alongside neural networks. Note that it is fairly simple to train a neural network architecture under the multitask framework. Indeed, as [Crawshaw, 2020] suggests one may design a shared block of parameters (such as convolutional layers), and add multiple independent paths in the network for training to specific tasks. Moreover, in the context of fault diagnosis, [Hasan et al., 2020] proposed a method for health and speed detection from bispectrum images.

As [Pan and Yang, 2009] puts, the unsupervised category is concerned with the application of transfer learning to unsupervised tasks, such as clustering, dimensionality reduction and density estimation. The last category of transfer learning is *transductive transfer learning*, and is the central problem of this work. As discussed by [Pan and Yang, 2009], this transfer learning setting is also known in the literature as Domain Adaptation (DA). In addition, being DA one of the major topics in this thesis, we will focus on this category instead of the other two. Note that since a domain $\mathcal{D}$ is a pair $(\mathcal{X}, P(X))$, the reason for which $\mathcal{D}_S \neq \mathcal{D}_T$ may be one of two: 1. the feature spaces differ ($\mathcal{X}_S \neq \mathcal{X}_T$), or 2. their probability distributions are different ($P_S(X) \neq P_T(X)$).

Note that we may assume without loss of generality that $\mathcal{X}_S = \mathcal{X}_T$, as the features are engineered before the learning process. Thus the case in which $P_S(X) \neq P_T(X)$ is investigated. Indeed, this is a particular case of a more general phenomenon called *distributional shift*, defined below.

Definition 4.7. (Distributional Shift) Let $\mathcal{D}_S$, $\mathcal{D}_T$, $\mathcal{T}_S$ and $\mathcal{T}_T$ define a transfer learning problem. The distributional shift assumption corresponds to assuming $\mathcal{X}_S = \mathcal{X}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$, while leaving the following distributions free to differ,

Feature distributions $P_S(X)$ and $P_T(X)$,

Label distributions $P_S(Y)$ and $P_T(Y)$,

Conditional distributions $P_S(Y|X)$ and $P_T(Y|X)$ or $P_S(X|Y)$ and $P_T(X|Y)$.

Measuring Distributional Shift

In this section we discuss how to measure the degree upon which $\hat{P}_S$ is different from $\hat{P}_T$. Note that we previously defined the Wasserstein distance. Thus, here we present three other distances between probability distributions: the Total Variation (TV) distance, the $\mathcal{H}-$distance, and the Maximum Mean Discrepancy (MMD) distance.

Definition 4.8. (Total Variation [Ben-David et al., 2010]) The Total Variation distance TV is given by:

$$\mathrm{TV}(P_S, P_T) = 2 \sup\Big\{|P_S(B) - P_T(B)| : B \in \mathcal{B}\Big\}, \tag{4.11}$$

where $\mathcal{B}$ is the set of measurable subsets under μ and ν.

Notice that, for $P_S = \delta_x$ and $P_T = \delta_y$, $\mathrm{TV}(P_S, P_T) = 2$ whenever $x \neq y$. For $x = y$, one has $\mathrm{TV}(P_S, P_T) = 0$. Therefore, by denoting $\mathrm{supp}(P_S)$ to the support of the distribution P_S, we have,

$$\mathrm{TV}(\hat{P}_S, \hat{P}_T) = \begin{cases} 2 & \text{if } \mathrm{supp}(\hat{P}_S) \neq \mathrm{supp}(\hat{P}_T) \\ 0 & \text{otherwise} \end{cases},$$

As pointed out by [Ben-David et al., 2010], the main issue for the TV distance is that it cannot be accurately estimated from finite samples. This result was first proved in [Batu et al., 2000]. Motivated by this limitation, [Kifer et al., 2004] proposed an alternative distance called proxy $\mathcal{A}$-distance. Their insight was to limit the family of sub-sets for which the supremum in Equation 4.11 is evaluated. As it turns out the $\mathcal{H}-$distance is defined in terms of the proxy $\mathcal{A}-$distance as follows.

Definition 4.9. ($\mathcal{H}-$distance [Ben-David et al., 2010]) Let $\mathcal{H}$ be a set of hypothesis functions. The $\mathcal{H}-$distance is calculated as follows,

$$d_{\mathcal{H}}(P_S, P_T) = 2 \sup\Big\{|P_S(I(h)) - P_T(I(h))| : h \in \mathcal{H}\Big\}. \tag{4.12}$$

where $I(h)$ is the so-called index set, that is $I(h) = \{\mathbf{x} : h(\mathbf{x}) = 1\}$.

Equation 4.12 is particularly easy to estimate from finite samples. Following [Ben-David et al., 2010], we may rewrite $P_S(I(h))$ and $P_T(I(h))$ using finite samples S and T,

$$d_{\mathcal{H}}(\hat{P}_S, \hat{P}_T) = 2\left(1 - \underset{h \in \mathcal{H}}{\mathrm{minimize}} \frac{1}{n}\left(\sum_{\mathbf{x}:h(\mathbf{x})=0} I(\mathbf{x} \in S) + \sum_{\mathbf{x}:h(\mathbf{x})=1} I(\mathbf{x} \in T)\right)\right). \tag{4.13}$$

As stated in [Ben-David et al., 2010], Equation 4.13 directly leads to a procedure for calculating the $\mathcal{H}$-distance: (i) Solve the binary optimization problem with distinguishing S (labeled as 0) from T (labeled as 1), hence finding $h \in \mathcal{H}$ that minimizes the right-hand-side of Equation 4.13, (ii) Calculate the accuracy ϵ of h. Then, $d_{\mathcal{H}}(\hat{P}_S, \hat{P}_T) = 2(1 - \epsilon)$. Notice that the $\mathcal{H}-$distance depends on the hypothesis class $\mathcal{H}$. Indeed, the aforementioned process for estimating $d_{\mathcal{H}}$ involves fitting a classifier to the data. This optimization problem depends on $\mathcal{H}$. In Figure 4.5 we show a comparative of the $\mathcal{H}-$distance for two different classes of hypothesis class. Moreover, Figure 4.5 shows an important issue with the $\mathcal{H}-$divergence: a low divergence value may not reflect close domains. In the example, by using $\mathcal{H}$ as the set of separating hyperplanes the discrepancy between the blue and red domain is close to 1.0.

Finally, we consider the MMD distance, which was originally proposed for checking if two sets of observations come from the same probability distribution [Borgwardt et al., 2006]. As follows, the MMD distance is defined using Reproducing Kernel Hilbert Space (RKHS) formalism [Saitoh and Sawano, 2016].

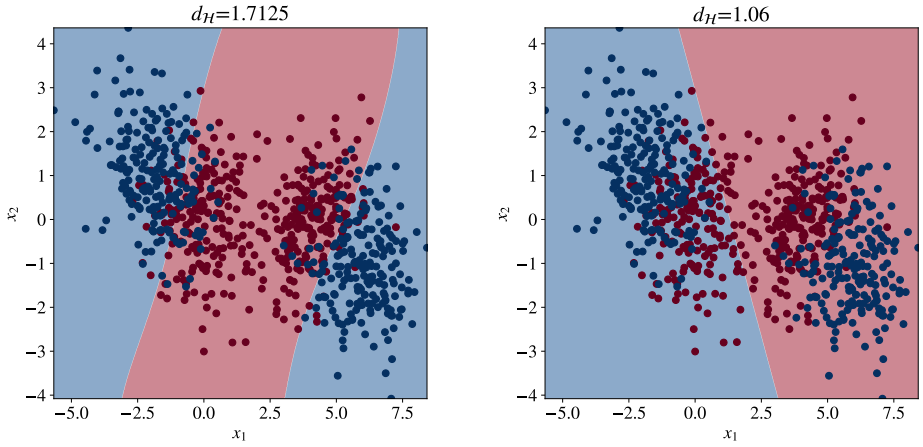


Figure 4.5: On the left, the $\mathcal{H}-$distance is calculated using $\mathcal{H}$ as the set of radial basis functions. On the right, the hypothesis class corresponds to the class of separating hyperplanes. Each domain is represented by a color. The areas represent the classifier's prediction to the respective color. Note that using the radial basis function kernel, the classifier manages to find the difference between the two domains, resulting in a high $\mathcal{H}-$distance.

Definition 4.10. (Maximum Mean Discrepancy [Gretton et al., 2007]) Let $\mathcal{H}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$ and let P_S and P_T be probability distributions on $\mathcal{X}$. The MMD between P_S and P_T is defined as,

$$\text{MMD}(P_S, P_T) = \sup_{f \in \mathcal{H}: ||f||_{\mathcal{H}}} \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[f(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[f(\mathbf{y})]. \tag{4.14}$$

Note that Equation 4.15, as defined, cannot be easily rewritten in terms of empirical distributions. The next lemma allows us to rewrite it in a more convenient form,

Lemma 4.1. *(Maximum Mean Discrepancy) With the same assumptions as those made in Definition 4.10, the MMD between P_S and P_T may be rewritten as,*

$$MMD = ||\mathop{E}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})] - \mathop{E}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})]||_{\mathcal{H}}.. \tag{4.15}$$

Proof. Available in Section D.3. □

As follows, Equation 4.15 further allows us to approximate the MMD distance from finite samples. By considering empirical approximations of the expectation, one has:

$$\text{MMD}(\hat{P}_S, \hat{P}_T) = \left|\left| \frac{1}{n_S} \sum_{i=1}^{n_S} \phi(\mathbf{x}_S^i) - \frac{1}{n_T} \sum_{i=1}^{n_T} \phi(\mathbf{x}_T^i) \right|\right|_{\mathcal{H}}$$

In addition, the next example draws a connection between the source of distribution shift with the calculation of the aforementioned distances.

Example 4.3. (Characterization of parameter mismatch in terms of distributional shift) In this example we will consider data sampled from a simulation environment with various parameter settings. Specifically, the default parameters are those introduced in Example 2.1, namely, $A_1 = A_2 = 0.1$ and $a_{12} = a_{13} = a_{23} = 0.1$. Thus 500 data points are sampled using these parameters for simulation, resulting in the support of the source distribution $\hat{\mu}$.

Moreover, let $\epsilon \in \{0.1, \cdots, 1.0\}$ be a perturbation in the previous parameters (e.g. $\epsilon = 0.3$ results in $A_1 = A_2 = 1.3$ and $a_{12} = a_{13} = a_{23} = 0.4$). This leads to target distributions $\hat{P}_T^{\epsilon}$ with an increasing degree of mismatch. For each ϵ, 500 data points are sampled. In Figure 4.6, the Wasserstein distance (left), the $\mathcal{H}-$distance (center) and the MMD (right) are calculated for each value of ϵ.

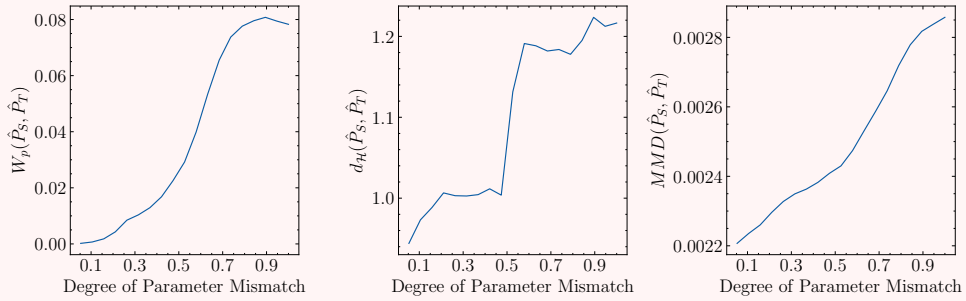


Figure 4.6: Statistical divergences for a variable degree of parameter mismatch. For the Wasserstein and MMD distances, an increase in the degree of mismatch implies an increase in the distance between empirical distributions. Moreover, $d_{\mathcal{H}}$ is noisy when compared to the other two divergences, as its values depends on the fit to the data.

4.4 How To Transfer

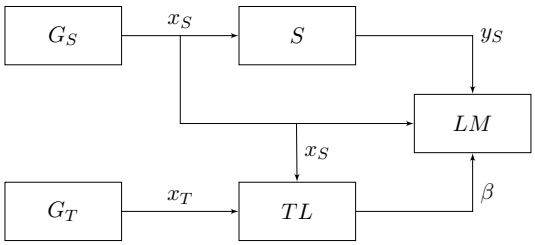
Following the approach discussed by [Pan and Yang, 2009], we now present the three different knowledge components used in transfer learning. These are: (i) instance-transfer, (ii) feature representation-transfer, and (iii) optimal transport-based transfer. In addition, considering Figure 3.7, when dealing with transfer learning problems one introduces two new elements: target samples generator, G_T, and a transfer learner TL. Note that G_T is distinguished from the source generator by denoting the latter as G_S.

Instance-Based Transfer

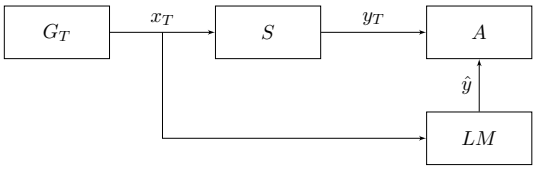
This category of transfer learning considers that the divergence between $P_S(X, Y)$ and $P_T(X, Y)$ can be reduced by giving weights β to instances in the source domain $\mathcal{D}_S$, so that the new distribution on this sub-set is similar to P_T. The theoretical setting for this problem is given by Figure 4.7,

As discussed in [Yang et al., 2020], this approach is motivated by the bias-variance trade-off. Indeed, consider data coming from two different distributions, with very few target domain samples. If a classifier is fit solely on target samples, there is a large variance due to the small number of samples. As source data is used when fitting the classifier, it becomes biased towards the source domain. By picking only data points in the source that follow a similar distribution to those in the target domain reduces the bias.

It remains the question on how to pick points in the source domain. This is done by assigning wights to these points, a procedure which is called importance estimation. Theoretically, consider



(1) Training phase



(2) Evaluation phase

Figure 4.7: Theoretical setting for the instance-based transfer case. Note that here a distinction is made upon the way samples are generated. During the learning phase, the unlabeled samples x_T are used to estimate the importance β of source samples x_S. During the evaluation phase, the samples x_T are labeled, yielding y_T, and compared to the LM predictions. Note that if the labels y_T are not available, it is impossible to assess how LM performs on the target samples.

the risk functional defined by Equation 3.8. The risk of a hypothesis h on the target domain is given by,

$$R_T(h) = \sum_{y=1}^{K} \int_{\mathcal{X}} \ell(h(\mathbf{x}), y) P_T(\mathbf{x}, y) d\mathbf{x}.$$

Notice that $P_T(\mathbf{x}, y) = P_T(\mathbf{x})P_T(y|\mathbf{x})$. As follows, we may introduce the factor $P_S(\mathbf{x}, y)/P_S(\mathbf{x}, y)$ without changing the equation,

$$\begin{aligned}
R_T(h) &= \sum_{y=1}^{K} \int_{\mathcal{X}} \ell(h(\mathbf{x}), y) P_T(\mathbf{x}, y) \frac{P_S(\mathbf{x}, y)}{P_S(\mathbf{x}, y)} d\mathbf{x}, \\
&= \sum_{y=1}^{K} \int_{\mathcal{X}} \ell(h(\mathbf{x}), y) P_S(\mathbf{x}, y) \frac{P_T(\mathbf{x}) P_T(y|\mathbf{x})}{P_S(\mathbf{x}) P_S(y|\mathbf{x})} d\mathbf{x}, \\
&= \sum_{y=1}^{K} \int_{\mathcal{X}} \ell(h(\mathbf{x}), y) P_S(\mathbf{x}, y) \frac{P_T(\mathbf{x})}{P_S(\mathbf{x})} d\mathbf{x}, \\
&= \sum_{y=1}^{K} \int_{\mathcal{X}} \ell(h(\mathbf{x}), y) P_S(\mathbf{x}, y) \beta(\mathbf{x}) d\mathbf{x}
\end{aligned}$$

where between the second-to-last and last equations the covariate shift hypothesis was used. The factor $\beta(\mathbf{x}) = P_T(\mathbf{x})/P_S(\mathbf{x})$ is the importance of $\mathbf{x}$, and can be used to weight the source risk towards the target risk. In terms of empirical risk, the weighting reads as,

$$\hat{R}_T(h) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(\mathbf{x}_T^i), y_T^i) \beta_i, \tag{4.16}$$

for $\beta_i = \beta(\mathbf{x}_i)$. This procedure has as additional hypothesis to covariate shift, the fact that $P_T(\mathbf{x}) \neq 0$ whenever $P_S(\mathbf{x}) \neq 0$. In the following discussion, three methods for the estimation of β are presented: Kullback-Leibler Importance Estimation Procedure (KLIEP) [Sugiyama et al., 2008], Least Squares Importance Fitting (LSIF) [Kanamori et al., 2009], and Kernel Mean Matching (KMM) [Gretton et al., 2009].

Kullback-Leibler Importance Estimation

Given the difficulties involved with the previous methods, KLIEP proposes to estimate β directly, using a linear model,

$$\beta(\mathbf{x}) = \sum_{i=1}^{b} \alpha_i \phi_i(\mathbf{x}),$$

where $\phi_i : \mathbb{R}^d \to \mathbb{R}$ are called basis functions and chosen beforehand. The estimation of α_i, as proposed by [Sugiyama et al., 2008] is done through the minimization of the Kullback-Leibler divergence between P_T and the weighted source distribution $P_S^\beta(\mathbf{x}) = \beta(\mathbf{x})P_S(\mathbf{x})$. The derivation can be found in Appendix B. The optimization algorithm defined by [Sugiyama et al., 2008] is,

$$\begin{aligned} \underset{\{\alpha_j\}_{j=1}^b}{\text{maximize}} \quad & \frac{1}{n_T} \sum_{i=1}^{n_T} \log\left(\sum_{j=1}^{b} \alpha_j \phi_j(\mathbf{x}_T^i) \right) \\ \text{subject to} \quad & \sum_{i=1}^{n_S} \sum_{j=1}^{b} \alpha_j \phi_j(\mathbf{x}_S^i) = n_S \\ & \alpha_j \geq 0, \forall j \end{aligned} \tag{4.17}$$

As follows, once α_i for $i = 1, \cdots, b$ is defined, one may retrieve the sample weights β_j, $j = 1, \cdots, n_S$ by querying the linear model $\beta_j = \sum_{i=1}^{b} \alpha_i \phi_i(\mathbf{x}_j)$. Note that the problem 4.17 is a convex optimization problem. As [Sugiyama et al., 2008] suggests, one may simply perform gradient descent with constraint satisfaction.

Kernel Mean Matching

This algorithm was proposed by [Gretton et al., 2009], and is inspired on finding β such that the MMD distance between P_S^β and P_T is minimized. As follows, the MMD between these probability distributions can be expressed as,

$$\text{MMD}(P_S^\beta, P_T)^2 = \left|\left| \frac{1}{n_S} \sum_{i=1}^{n_S} \beta(\mathbf{x}_S^i)\phi(\mathbf{x}_S^i) - \frac{1}{n_T} \sum_{i=1}^{n_T} \phi(\mathbf{x}_T^i) \right|\right|_{\mathcal{H}}^2.$$

Furthermore, as shown in [Gretton et al., 2009], the optimization problem that one gets by following their derivation is a quadratic problem similar to SVM,

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2}\beta^T K \beta - \kappa^T \beta \\ \text{subject to} \quad & |\sum_{i=1}^{n_S} \beta_i - n_S| \leq B\sqrt{n_S} \\ & \beta_i \in [0, B], \forall j \end{aligned}$$

A complete derivation of this optimization problem can be found in Appendix B. This algorithm has as hyper-parameters the kernel function choice $K_{ij} = k(\mathbf{x}_S^i, \mathbf{x}_T^j)$, and the bound B for the weights.

Least Squares Importance Fitting

As KLIEP, the LSIF procedure assumes a linear model for the weights β. The purpose of this method is estimating the parameters α using least squares strategy.

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^b}{\text{minimize}} \quad & \frac{1}{2}\alpha^T \mathbf{H} \alpha - \mathbf{h}^T \alpha + \lambda \mathbb{1}_b^T \alpha, \\ \text{subject to} \quad & \alpha_i \geq 0 \end{aligned}$$

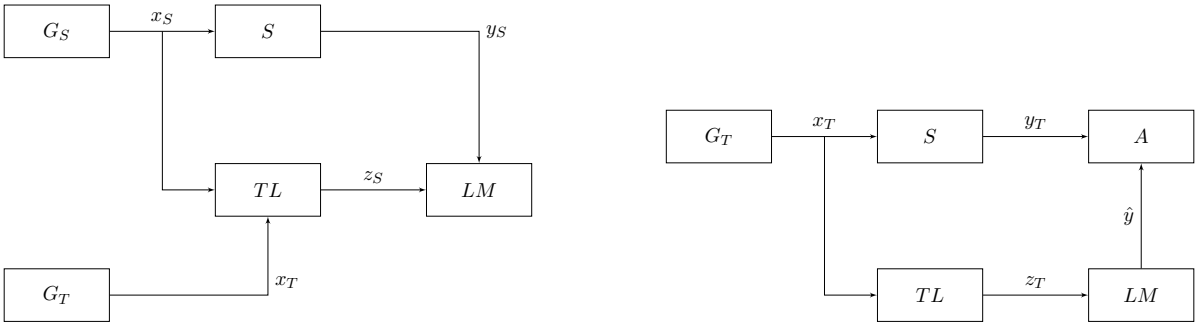
This is a quadratic optimization problem with convex constraints, hence, it may be solved with optimization methods similar to those employed for KLIEP or KMM. In addition, [Kanamori et al., 2009] also proposes an unconstrained variant called unconstrained Least Squares Importance Fitting (uLSIF). This method consists on solving the following optimization problem,

$$\underset{\beta \in \mathbb{R}^b}{\text{minimize}} \ \frac{1}{2}\alpha^T \mathbf{H}\alpha - \mathbf{h}^T\alpha + \frac{\lambda}{2}\alpha^T\alpha, \tag{4.18}$$

whose solution can be found analytically as $\alpha^* = (\mathbf{H} + \lambda I)^{-1}\mathbf{h}$. To guarantee that $\alpha_i \geq 0$ for all indices i, [Kanamori et al., 2009] further proposes the projection operation $\alpha_i^* = \max(0, \alpha_i^*)$. Again, once one has determined α, the weights β_i may be retrieved by querying the linear model.

Feature-based Transfer

This category of transfer learning considers the construction of a mapping $g : \mathcal{X} \to \mathcal{Z}$, from data, which transforms samples from the original feature space into a second space $\mathcal{Z}$, also called latent space. Thus, as shown in Figure 4.8, TL transform source and target features into latent representations z_S and z_T, respectively.



(1) Training phase (2) Evaluation phase

Figure 4.8: Theoretical setting for the feature-based transfer case. In this case, TL is used to transform feature vectors x_S and x_T into representations z_S and z_T respectively.

The goal of feature-based transfer learning is that samples coming from the source and target, upon the transformation, are indistinguishable from each other with respect to their distributions. To do so, statistical divergence measures, such as the MMD, are minimized. Three approaches fall under this previous description:

- Transfer Component Analysis (TCA) [Pan et al., 2010], which constructs g through the minimization of the MMD distance between probability distributions P_S and P_T.
- Geodesic Flow Kernel (GFK) [Gong et al., 2012], which constructs a flow between the source and target sub-spaces through geodesics in a Grassmanian manifold. The original feature space is then projected into the (infinitely many) sub-spaces.
- Domain-Adversarial Neural Network (DANN), which is a neural network constituted by a feature extractor g, composed with a class classifier f_c and a domain classifier f_d. The neural network is trained in a adversarial fashion, namely, source samples are used to fit $g \circ f_c$, while artificially labeled samples $x = [x_S, x_T]$, $\tilde{y} = [0, 1]$ are used to fit $g \circ f_d$.

Transfer Component Analysis

As KMM, the TCA algorithm is based on the minimization of the MMD divergence. However, instead of reweighting P_S, they seek to reduce the distance in distribution of $\phi(\mathbf{X}_S)$ and $\phi(\mathbf{X}_T)$.

Notice that, as before, we may use the kernel trick to express the MMD as,

$$\begin{aligned}\text{MMD}(\hat{P}_S, \hat{P}_T) &= \left\| \frac{1}{n_S}\sum_{i=1}^{n_S}\phi(\mathbf{x}_S^i) - \frac{1}{n_T}\sum_{i=1}^{n_T}\phi(\mathbf{x}_T^i) \right\|_{\mathcal{H}}^2, \\ &= \left\langle \frac{1}{n_S}\sum_{i=1}^{n_S}\beta_i\phi(\mathbf{X}_S^i) - \frac{1}{n_T}\sum_{i=1}^{n_T}\phi(\mathbf{x}_T^i), \frac{1}{n_S}\sum_{i=1}^{n_S}\beta_i\phi(\mathbf{X}_S^i) - \frac{1}{n_T}\sum_{i=1}^{n_T}\phi(\mathbf{x}_T^i) \right\rangle_{\mathcal{H}}, \\ &= \frac{1}{n_S^2}\sum_{i,j}K_{SS}(i,j) - \frac{2}{n_S n_T}\sum_{i,j}K_{ST}(i,j) + \frac{1}{n_T^2}\sum_{i,j}K_{TT}(i,j).\end{aligned}$$

Using this fact, [Pan et al., 2010] proposes matching distributions $\hat{P}_S$ and $\hat{P}_T$ in a latent sub-space. This is done by introducing a linear transformation through the matrix $\mathbf{W} \in \mathbb{R}^{(n_S+n_T)\times d'}$, for a dimensionality value $d' \ll d$, where d is the number of features in the original space. In this sense, the TCA optimization problem is defined as,

$$\begin{aligned}\underset{\mathbf{W}}{\text{minimize}} \quad & \text{Tr}(\mathbf{W}^T\mathbf{W}) + \mu\text{Tr}(\mathbf{W}^T\mathbf{KLKW}) \\ \text{subject to} \quad & \mathbf{W}^T\mathbf{KHKW} = \mathbf{I}_n\end{aligned} \tag{4.19}$$

where $K \in \mathbb{R}^{(n_S+n_T)^2}$ is the kernel matrix containing entries $K_{SS}(i,j) = k(\mathbf{x}_S^i, \mathbf{x}_S^j)$, $K_{TT}(i,j) = k(\mathbf{x}_T^i, \mathbf{x}_T^j)$, $K_{ST}(i,j) = k(\mathbf{x}_S^i, \mathbf{x}_T^j)$ arranged properly, $\mathbf{L}$ is the scaling matrix and $\mathbf{H}$ is the centering matrix, and μ a penalty term. A complete derivation of the latter equation, with further details on the aforementioned matrices is given in the Appendix B. The minimization of Equation 4.19 is done by performing the eigenvalue decomposition of $(\mathbf{I}_n + \lambda\mathbf{KLK})^{-1}\mathbf{KHK}$. The matrix $\mathbf{W}$ is the matrix whose columns are the m eigenvectors associated with the largest m eigenvalues. As stated in [Matasci et al., 2011], once $\mathbf{W}$ has been calculated, the samples may be transformed using $\tilde{\mathbf{X}} = \mathbf{KW}$.

Geodesic Flow Kernel

As TCA, this algorithm assumes that there exists low-dimensional sub-spaces in the source and target domains that share common characteristics. GFK considers all sub-spaces of a given dimension d' as candidates. These sub-spaces form a manifold, called the Grassmanian $\mathbb{G}(d', d)$, where d is the dimensionality of the original space. Therefore, GFK builds a flow $\mathbf{\Phi} : [0,1] \to \mathbb{G}(d', d)$, where for each $t \in [0,1]$, $\mathbf{\Phi}(t)$ as a sub-space on the Grassmanian.

In addition, GFK assumes that initial sub-spaces S_S and S_T are extracted which is done by either applying PCA or Partial Least Squares (PLS). Especially, let $\mathbf{B}_S \in \mathbb{R}^{d\times d'}$ and $\mathbf{B}_T \in \mathbb{R}^{d\times d'}$ be the basis of S_S and S_T, respectively. The flow $\mathbf{\Phi}$ is such that $\mathbf{\Phi}(0) = \mathbf{B}_S$ and $\mathbf{\Phi}(1) = \mathbf{B}_T$. Moreover, notice that for any $t \in [0,1]$, $\mathbf{\Phi}(t)$ forms the basis of a given sub-space in $\mathbb{G}(d, D)$. Let S_t be such sub-space. Note that we may project any vector $\mathbf{x} \in \mathbb{R}^D$ into S_t by using $\mathbf{z} = \mathbf{\Phi}(t)^T\mathbf{x}$.

The principle of GFK is to project $\mathbf{x}$ into $\mathbf{\Phi}(t)$, for t varying continuously from 0 to 1. This generates a vector with infinitely many dimensions, $\mathbf{z}^\infty$. In practice, $\mathbf{z}^\infty$ does not need to be computed. Indeed, let $\mathbf{x}_i$ and $\mathbf{x}_j$ be vectors in $\mathbb{R}^d$, with respective projection $\mathbf{z}_i^\infty$ and $\mathbf{z}_j^\infty$. The inner product of the projections can be expressed as,

$$\langle \mathbf{z}_i^\infty, \mathbf{z}_j^\infty \rangle = \int_0^1 (\mathbf{\Phi}(t)^T\mathbf{x}_i)^T\mathbf{\Phi}(t)^T\mathbf{x}_i dt = \mathbf{x}_i\mathbf{G}\mathbf{x}_j, \tag{4.20}$$

where $\mathbf{G} = \int_0^1 \mathbf{\Phi}(t)\mathbf{\Phi}(t)^T dt$. As remarked by [Gong et al., 2012], this is a step similar to the Kernel Trick. As follows the integral involved in the calculation of $\mathbf{G}$ can be analytically computed, and is dependent on the construction of the geodesic flow $\mathbf{\Phi}(t)$. In Appendix B we discuss the complete derivation of $\mathbf{G}$, as presented by [Gong et al., 2012]. For now, let us assume that $\mathbf{G}$ has been calculated. For performing the adaptation step, one may substitute the kernel matrix $\mathbf{K}$, in the SVM algorithm, by $\mathbf{G}$.

Domain Adversarial Training of Neural Networks

The DANN was proposed by [Ganin et al., 2016] as a way to train DNNs in cases of unsupervised domain adaptation. Let g denote the feature extractor and f_c the classifier in a CNN. The authors have propose to include another classifier f_d, sharing the feature extractor g, for distinguishing to which domain a sample belongs.

Recalling the definition of $\mathcal{H}-$distance, the feature distributions P_S and P_T is zero if and only if a classifier cannot distinguish the source from target samples. Hence, the domain-adversarial training of neural networks consists on a mini-max game between the class classifier and the domain classifier. Let θ_g (resp. θ_c, θ_d) correspond to the parameters of the feature-extractor (resp. class and domain classifiers). The mini-max game can be expressed as,

$$(\hat{\theta}_g, \hat{\theta}_c) = \underset{\theta_g, \theta_c}{\operatorname{argmin}} \frac{1}{n_S} \sum_{i=1}^{n_S} \ell_c^i(\theta_g, \theta_c) - \lambda \left(\frac{1}{n_S} \sum_{i=1}^{n_S} \ell_d^i(\theta_g, \hat{\theta}_d) + \frac{1}{n_T} \sum_{i=n_S+1}^{n_S+n_T} \ell_d^i(\theta_g, \hat{\theta}_d) \right), \tag{4.21}$$

$$\hat{\theta}_d = \underset{\theta_d}{\operatorname{argmax}} \frac{1}{n_S} \sum_{i=1}^{n_S} \ell_c^i(\hat{\theta}_g, \hat{\theta}_c) - \lambda \left(\frac{1}{n_S} \sum_{i=1}^{n_S} \ell_d^i(\hat{\theta}_g, \theta_d) + \frac{1}{n_T} \sum_{i=n_S+1}^{n_S+n_T} \ell_d^i(\hat{\theta}_g, \theta_d) \right), \tag{4.22}$$

where in the last equation ℓ_c^i is a shorthand for the loss incurred with the class prediction $\hat{y}_i$ and ground-truth y_i. The term ℓ_d^i is defined in the same way, for the domain prediction. The optimization procedure defined by the relations in Equations 4.21 and 4.22 can be implemented using automatic differentiation software, such as Tensorflow[1], with the gradient updates defined in [Ganin et al., 2016].

Optimal Transport-based Transfer

The class of methods based on optimal transport, whose theory was covered early on this chapter, is constituted by algorithms that estimate a mapping $\phi : \mathcal{X}_s \to \mathcal{X}_t$. In light of the covariate hypothesis, the mapping T is supposed to preserve the predictive distribution,

$$P_S(Y|X) = P_T(Y|\phi(X)),$$

moreover, one expects that through the application of T to X, one has $P_S(X) \approx P_T(\phi(X))$. In the following discussion, we will analyze three algorithms for transfer learning: OTDA, mapping estimation and Joint Distribution Optimal Transport (JDOT).

Optimal Transport for Domain Adaptation

The standard OTDA approach consists on estimating the optimal transport plan γ through the linear program posed by Equation 4.2, or more efficiently through the Sinkhorn algorithm, which minimizes Equation 4.5. In either case, the algorithm outputs a matrix $\gamma \in \mathbb{R}^{n_S \times n_T}$, for n_S, the number of samples in the source domain, and n_T, the number of samples in the target domain. The elements γ_{ij} of the transportation matrix can be seen as the amount of mass that is transferred from sample $\mathbf{x}_S^i$ to $\mathbf{x}_T^j$.

Following [Courty et al., 2016], once γ has been estimated, one may transport the samples from μ_S to μ_T through interpolation. Let $\alpha \in [0, 1]$, then the interpolation of μ between the two distributions is given by the geodesic on the Wasserstein space,

$$\mu_\alpha = \underset{\mu \in \mathcal{P}_p(\mathcal{X})}{\operatorname{argmin}} \ (1-\alpha) W_p(\mu_S, \mu)^p + \alpha W_p(\mu_T, \mu)^p.$$

[1] https://www.tensorflow.org/

The solution to this interpolation problem is a probability measure μ supported on $n_s + n_t - 1$ points [Peyré et al., 2019]. Particularly, for $p = 2$ one has a simple closed formula for μ_α,

$$\mu_\alpha = \sum_{i=1}^{n_S} \sum_{j=1}^{n_T} \gamma_{ij} \delta_{(1-\alpha)\mathbf{x}_S^i + \alpha \mathbf{x}_T^j}.$$

In the context of OTDA one rather wants the transport into the target domain, hence α is set to 1. This results in the empirical measure $\hat{\mu} = \sum_{j=1}^{n_t} \omega_j \delta_{\mathbf{x}_j^t}$, for $\omega_j = \sum_{i=1}^{n_s} \gamma_{ij}$. Thus, the weight assigned to each $\mathbf{x}_j^t$ is the sum of probability mass coming from the samples $\mathbf{x}_i^s$. Moreover, one can devise a transformation ϕ_γ on the source samples using this previous discussion,

$$T_\gamma(\mathbf{x}_S^i) = \underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \sum_j \gamma_{ij} c(\mathbf{x}, \mathbf{x}_T^j). \tag{4.23}$$

This transformation is known in the literature as barycentric mapping [Courty et al., 2016]. In general one estimates $\hat{\mathbf{X}}_S = [T_\gamma(\mathbf{x}_S^1), \cdots, T_\gamma(\mathbf{x}_S^{n_S})]$ rather than T_γ directly. This estimation can have a closed-formula, such as when the ground cost c is based on the Euclidean distance. In this case $\hat{\mathbf{X}}_S = n_S \gamma \mathbf{X}_T$, or one may employ gradient-based optimization, provided that c is differentiable.

In addition to this discussion, note that in classification problems γ is desirably class sparse. This means that whenever $\mathbf{x}_S^i$ and $\mathbf{x}_T^j$ belong to different classes, the mass γ_{ij} that is transferred between these samples should be zero. With this insight in mind, [Courty et al., 2016] proposed to add another regularization term, concerning class sparsity. In this work we will explore the following regularization strategy,

$$\Omega_c(\gamma) = \sum_{j=1}^{n_T} \sum_{k=1}^{K} ||\gamma(\mathcal{I}_k, j)||_q^p,$$

this is known as $\ell_p - \ell_q$ regularizer. In the last equation, note that $\gamma(\mathcal{I}_k, j)$ is a vector. In this notation, $\mathcal{I}_k$ contains the indices of all source samples who belong to class k. Thus the vector $\gamma(\mathcal{I}_k, j)$ has the one each coordinate the mass transferred from samples of class k, to the target sample j. In addition, its norm gives a measure of how much mass is transported to class k to the sample $\mathbf{x}_T^j$. Thus, Ω_c induces class sparsity. In addition, two settings for p and q are used in [Courty et al., 2016]: $p = 1$, $q = 2$, which is referred as the group-lasso, and $p = 0.5$ and $q = 1$, which is referred as $\ell_p - \ell_1$ regularizer.

Mapping Estimation

The barycentric mapping suffers from a major drawback: for new samples $\mathbf{x}_S^{i'}$, a new transportation matrix γ' from $\mathbf{X}_S'$ to $\mathbf{X}_T$, in order to transform the new source samples. Moreover, the original Monge problem does not always have solution for empirical measures. In face of these problems, [Perrot et al., 2016] proposed an approach for estimating γ and T jointly, through the following minimization problem,

$$(\gamma, T) = \underset{\gamma \in \Pi(\mu_S, \mu_T), T \in \mathcal{H}}{\text{argmin}} \frac{1}{n_S d_T} ||T(\mathbf{X}_S) - n_S \gamma \mathbf{X}_T||_2^2 + \frac{\lambda_\gamma}{max(C)} \langle \gamma, C \rangle_F + \frac{\lambda_T}{d_S d_T} R(T). \tag{4.24}$$

where $\mathcal{H}$ is a family of transformations, λ_γ and λ_T are weighting parameters determining the trade-off between the minimization with respect to γ, and with respect to T. Additionally, to solve Equation 4.24 one needs to define the family of transformations $\mathcal{H}$. Two families were suggested in [Perrot et al., 2016], a family of linear transforms,

$$\mathcal{L} = \{T : \exists \mathbf{L} \in \mathbb{R}^{d_S \times d_T}, \forall \mathbf{x}^S \in \mathcal{X}_S, T(\mathbf{x}_S) = (\mathbf{x}_S)^T \mathbf{L}\},$$

Algorithm 2 Mapping Estimation Optimal Transport

Input: Source and target samples $\mathbf{X}_S, \mathbf{X}_T$, and hyper-parameters $\lambda_\gamma, \lambda_T$.
Initialize $k = 0$, $\gamma^0 \in \Pi(\hat{\mu}_S, \hat{\mu}_T)$ and $\mathbf{L}^0 = \mathbf{I}$
while Not converged **do**
$\gamma^{k+1} \leftarrow$ solution to 4.24 for fixed $\mathbf{L}^k$ using the Frank-Wolfe Algorithm [Jaggi, 2013].
$\mathbf{L}^{k+1} \leftarrow$ solution to 4.24 for fixed γ^{k+1}, using Equations 4.25 or 4.26.
$k \leftarrow k + 1$
end while
Output: $\mathbf{L}, \gamma$.

and a family of kernelized linear transformations. In this case, let ϕ be a non-linear function with associated kernel $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$. Moreover, let $k_{\mathbf{X}_S}(\mathbf{x}^\mathbf{S}) = [k(\mathbf{x}_S, \mathbf{x}_S^i)]_{i=1}^N$ be a vector, then we may write the family of transformations as,

$$\mathcal{K} = \{T : \exists L \in \mathbb{R}^{n_S \times d_T}, \forall \mathbf{x}_S \in \mathcal{X}_S, T(\mathbf{x}_S) = k_{X_S}(\mathbf{x}_S)^T \mathbf{L}\}.$$

With these ingredients, [Perrot et al., 2016] further proposed to use the Block Coordinate Descent (BCD) algorithm [Tseng, 2001] for solving Equation 4.24. Using BCD, one divides the minimization step in two parts: (1) minimizing Equation 4.24 for fixed T, and (2) for fixed γ. This first minimization step consists on using the Frank-Wolfe algorithm [Jaggi, 2013] for finding the optimal γ, while the second consists on finding $\mathbf{L}$ that minimizes the previous equation. Such matrix has closed form for each family [Perrot et al., 2016],

$$\mathcal{L} \rightarrow \mathbf{L} = \left(\frac{1}{n_S d_T} \mathbf{X}_S^T \mathbf{X}_S + \frac{\lambda_T}{d_S d_T} \mathbf{I} \right)^{-1} \left(\frac{1}{d_T} \mathbf{X}_S^T \gamma \mathbf{X}_S + \frac{\lambda_T}{d_S d_T} \mathbf{I} \right), \tag{4.25}$$

$$\mathcal{K} \rightarrow \mathbf{L} = \left(\frac{1}{n_S d_T} K_{SS} + \frac{\lambda_T}{d^2} \mathbf{I} \right)^{-1} \frac{n_S \gamma \mathbf{X}_T}{n_S d_T}. \tag{4.26}$$

Therefore, the Mapping Estimation algorithm, as presented by [Perrot et al., 2016], is summarized in Algorithm 2.

Joint Distribution Optimal Transport

A major assumption for the application of OTDA is the covariate shift hypothesis. That is, one supposes A condition upon the marginals, $P_S(X) \neq P_T(X)$, and another upon the conditional, $P_S(Y|X) = P_T(Y|X)$. However, this is not always true. In that spirit [Courty et al., 2017] has proposed an alternative formulation using the joint distribution $P(X, Y)$.

The principle is to suppose $P_S(X, Y) \neq P_T(X, Y)$, without further assumptions on $P(X)$ nor $P(Y|X)$. Therefore, let $\mu_S = P_S(X, Y)$ (resp. μ_T and $P_T(X, Y)$). With $\hat{\mu}_S = \frac{1}{n_S} \sum_{i=1}^{n_S} \delta_{\mathbf{x}_S^i, y_S^i}$ (resp. $\hat{\mu}_T$), we may formulate the Kantorovich problem in terms of these distributions,

$$\gamma^* = \underset{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)}{\operatorname{argmin}} \sum_{i,j} D(\mathbf{x}_S^i, y_S^i; \mathbf{x}_T^j, y_T^j) \gamma_{ij}$$

where $D(\mathbf{x}_i^s, y_i^s; \mathbf{x}_j^t, y_j^t) = \alpha d(\mathbf{x}_i^s, \mathbf{x}_j^t) + \ell(y_i^t, y_j^t)$ is a loss function over features and labels. In [Courty et al., 2017], the authors have considered different choices for ℓ, such as,

$$\ell_{hinge}(y, f(\mathbf{x})) = \max(0, y f(\mathbf{x})),$$

$$\ell_{MSE}(y, f(\mathbf{x})) = \frac{1}{K} \sum_{i=1}^{K} (y_i - f(\mathbf{x})_i)^2.$$

In our experiments we will further consider the usage of the cross-entropy loss $\ell_{entropy}$ as defined in Equation 3.17. Notice, however, that this formulation assumes that the labels y_j^t

are available. Since this is not true for unsupervised domain adaptation, [Courty et al., 2017] proceeds into considering the following surrogate distribution, exploiting the predictive function f,

$$\mu_T^f = \frac{1}{n_T}\sum_{j=1}^{n_t}\delta_{\mathbf{x}_T^j, f(\mathbf{x}_T^j)}.$$

As follows, JDOT proposes an optimization problem over the coupling γ, and the predictive function f,

$$(f,\gamma) = \underset{f\in\mathcal{H}_\ell,\gamma\in\Pi(\hat{\mu}_S,\hat{\mu}_T^f)}{\operatorname{argmin}} \sum_{i,j}(\alpha d(\mathbf{x}_S^i,\mathbf{x}_T^j) + \ell(y_S^i, f(\mathbf{x}_T^j)))\gamma_{ij} + \lambda\Omega(f). \tag{4.27}$$

As the previous method, [Courty et al., 2017] proposes the usage of BCD algorithm for solving the optimization posed by problem 4.27. Especially, one performs two independent steps: solving for γ, with f fixed,

$$\gamma^{k+1} = \underset{\gamma\in\Pi(\hat{\mu}_S,\hat{\mu}_T^{f^k})}{\operatorname{argmin}} \sum_{i,j} C_{ij}^k\gamma_{ij}, \tag{4.28}$$

using $C_{ij} = \alpha d(\mathbf{x}_i^s,\mathbf{x}_j^t) + \ell(y_i^s, f^k(\mathbf{x}_j^t))$, and (2) Solving for f with γ fixed. The first is step is done, as previously mentioned, through the Frank-Wolfe algorithm [Jaggi, 2013], while the second amounts to solving,

$$f^{k+1} = \underset{f\in\mathcal{H}_\ell}{\operatorname{argmin}} \sum_{i,j}\gamma_{ij}^{k+1}\ell(y_S^i, f(\mathbf{x}_T^j)) + \lambda\Omega(f). \tag{4.29}$$

Thus, the BCD iterations may be summarized in algorithm 3.

Algorithm 3 BCD iterations for the JDOT algorithm

Input: Source and target samples $\mathbf{X}_S, \mathbf{X}_T$.
  Initialize $k = 0$, $\gamma^0 \in \Pi(\hat{\mu}_S, \hat{\mu}_T^f)$,
  while Not converged **do**
    $\gamma^{k+1} \leftarrow$ solution to 4.28 for fixed f^k using the Frank-Wolfe algorithm [Jaggi, 2013].
    $f^{k+1} \leftarrow$ solution to 4.29 for fixed γ^{k+1},
    $k \leftarrow k+1$
  end while
Output: $\mathbf{L}, \gamma$.

4.5 When to Transfer

Theoretical Guarantees of Domain Adaptation Algorithms

To understand the implications of having a source domain S, and a target domain T from which the data is generated, we will use extensively the notion of risk of a hypothesis. An important property of the probability of disagreement was stated in [Crammer et al., 2008], which is the triangle inequality,

Lemma 4.2. *(Triangle Inequality for Hypothesis Risk) Let h_1, h_2 and h_3 be three hypothesis in $\mathcal{H}$. Then,*

$$\mathcal{R}_S(h_1,h_3) \leq \mathcal{R}_S(h_1,h_2) + \mathcal{R}_S(h_2,h_3). \tag{4.30}$$

Proof. Available in Section D.3. □

Since now $\mathbf{x}$ comes from two domains, let f be a fixed labeling function. For a given hypothesis $h \in \mathcal{H}$, h may disagree with f for samples coming from P_S and from P_T. As follows, the concept of ideal joint hypothesis will be useful,

Definition 4.11. (Ideal Joint Hypothesis [Ben-David et al., 2010]) Given two domains $\mathcal{D}_S = (\mathcal{X}, P_S)$, $\mathcal{D}_T = (\mathcal{X}, P_T)$, and a labeling function f, the ideal joint hypothesis is defined as,

$$h^* = \underset{h \in \mathcal{H}}{argmin}\ \mathcal{R}_S(h, f) + \mathcal{R}_T(h, f). \tag{4.31}$$

Moreover, the combined error is given by,

$$\lambda = \mathcal{R}_S(h^*) + \mathcal{R}_T(h^*). \tag{4.32}$$

As mentioned previously, most domain adaptation algorithms try to ease the burden of the distributional shift by minimizing the divergence between P_S and P_T. If, for a statistical divergence φ, one can bound the difference $\mathcal{R}_T(h, h') - \mathcal{R}_S(h, h')$, then minimizing $\varphi(P_S, P_T)$ yields risks that are closer in value. This motivates the following definition,

Definition 4.12. (Bounding Distances) Let $\mathcal{D}_S = (\mathcal{X}, P_S)$ and $\mathcal{D}_T = (\mathcal{X}, P_T)$ be two domains. A statistical divergence $\varphi : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ is said to bound $\mathcal{R}_T$ by $\mathcal{R}_S$ with constant $\alpha > 0$ if, $\forall h, h' \in \mathcal{H}$

$$\mathcal{R}_T(h, h') - \mathcal{R}_S(h, h') \leq \alpha\varphi(P_S, P_T). \tag{4.33}$$

With these definitions, an important theoretical result is relating the risk of a hypothesis on the target domain $\mathcal{R}_T(h)$, with the risk on the source domain $\mathcal{R}_S(h)$. As already mentioned, such bound has been extensively discussed on the literature, and it exists for various different types of statistical divergences. Indeed, various versions exists, depending on the problem context and on the chosen statistical divergence. Here, we present a slightly more general version,

Theorem 4.3. *(A bound for the target risk) Let $\mathcal{D}_S = (\mathcal{X}, P_S)$ and $\mathcal{D}_T = (\mathcal{X}, P_T)$ be two domains. Moreover, let $\mathcal{R}_S$ and $\mathcal{R}_T$ be bounded by φ with constant α, and let λ be the combined error defined by Equation 4.32. Then, for any hypothesis $h \in \mathcal{H}$,*

$$\mathcal{R}_T(h) \leq \mathcal{R}_S(h) + \alpha\varphi(P_S, P_T) + \lambda$$

Proof. Available in Section D.3. □

Therefore, to justify theoretically that a divergence-minimization strategy can improve a model's performance on target domain is equivalent to prove that such divergence fits Definition 4.12. The next theorem is based on the results of [Ben-David et al., 2010] and [Redko et al., 2017], and claims that the three distances used in this work bound the target risk by the source risk.

Theorem 4.4. *(Bounding Distances) The following distances between probability distributions fit into Definition 4.12: (i) the $\mathcal{H}-$distance, (ii) the Wasserstein distance, and (iii) the MMD distance.*

Proof. Available in Section D.3. □

Thus, the theoretical justification of domain adaptation algorithms is as follows. First, one proposes some distance φ between probability distributions and an algorithm for its minimization. Second, one may show that φ fits into Definition 4.12. Third, it must be shown that $\varphi(P_S, P_T)$ can be accurately estimated from finite samples.

Negative Transfer

Negative transfer refers to the case where transfer learning deteriorates the performance of the learning machine. This problem has long received attention from the domain adaptation community, but only recently it has been treated formally. The starting point is the formal definition of a transfer learner,

Definition 4.13. (Adapted from [David et al., 2010]) A Transfer Learner (TL) is a function,

$$\text{TL} : \bigcup_{n_S=1}^{\infty} \bigcup_{n_T=1}^{\infty} (\mathcal{X} \times \{0,1\})^{n_S} \times \mathcal{X}^{n_T} \to \{0,1\}^{\mathcal{X}}$$

The previous definition assumed, without loss of generality, a binary classification problem. Moreover, it is formulated in the context of unsupervised domain adaptation. Intuitively, TL takes labeled data from the source domain and unlabeled data from the target domain, and outputs a hypothesis h_{TL}. This definition allows he characterization of the Unsupervised Negative Transfer Gap (UNTG),

Definition 4.14. (Unsupervised Negative Transfer Gap) Let $S = \{\mathbf{x}_S^i, y_S^i\}_{i=1}^{n_S}$ be a labeled sample from the source domain, and $T = \{\mathbf{x}_T^j\}_{j=1}^{n_T}$ be a unlabeled sample from the target domain. Let $h_{TL} = TL(S,T)$, and $h_S = LM(S)$. For a given **unsupervised** transfer learner TL, the UNTG is given by,

$$\text{UNTG(TL)} = \mathcal{R}_T(h_{TL}) - \mathcal{R}_T(h_S).$$

The previous definition characterizes the UNTG as the difference in terms of target domain risk, of the hypothesis learned by the transfer learner h_{TL}, and the hypothesis learned by the learning machine h_S, using solely source domain data. Note that Definition 4.14 is slightly adapted from [Wang et al., 2019], who have defined the negative transfer gap as,

$$\text{NTG}(TL) = \mathcal{R}_T(h_{TL}) - \mathcal{R}_T(h_T),$$

where h_T is the hypothesis learned by the learning machine using solely target data. Note that in unsupervised domain adaptation this gap is overly pessimistic, and contradicts the objectives

of domain adaptation as set in Definition 4.6, that is, to help learning a predictive function h on the target domain, using source domain knowledge. Using the work of [David et al., 2010], the authors have discussed the necessity and sufficiency of the following hypothesis in the context of instance-based transfer learning,

H1 The training and target distributions are close with respect to the $\mathcal{H}\Delta\mathcal{H}-$distance,

H2 There exists a hypothesis in $\mathcal{H}$ that has low error on both domains,

H3 The covariate shift assumption.

The conclusion of [David et al., 2010] is that neither H1+H3 nor H2+H3 suffices to guarantee successful transfer. Nonetheless, [Ben-David et al., 2007] previously shown that H1+H2 imply positive transfer. In addition to the results of [David et al., 2010], Theorem4.4 imply that the $\mathcal{H}-$divergence may be substituted for the Wasserstein or the MMD distance in H1. The next example shows a comparison on the minimization of the Wasserstein distance between P_S nd P_T in two scenarios: low and high λ_H.

Example 4.4. (Negative Transfer under OTDA) This example illustrates OTDA under low and high λ. Being so, consider labeled samples $S = \{\mathbf{x}_S^i, y_S^i\}_{i=1}^{n_S}$ and $T = \{\mathbf{x}_T^i, y_T^i\}_{i=1}^{n_T}$. Under this assumption the combined error may be directly calculated. Figure 4.9 presents an example for hich there is an hypothesis that has low risk on both domains.

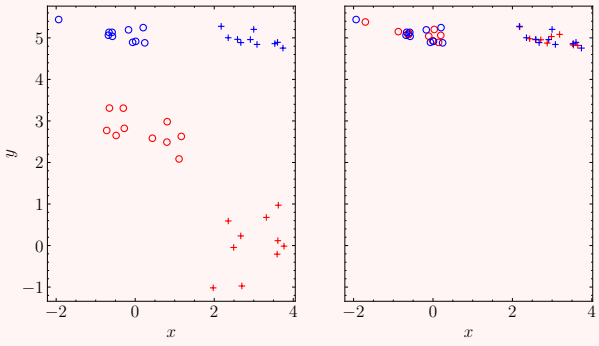


Figure 4.9: On the left, original domain adaptation problem, with source and target domains shown in red and blue, respectively. On the right, the source samples (black) are transported to locations in red, next to the target domain in blue. The circles correspond to class 1, while the crosses correspond to class 2.

Note that by fitting a classifier (black thick line in Figure 4.9) with data coming from the source domain yields a high risk on the target domain ($R_T(h_S) = 0.5$), while there exists an hypothesis h^*, shown in Figure 4.9 as a dotted black line, that can classify both domains $R_T(h^*) = 0.0$ and $R_S(h^*) = 0.0$). Hence $\lambda = R_T(h^*) + R_S(h^*) = 0.0$.

In this case optimal transport works well, as can be seen in the right part of Figure 4.9. Especially, the hypothesis discovered once the data is transported is close to h^*. In contrast with this case, Figure 4.10 presents a negative transfer case, especially remarking that λ for his problem is high.

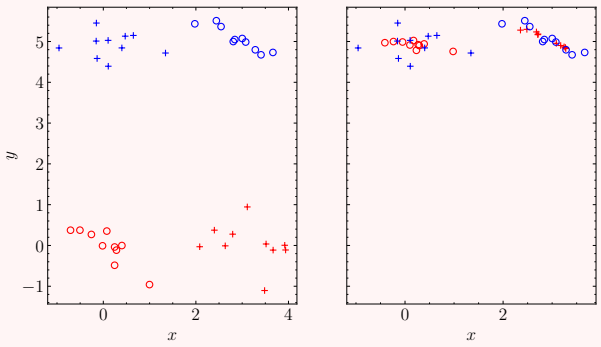


Figure 4.10: On the left, original domain adaptation problem. On the right, the optimal transport result. The otation for this figure is similar to that of Figure 4.9.

Note that, since the hypothesis space is linear, there is no h for which the joint risk is 0. Even though, on each domain the classes are separable, which means that we may find h_S and h_T such that $\mathcal{R}_S(h_S) = \mathcal{R}_T(h_T) = 0$. In the best case, however, $\lambda = 0.5$. In addition note that the optimal transport solution mix the two classes, exhibiting a severe case of negative transfer. To further illustrate the issue, Figure 4.11 shows a comparison between the transport plans for the previous two cases.

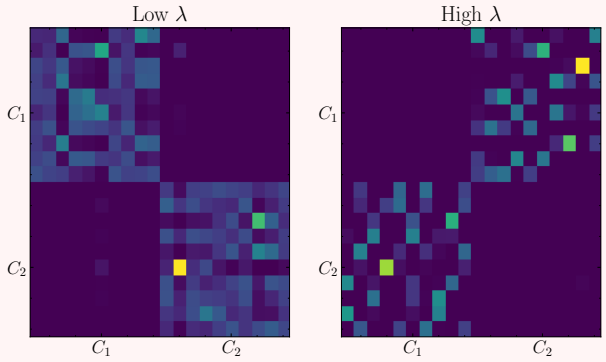


Figure 4.11: On the left, transport plan γ fit to the data in Figure 4.9. On the right, ransport plan γ fit to the data in Figure 4.10.

Note that the transport plan on the left of Figure 4.11 correctly matches points within the same classes, with few exceptions. On the other hand, when λ is high the transport plan matches data points from opposing classes. As consequence, the marginals $P_S(X)$ and $P_T(X)$ are indeed matched, that is, $W_1(\hat{P}_S, \hat{P}_T)$ is minimized, but λ grows to its maximum value.

In the previous example notice that, on the right part of Figure 4.11, γ transports mass from opposing classes. As highlighted by [Courty et al., 2016], this is an undesirable characteristic that may appear when applying OT for domain adaptation. Thus, we propose an index of Undesired Mass Flow (UMF), that measures the flow of mass between different classes.

Definition 4.15. (Undesired Mass Transfer) Suppose that $S = \{\mathbf{x}_S^i, y_S^i\}$ and $T = \{\mathbf{x}_T^j, y_T^j\}$ are labeled samples corresponding to the source and target domains. For classes k_1 and k_2, the mass

flow from class k_1 to k_2 upon γ, is defined as,

$$\Phi_{k_1,k_2}(\gamma) = \sum_{i:y_S^i} \sum_{j:y_T^j} \gamma_{ij}. \tag{4.34}$$

Thus, the UMF index is defined by the amount of mass transferred between different classes, namely,

$$\text{UMF}(\gamma) = \sum_{k_1 \neq k_2} \Phi_{k_1,k_2}(\gamma). \tag{4.35}$$

Note that the previous definition can help measuring the quality of a transport plan γ, but it cannot be used for learning better plans in unsupervised domain adaptation. This is mainly due the fact that UMF(γ) needs labeled samples in the target domain. Nonetheless, if a semi-supervised scenario is assumed, that is, if few samples in the target domain are available, then the UMF index can be used for penalizing transport plans that transfer mass between different classes. In this case, the UMF term is very similar to the semi-supervised penalty proposed by [Courty et al., 2016].

Chapter 5

Results and Discussion

This chapter seeks to apply the theoretical methodology presented so far for the analysis of a benchmark system in fault diagnosis: the Continuous Stirred Tank Reactor (CSTR). As follows, in Section 5.1 we perform the dynamical analysis for the system, following the concepts of Chapter 2. We then proceed to sampling and extracting features from the described simulation environment, as presented in Chapter 3. Moreover, Sections 5.2 through 5.4 are concerned with our experiments concerning cross-domain fault diagnosis.

5.1 Case Study: Continuous Stirred Tank Reactor

The CSTR is a widely used benchmark for FDD in the context of chemical plants. It has been defined by numerous studies in different settings for its configurations and faults. In general, a continuously stirred tank with temperature T carries an exothermic reaction $A \rightarrow B$, for which C_i refers to the concentration of A in the inlet, that has temperature T_i. The tank is supposed to be jacketed. Inside the jacket flows a coolant with flow-rate Q_c and temperature T_{ci} that exchanges heat with the tank. Furthermore, the concentration of B in the outlet is denoted by C, and temperature T. The jacket has an outlet of coolant, with temperature T_c. This description is shown in Figure 5.1.

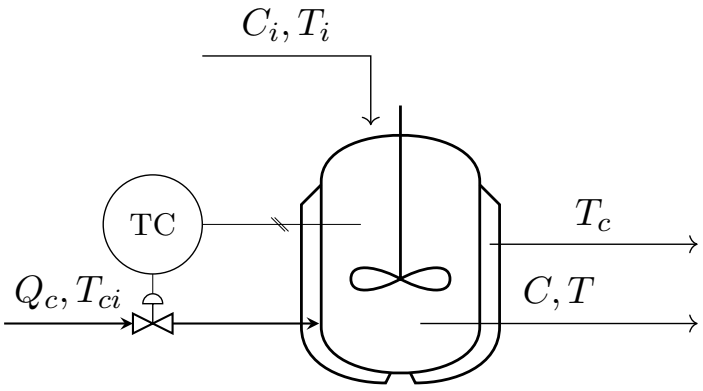


Figure 5.1: Figure reproduced from [Li et al., 2020] representing the closed-loop CSTR system. Notice that the flow-rate Q_c is controlled by the temperature measurements.

Concerning the settings existing in the literature, [Choi et al., 2005] models the system dynamics in terms of state variables C and T, whereas [Juricek et al., 2004], [Pilario and Cao, 2018] and [Li et al., 2020] use T_c as a state variable as well. The fault scenarios are different as well. For instance, [Choi et al., 2005] only considers sensor biases, whereas the other works consider more complex fault such as catalyst decay, heat fouling and even disturbances, such as

in [Li et al., 2020]. Concerning our approach, we adopt the same as [Li et al., 2020], which was inspired by [Pilario and Cao, 2018].

Dynamical Analysis

Following [Pilario and Cao, 2018], the CSTR system is modeled through a set of ODEs,

$$\begin{cases} \dot{C} &= \frac{Q}{V}(C_i - C) - akC^N + \nu_1, \\ \dot{T} &= \frac{Q}{V}(T_i - T) - a\frac{\Delta H_r k C^N}{\rho C_p} - b\frac{UA}{\rho C_p V}(T - T_c) + \nu_2, \\ \dot{T}_c &= \frac{Q_c}{V_c}(T_{ci} - T_c) + b\frac{UA}{\rho_c C_{pc} V_c}(T - T_c) + \nu_3 \end{cases} \tag{5.1}$$

where $\nu_k, k = 1 \cdots, 3$ is the process noise, sampled from $\mathcal{N}(0, 10^{-2})$. Following the diagram in Figure 5.1, there are seven variables involved with the CSTR system: the temperatures T_{ci}, T_c, T_i and T, the concentrations C_i and C, and the flow-rate Q_c. The subscript c indicate that the variable is associated with the coolant (e.g. T_c, the coolant outlet temperature). Whenever the subscript i is present, the variable is associated with inlet (e.g. T_{ci}, the coolant inlet temperature). Moreover, for purposes of fault diagnosis all process variables are measured, namely $y_m = [C_i, T_i, T_{ci}, Q_c, C, T, T_c]^T \in \mathbb{R}^7$. We thus assume a measurement noise $\eta_j \sim \mathcal{N}(0, 10^{-2})$ $j = 1, \cdots, 7$. A summary of each variable involved in this latter equation may be found in Table 5.1.

Type	Symbol	Description	Units	Value
Input	C_i	Concentration of A in the inlet.	mol/L	0.1
	T_i	Temperature of the inlet	K	350
	T_{ci}	Temperature of the coolant inlet	K	350
	Q_c	Inlet flow-rate	L/min	-
State Variable	C	Concentration of B in the outlet	mol/L	-
	T	Temperature of the outlet	K	-
	T_c	Temperature of the coolant outlet	K	-
Output	T	Temperature of the outlet	K	-
Parameter	Q	Inlet flow-rate	L/min	100
	V	Tank volume	L	150
	V_c	Jacket volume	L	10
	ΔH_r	Reaction heat	cal/mol	-2×10^{-5}
	UA	Heat transfer coefficient	K.cal/min	7×10^5
	k_0	Pre-exponential factor of k	min^{-1}	7.2×10^{10}
	E/R	Activation energy over gas constant	K	10^4
	ρ, ρ_c	Fluid density	g/L	1000
	C_p, C_{pc}	Fluid heat capacity	K.cal/g	1
	N	Reaction order	-	1

Table 5.1: Description of parameters, inputs and variables in the CSTR system.

Furthermore the dynamical system represented by Equation 5.1 is acquired by using mass and energy conservation. These two laws are based on first principles, as stated in Equation 2.1. As follows, for simplicity let us omit the process noise ν. Each term in the aforementioned system of equation is related to a physical quantity. A complete derivation of these equations is

available in Section D.4.

$$\dot{C} = \underbrace{\frac{QC_i}{V}}_{\text{Inlet flow of mass}} - \underbrace{\frac{QC}{V}}_{\text{Outlet flow of mass}} - \underbrace{akC^N}_{\text{Mass consumed}},$$

$$\dot{T} = \underbrace{\frac{QT_i}{V}}_{\text{Inlet flow of energy}} - \underbrace{\frac{QT}{V}}_{\text{Outlet flow of energy}} - \underbrace{a\frac{\Delta H_r k C^N}{\rho C_p}}_{\text{Reaction heat}} - \underbrace{b\frac{UA}{\rho C_p V}(T - T_c)}_{\text{Heat Exchanged}},$$

$$\dot{T}_c = \underbrace{\frac{Q_c T_{ci}}{V_c}}_{\text{Inlet flow of Energy}} - \underbrace{\frac{Q_c T_c}{V_c}}_{\text{Outlet flow of Energy}} + \underbrace{b\frac{UA}{\rho_c C_{pc} V_c}(T - T_c)}_{\text{Heat Exchange}}$$

Following [Pilario and Cao, 2018], to investigate the effect of Q_c on the operation of the CSTR system we simulate Equations 5.1 for flow-rates ranging from 20 to 200 L/min. Note that outside this range, the flow-rate is assumed to saturate. As expected, since the reactor carries an exothermic reaction, when there is not enough flow-rate the temperature is expected to rise. This is shown in Figure 5.2,

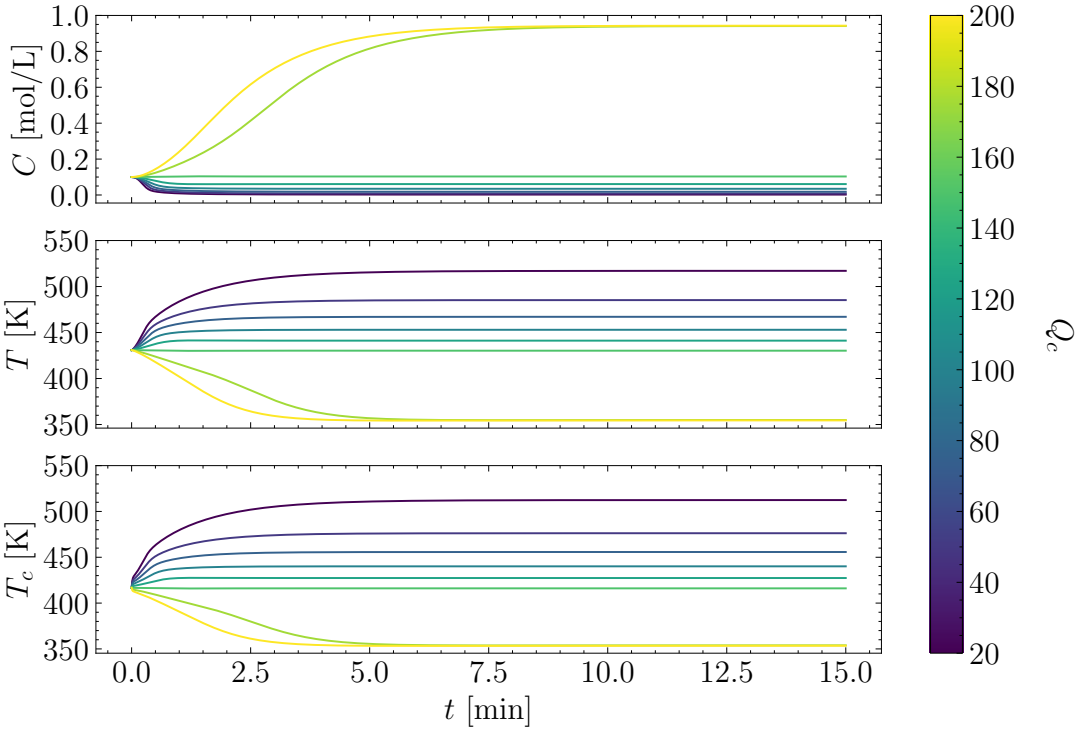


Figure 5.2: Dynamic of state variables for Q_c ranging from 20 to 200 L/min.

The analysis of Figure 5.2 shows three different scenarios: (i) for values of Q_c under 148.55 L/min, the system converges to a high temperature and low concentration scenario (as $t \to \infty$, $Q_c \to 20$, and $C \to 0$). In addition, as Q_c shrinks, the temperatures rise progressively. (ii) for $Q_c \approx 148.55$ L/min, the system remains at the initial condition. As we will see in the next section, this happens because the initial settings chosen with this particular value of Q_c is an equilibrium point. (iii) for Q_c higher than 148.55 L/min, the system converges to a low temperature and high concentration scenario, that is, the system converges to $T \approx T_c \approx 353.5K$ and $C = 0.943 mol/L$.

Finally, following this insight, [Pilario and Cao, 2018] proposes manipulating the coolant flow-rate Q_c for controlling the reactor temperature T. This is done by using a set-point $T_{sp} = 430.9$ K, and a proportional-integral controller with $K_p = 1.0$ and $K_i = 5.0$.

Fault Analysis

As mentioned before, various studies propose different fault settings for benchmarking tasks of detection and diagnosis. For instance [Choi et al., 2005] considered only sensor faults, categorized as sensor bias and shift, whereas [Juricek et al., 2004] considered sensor bias, process disturbance, catalyst deactivation and heat fouling. This choice of faults is very similar to that of [Li et al., 2020] which is the one adopted in this work. As follows, we present in Table 5.2 each category of fault considered.

Type	Fault Class	Model	Value of δ	Description
Process Faults	1	$a = a_0 exp(-\delta t)$	0.004	Catalyst decay
	2	$b = b_0 exp(-\delta t)$	0.005	Heat transfer fouling
Sensor Faults	3	$\tilde{C}_i = C_i + \delta t$	0.005	Sensor bias
	4	$\tilde{T}_i = T_i + \delta t$	0.1	Sensor bias
	5	$\tilde{T}_{ci} = T_{ci} + \delta t$	0.1	Sensor bias
	6	$\tilde{C} = C + \delta t$	0.005	Sensor bias
	7	$\tilde{T} = T + \delta t$	0.1	Sensor bias
	8	$\tilde{T}_c = T_c + \delta t$	0.1	Sensor bias
	9	$\tilde{Q}_c = Q_c + \delta t$	-0.2	Sensor bias
Process Disturbance	10	$\Delta C_i \sim \mathcal{N}(0, \delta)$	0.005	Concentration disturbance
	11	$\Delta T_i \sim \mathcal{N}(0, \delta)$	5	Reactant temperature disturbance
	12	$\Delta T_{ci} \sim \mathcal{N}(0, \delta)$	5	Coolant temperature disturbance

Table 5.2: The 12 distinct types of faults. Each fault is modeled according its description. Disturbances are sampled from Gaussian distributions, while sensor shift corresponds to a ramp.

We thus discuss each type of fault. First, we consider sensor faults. As seen in Table 5.2, this malfunction is described as sensor bias, in which a linear trend is added to the sensor measurements [Juricek et al., 2004]. Therefore, let us illustrate the sensor bias fault for C_i, as for the other process variables the description is the same. Thus, this fault is modeled mathematically as follows,

$$\tilde{C}_i(t) = C_i(t) + \delta t. \tag{5.2}$$

where δ to the fault parameter, in this case, the trend slope. The second type of fault we will explore is process disturbance. As considered in [Li et al., 2020], the inputs C_i, T_i and T_{ci} are perturbed by ΔC_i, ΔT_i, and ΔT_{ci}, respectively. Even though this case is not considered as a fault by [Pilario and Cao, 2018], [Li et al., 2020] used the same model in their simulation. Thus, for C_i (resp. T_i and T_{ci}),

$$\tilde{C}_i(t) = C_i(t) + \Delta C_i,$$

where ΔC_i (resp. T_i and T_{ci}) is sampled randomly from a Gaussian distribution at each 60 minutes [Pilario and Cao, 2018]. A comparison between the normal operation, sensor bias and process disturbance for the C_i variable is presented in Figure 5.3.

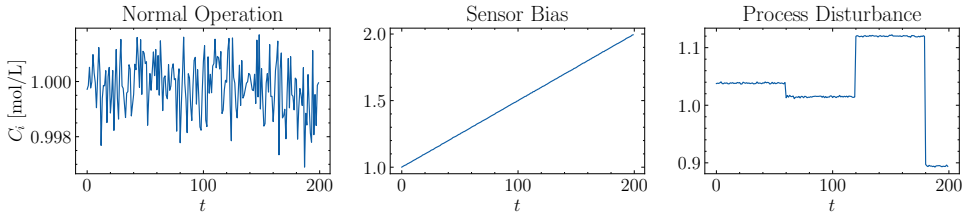


Figure 5.3: Comparison between normal operation, sensor shift, and disturbance faults on C_i.

Finally, the third group is composed by internal faults, and comprehends deviations in the process parameters, such as catalyst decay and heat transfer fouling. The former affects the availability of catalyst in the reaction, while the latter affects the heat transfer between the tank and the jacket. As shown in Table 5.2, we adopt an exponential decay model for the fault's associated parameters, namely a and b. Note that a affects the change in the concentration of A, thus also interfering in the heat generated by the reaction. On the other hand by decaying b one affects the heat exchange between the tank and the jacket. Note that in the extreme case where $a \to 0$, A is no more consumed, and no heat is exchanged between tank and jacket. The effect of these faults is shown in Figure 5.4.

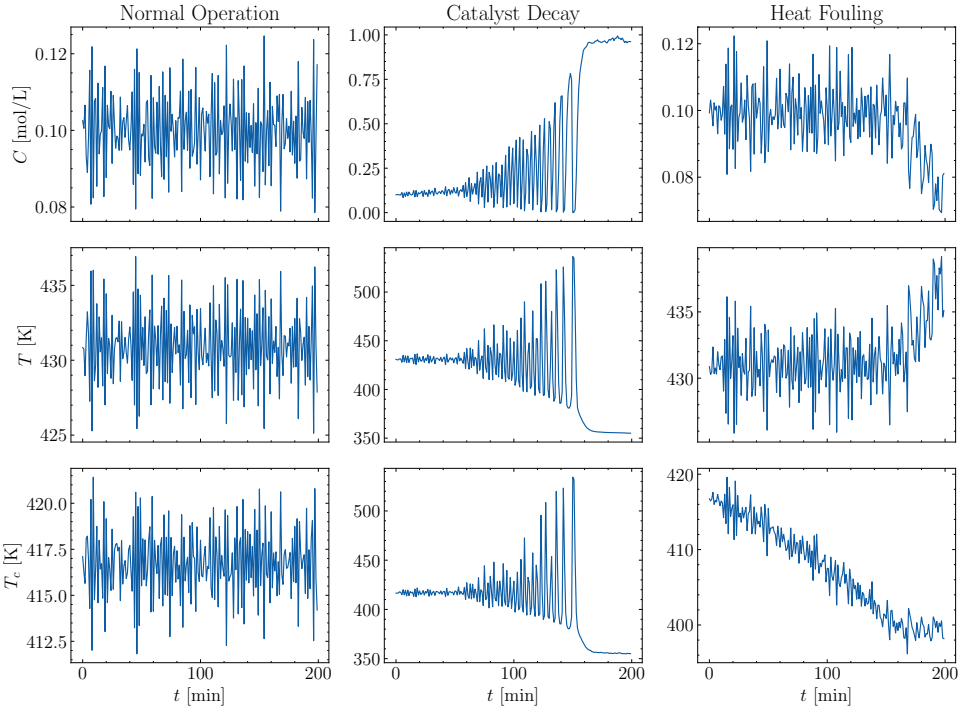


Figure 5.4: Comparison of the effect of catalyst decay and heat fouling faults with normal operation on the state variables.

Cross-Domain Sampling

The problematic of this work is related to the diagnostic of faults when data is sampled from different probability distributions. This, as defined in Chapter 1, corresponds to the cross-domain fault diagnosis problem. In this discussion we explore the motivations and methodology of [Li et al., 2020]. One of the main motivations for this line of work is models trained on simulation data for deployment in real plants. Furthermore, a common choice for emulating model-process mismatch is by introducing parameter errors. Adopting the notation introduced in Chapter 2, we will denote the set of parameters as Θ. Thus, $\Theta = [V, V_c, \Delta H_r, UA, k_0, E/R]$ is a vector of parameters.

Therefore, we describe the sampling methodology. First, the system is simulated 100 times using the parameters described in Table 5.1 for each fault in Table 5.2, corresponding to the source domain. This generates a data matrix $\mathbf{X}_S \in \mathbb{R}^{1300\times1400}$, where $n_S = 1300$ is the number of source domain samples, whereas 1400 corresponds to each process variable sampled at 200 time-steps, with $h = 1$ min. Moreover, we introduce an error vector $\Delta\Theta$ in the parameters. Each component is sampled independently using an uniform distribution according to a degree of

parameter mismatch ϵ, that is,

$$\Delta\Theta_i \sim \mathcal{U}(-\epsilon, \epsilon),$$
$$\tilde{\Theta}_i = \Theta_i + \Delta\Theta_i.$$

Thus, a target domain is defined by sampling data points $\mathbf{X}_T \in \mathbb{R}^{260\times1400}$ using $\tilde{\Theta}$ for different degrees ϵ. Note that $n_T = 260$ corresponds to 20 samples collected from each fault category. This procedure generates three different targets domains, with $\epsilon = 0.1, 0.15, 0.2$, thus having an increasing degree of mismatch. We also consider a second type of mismatch, characterized by a change in the system's reaction order N. For this latter setting, $\epsilon = 0.15$ is fixed, and $N = 0.5, 1.5$, and 2.0, thus resulting in three more target domains, thus six in total. Furthermore, Figure 5.5 shows how the mismatch influences the process dynamics. Note that changing the reaction order has a heavier impact on the dynamics than changing the parameters.

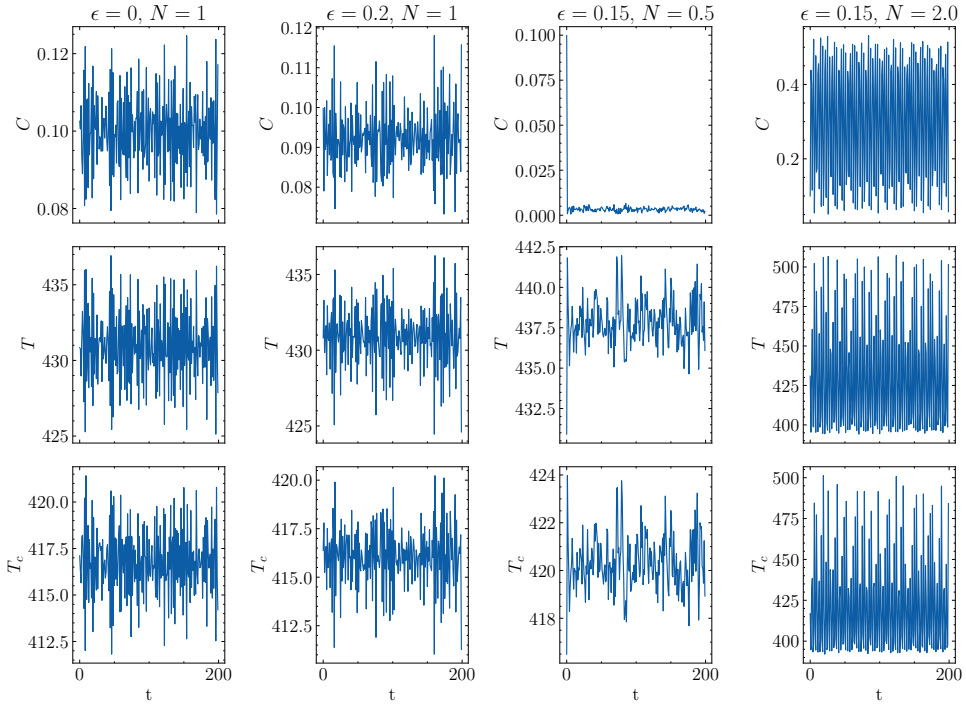


Figure 5.5: Change in the system's behavior for different degree of mismatch ϵ, and reaction order value N.

As follows, six experiments are carried, corresponding the adaptation of each source with a selected target. These experiments have a different degree of difficulty, corresponding to the degree upon which $\hat{P}_S$ differs from $\hat{P}_T$. To further illustrate this idea Figure 5.6 shows a comparison for the Wasserstein distance, as in Definition 4.4, between $\hat{P}_S$ and $\hat{P}_T$, the empirical probability distributions of source and target domains respectively. We present those distances as a function of the value's choice for ϵ and N. Note that when changing the reaction order, one has larger distributional shift, thus confirming that his parameter has more impact on the distributional shift phenomenon.

Feature Extraction and Classification

In this section the environment for the CSTR system simulation is described, as well as the features extracted from raw-data are analyzed. For the simulation, the plant of [Pilario and Cao, 2018] was used, with minor adjustments: (i) The simulation time was changed from 1200 minutes to 200, and (ii) All faults are assumed to happen at $t = 0$ min, instead of $t = 200$ min.

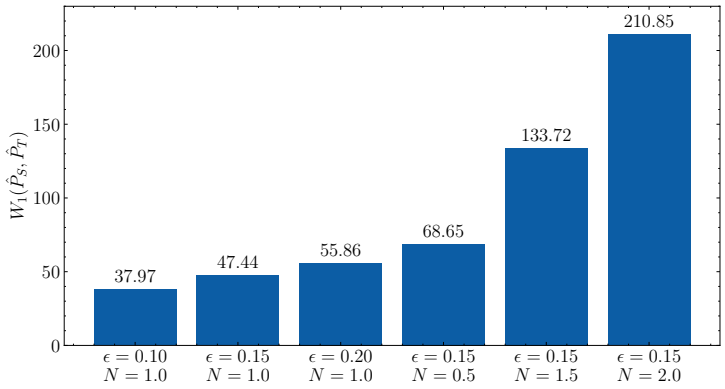


Figure 5.6: Wasserstein distance between source distribution ($\epsilon = 0.0$ and $N = 1.0$) and target distribution, for various values of degree of mismatch ϵ and reaction order N. Each combination of ϵ and N yields a different target domain. Note that the reaction order has a higher impact on the distributional shift.

As discussed in the previous section, raw signal data is collected, yielding samples of dimension $200 \times 7 = 1400$.

Concerning the feature extraction procedure, three kinds of features are analyzed: ACF, raw signals and CNN features. For the first type, we proceed as in Chapter 3 and calculate ACF each sample and process variable independently, for lags ranging from $\tau = 0$ through $\tau_{max} = 40$ time steps. This corresponds to considering 40 minutes of lag in the computation. Moreover, the final features are robust to the choice of τ_{max}. For constructing the final feature vector, the mean and variance of $R_{xx}(\tau)$ are used as features, where x corresponds to each one of the process variables. This yields a feature vector with 14-dimensions for classification. Hence, to visualize each three feature setting we employ the nonlinear dimensionality reduction algorithm t-SNE [Van der Maaten and Hinton, 2008], as done in Example 3.6. The visualization is shown in Figure 5.7.

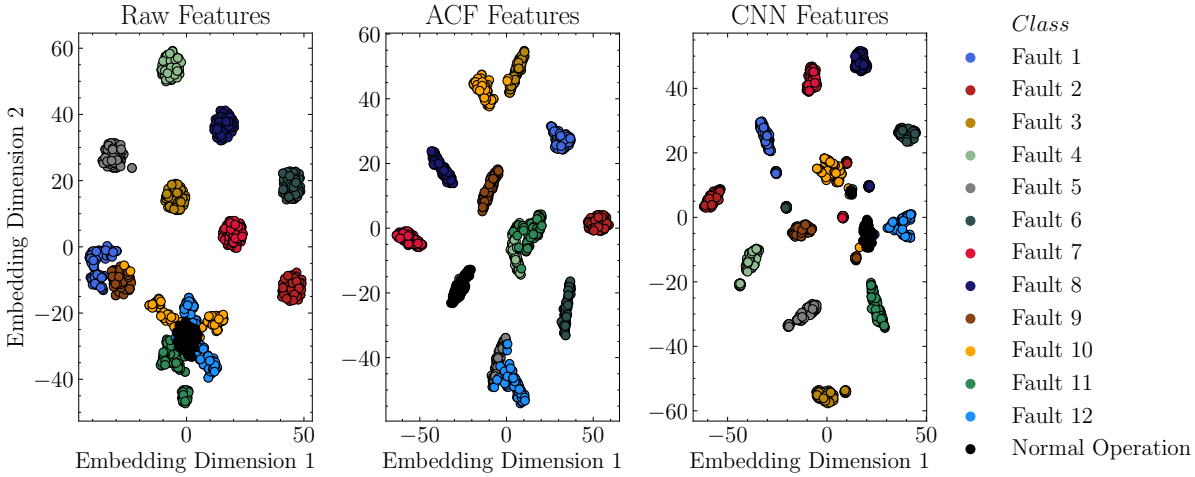


Figure 5.7: t-SNE embeddings for fault diagnosis data with raw, ACF, and CNN features. Note that for raw features faults 10, 11 and 12 and the normal operation are close together in the embedding space.

Once the features have been extracted, a classifier may be trained. To illustrate the performance on the source data, a linear SVM is trained on the ACF and raw features, and a CNN is trained on raw features. The CNN architecture is a modified version of that shown in Table 3.1. Especially, we change the input shape to $n_{batch} \times 1400 \times 1$, and the output shape to $n_{batch} \times 13$, following [Li et al., 2020].

The classification results are presented in Table 5.3, the respective classifiers are evaluated using 5-fold cross-validation through their classification performance or accuracy. For a classifier h, This metric is defined as,

$$\text{Accuracy}(h) = \frac{1}{N}\sum_{i=1}^{N} I[h(\mathbf{x}_i) = y_i],$$

where $I[\cdot]$ is the indicator function. With respect to the accuracy, The CNN classifier outperforms the SVM trained on ACF and raw features. Note that the CNN classifier is non-linear, as it has three dense layers, while the SVM is linear. Nonetheless the gap between ACF-based SVM and the CNN is very small, thus highlighting the potential of these features.

Features	Accuracy
ACF	99.615 ± 0.344
Raw	78.576 ± 1.299
CNN	99.800 ± 0.002

Table 5.3: Fault diagnosis performance on each set of features.

To further explore this result, we consider the confusion matrix M of the ground truth labels y with the predictions $\hat{y}$. The confusion matrix gives a deeper view on how the classifier is miss-classifying samples.. This is a 13×13 matrix whose i-th line corresponds to the number of samples having true label i, and the j-th column to the number of samples being predicted with class j. As consequence the element M_{ij} corresponds to the number of samples with class i, predicted as class j. Essentially, a 100% accuracy corresponds to M being a diagonal matrix. As follows, Figure 5.8 shows the confusion matrix for the SVM classifier trained on raw features. Note that the classifier is confusing classes 10, 11 and 12.

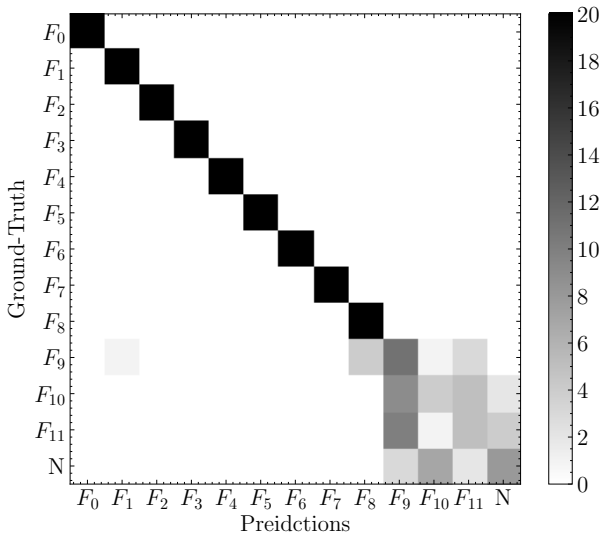


Figure 5.8: Confusion matrix for the SVM classifier trained on raw features.

5.2 Comparative Study of Transfer Learning Algorithms

We begin our discussion by detailing the process of model selection. We use a strategy similar 5-fold to a cross validation scheme. Especially, note that the source data is used integrally for training, while the target data is used for evaluation (testing), so these sets are independent.

During training, we assume that no label in the target domain is available. The adopted validation strategy is illustrated in Figure 5.9. As it suggests, by leaving one partition out at each iteration generates five different sets of data, each corresponding to an experiment. The assessment is made by training a classifier (linear SVM or neural network) on the available source data (4 out of 5 source folds), then evaluating it on the available target samples (4 out of 5 target folds). Note that overfitting is avoided since no label information is used during the transfer.

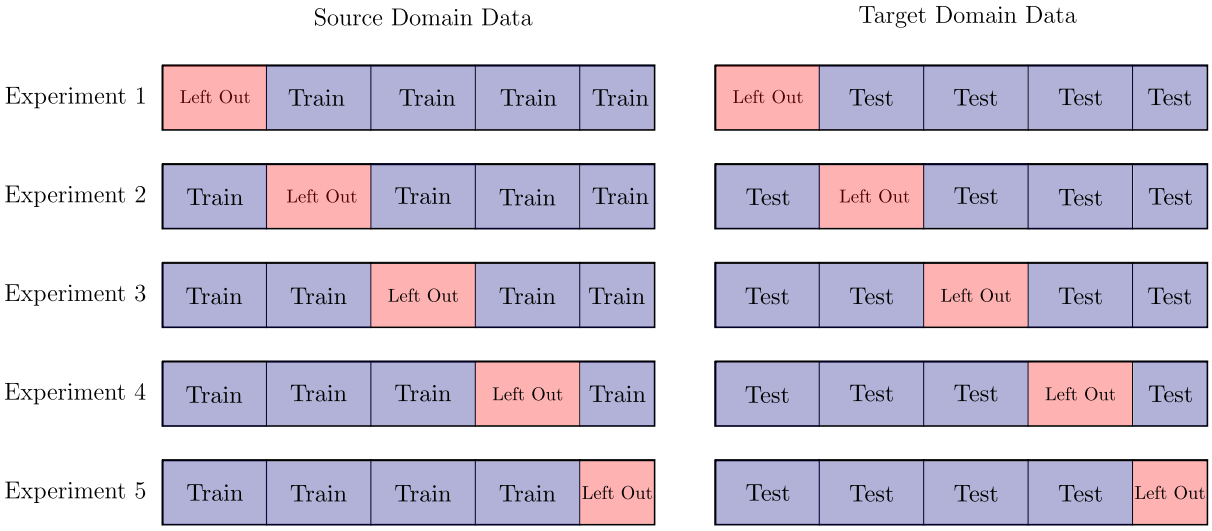


Figure 5.9: Illustration of the cross-validation procedure, carried out as 5 experiments with different sections of the dataset. Source domain data is used solely for training, while target domain data is used only for evaluation/transfer learning.

For the comparative study, each algorithm described in Chapter 4 is evaluated using the cross-validation procedure. All hyper-parameters are selected with respect to the average domain accuracy, here denoted as score. Let n_{T_m} denote the number of samples on the m-th target domain. The score that a given transfer learning algorithm has is given by,

$$\text{score} = \frac{1}{n_{domains}} \sum_{m=1}^{n_{domains}} \frac{1}{n_{T_m}} \sum_{i=1}^{n_{T_m}} I[h_{S,T_m}(\mathbf{x}_i) = f_m(\mathbf{x}_i)],$$

where h_{S,T_m} is the hypothesis learned by the transfer learner TL using source samples S and target samples T_m. In addition, f_m is the ground-truth labeling function of the m-th target domain.

Baseline Assessment

In this section we compare the performance of a SVM classifier on ACF and raw features with the performance of a CNN under distributional shift. No transfer learning algorithm is employed in this step.

Reaction Order	1.0			0.5	1.5	2.0	Score
Degree of Mismatch	10%	15%	20%	15%			
Raw Features	68.654 ± 0.769	64.519 ± 1.231	62.404 ± 1.154	56.923 ± 0.892	46.346 ± 1.709	32.981 ± 0.942	55.304
ACF Features	**97.692 ± 0.360**	**97.308 ± 0.385**	**92.019 ± 0.490**	**85.096 ± 3.145**	62.981 ± 0.912	59.808 ± 0.942	**82.484**
CNN Features	80.192 ± 3.001	75.288 ± 2.031	74.135 ± 3.815	62.500 ± 2.544	**69.808 ± 5.400**	**60.288 ± 5.830**	70.000

Table 5.4: Comparison of classification results without domain adaptation.

Note that using raw features yields poor results in terms of accuracy. This suggests that using a mathematical model for features (such as the ACF model), or employing a convolutional feature

extractor reduces the effect of distributional shift. In addition, the fact that the performance of ACF features is better than CNN features is a consequence of the diversity of data employed. Especially, CNNs are known for needing large amounts of diverse data to generalize well. Thus, since the sample size for the training set upon which the CNN is trained is of only 1300 samples, the network overfits this domain, not being able to generalize to the target domains.

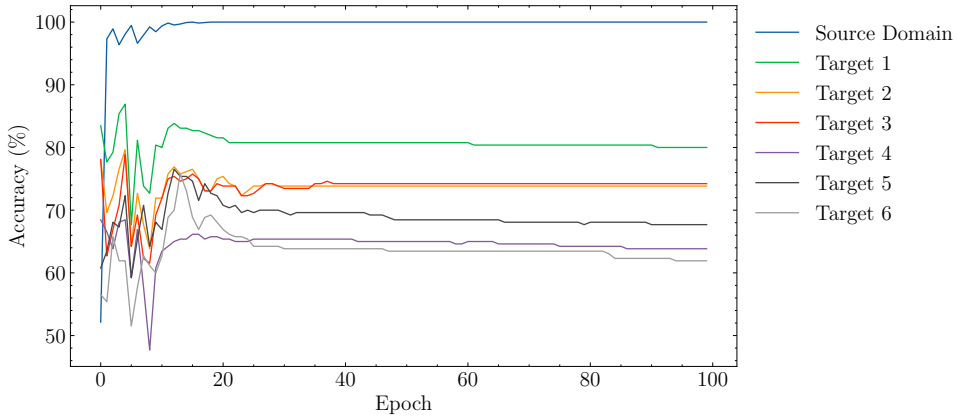


Figure 5.10: Learning curve for the CSTR fault diagnosis task.

To further justify the latter claim, Figure 5.10 shows the learning curve of the proposed CNN trained on source data only, and evaluated on each domain. Note that the data perfectly classifies the source domain, while slowly degrading the performance on the target domains, characterizing an overfitting scenario.

In addition, we compare the mistakes that the different classifiers are making by inspecting the full set of confusion matrix reported in . This is shown in Figures A.1 through A.3. Note that the SVM with raw features has trouble distinguishing between faults 9 through 12, and the normal operation. These faults correspond respectively to Q_c's sensor shift, and disturbances in C_i, T_i, and T_{ci}. This result is similar to that of Figure 5.8. Furthermore, by employing a CNN one manages the build a feature extractor more robust to distributional shift. This is supported by Table 5.4 and a comparison between the confusion matrices in Figures A.1 and A.3. Especially, the errors are less concentrated on faults 10, 11 and 12, yielding to better classification results.

Finally, the ACF results show that these features are very robust to changes in the probability distribution generating the data. For instance, for low mismatch levels ($\epsilon = 0.1$ and $N = 1$), only a few samples from fault 2 are mistaken for fault 1. This evidences the fact that having a mathematical modeling for extracting features can yield a better classifier in situations where the sample size is reduced.

Instance-Based Algorithms

For instance-based domain adaptation, three algorithms are evaluated: KMM, KLIEP and uLSIF. Note that since the weights β_i affect the parameter C on the SVM classifier, it is important to cross-validate C as well. Therefore, it is searched for all algorithms using SVM on the grid $[1, 15, 25]$. In addition, for all importance estimation algorithms we use a Gaussian kernel, which has as hyper-parameter the bandwidth γ. This parameter is searched on the grid $[10^{-3}, 10^{-2}, 10^{-1}]$, plus a scaling choice corresponding to $\gamma = \text{median}(D_{ij})^{-1}$, where D_{ij} correspond to the pairwise distances between source domain samples.

In general, instance-based algorithms work by finding a weight vector $\beta \in \mathbb{R}^{n_S}$. This weight vector is used for re-weighting the samples in the source domain, so that the distributions $\beta P_S = \sum_{i=1}^{n_S} \beta_i \delta(x - x_i)$ and P_T match. In short, the difference between the three algorithms used in this section relies on the strategy of finding β.

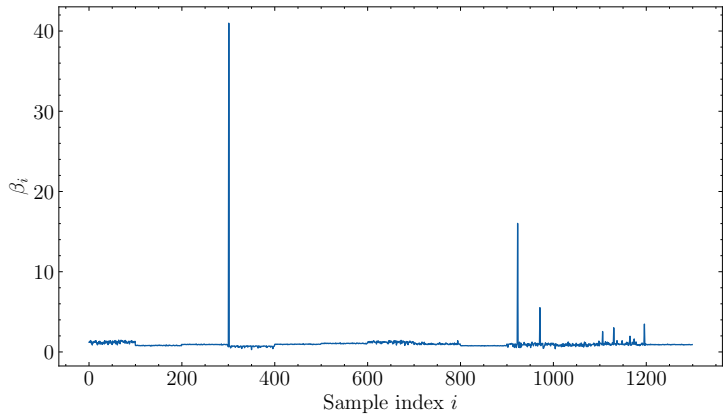


Figure 5.11: Estimated weights for the source domain samples using the KMM algorithm.

For KMM, the only parameter to tune is the importance constraint B, for which $\beta_i \in [0, B]$. The values for B are searched on the grid $[1, 10, 100, 1000]$. For KLIEP, the hyper-parameters are the choice of basis functions ϕ and learning rate ϵ. For ϕ we use a Gaussian kernel centered around 100 randomly chosen samples from the target domain. ϵ is searched on the grid $[10^{-1}, 10^{-2}, \cdots, 10^{-4}]$. Finally, for uLSIF, the only parameter left for tuning is the ℓ^2 regularization term, which is searched on the grid $[10^{-1}, 10^{-2}, \cdots, 10^{-9}]$.

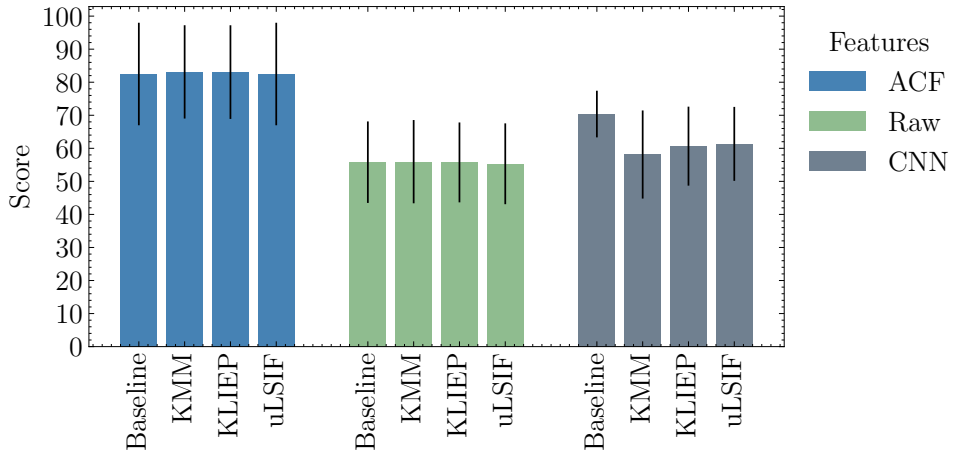


Figure 5.12: Performance comparison for instance-based domain adaptation. For each method, the best acquired accuracy is shown. Results are grouped in terms of used features.

All of these strategies were devised for solving a particular case of domain adaptation, also called *sample selection bias*. In this latter case, as [Cortes and Mohri, 2014] defines, in addition to the hypothesis of covariate shift one needs to have $\text{supp}(P_T) \subset \text{supp}(P_S)$, otherwise a few extreme points will dominate the weights, and thus have a huge influence on the selection of the hypothesis by the algorithm [Cortes et al., 2010]. An example of this issue can be found on the estimated weights for the ACF features, with the KMM algorithm, as shown in Figure 5.11.

The previous arguments reflect on the performance results of instance-based algorithms, as shown in Figure 5.12. Especially, whenever SVM is used β_i weights the penalty C. As consequence, since C is searched alongside each algorithm's parameters, the cross-validation procedure chooses high values for C so as to compensate the small weights assigned to the samples. As consequence the SVM classifier has equivalent performance, with or without adaptation. For CNN features the scenario is worse. By weighting the loss using the weights one gives too much weight to a few samples, harming the process of learning.

Feature-Based Algorithms

For feature-based domain adaptation, four algorithms are evaluated, namely: PCA, TCA, GFK and DANN. Note that this last algorithm is only evaluated for CNN features, since its formulation involves a deep neural network. Since PCA, TCA and GFK involve a dimensionality reduction step, we search the optimal dimensionality d of the latent space on a grid dependent on the type of features. For instance, for ACF features, d is searched on $[1, 2, 5, 7, 10, 13]$. For raw and CNN features, d is searched on $[2, 5, 10, 15, 20, 25, 30, 100, 200, 300]$. In addition, note that TCA and GFK are kernel-based algorithms, hence a Gaussian kernel is used. In addition, the bandwidth γ is searched on $[10^{-1}, 10^{-2}, 10^{-3}]$, plus the scaling choice mentioned early. Finally, the projection step in GFK may be done either through PCA or through PLS. We validate through these two choices as well.

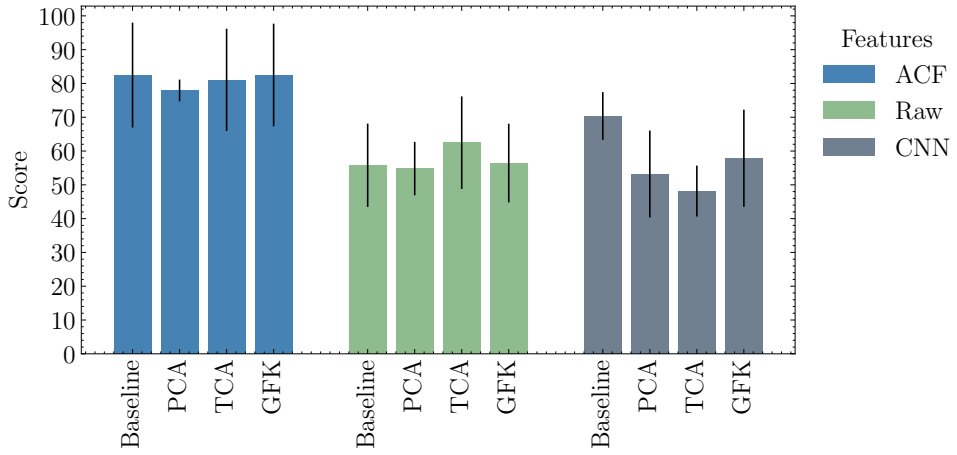


Figure 5.13: Performance comparison for instance-based domain adaptation. For each method, the best acquired accuracy is shown. Results are grouped in terms of used features.

The performance of feature-based domain adaptation differs accordingly the set of features used. First, for ACF features, the best performing method is GFK, with an improvement of 0.032% over the baseline score. Such small improvement indicates that there is no sub-space where the domains share common features. Especially, our results agree with previous experiments in the literature. For instance, [Courty et al., 2016] mentions that for low-dimensional problems, sub-space alignment algorithms have poor performance.

In contrast with this previous result, using TCA on raw features improves the baseline score by 7.164%. This further reinforces the aforementioned claim, indicating that there exists a sub-space where the domains share common characteristics. Note that this was expected, since the signals have are highly redundant in time. Nonetheless GFK performs poorly. In our experiments we noted that the matrices $\mathbf{B}_S$ and $\mathbf{B}_T$ are rank deficient, harming the algorithm's performance.

Finally, the performance of feature-based methods on CNN-extracted features actually harms the classifier's performance on the target domain. We remark that by applying dimensionality reduction before the densely connected layers, on reduces the number of units in this sub-network. As consequence, this reduces the complexity of the space of functions the neural network can learn, thus harming the classification performance.

Optimal Transport-Based Algorithms

For optimal transport-based domain adaptation, several algorithms are tested. First, under the name OTDA, the barycentric mapping presented by Equation 4.23 is used. This mapping relies on the estimation of the transport plan γ, which is done through three approaches, being them:

(i) linear programming, (ii) Sinkhorn algorithm, and (iii) Sinkhorn algorithm with class-based regularization. For class-based regularization, we tested the group-lasso and $\ell_p - \ell_1$ regularizers.

In all the algorithms labeled as OTDA, two parameters come into play: (i) the entropic regularization term λ, and (ii) the class-regularization penalty η. In this regard, both parameters are searched over the grid $[10^{-1}, 10^{-2}, 10^{-3}, 0.0]$. Note that whenever λ or η is zero, that means no regularization. Furthermore, for using such small regularization terms we have normalized the ground-cost matrix by its maximum value, namely $C_{ij} = \frac{D_{ij}}{\max_{ij} D_{ij}}$. This step was done since it yields more stable results computationally.

For mapping estimation procedure proposed by [Perrot et al., 2016], which we refer in short as MP due to its connections to the Monge Problem, we adopt the family of kernelized transformations with Gaussian kernel. In particular, it was verified that the results are robust to the bandwidth γ, so it was set to a default value of 1. This leads to two parameters for validation: (i) the OT-loss weight λ_T, and (ii) the regularization λ_T. Both of these parameters were searched on the grid $[1, 10^{-1}, \cdots, 10^{-3}]$.

Finally, for JDOT 3 parameters are tuned. These are: the entropic regularization term, searched on the grid $[0.0, 10^{-2}, \cdots, 10^{2}]$, the ground cost weight α, searched on the grid $[10^{-1}, 1, \cdots, 10^{2}]$, and the class loss ℓ. For this latter parameter, three types of losses are evaluated: (i) The hinge loss, (ii) The cross-entropy loss, and (iii) The mean squared error loss, all of them previously defined in Chapter 4.

We begin our discussion by comparing the results for the OTDA methods, which rely on the barycentric mapping for transporting samples from source to target domain. Figure 5.14 shows a comparative between four estimation strategies for the transport plan: exact linear programming solver (OT-Ex), Sinkhorn algorithm (OT-IT), and Sinkhorn algorithm with group-lasso (OT-GL) and $\ell_p - \ell_1$ (OT-$\ell_p - \ell_1$) regularization.

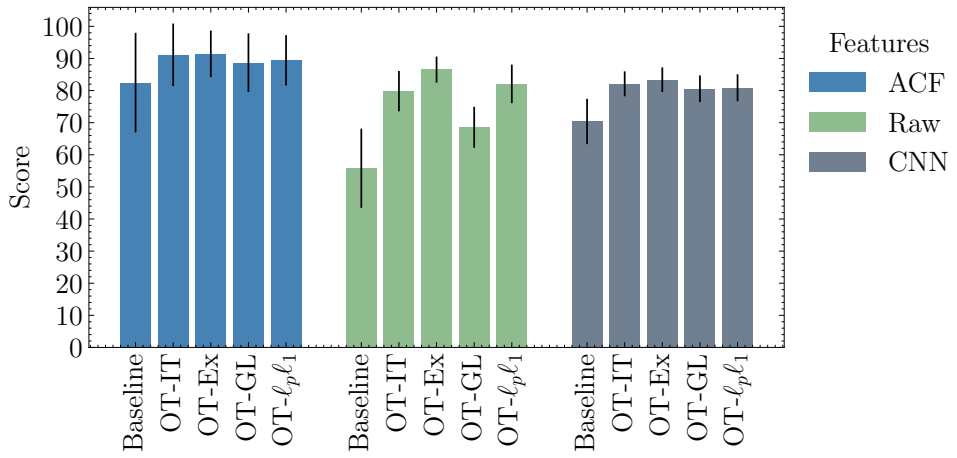


Figure 5.14: Performance comparison for OTDA methods. Results are grouped for each type of feature.

In comparison with previous literature, Sinkhorn estimation of γ was expected to perform better in terms of classification accuracy. Nevertheless, this was not verified, as the exact estimation surpasses the performance for all sets of features.

We remark, in contrast with the experiments in [Courty et al., 2016], our problem has significantly lower dimensionality and number of samples. For instance, when using ACF features we work in $\mathbb{R}^{14}$, whereas the datasets considered in [Courty et al., 2016] have at least 256 features. The number of samples is also important, as the linear programming time complexity depends on it. For the CSTR data, one has 1560 samples per problem (1300 source and 260 target samples), while in [Courty et al., 2016] they have considered problems of over 3800 samples.

To further explore how the exact OT solver performs better than the other settings, we analyze

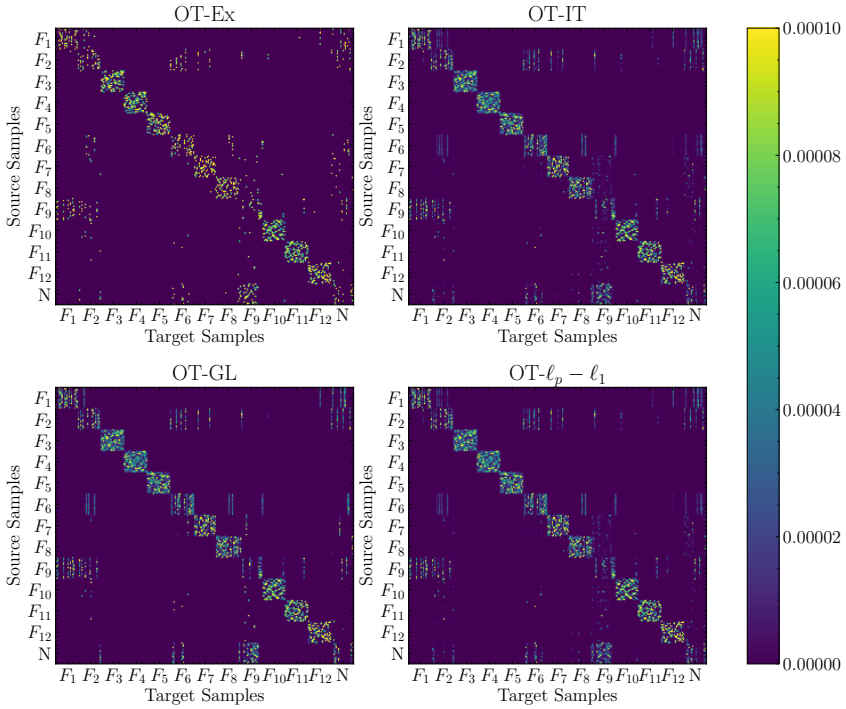


Figure 5.15: Transport plan for each OT solver tested.

the transport plans γ_{Ex}, γ_{IT}, γ_{GL}, and $\gamma_{\ell_p\ell_1}$, with their respective solver. Figure 5.15 shows a comparison between the estimated transport plans when the target domain is sampled with $\epsilon = 0.15$ and $N = 2.0$. Note that with entropic regularization, the transference of mass between samples with different classes become more evident. This helps explaining why classification performance drops, since whenever mass is transferred between samples belonging to different faults, the classes become mixed in the target domain.

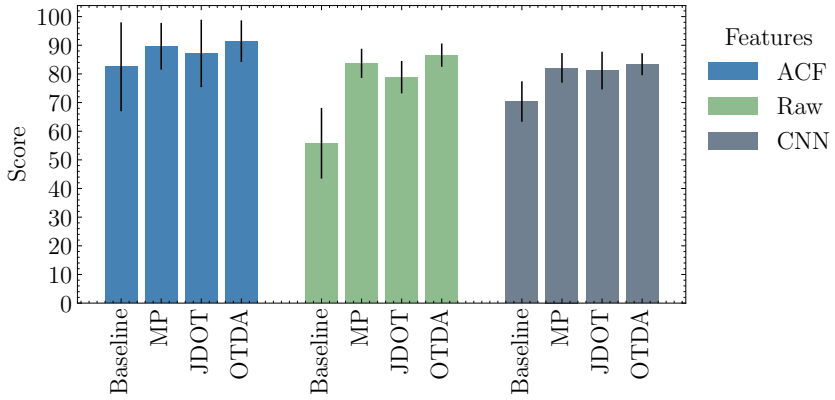


Figure 5.16: Performance comparison for instance-based domain adaptation. For each method, the best acquired accuracy is shown. Results are grouped in terms of used features.

Therefore, we use the exact linear programming for estimating the transport plan γ for the OTDA technique. As mentioned previously, this approach is compared with the mapping estimation strategy proposed by [Perrot et al., 2016] and the JDOT algorithm of [Courty et al., 2017]. A comparison between these three algorithms is shown in Figure 5.16. Concerning Figure 5.16, it is important to mention that among the OT-based algorithms, there was no algorithm that was better for all features. The OT-Ex performed better on ACF features, while the mapping estimation and JDOT approaches performed better on raw and CNN features.

This indicates that simpler approaches (such as the barycentric mapping) can still achieve state-of-the-art performance when the number of samples is limited.

Overall Analysis

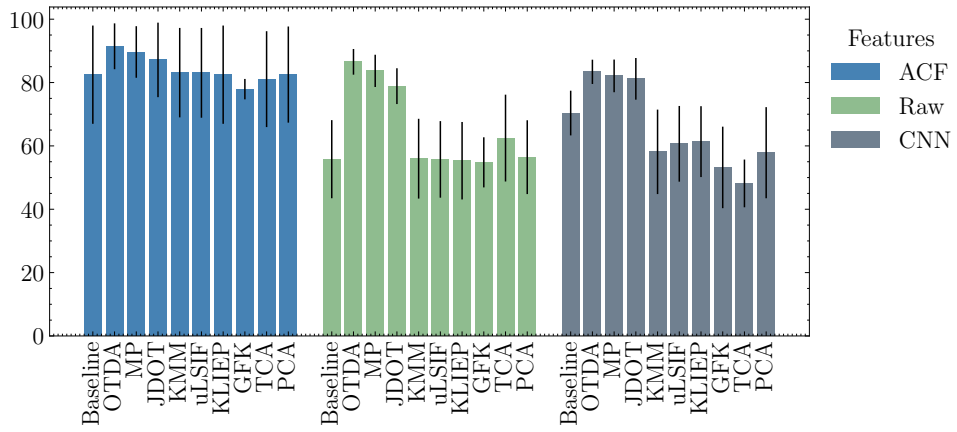


Figure 5.17: Result summary for the comparative study between domain adaptation algorithms.

In this section we present a summary of our comparative study. The detailed results can be seen in Table A.1, in Appendix A whereas Figure 5.17 presents the table's content in summary. In the following discussion, we analyze these results.

First, considering the choice for feature extraction, we highlight that as stated previously, ACF have better performance than CNN features or raw signals. This can be verified when comparing the baseline's score (no adaptation) and when using domain adaptation algorithms. For instance, the best score acquired for each feature choice is 91.442%, 82.917% and 83.686% for ACF, CNN and raw signals respectively. This makes clear the generalization power that ACF features have.

In addition, concerning the latter performance comparison, note that domain adaptation on raw signals perform better than on CNN features, even though the CNN extractor has a higher baseline score than raw signal (70% for the former versus 55% for the latter). This is mainly due the fact that the CNN is trained on a very small dataset. To illustrate how small is the training set we compare the number of parameters in the CNN proposed by [Li et al., 2020] with approaches trained on the Imagenet dataset [Russakovsky et al., 2015]. Note that while the dataset used in this work has over 1300 samples, the Imagenet dataset has over 1.2 million images. In addition, our CNN has over 27 million parameters, whereas the earliest CNNs trained on Imagenet, such as Alexnet [Krizhevsky et al., 2012], has over 60 million parameters. Thus, comparing the number of samples on each dataset, and the size of each network, one can clearly see that the number of collected samples is not enough.

Second, considering the categories for domain adaptation algorithms, optimal-transport based strategies perform consistently better for all feature choices, when compared with feature or instance-based approaches. Note that this conclusion agrees with previous applications of OTDA for cross-domain fault diagnosis, such as in [Liu et al., 2021] and [Cheng et al., 2019], whereas it disagrees with previous results in visual adaptation, as in [Courty et al., 2016], [Perrot et al., 2016], and [Courty et al., 2017]. This indicates that OT-based approaches are better suited for transferring knowledge in time-series classification.

Therefore we remark that using a mathematical modeling step for feature extraction, such as the correlation analysis used in this work, yields the best results. This kind of approach has two positive points. On one hand, it provides an analytical description to the features, with guarantees that they indeed distinguish well the classes. On the other hand, it gives a

simpler model, in terms of number of features. Finally, the superiority of this approach is more prominent when mathematical models for the problem are known, such as dynamical systems, and the number of samples is limited, such as this case study.

5.3 Distributional Shift Characterization

In this section we analyze the relation between a given classifier's accuracy, and the measure of distributional shift between source and target distributions. Note that by Theorem 4.4, the $\mathcal{H}-$distance, the Wasserstein distance and the MMD distance serve as a bound between the miss-classification rate on the target and source domain. Indeed, by Theorem 4.3, one has,

$$\mathcal{R}_T(h) \leq \mathcal{R}_S(h) + \alpha\varphi(P_S, P_T) + \lambda,$$

where φ is a divergence between probability distributions. Assuming the hypothesis space to be fixed, λ is constant. Thus, for greater values of $\varphi(P_S, P_T)$ one gets a looser bound between $\mathcal{R}_T$ and $\mathcal{R}_S$, allowing the target-risk to become larger. To start our discussion, we will consider how the degree of parameter mismatch ϵ, and the reaction order N affect $\varphi(\hat{P}_S, \hat{P}_T)$, when φ is the Wasserstein distance W_2 with euclidean distance as ground cost.

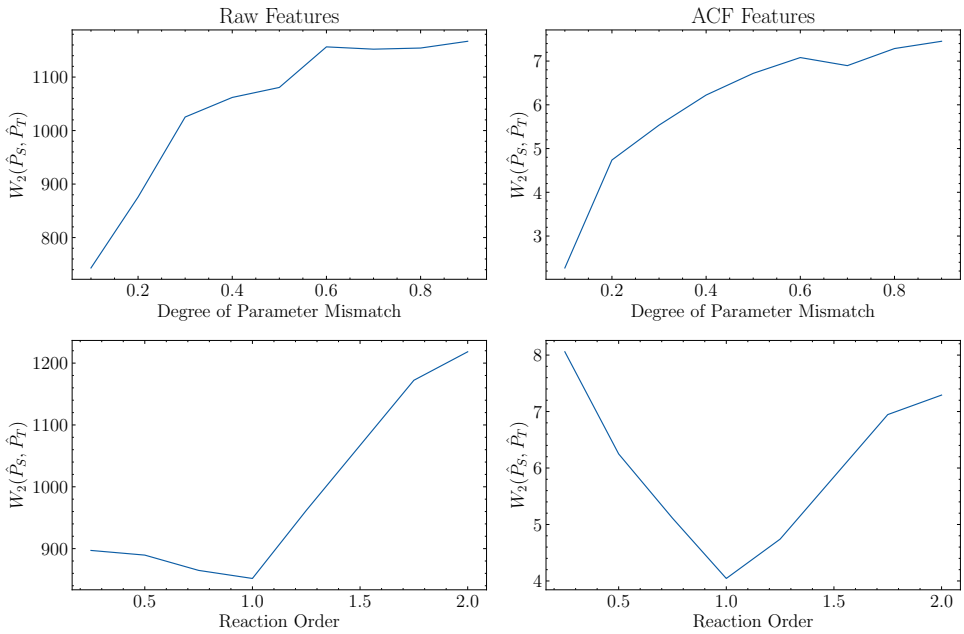


Figure 5.18: Wasserstein distance as a function of degree of parameter mismatch ϵ and reaction order N for ACF and raw features.

Following this idea, we expand the sampling procedure of [Li et al., 2020] for higher values of parameter mismatch ϵ and reaction order N. Therefore, we generate new target domains for a degree of parameter mismatch with values on $[0.1, 0.2, 0.3, \cdots, 0.9]$, and reaction order on $[0.25, 0.5, 0.75, \cdots, 2.0]$. Being aware that the values picked for the reaction order are not realistic, here we defined them only for the sake of further investigating the proposed methodology. Thus, this yields 17 domains, for which we measure $W_2(\hat{P}_S, \hat{P}_T)$. Figure 5.18 shows how changes in these parameters affect the distance between source and target distributions. As expected, $W_2(\hat{P}_S, \hat{P}_T)$ grows for increasing values of ϵ. Moreover $\epsilon > 0.6$ the distributional shift saturates. This indicates that the two domains are so different in terms of distributions that they become unrelated. Furthermore, for changes in the reaction order N the shift behaves differently according to the features. Namely, for raw features $N > 1$ implies on a higher distributional

shift than $N < 1$, while for ACF features it is the contrary. This indicates that the shift behaves differently in light of different features.

In addition to the previous analysis, we also consider the relationship between the empirical risk $\hat{\mathcal{R}}_T$ and the Wasserstein distance $W_2(\hat{P}_S, \hat{P}_T)$. Following the bound given by Theorem 4.3, we expect an increase in $\hat{\mathcal{R}}_T$ for increasing values of $W(\hat{P}_S, \hat{P}_T)$. We remark that the definition of $\hat{\mathcal{R}}_T$ relies on the choice of loss function ℓ. Being so we use the $0-1$ loss, so that $\hat{\mathcal{R}}_T$ is equal to the miss-classification rate. Figure 5.19 presents $\hat{\mathcal{R}}_T$ as a function of $W_2(\hat{P}_S, \hat{P}_T)$ for various choices of classifiers. Each point in Figure 5.19 represents an experiment with fixed ϵ, N and classification algorithm. For comparison, we have considered three algorithms: linear and RBF SVMs and a MLP. Alongside the scatter plot with the experiments results, a regression line was fitted to the data for each classifier.

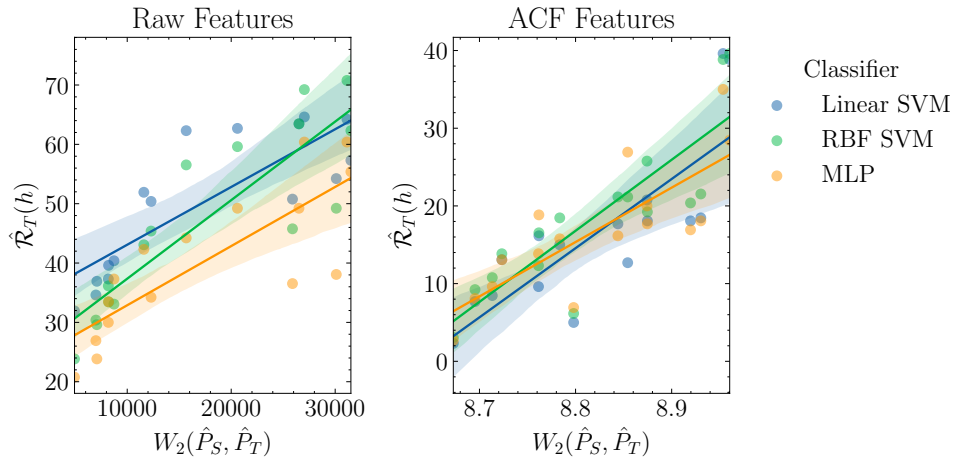


Figure 5.19: Comparison between the empirical risk on target domain as a function of the Wasserstein distance for three distinct classifiers, for raw and ACF features.

Finally, this reasoning allows a comparison concerning the robustness of classifiers with respect to distributional shift for each feature choice. This can be done through the regression line slope, that represents the increase on the empirical risk due an increase in the distance between distributions, measured through the Wasserstein distance. In Table 5.5 the regression line slope and its R^2 statistic are shown for each tested setting. Note that we may conclude that, for instance, for raw features the linear SVM is more robust to distributional shift, while for ACF features, the MLP is more robust.

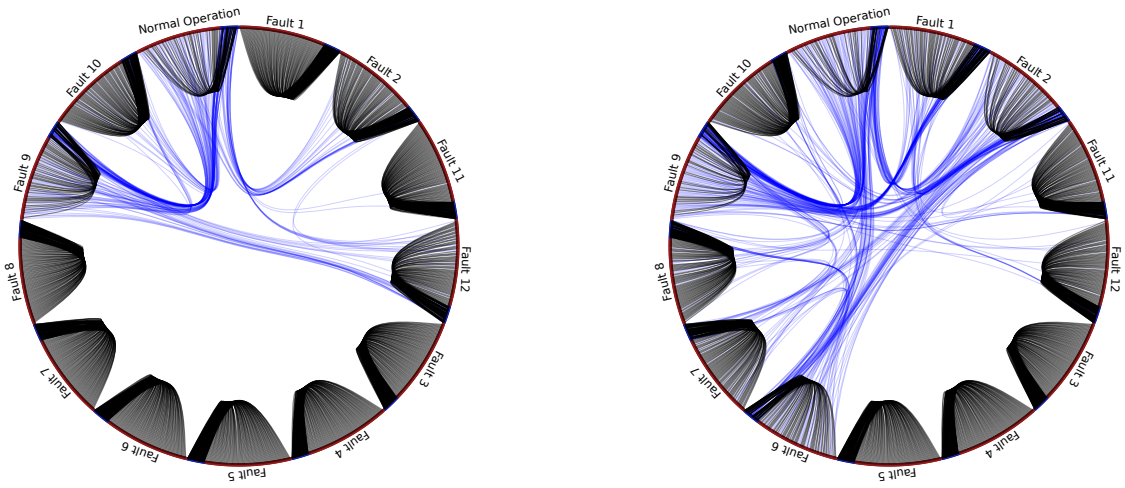
Feature Choice	Statistic	Linear SVM	RBF SVM	MLP
Raw	Slope	9.297×10^{-4}	13.236×10^{-4}	9.969×10^{-4}
	R^2	0.822	0.868	0.807
ACF	Slope	89.034	91.445	70.012
	R^2	0.811	0.852	0.796

Table 5.5: Classifier robustness to distributional shift. The slope gives how much an increase in $W_2(\hat{P}_S, \hat{P}_T)$ increases the miss-classification rate given by $\hat{\mathcal{R}}_T$.

5.4 Negative Transfer under Optimal Transport

In this section we investigate the phenomenon of negative transfer when applying OTDA methods for domain adaptation. Note that, as defined in Chapter 4, the negative transfer phenomenon happens whenever $\text{UNTG} = \mathcal{R}_T(h_{TL}) - \mathcal{R}_T(h_S) > 0$, being h_{TL} the hypothesis learned by the

transfer learner, and h_S the hypothesis learned by the learning machine using solely source domain data. We base our discussion on the fact that transportation plans γ for which $\gamma_{ij} > 0$ whenever $y_S^i \neq y_T^j$ may yield poor classification results. This was previously noted by [Courty et al., 2016], who proposed various class-based regularizers to avoid such phenomenon. To illustrate the issue, we present in Figure 5.20 a visualization of the mass transfer under γ, highlighting the undesired connections γ_{ij} in blue.



(1) Visualization of mass transfer under the transport plan γ for $\epsilon = 0.1$ and $N = 1.0$.

(2) Visualization of mass transfer under the transport plan γ for $\epsilon = 0.15$ and $N = 2.0$.

Figure 5.20: Comparison between the mass transferred across different faults (edges in blue) for different degrees of parameter mismatch. Source domain samples are represented as continuous red arcs, whereas target domain samples are continuous blue arcs. Whenever source samples are connected to target samples of the same fault, a gray edge is drawn between the locations.

Therefore, we propose to analyze the distributional shift and UNTG by using the UMF index proposed in Definition 4.15. Initially, we assess the relationship between distributional shift and the UMF. Thus, for each target domain with $\epsilon \in \{0.1, 0.15, 0.2\}$ and $N \in \{0.5, 1.0, 1.5, 2.0\}$, we calculate γ using both the exact and Sinkhorn solvers. This yields 12 transport plans, for which we evaluate the proposed UMF index. The results are shown in Figure 5.21. We highlight that these results should be compared with those of Figure 5.6, which shows the Wasserstein distance for the same settings.

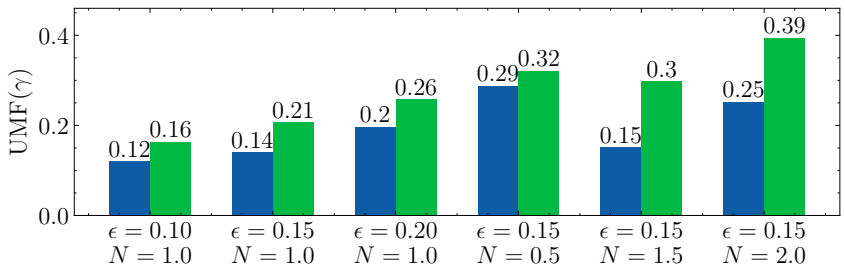


Figure 5.21: UMF index for different target domains.

Note that when increasing the degree of mismatch ϵ, one has an increasing UMF. In addition, when changing the reaction order one from 0.5 to 1.5 has a drop in the UMF of γ. This helps explaining why the classification performance improves from domain 4 to domain 5, as verified in Table A.1, whereas the Wasserstein distance increases. Note that this is not verified for other

algorithms (e.g. instance-based, baseline and feature-based), since these algorithms do not rely on the transportation of samples. This is a strong indicative that an increase in the UMF reflects in bad transportation plans, potentially causing the negative transfer phenomenon.

To further verify the previous claim in practice, we perform an analysis similar to that of Figure 5.19, that is, the target empirical risk is assessed for each value of UMF. The results are shown in Figure 5.22. We highlight that, while in Figure 5.19 we presented the target empirical risk of the source hypothesis h_S, in this analysis we presented the target empirical risk of the hypothesis h_{TL} acquired by the OTDA transfer learner.

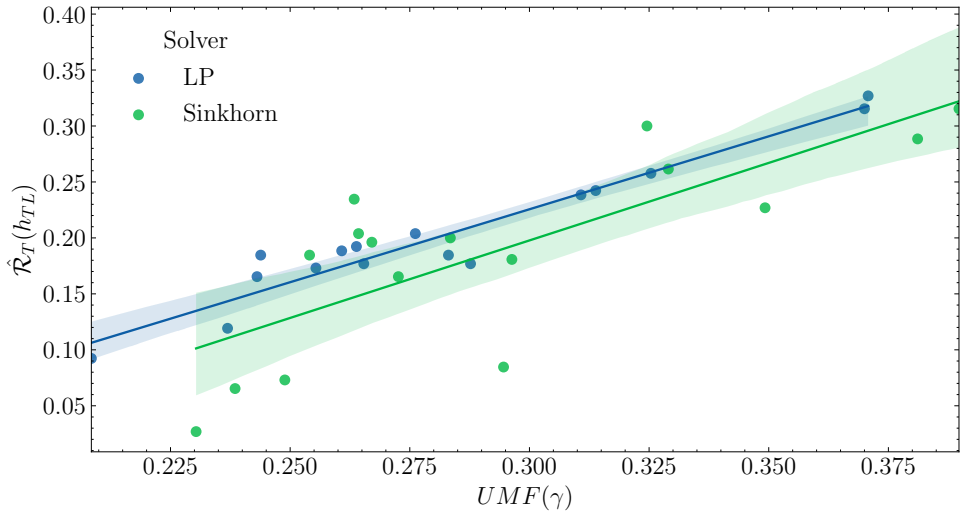


Figure 5.22: The relationship between UMF index and empirical risk for linear programming and Sinkhorn solvers

From Figure 5.22 we may infer that the target empirical risk is positively correlated with the UMF index here proposed. This further sheds a light on the reason why OTDA fails: whenever γ transfer masses between different classes, the target domain accuracy tends to drop. This further justifies the class-based regularization approach proposed by [Courty et al., 2016].

Chapter 6

Conclusion

This thesis has presented an end-to-end analysis of a fault diagnosis system, comprising mathematical modeling, feature engineering, and classification algorithm choice. The central problem treated in this work is known as cross-domain fault diagnosis [Zheng et al., 2019], which corresponds to the problem of fault diagnosis when data is sampled in heterogeneous conditions. This latter condition is known as domain adaptation in the literature. We explore this problem through parameter mismatch in simulation environments as done in [Li et al., 2020]. Our analysis relies heavily on the theory of optimal transport developed by [Villani, 2008] and [Peyré et al., 2019], using these concepts both as a tool for analysis, and for overcoming distributional shift. Therefore, we exemplify our methodology through a case study concerning the CSTR benchmark system, providing three lines of analysis:

Line 1. A comparative study between feature extraction methods and domain adaptation algorithms, covering instance, feature and optimal transport-based methods.

Line 2. An analysis on the relation between the degree of model wrongful specification and the Wasserstein distance between source and target probability distributions. This is further expanded by a study on the relationship between the latter distance, and the probability of classification error.

Line 3. An analysis on the degree upon which optimal transportation plans γ transfer mass between different classes, and how this impacts the probability of classification errors under domain adaptation.

Concerning Line 1, we performed comprehensive experiments comparing feature extraction approaches, and transfer learning algorithms. The comparison was established by comparing a score, corresponding to the average classification accuracy on all domains. For feature extraction methods, three choices were explored: raw signals, ACF coefficients, and CNN latent representations. Among these choices, whenever training and test data are sampled from the same probability distribution, the latent representations have the best performance.

Moreover, by relaxing the assumption upon training and test probability distributions, we compare algorithms from three major frameworks in transfer learning: instance, feature and OT-based algorithms. In this alternative learning setting we found that ACF features have a remarkable robustness to distributional shift. Comparing ACF, CNN and raw signals, these have a score of 82.484%, 55.304% and 70.000% respectively.

In addition, among the tested frameworks, OT-based algorithms surpass other approaches by margins of 8.926%, 5.917%, 21.218%, for ACF, CNN and raw signals respectively, with respect to the aforementioned score. Such loose margins agree with previous work on cross-domain fault diagnosis such as [Cheng et al., 2019] and [Liu et al., 2021], but contrast with visual adaptation benchmarks such as the Office-Caltech dataset explored in [Courty et al., 2016], [Perrot et al.,

2016] and [Courty et al., 2017] for which the margins are more tight. This highlights that OT-based transport may be better suited for time-series domain adaptation.

Concerning Line 2, our experiments aim at verifying two hypothesis: 1. That sampling from systems with wrongful specified parameters lead to data following different probability distributions, and 2. A larger distance between these distributions leads to higher classification error. We verify the first hypothesis by sampling from systems with an increasing degree of parameter mismatch. We also examine the effect of modifying more specific parameters, such as the reaction order in the tank reactor. The second hypothesis is verified by training classification models on the data acquired in the previous conditions, thus showing that distributional shift and parameter mismatch tends to increase classification error.

Moreover, concerning Line 3, we propose an index for measuring the goodness of a optimal transportation plan in a domain adaptation problem. The index is called UMF, and it measures how much probability mass is transported between different classes. As our experiments show, a higher degree of UMF may lead to higher classification error, which further justifies previous studies, such as the class-based regularization for OTDA proposed by [Courty et al., 2016], and the theoretical discussion presented in [Redko et al., 2017]. In the following, we discuss possible improvements to our methodology, as well as future works.

6.1 Future Research Directions

We highlight at least 5 possible directions in which our methodology may be improved or expanded,

1. Explore the problem of detection, and possibly the design of a FDD system without separating the two tasks.

2. Explore other types of faults, e.g. other types of process disturbance, other functional models for catalyst decay and heat fouling. Concerning the former type, one could consider sinusoidal perturbations. Concerning the latter type, one could use a linear decay as in [Juricek et al., 2004].

3. Explore other types of DNN architecture. For instance, one could design an alternative input to the CNN, such as a tensor of shape $(n_{batch}, 200, 7)$, where 200 is the number of time-steps and 7 is the number of channels, corresponding to the process variables. Another possibility is considering other types of neural network architecture, such as Recurrent Neural Networks (RNNs).

4. Expand the sampling procedure, collecting more samples and considering data augmentation for a more reliable training of a DNN.

5. Explore the relationship between the UMF and the semi-supervised regularization term proposed by [Courty et al., 2016]. Especially, the UMF index may be used as a regularization term in semi-supervised domain adaptation.

6. Explore the multi-source domain adaptation setting for cross-domain fault diagnosis. First, one could set a single domain as the target, corresponding to the real or emulated physical domain. Second, one samples data from different parameter settings, within a confidence range, generating various source-domains. Usually, this setting has higher performance since much more information about the problem is available (e.g. [Peng et al., 2019] and [Turrisi et al., 2020]), but it is also more challenging due to the complexity of optimization problems involved.

Bibliography

[Batu et al., 2000] Batu, T., Fortnow, L., Rubinfeld, R., Smith, W. D., and White, P. (2000). Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE.

[Ben-David et al., 2010] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

[Ben-David et al., 2007] Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. (2007). Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[Blondel et al., 2018] Blondel, M., Seguy, V., and Rolet, A. (2018). Smooth and sparse optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pages 880–889. PMLR.

[Bolley et al., 2007] Bolley, F., Guillin, A., and Villani, C. (2007). Quantitative concentration inequalities for empirical measures on non-compact spaces. *Probability Theory and Related Fields*, 137(3-4):541–593.

[Borgwardt et al., 2006] Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57.

[Box et al., 2011] Box, G. E., Jenkins, G. M., and Reinsel, G. C. (2011). *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons.

[Boyd et al., 2004] Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Burden and Faires, 2011] Burden, R. L. and Faires, J. D. (2011). Numerical analysis.

[Casella and Berger, 2002] Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

[Chen, 1984] Chen, C.-T. (1984). *Linear system theory and design*, volume 301. Holt, Rinehart and Winston New York.

[Chen and Seborg, 2002] Chen, D. and Seborg, D. E. (2002). Pi/pid controller design based on direct synthesis and disturbance rejection. *Industrial & engineering chemistry research*, 41(19):4807–4822.

[Cheng et al., 2019] Cheng, C., Zhou, B., Ma, G., Wu, D., and Yuan, Y. (2019). Wasserstein distance based deep adversarial transfer learning for intelligent fault diagnosis. *arXiv preprint arXiv:1903.06753*.

[Chiang et al., 2000] Chiang, L. H., Russell, E. L., and Braatz, R. D. (2000). *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media.

[Choi et al., 2005] Choi, S. W., Lee, C., Lee, J.-M., Park, J. H., and Lee, I.-B. (2005). Fault detection and identification of nonlinear processes based on kernel PCA. *Chemometrics and intelligent laboratory systems*, 75(1):55–67.

[Cortes et al., 2010] Cortes, C., Mansour, Y., and Mohri, M. (2010). Learning bounds for importance weighting. In *Advances in neural information processing systems*, pages 442–450.

[Cortes and Mohri, 2014] Cortes, C. and Mohri, M. (2014). Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126.

[Courty et al., 2017] Courty, N., Flamary, R., Habrard, A., and Rakotomamonjy, A. (2017). Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739.

[Courty et al., 2016] Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2016). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865.

[Crammer et al., 2008] Crammer, K., Kearns, M., and Wortman, J. (2008). Learning from multiple sources. *Journal of Machine Learning Research*, 9(Aug):1757–1774.

[Crawshaw, 2020] Crawshaw, M. (2020). Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*.

[Cuturi, 2013] Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300.

[Cuturi and Doucet, 2014] Cuturi, M. and Doucet, A. (2014). Fast computation of wasserstein barycenters. *Journal of Machine Learning Research*.

[Dai et al., 2008] Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2008). Self-taught clustering. In *Proceedings of the 25th international conference on Machine learning*, pages 200–207.

[Dantzig and Thapa, 2006] Dantzig, G. B. and Thapa, M. N. (2006). *Linear programming 1: introduction*. Springer Science & Business Media.

[David et al., 2010] David, S. B., Lu, T., Luu, T., and Pál, D. (2010). Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings.

[Erhan et al., 2010] Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

[Ganin et al., 2016] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

[George et al., 2007] George, A. J., Marcus, A. R. B., Isabela, S. V., and Péricles, R. B. (2007). Identification of first-order plus dead-time continuous-time models using time-frequency information. *VII Simpósio Brasileiro de Automação Inteligente*.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

[Gong et al., 2012] Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[Gretton et al., 2007] Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2007). A kernel approach to comparing distributions. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1637. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Gretton et al., 2009] Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

[Hasan et al., 2020] Hasan, M. J., Sohaib, M., and Kim, J.-M. (2020). A multitask-aided transfer learning-based diagnostic framework for bearings under inconsistent working conditions. *Sensors*, 20(24):7205.

[Isermann, 1997] Isermann, R. (1997). Supervision, fault-detection and fault-diagnosis methods—an introduction. *Control engineering practice*, 5(5):639–652.

[Isermann, 2006] Isermann, R. (2006). *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media.

[Jaggi, 2013] Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR.

[Jegadeeshwaran and Sugumaran, 2015] Jegadeeshwaran, R. and Sugumaran, V. (2015). Fault diagnosis of automobile hydraulic brake system using statistical features and support vector machines. *Mechanical Systems and Signal Processing*, 52:436–446.

[Jiang et al., 2017] Jiang, L., Ge, Z., and Song, Z. (2017). Semi-supervised fault classification based on dynamic sparse stacked auto-encoders model. *Chemometrics and Intelligent Laboratory Systems*, 168:72–83.

[Juricek et al., 2004] Juricek, B. C., Seborg, D. E., and Larimore, W. E. (2004). Fault detection using canonical variate analysis. *Industrial & engineering chemistry research*, 43(2):458–474.

[Kanamori et al., 2009] Kanamori, T., Hido, S., and Sugiyama, M. (2009). A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445.

[Kantorovich, 2006] Kantorovich, L. V. (2006). On the translocation of masses. *Journal of Mathematical Sciences*, 133(4):1381–1382.

[Kifer et al., 2004] Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[Li et al., 2020] Li, W., Gu, S., Zhang, X., and Chen, T. (2020). Transfer learning for process fault diagnosis: Knowledge transfer from simulation to physical processes. *Computers & Chemical Engineering*, page 106904.

[Liu et al., 2021] Liu, Z.-H., Jiang, L.-B., Wei, H.-L., Chen, L., and Li, X.-H. (2021). Optimal transport-based deep domain adaptation approach for fault diagnosis of rotating machine. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12.

[Lyons et al., 1998] Lyons, M., Akamatsu, S., Kamachi, M., and Gyoba, J. (1998). Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE international conference on automatic face and gesture recognition*, pages 200–205. IEEE.

[Matasci et al., 2011] Matasci, G., Volpi, M., Tuia, D., and Kanevski, M. (2011). Transfer component analysis for domain adaptation in image classification. In *Image and Signal Processing for Remote Sensing XVII*, volume 8180, page 81800F. International Society for Optics and Photonics.

[Nise, 2020] Nise, N. S. (2020). *Control systems engineering*. John Wiley & Sons.

[Noura et al., 2009] Noura, H., Theilliol, D., Ponsart, J.-C., and Chamseddine, A. (2009). *Fault-tolerant control systems: Design and practical applications*. Springer Science & Business Media.

[Pan et al., 2010] Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Papoulis and Pillai, 2002] Papoulis, A. and Pillai, S. U. (2002). *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education.

[Patton et al., 2013] Patton, R. J., Frank, P. M., and Clark, R. N. (2013). *Issues of fault diagnosis for dynamic systems*. Springer Science & Business Media.

[Peng et al., 2019] Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415.

[Perrot et al., 2016] Perrot, M., Courty, N., Flamary, R., and Habrard, A. (2016). Mapping estimation for discrete optimal transport. In *Advances in Neural Information Processing Systems*, pages 4197–4205.

[Peyré et al., 2019] Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

[Pilario and Cao, 2018] Pilario, K. E. S. and Cao, Y. (2018). Canonical variate dissimilarity analysis for process incipient fault detection. *IEEE Transactions on Industrial Informatics*, 14(12):5308–5315.

[Raina et al., 2007] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766.

[Redko et al., 2017] Redko, I., Habrard, A., and Sebban, M. (2017). Theoretical analysis of domain adaptation with optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 737–753. Springer.

[Redko et al., 2020] Redko, I., Morvant, E., Habrard, A., Sebban, M., and Bennani, Y. (2020). A survey on domain adaptation theory. *arXiv preprint arXiv:2004.11829*.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Saitoh and Sawano, 2016] Saitoh, S. and Sawano, Y. (2016). *Theory of reproducing kernels and applications*. Springer.

[Solomon et al., 2015] Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. (2015). Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):1–11.

[Särkkä, 2013] Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

[Strogatz, 2018] Strogatz, S. H. (2018). *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press.

[Su et al., 2014] Su, Z., Tang, B., Ma, J., and Deng, L. (2014). Fault diagnosis method based on incremental enhanced supervised locally linear embedding and adaptive nearest neighbor classifier. *Measurement*, 48:136–148.

[Sugiyama et al., 2008] Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., and Kawanabe, M. (2008). Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746.

[Tseng, 2001] Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494.

[Turrisi et al., 2020] Turrisi, R., Flamary, R., Rakotomamonjy, A., and Pontil, M. (2020). Multi-source domain adaptation via weighted joint distributions optimal transport. *arXiv preprint arXiv:2006.12938*.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[Vapnik, 2013] Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

[Villani, 2008] Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

[Wang et al., 2019] Wang, Z., Dai, Z., Póczos, B., and Carbonell, J. (2019). Characterizing and avoiding negative transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11293–11302.

[Wang et al., 2017] Wang, Z., Zhang, Q., Xiong, J., Xiao, M., Sun, G., and He, J. (2017). Fault diagnosis of a rolling bearing using wavelet packet denoising and random forests. *IEEE Sensors Journal*, 17(17):5581–5588.

[Yan et al., 2014] Yan, R., Gao, R. X., and Chen, X. (2014). Wavelets for fault diagnosis of rotary machines: A review with applications. *Signal processing*, 96:1–15.

[Yang et al., 2020] Yang, Q., Zhang, Y., Dai, W., and Pan, S. J. (2020). *Transfer Learning*. Cambridge University Press.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

[Zhang et al., 2015] Zhang, X., Chen, W., Wang, B., and Chen, X. (2015). Intelligent fault diagnosis of rotating machinery using support vector machine with ant colony algorithm for synchronous feature selection and parameter optimization. *Neurocomputing*, 167:260–279.

[Zheng et al., 2019] Zheng, H., Wang, R., Yang, Y., Yin, J., Li, Y., Li, Y., and Xu, M. (2019). Cross-domain fault diagnosis using knowledge transfer strategy: a review. *IEEE Access*, 7:129260–129290.

[Zhu and Knyazev, 2013] Zhu, P. and Knyazev, A. V. (2013). Angles between subspaces and their tangents. *Journal of Numerical Mathematics*, 21(4):325–340.

[Ziegler et al., 1942] Ziegler, J. G., Nichols, N. B., et al. (1942). Optimum settings for automatic controllers. *trans. ASME*, 64(11).

Appendix A

Detailed Results

A.1 Baseline Assessment

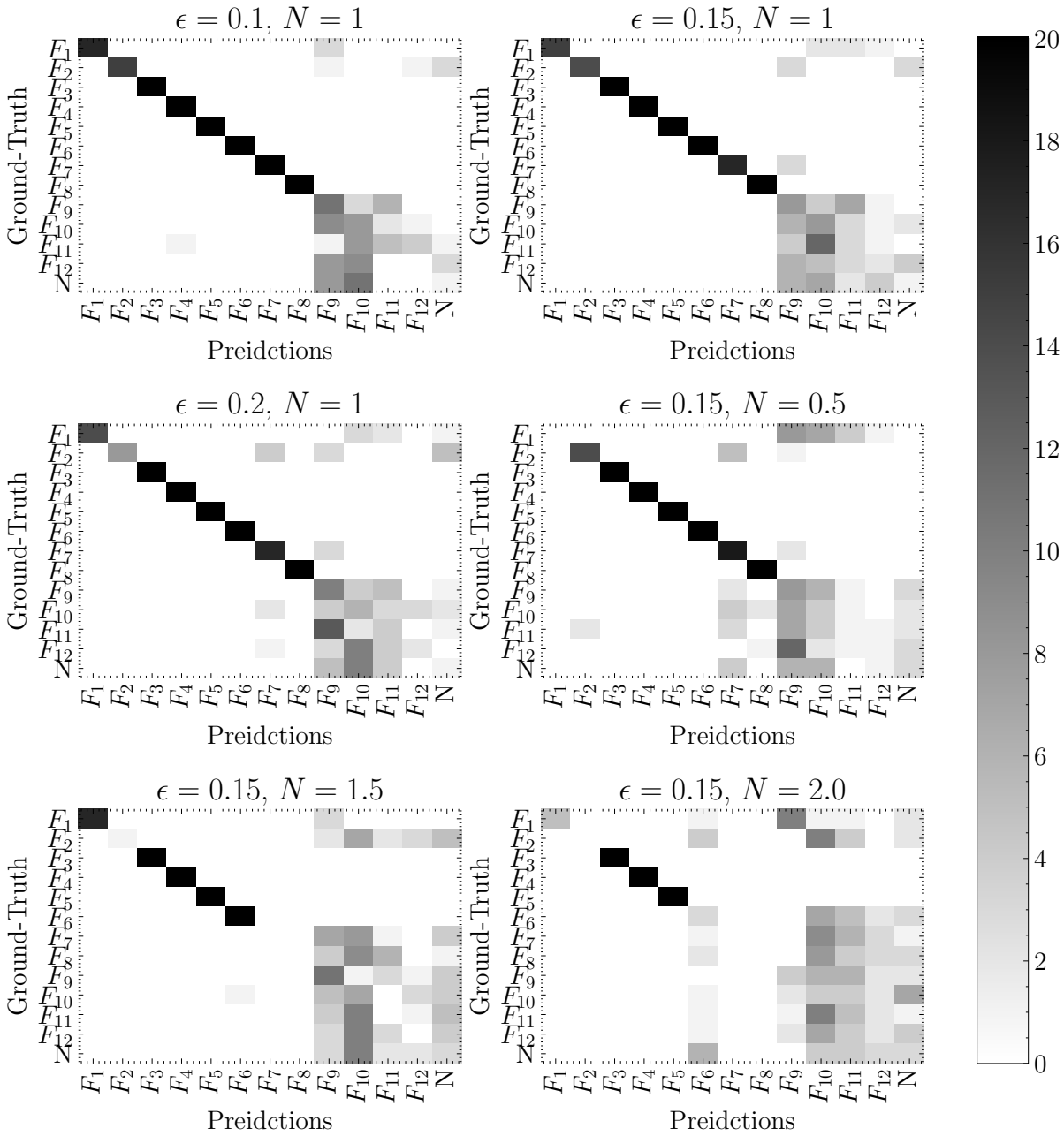


Figure A.1: Comparison between confusion matrices of SVM classification with raw features. Each confusion matrix corresponds to a target domain sampled using a different setting for parameter mismatch.

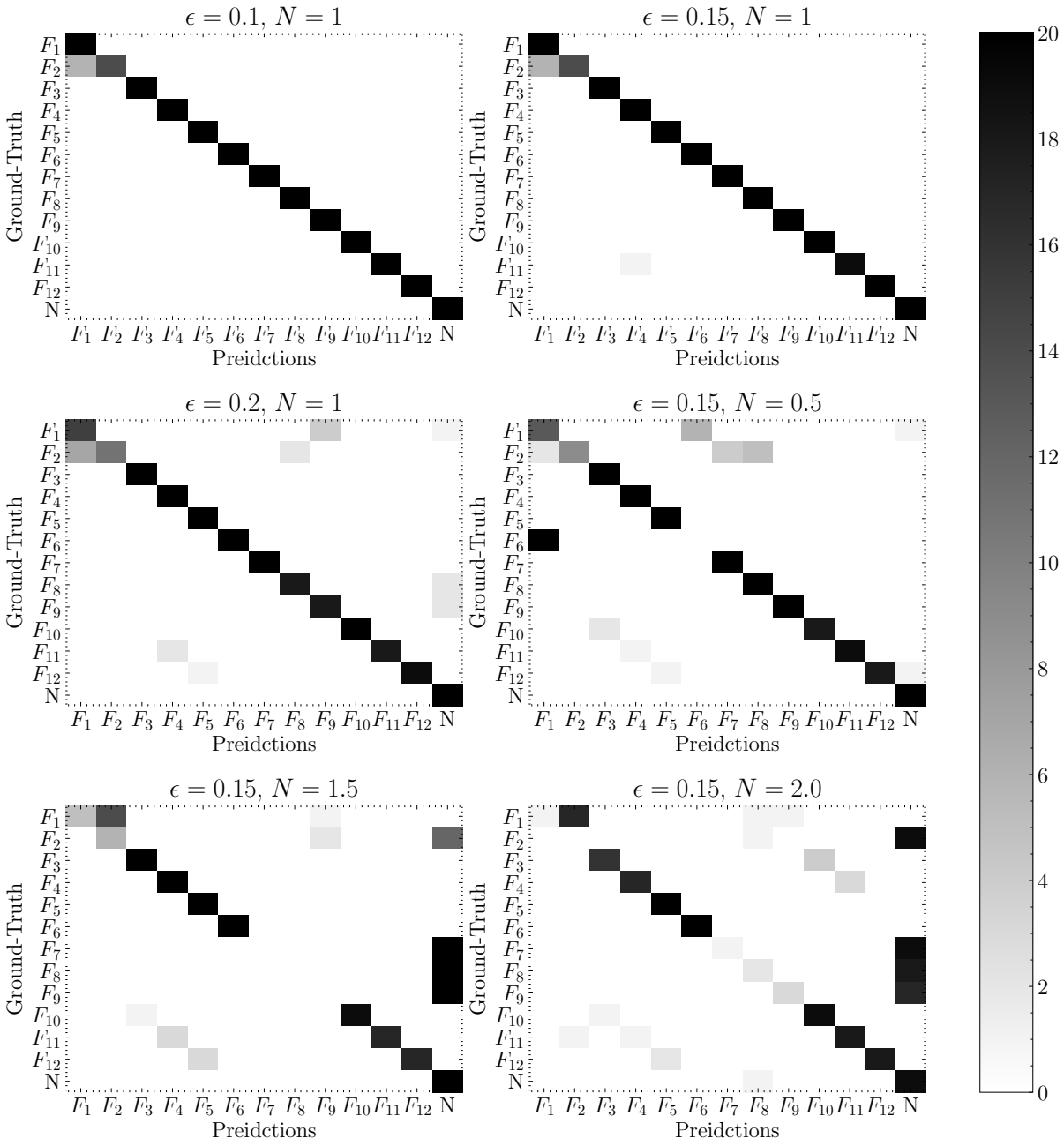


Figure A.2: Comparison between confusion matrices of SVM classification with ACF features. Each confusion matrix corresponds to a target domain sampled using a different setting for parameter mismatch.

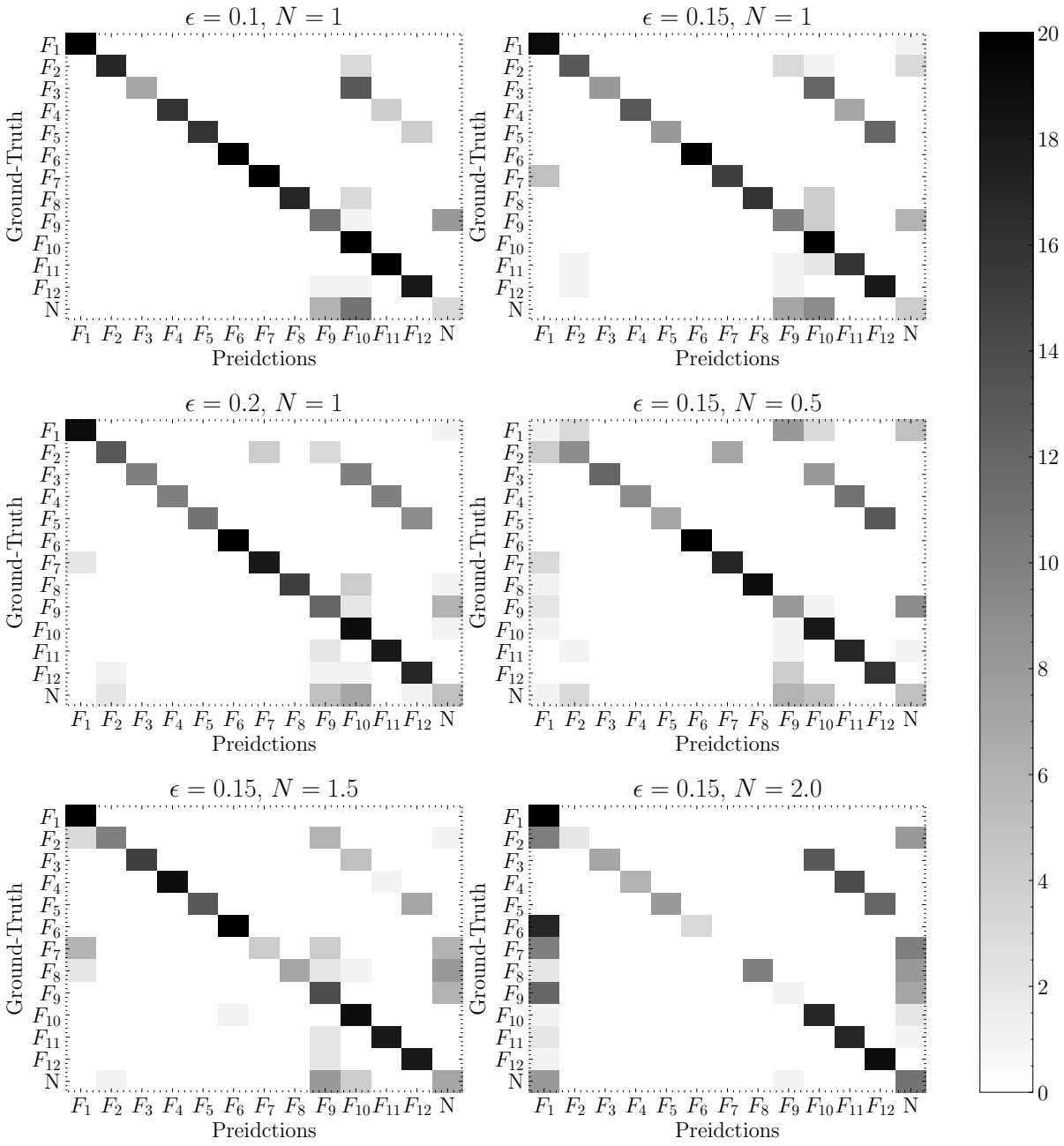


Figure A.3: Comparison between confusion matrices of CNN classification with Raw features. Each confusion matrix corresponds to a target domain sampled using a different setting for parameter mismatch.

A.2 Comparative Study of Transfer Learning Algorithms

Reaction Order	1.0			0.5	1.5	2.0	Score
Degree of Mismatch	10%	15%	20%	15%			
Raw Features							
SVM	68.654 ± 0.769	64.519 ± 1.231	62.404 ± 1.154	56.923 ± 0.892	46.346 ± 1.709	32.981 ± 0.942	55.304
KMM	70.769 ± 0.638	64.423 ± 1.609	63.173 ± 1.889	57.212 ± 1.393	47.308 ± 0.490	32.885 ± 1.413	55.962
KLIEP	68.750 ± 0.745	64.519 ± 1.231	62.404 ± 1.153	56.923 ± 0.892	46.346 ± 1.709	32.981 ± 0.942	55.321
uLSIF	69.423 ± 1.346	63.846 ± 0.360	62.596 ± 1.860	56.731 ± 1.520	47.596 ± 0.527	27.692 ± 9.860	54.647
PCA	69.423 ± 0.490	64.519 ± 1.027	64.615 ± 0.577	57.212 ± 1.29	46.635 ± 0.804	36.058 ± 0.608	56.410
TCA	81.442 ± 0.990	71.923 ± 1.477	65.192 ± 0.990	66.346 ± 1.799	49.519 ± 1.747	40.385 ± 0.860	62.468
GFK	65.385 ± 1.290	59.135 ± 1.178	58.269 ± 0.561	53.173 ± 1.380	49.423 ± 1.027	42.692 ± 1.909	54.679
OTDA	**89.423 ± 0.680**	87.596 ± 0.769	80.673 ± 1.532	72.404 ± 1.380	85.769 ± 1.201	77.308 ± 1.113	82.196
Monge Mapping	89.327 ± 0.360	**87.981 ± 1.008**	**84.038 ± 1.757**	**74.808 ± 0.892**	**86.538 ± 0.804**	**79.423 ± 0.707**	**83.686**
JDOT	86.346 ± 0.838	83.846 ± 1.654	76.058 ± 2.284	70.000 ± 1.840	81.923 ± 0.942	75.000 ± 0.527	78.862
ACF Features							
SVM	97.692 ± 0.360	97.308 ± 0.385	92.019 ± 0.490	85.096 ± 3.145	62.981 ± 0.912	59.808 ± 0.942	82.484
KMM	97.692 ± 0.360	97.308 ± 0.385	92.019 ± 0.490	85.096 ± 3.145	62.981 ± 0.912	59.808 ± 0.942	82.484
KLIEP	97.692 ± 0.360	97.308 ± 0.385	92.019 ± 0.490	85.096 ± 3.145	62.981 ± 0.912	59.808 ± 0.942	82.484
uLSIF	97.956 ± 0.527	97.212 ± 0.360	91.538 ± 0.720	83.750 ± 4.504	62.122 ± 1.508	61.827 ± 1.985	82.356
PCA	97.596 ± 0.527	**97.404 ± 0.490**	91.923 ± 1.071	84.135 ± 0.304	62.885 ± 0.881	61.154 ± 1.704	82.516
TCA	96.923 ± 0.720	94.808 ± 0.827	89.808 ± 0.769	84.519 ± 0.707	61.827 ± 1.162	55.385 ± 1.407	80.545
GFK	95.962 ± 1.380	91.154 ± 0.892	81.250 ± 2.803	69.519 ± 6.589	60.288 ± 2.463	52.692 ± 4.736	75.144
OTDA	98.365 ± 0.385	95.000 ± 1.162	89.231 ± 1.709	94.038 ± 0.490	**95.577 ± 1.439**	76.442 ± 2.085	**91.442**
Monge Mapping	**99.615 ± 0.192**	96.538 ± 1.704	**92.596 ± 1.736**	82.500 ± 0.720	90.192 ± 0.838	**76.635 ± 1.346**	89.679
JDOT	96.923 ± 0.892	95.481 ± 0.838	90.288 ± 0.638	**97.788 ± 0.490**	75.288 ± 2.910	67.115 ± 1.508	87.147
CNN Features							
CNN	80.192 ± 3.001	75.288 ± 2.031	74.135 ± 3.815	62.500 ± 2.544	69.808 ± 5.400	60.288 ± 5.830	70.000
DANN	86.219 ± 2.745	76.594 ± 3.284	75.219 ± 2.331	71.875 ± 3.168	79.969 ± 1.457	69.844 ± 3.205	77.000
KMM	70.673 ± 4.641	67.981 ± 3.065	59.808 ± 2.878	35.962 ± 1.532	67.596 ± 5.237	44.712 ± 7.111	57.788
KLIEP	74.231 ± 3.019	70.385 ± 2.120	64.038 ± 1.783	44.327 ± 3.485	68.462 ± 4.392	47.308 ± 6.568	61.458
uLSIF	72.788 ± 2.763	71.154 ± 3.454	65.673 ± 2.679	49.135 ± 4.821	67.981 ± 5.512	44.615 ± 5.930	61.891
PCA	75.096 ± 5.137	69.904 ± 4.142	63.269 ± 4.242	43.558 ± 5.011	63.558 ± 4.888	37.308 ± 7.477	58.782
TCA	51.442 ± 9.035	52.308 ± 9.169	49.712 ± 7.446	39.231 ± 10.305	55.962 ± 5.048	39.519 ± 9.841	48.029
GFK	65.577 ± 5.315	59.615 ± 7.776	57.596 ± 7.646	38.077 ± 5.485	52.115 ± 5.854	33.846 ± 5.039	51.138
OTDA	**89.327 ± 0.360**	84.519 ± 3.803	**77.981 ± 2.810**	77.212 ± 2.288	84.135 ± 2.995	83.269 ± 1.113	82.740
Monge Mapping	89.808 ± 1.676	82.981 ± 2.862	78.077 ± 2.076	74.712 ± 2.518	**87.115 ± 2.605**	79.423 ± 2.777	82.019
JDOT	**89.327 ± 0.932**	**85.673 ± 3.062**	77.596 ± 2.076	**77.692 ± 1.654**	82.692 ± 2.107	**84.519 ± 0.471**	**82.917**

Table A.1: Results for the comparative study of domain adaptation algorithms. The experiments are grouped in terms of features employed, and algorithm's nature. For each feature choice and target domain, the best acquired accuracy/score is shown in bold.

Appendix B

Optimization

In this appendix we briefly discuss concepts of mathematical optimization. In Section B.1 we introduce the subject, giving as examples the least squares and linear programming problems. In Section B.2 we introduce the Lagrange formalism for duality, and exemplify on the SVM algorithm. Finally, B.3 presents applications related to the thesis subject, such as system identification, controller design and the Franke-Wolfe optimization algorithm. This latter algorithm is useful for optimal transport algorithms (e.g. Mapping Estimation [Perrot et al., 2016] and JDOT [Courty et al., 2017]).

B.1 Introduction

Various aspects of this thesis are related to optimization of quantities. For instance, in Chapter **??** we described a mathematical theory for transportation of masses under least effort. The optimal transport problem, be it in Monge or Kantorovich formulation, is essentially a minimization problem. Mentions to optimization algorithms of the theory behind optimization problems are ubiquitous in this thesis, hence the need for a brief introduction on the subject. Here we follow the approach of [Boyd et al., 2004] on introducing optimization problems,

Definition B.1. A mathematical optimization problem (optimization problem in short) has the form,

$$\begin{array}{ll} \underset{\mathbf{x}}{\text{minimize}} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \leq 0 \\ & h_i(\mathbf{x}) = 0 \end{array}$$

where $\mathbf{x}$ is called optimization variable, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is called objective function and the functions f_i, $i = 1, \cdots, m$ are called inequality constraint functions, whereas h_i $i = 1, \cdots, p$ are called equality constraints. A vector $\mathbf{x}^*$ is called a solution to the optimization problem if: 1. It suffices the constraints, $f_i(\mathbf{x}^*) \leq 0$, $h_i(\mathbf{x}^*) = 0$, and 2. Among the points $\mathbf{x}$ that suffice the constraint, $\mathbf{x}^*$ has the smallest value with respect to the objective f_0.

In particular, two optimization problems are of high importance in this work: the least squares, and linear programming problems. The former is related to parameter estimation through data, and is used in the uLSIF algorithm derivation. The latter is of central importance to optimal transport, since the Kantorovich problem is a linear program.

Example B.1. (Least Squares) The Least Squares problem is related to the process of data fitting, that is, one has data $(\mathbf{x}_i, y_i)_{i=1}^n$, and a model $h_\theta(\mathbf{x})$ indexed by a parameter θ. One wants to determine θ, such that,

$$\hat{\theta} = \underset{\theta}{\text{argmin}}\ f_0(\theta) = \sum_{i=1}^{n}(y_i - h_\theta(\mathbf{x}_i))^2.$$

Note that this is an unconstrained optimization problem, namely, there are no constraints imposed on the optimized variable θ. Moreover, assuming h_θ to be a linear function, $h_\theta(\mathbf{x}_i) = \theta^T\mathbf{x}_i$, the objective may be expressed in matrix form,

$$f_0(\theta) = ||\mathbf{y} - \theta^T\mathbf{X}||_2^2,$$

where $\mathbf{y} = [y_1, \cdots, y_n]^T \in \mathbb{R}^{n\times 1}$ and $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$ are the data matrices. Note that the objective function is differentiable with respect to θ, hence a necessary condition for θ to be optimal is $\frac{\partial f_0}{\partial \theta} = 0$. This leads to the following condition,

$$(\mathbf{X}^T\mathbf{X})\theta = \mathbf{X}^T\mathbf{y}$$

which in turn implies that $\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. This condition is sufficient if, in addition, the second derivative of f_0 with respect to θ has negative determinant. Indeed,

$$\frac{\partial^2 f_0}{\partial \theta^2} = -\mathbf{X}^T\mathbf{X},$$

by noting that $\mathbf{X}^T\mathbf{X}$ is positive definite (thus have positive determinant), the matrix $\frac{\partial^2 f_0}{\partial \theta^2}$ has a negative determinant, thus the found solution is indeed the minimum of f_0.

Example B.2. (Linear Programming) A linear program is simply an optimization problem for which f_0 and f_i are linear, that is,

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{a}_i^T\mathbf{x} \leq b_i \end{aligned}$$

Thus a linear program is defined by vectors $\mathbf{c}$, $\mathbf{b}$ and matrix $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_m]$. In this terminology, the linear program has n variables and m constraints. As remarked by [Boyd et al., 2004], there are reliable algorithms for solving linear programs, although no closed form solution exists. These are the Simplex algorithm [Dantzig and Thapa, 2006], and the interior point method [Boyd et al., 2004].

B.2 Duality

We follow the approach of [Boyd et al., 2004], that is, first we define the Lagrangian, then the Lagrange dual function, then the Lagrange dual problem, and finally defining the Karush-Kuhn-

Tucker (KKT) conditions. We illustrate each concept using the SVM optimization problem defined in Chapter 3.

Definition B.2. (Lagrangian and Lagrange Dual Function [Boyd et al., 2004]) Let f_0 be an objective function, and f_i, h_i define the constraints for an optimization problem. The Lagrangian is denoted by $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, and for multipliers $\{\lambda_i\}_{i=1}^m$ and $\{\nu_j\}_{j=1}^p$, it has the following formula,

$$L(\mathbf{x}, \lambda, \nu) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{j=1}^{p} \nu_j h_j(\mathbf{x}),$$

where λ_i is the (Lagrange) multiplier associated with constraint f_i while ν_j is the multiplier associated with constraint h_j. In addition, for $\mathcal{F}$ being the set of feasible points, the Lagrange dual function is defined as follows,

$$g(\lambda, \nu) = \inf_{\mathbf{x} \in \mathcal{F}} L(\mathbf{x}, \lambda, \nu).$$

The Lagrangian is useful in a constrained optimization problem because it is a function that takes into account the objective function and the constraints. As it turns out, in the next definition we will use this function for rewriting the primal optimization problem into dual form. In addition, the Lagrange dual function represents a lower bound on the value of the objective function for feasible points [Boyd et al., 2004]. Still following [Boyd et al., 2004], the Lagrange dual problem arises from the question: what is the best lower bound that we can get?

Definition B.3. (Lagrange Dual Problem [Boyd et al., 2004]) Let f_0 be an objective function, and $\{f_i\}_{i=1}^m$, $\{h_j\}_{j=1}^p$ be constraints with multipliers λ_i and ν_j, respectively. The Lagrange dual problem is defined in terms of the Lagrange dual function,

$$\begin{aligned} &\text{maximize} && g(\lambda, \nu) \\ &\text{subject to} && \lambda \geq 0. \end{aligned}$$

The Lagrange dual problem is useful because sometimes it allows us to rewrite the primal problem in a nicer form. This is particularly useful for the SVM algorithm, as it allows the application of the so-called Kernel Trick.

Example B.3. (Dual SVM) The starting point for our analysis is the SVM primal problem. As derived by [Bishop, 2006], and stated in Chapter 3, it is a quadratic minimization problem under linear constraints,

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{argmin}} && \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to} && y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{aligned}$$

As follows, note that there are no equality constraints, hence we introduce Lagrange multipliers

λ_i for the inequality constraints. Therefore, the problem's Lagrangian is written as,

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \lambda_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1).$$

To define the Lagrange dual function, we may profit from the fact that L is differentiable with respect to $\mathbf{w}$ and b. We may thus find $\mathbf{w}^*$ and b^* by taking the derivatives of L with respect to $\mathbf{w}$ and setting them to zero,

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i, \frac{\partial L}{\partial b} = \sum_{i=1}^{n} \lambda_i y_i,$$

thus $\mathbf{w} = \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{n} \lambda_i y_i = 0$ are the first two conditions for optimality. We may thus plug these back into L, to get the Lagrange dual function,

$$g(\lambda) = \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j \mathbf{x}_i^T \mathbf{x}_j.$$

The dual Lagrange problem may be thus expressed in matrix form as,

$$\begin{aligned} \underset{\lambda}{\text{argmin}} \quad & \lambda^T \mathbb{1} - \frac{1}{2} \lambda^T \mathbf{K} \lambda \\ \text{subject to} \quad & \mathbf{y}^T \lambda = 0 \\ & \lambda \geq 0 \end{aligned}$$

As we just shown, the fact that the constraints were both differentiable allowed us to easily rewrite the primal optimization problem in terms of the Lagrange dual function, thus resulting in the dual problem. As follows, whenever both objective and constraints are differentiable, the conditions for optimality may be summarized in the so-called KKT conditions,

Definition B.4. (KKT Conditions for Optimality [Boyd et al., 2004]) Let $\mathbf{x}^*$ be solution of the primal problem, and (λ^*, ν^*) the solution of the dual. The KKT conditions for optimality are expressed as,

Equation	Description
$f_i(\mathbf{x}^*) \leq 0$ $h_i(\mathbf{x}^*) = 0$	Primal Feasibility
$\lambda_i \geq 0$ $\lambda_i f_i(\mathbf{x}^*) \geq 0$	Dual Feasibility
$\nabla f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\mathbf{x}) + \sum_{j=1}^{p} \nu_j \nabla h_j(\mathbf{x}) = 0$	Optimality Condition

Example B.4. (KKT Conditions for SVM) The KKT conditions for the SVM algorithm are

written as follows,

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i, \\ \sum_{i=1}^{n} \lambda_i y_i &= 0, \\ \lambda_i &\geq 0, \\ y_i(\mathbf{w}^T \mathbf{x}_i + b) &\geq 0, \\ \lambda_i(y_i h(\mathbf{x}_i) - 1) &= 0. \end{aligned}$$

Note that, as consequence, one has either $\lambda_i = 0$, or $y_i h(\mathbf{x}_i) = 1$. Hence, only a few multipliers $\lambda_i \neq 0$, and for those, the constraint $y_i h(\mathbf{x}_i) - 1 = 0$ is said to be active. The vectors $\mathbf{x}_i$ for which $\lambda_i \geq 0$ are said to be *support vectors*.

B.3 Applications

System Identification

System identification is a broad field whose purpose is to create a mathematical model for a system. This definition is rather general, and comprehends the mathematical modeling done in Section 2.1. In this section we are particularly interested in estimating the transfer function $G_p(s)$ from samples $\{(u(t_i), y(t_i))\}_{i=1}^{N}$.

For estimating G_p, we will further assume a particular model, called First Order plus Time Delay (FOPTD). This model is represented by the following transfer function,

$$G_p(s) = \frac{Ke^{-\theta s}}{\tau s + 1}, \tag{B.1}$$

this representation is particularly for non-linear systems with behavior similar to a first-order process, such as the two-tanks system. From the properties of the Laplace transform, Equation B.1 is associated with the following differential equation,

$$\tau \frac{d\hat{y}}{dt}(t) + \hat{y}(t) = Ku(t - \theta).$$

Notice that we denote the model's output as $\hat{y}(t)$ to make the difference between the true output $y(t)$. Especially when the true system is non-linear, $y(t)$ and $\hat{y}(t)$ may differ. Proceeding as in [George et al., 2007], we may isolate the derivative in the left-hand-side, and integrate the right-hand-side, obtaining,

$$\hat{y}(t) = -\frac{1}{\tau} \int_0^t \hat{y}(r)dr + \frac{K}{\tau} \int_0^t u(r - \theta)dr. \tag{B.2}$$

The goal of system identification in this particular context is estimating parameters K, τ and θ. We will further assume that $u(t)$ is a step function with unitary magnitude. As follows, rewriting

$\alpha_1 = -\tau^{-1}$, $\alpha_2 = K\tau^{-1}$, $\alpha_3 = \theta$, Equation B.2 may be expressed as,

$$\begin{aligned}
\hat{y}(t) &= \alpha_1 \int_0^t \hat{y}(r)dr + \alpha_2 \int_{-\alpha_3}^{t-\alpha_3} u(w)dw, \\
&= \alpha_1 \int_0^t \hat{y}(r)dr + \alpha_2 \int_0^{t-\alpha_3} dw, \\
&= \alpha_1 \int_0^t \hat{y}(r)dr + \alpha_2(t-\alpha_3), \\
&= \alpha_1 \int_0^t \hat{y}(r)dr + \alpha_2 t - \alpha_2\alpha_3. \\
&= \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_2\alpha_3 \end{bmatrix} \begin{bmatrix} \int_0^t y(r)dr \\ t \\ -1 \end{bmatrix}, \\
&= \alpha^T \phi(t).
\end{aligned}$$

Since we assumed previously that $u(t)$ is known (it is a step function), we only need samples $y_i = y(t_i)$, which are acquired either through the simulation of the true system, or through sensors. In this setting, we may as well denote use the notation,

$$\begin{aligned}
\Phi &= \begin{bmatrix} \phi(t_0) & \cdots & \phi(t_{N-1}) \end{bmatrix}^T, \\
\mathbf{Y} &= \begin{bmatrix} y_0 & \cdots & y_{N-1} \end{bmatrix}^T.
\end{aligned}$$

therefore we may use the following objective function to assess how well the choice of parameters α approximates the system's true output,

$$\begin{aligned}
J(\alpha) &= \sum_{i=0}^{N-1} (y(t_i) - \alpha^T \phi(t_i))^2, \\
&= \sum_{i=0}^{N-1} (\mathbf{Y}_i - \alpha^T \Phi_i)^2, \\
&= ||\mathbf{Y} - \alpha^T \Phi||_2^2,
\end{aligned}$$

notice that the minimization of the objective J with respect to α is a ordinary least squares problem, whose solution is given by,

$$\hat{\alpha}_{OLS} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{Y}.$$

The aforementioned methodology is applied to the two-tanks system in the following example.

Example B.5. (Direct Synthesis of PI Controller for Two Tanks System) Using the previous discussion, we consider the filling scenario for the two-tanks system, with an unitary step function. In this case, we estimate the following parameters for τ, K and θ,

$$\tau = 11.281s^{-1} \quad K = 1.276 \quad \theta = 0.091s^{-1}$$

We may evaluate the fidelity of this representation by comparing the FOPTD approximation with the mathematical model we built throughout Section 2.1. We recall that we may compare these two responses through the RMSE, defined by Equation 2.10. In Figure B.1, we show a comparison between these two models.

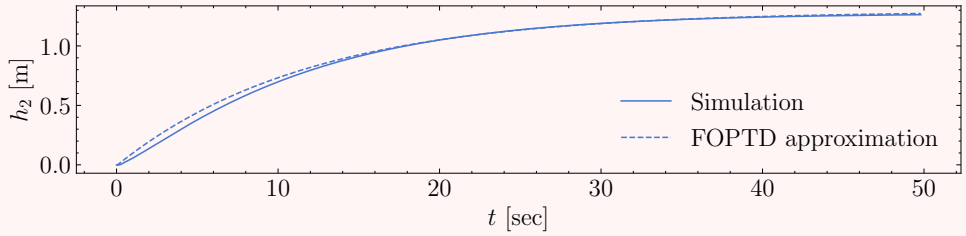


Figure B.1: Simulation of non-linear and estimated FOPTD models for the two-tanks system. Visually, the FOPTD model, which is linear, is much more close to the non-linear response than the linearization in Figure 2.4.

Notice that, when comparing the FOPTD model with the linearization obtained in Section 2.2. Especially, the RMSE between the non-linear and the FOPTD solution is 0.021, over 95.9% smaller than the linearized response.

Controller Design

Controller design is a broad field concerned with the definition of a control rule for G_c. As noted at the beginning of this section, a widespread form of controller is the PID controller, which is defined through three constants, K_p, K_i and K_d. These constants can be selected in a variety of ways. For instance, a common practice is to fine-tune these parameters by hand, or by using a heuristic mechanism such as Ziegler-Nichols method [Ziegler et al., 1942].

In this section we present the direct synthesis method devised by [Chen and Seborg, 2002]. This method relies on process models to directly find the PID parameters. Especially, the authors have considered the most commonly used models in the literature, such as the FOPTD model, which we described and estimated in the previous section for the two-tanks system.

In the following discussion, we will assume that there are no disturbances in the system. Using the block diagram exposed in Figure 2.6, notice that we may write the relationship between the output $Y(s)$ and the set-point signal $R(s)$ as,

$$\frac{Y(s)}{R(s)} = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)},$$

by rearranging the terms in this last equation, we may express $G_c(s)$ in closed form as follows,

$$G_c(s) = \frac{(Y(s)/R(s))}{G_p(s)(1 - Y(s)/R(s))}.$$

Notice that the closed-loop transfer function $Y(s)/R(s)$ directly influence the controller's transfer function. Therefore, as noted by [Chen and Seborg, 2002], one should specify a desired transfer function $G_\ell(s) = (Y(s)/R(s))_d$ in order to define $G_c(s)$. Hence, with G_p and G_ℓ at hand, G_c can be defined directly.

In this section we will discuss the design of a PI controller using the direct synthesis method. Especially, we recall that $G_p(s)$ is a FOPTD process with parameters τ, K and θ. Furthermore, let the desired closed-loop transfer function be a FOPTD as well, with $K = 1$, same delay θ,

and different time constant τ_c. In these conditions,

$$\begin{aligned}
G_c(s) &= \frac{\frac{e^{-\theta s}}{\tau_c s+1}}{\frac{Ke^{-\theta s}}{\tau s+1}(1-\frac{e^{-\theta s}}{\tau_c s+1})}, \\
&= \frac{\frac{1-\theta s}{\tau_c s+1}}{\frac{K(1-\theta s)}{\tau s+1}(1-\frac{1-\theta s}{\tau_c s+1})}, \\
&= \frac{\frac{1-\theta s}{\tau_c s+1}}{\frac{K(1-\theta s)}{\tau s+1}(\frac{\tau_c s+1}{\tau_c s+1}-\frac{1-\theta s}{\tau_c s+1})}, \\
&= \frac{\tau s+1}{K(\tau_c+\theta)s}, \\
&= \frac{\tau}{K(\tau_c+\theta)}+\frac{1}{K(\tau_c+\theta)}\frac{1}{s}, \\
&= K_p + K_i\frac{1}{s}.
\end{aligned}$$

As follows, G_c is a PI controller with constants K_p and K_i dependent on the parameters of the FOPTD model, and the desired closed-loop time constant τ_c. Notice that the direct synthesis method still has a parameter to tune, namely τ_c, but this is significantly easier than tuning all PID parameters. In the following example, we explore the direct synthesis method in the context of two-tanks system.

Example B.6. In this example we apply the direct synthesis method for the two-tanks system. Recall that we have previously fit a FOPTD model to the two-tanks system data using the ordinary least squares procedure. This yielded the following transfer function,

$$G_p(s) = \frac{1.276e^{-0.091s}}{11.281s+1}.$$

Moreover, we will define $r(t) = 0.75$ meters, which corresponds to the half of the maximum height. This way, we manipulate the input flow-rate $u(t)$ so that the second tank level stays at 0.75 meters. As follows, we may write the constants K_p and K_i as,

$$K_p = \frac{11.281}{1.276(\tau_c+0.091)}, \tag{B.3}$$

$$K_i = \frac{1}{1.276(\tau_c+0.091)}, \tag{B.4}$$

nonetheless, this is not sufficient for determining an ideal value for τ_c. To understand graphically the effect of τ_c on the close-loop response, Figure B.2 shows $y(t)$ for values of τ_c ranging from 0 to 10.

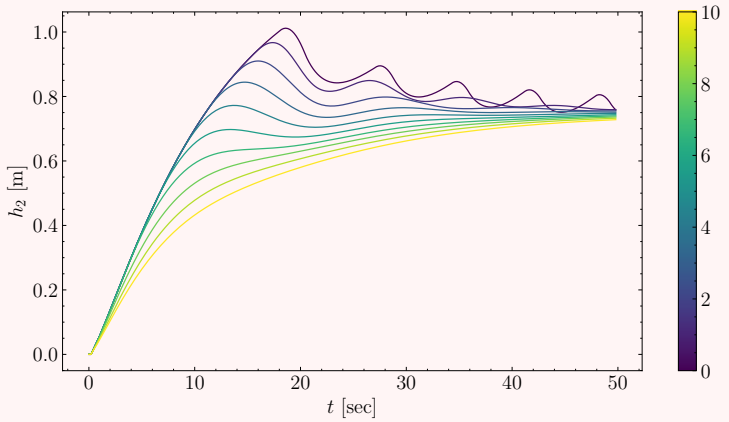


Figure B.2: Output $h_2(t) = y(t)$ for various values of τ_c, ranging from 1 to 10. As $\tau_c \to 0$, the response becomes more aggressive, with increasing overshoot. When reaching 0, the response takes too long to settle, and does not converge within the specified time range.

To choose a reasonable value, we will analyze the following statistics for the time response $y(t)$ as a function of τ_c,

- The overshoot as a percentage of the final value. Let y_{max} denote $\max_{t\in[t_0,t_f]} y(t)$, the previously mentioned percentage is denoted by,

$$\text{OS}\% = \frac{y_{max} - y(t_f)}{y(t_f)} \times 100.$$

- The rise time T_r. This quantity denotes the time passed for the system to go from 10% to 90% of the final value.

- Following [Chen and Seborg, 2002], we may evaluate the set-point tracking of each solution using the Integral Absolute Error (IAE). Moreover, the TV may be used to assess the required control effort. These two metrics are defined below,

$$IAE = \int_{t_0}^{t_f} |r(t) - y(t)|dt,$$
$$TV = \sum_{k=0}^{N-1} |u(t_{k+1}) - u(t_k)|.$$

The procedure for calculating each of these values is as follows: first, a range of 500 values for τ_c is created, between 0.5 and 10.0. For each τ_c in the said interval, the parameters K_p, K_i, and K_d are calculated using Equations B.3 and B.4. The obtained curves are shown in Figure B.3, and serve as a graphic method for choosing an optimal value for τ_c.

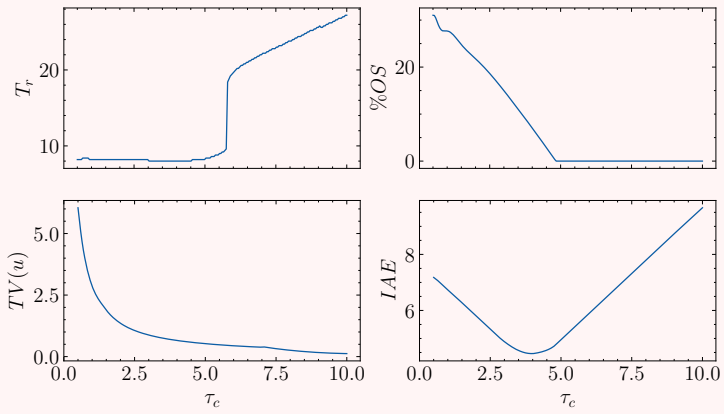


Figure B.3: Rise time T_r, overshooting as a percent of final value $OS\%$, input total variation $TV(u)$ and IAE as a function of τ_c.

Optimally, the τ_c that yields the smallest IAE value should be chosen. However, the yielded controller needs to suffice overshooting and rise time constraints. Nonetheless, Figure B.3 indicates that for $\tau_c = 4$ one has a small rise-time, and a reasonable overshooting (7%, while the minimum is 0%). The value for the total variation of the input signal is also reasonable, being 8% greater than the minimum value. In addition to this discussion, Figure B.4 shows the synthesized controller. Note that both h_1 and h_2 converge to the reference $r(t)$.

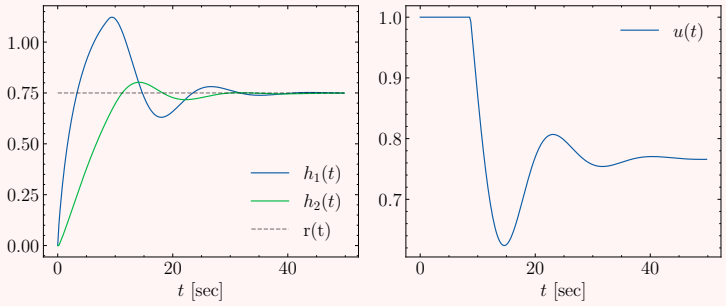


Figure B.4: Two tanks system controlled through a PID controller G_c designed using the direct synthesis method. Notice that, in the absence of G_c, the step-response converges to $h_2 = 1.2$ meters.

Derivation of Transfer Learning algorithms

Kullback-Leibler Importance Estimation

In this discussion, we follow the steps of [Sugiyama et al., 2008]. Our goal is deriving Equation 4.17. We recall that KLIEP works by assuming a linear model for the weight function β,

$$\beta(\mathbf{x}) = \sum_{i=1}^{b} \alpha_i \phi(\mathbf{x}),$$

Thus, the optimization algorithm in Equation 4.17 is derived by considering the Kullback-

Leibler divergence between distributions P_T and P_S^β,

$$\begin{aligned}\mathrm{KL}[P_T|P_S^\beta] &= \int P_T(\mathbf{x}) \log \frac{P_T(\mathbf{x})}{\beta(\mathbf{x})P_S^\beta(\mathbf{x})} d\mathbf{x}, \\ &= \int P_T(\mathbf{x}) \log \frac{P_T(\mathbf{x})}{P_S^\beta(\mathbf{x})} d\mathbf{x} - \int P_T(\mathbf{x}) \log \beta(\mathbf{x}) d\mathbf{x}, \\ &= KL[P_T||P_S^\beta] - \mathop{\mathbb{E}}_{\mathbf{x}\sim P_T}[\log(\beta(\mathbf{x}))].\end{aligned}$$

Notice that this latter equation is independent of β, and hence may be omitted during optimization. The second term may be approximated using the empirical mean, resulting in the objective function,

$$\begin{aligned}J(\mathbf{x}) &= \frac{1}{n_T} \sum_{i=1}^{n_T} \log(\beta(\mathbf{x}_T^i)), \\ &= \frac{1}{n_T} \sum_{i=1}^{n_T} \log\left(\sum_{j=1}^{b} \alpha_j \phi_j(\mathbf{x}_T^i) \right).\end{aligned} \tag{B.5}$$

Additionally to the objective defined by Equation B.5, we may further express the constraints for the maximization of J in terms of the normalization of $\hat{P}_T$,

$$\begin{aligned}1 &= \int \beta(\mathbf{x}) P_S(\mathbf{x}) d\mathbf{x}, \\ &= \mathop{\mathbb{E}}_{\mathbf{x}\sim P_S}[\beta(\mathbf{x})],\end{aligned}$$

again, the expectation may be approximation using the empirical mean. Therefore, the constraints may be expressed as,

$$\begin{aligned}1 &= \frac{1}{n_S} \sum_{i=1}^{n_S} \beta(\mathbf{x}_S^i), \\ &= \frac{1}{n_S} \sum_{i=1}^{n_S} \sum_{j=1}^{b} \alpha_j \phi_j(\mathbf{x}_S^i).\end{aligned} \tag{B.6}$$

Putting together the objective function given by B.5, and the constraints expressed by B.6, the KLIEP optimization problem can be expressed as,

$$\begin{aligned}\underset{\{\alpha_j\}_{j=1}^b}{\text{maximize}} \quad & \frac{1}{n_T} \sum_{i=1}^{n_T} \log\left(\sum_{j=1}^{b} \alpha_j \phi_j(\mathbf{x}_T^i) \right) \\ \text{subject to} \quad & \sum_{i=1}^{n_S} \sum_{j=1}^{b} \alpha_j \phi_j(\mathbf{x}_S^i) = n_S \\ & \alpha_j \geq 0, \forall j\end{aligned}$$

unconstrained Least Squares Importance Fitting

In this discussion, we adopt the steps of [Kanamori et al., 2009]. Our goal is deriving Equation 4.18. Thus, using the same linear model as in the previous example, the purpose of uLSIF is estimating the parameters α using least squares strategy. Under the assumption that there exists β such

that $P_T(\mathbf{x}) = \beta(\mathbf{x})P_S(\mathbf{x})$, the LSIF objective function may be written as,

$$\begin{aligned} J(\alpha) &= \frac{1}{2}\int (\hat{\beta}(\mathbf{x}) - \beta(\mathbf{x}))^2 P_S(\mathbf{x})d\mathbf{x}, \\ &= \frac{1}{2}\int \hat{\beta}(\mathbf{x})^2 P_S(\mathbf{x})d\mathbf{x} + \frac{1}{2}\int \beta(\mathbf{x})^2 P_S(\mathbf{x})d\mathbf{x} - \int \hat{\beta}(\mathbf{x})\beta(\mathbf{x})P_S(\mathbf{x})d\mathbf{x}, \\ &= \frac{1}{2}\int \hat{\beta}(\mathbf{x})^2 P_S(\mathbf{x})d\mathbf{x} - \int \hat{\beta}(\mathbf{x})\beta(\mathbf{x})P_S(\mathbf{x})d\mathbf{x} + cte, \end{aligned}$$

where from the second to third line $\beta(\mathbf{x})P_S(\mathbf{x})$ was substituted by $P_T(\mathbf{x})$. Moreover, cte corresponds to the term $\frac{1}{2}\int \beta(\mathbf{x})^2 P_S(\mathbf{x})d\mathbf{x}$, which does not depend on α. By expanding the third expression in terms of α, one has,

$$J(\alpha) = \frac{1}{2}\sum_{i=1}^{b}\sum_{j=1}^{b}\alpha_i\alpha_j\left(\int \phi_i(\mathbf{x})\phi_j(\mathbf{x})P_S(\mathbf{x})d\mathbf{x}\right) - \sum_{i=1}^{b}\alpha_i\left(\int \phi_i(\mathbf{x})P_T(\mathbf{x})d\mathbf{x}\right),$$

thus, by approximating the expectations in this latter equation with empirical averages, the objective may be expressed in matrix form using,

$$\begin{aligned} H_{ij} &= \frac{1}{n_S}\sum_{k=1}^{n_S}\phi_i(\mathbf{x}_S^k)\phi_j(\mathbf{x}_S^k), \\ h_i &= \frac{1}{n_T}\sum_{k=1}^{n_T}\phi_i(\mathbf{x}_T^k), \end{aligned}$$

which implies,

$$J(\alpha) = \frac{1}{2}\alpha^T\mathbf{H}\alpha - \mathbf{h}^T\alpha.$$

Thus, with addition to the Lagrange multiplier $\lambda \geq 0$, one has the following optimization problem [Kanamori et al., 2009]:

$$\begin{array}{ll} \underset{\alpha\in\mathbb{R}^b}{\text{minimize}} & \frac{1}{2}\alpha^T\mathbf{H}\alpha - \mathbf{h}^T\alpha + \lambda\mathbb{1}_b^T\alpha, \\ \text{subject to} & \alpha_i \geq 0 \end{array}$$

Kernel Mean Matching

In the following discussion we adopt the steps of [Gretton et al., 2009]. Thus, KMM seeks to minimize $MMD(P_S^\beta, P_T)$. By noticing that $P_S^\beta(\mathbf{x}) = \beta(\mathbf{x})P_S(\mathbf{x})$, one has:

$$\text{MMD}(P_S^\beta, P_T)^2 = \left|\left|\frac{1}{n_S}\sum_{i=1}^{n_S}\beta(\mathbf{x}_S^i)\phi(\mathbf{x}_S^i) - \frac{1}{n_T}\sum_{i=1}^{n_T}\phi(\mathbf{x}_T^i)\right|\right|_{\mathcal{H}}^2,$$

let $K_{SS}(i,j) = K(\mathbf{x}_S^i, \mathbf{x}_S^j)$, $K_{ST}(i,j) = K(\mathbf{x}_S^i, \mathbf{x}_T^j)$, and $K_{TT}(i,j) = K(\mathbf{x}_T^i, \mathbf{x}_T^j)$. Moreover, let $\kappa_i = \frac{n_T}{n_S}\sum_{j=1}^{n_T}K_{ST}(i,j)$ and $\beta_i = \beta(\mathbf{x}_S^i)$, using the properties of the RKHS,

$$\begin{aligned} MMD(P_S^\beta, P_T)^2 &= \left\langle \frac{1}{n_S}\sum_{i=1}^{n_S}\beta_i\phi(\mathbf{X}_S^i) - \frac{1}{n_T}\sum_{i=1}^{n_T}\phi(\mathbf{x}_T^i), \frac{1}{n_S}\sum_{i=1}^{n_S}\beta_i\phi(\mathbf{X}_S^i) - \frac{1}{n_T}\sum_{i=1}^{n_T}\phi(\mathbf{x}_T^i)\right\rangle_{\mathcal{H}}, \\ &= \frac{1}{n_S^2}\sum_{i,j}^{n_S}\beta_i K_{SS}(i,j)\beta_j - \frac{2}{n_T n_S}\sum_{i=1}^{n_S}\beta_i\sum_{j=1}^{n_T}\phi(\mathbf{x}_T^j) + \frac{1}{n_T^2}\sum_{i,j}^{n_T}K_{TT}(i,j), \\ &= \frac{1}{n_S^2}\beta^T K_{SS}\beta - \frac{2}{n_T^2}\kappa^T\beta + cte, \end{aligned}$$

which is quadratic on β. Notice that cte represents a term that does not depend on β, and hence can be neglected during optimization. Following [Gretton et al., 2009], the constraints on β are slightly different from KLIEP. They introduce a bound term B for which $\beta_i \in [0, B]$. Hence, the weights β_i are found using KMM through the optimization problem:

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2}\beta^T K\beta - \kappa^T\beta \\ \text{subject to} \quad & |\sum_{i=1}^{n_S}\beta_i - n_S| \leq B\sqrt{n_S} \\ & \beta_i \in [0, B], \forall j \end{aligned}$$

Transfer Component Analysis

We recall that, as discussed in Chapter 4, the MMD distance between empirical probability distributions $\hat{P}_S$ and $\hat{P}_T$ can be written as,

$$\text{MMD}(\hat{P}_S, \hat{P}_T)\frac{1}{n_S^2}\sum_{i,j}K_{SS}(i,j) - \frac{2}{n_S n_T}\sum_{i,j}K_{ST}(i,j) + \frac{1}{n_T^2}\sum_{i,j}K_{TT}(i,j),$$

in terms of kernel matrix $K_{SS}(i,j) = k(\mathbf{x}_S^i, \mathbf{x}_S^j)$, $K_{ST}(i,j) = k(\mathbf{x}_S^i, \mathbf{x}_T^j)$, $K_{TT}(i,j) = k(\mathbf{x}_T^i, \mathbf{x}_T^j)$. In this section we follow [Pan et al., 2010] for the derivation of the TCA algorithm.

First, we may rewrite the intiial equation in matrix form by defining matrices $\mathbf{K} \in \mathbb{R}^{n\times n}$ and $\mathbf{L} \in \mathbb{R}^{n\times n)}$, where $n = n_S + n_T$ is the total number of samples. These matrices are expressed as follows,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{SS} & \mathbf{K}_{ST} \\ \mathbf{K}_{TS} & \mathbf{K}_{TT} \end{bmatrix},$$

$$\mathbf{L}_{ij} = \begin{cases} 1/n_S^2 & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_S \\ 1/n_T^2 & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_T \\ -2/n_S n_T & \text{otherwise} \end{cases}$$

Hence, $\text{MMD}(P_S, P_T) = \text{Tr}(\mathbf{KL})$. In this fashion, the feature transfer process is made by an embedding matrix $W \in \mathbb{R}^{n\times m}$, for $m \ll n$. This embedding may be incorporated in the MMD formula by substituting $\mathbf{K}$ for $\mathbf{KWW}^T\mathbf{K}$. As noted by [Pan et al., 2010], an additional constraint needs to be ensured to avoid the trivial solution $\mathbf{W} = \mathbf{0}$. Let $\mathbf{H}$ be the centering matrix, given by,

$$\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbb{1}_n\mathbb{1}_n^T,$$

the TCA optimization problem is defined as,

$$\begin{aligned} \underset{\mathbf{W}}{\text{minimize}} \quad & \text{Tr}(\mathbf{W}^T\mathbf{W}) + \mu\text{Tr}(\mathbf{W}^T\mathbf{KLKW}) \\ \text{subject to} \quad & \mathbf{W}^T\mathbf{KHKW} = \mathbf{I}_n \end{aligned}$$

This trace minimization problem, however, involves a non-convex constraint term. This may be avoided by rewriting the previous problem using Lagrange multipliers. Consider the multipliers $Z \in \mathbb{R}^{n\times n}$, then, the previous objective function becomes:

$$J(\mathbf{W}) = \text{Tr}(\mathbf{W}^T(\mathbf{I}_n + \lambda\mathbf{KLK})\mathbf{W}) - \mathbf{Tr}((\mathbf{W}^T\mathbf{KHKW} - \mathbf{I}_n)\mathbf{Z}), \tag{B.7}$$

as follows, since the trace is a linear operator we may state the optimality condition as,

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{W}} = 0 &\iff (\mathbf{I}_n + \lambda \mathbf{KLK})\mathbf{W} - \mathbf{KHKWZ} = 0, \\ &\iff \mathbf{W}^T(\mathbf{I}_n + \lambda \mathbf{KLK})\mathbf{W} = \mathbf{W}^T\mathbf{KHKWZ},\end{aligned} \tag{B.8}$$

thus, plugging Equation B.8 back into B.7, one may easily see that $J(\mathbf{W}) = \text{Tr}(\mathbf{Z})$. Finally, $\mathbf{Z}$ may be isolated in Equation B.8, resulting in,

$$\begin{aligned}J(\mathbf{W}) &= (\mathbf{W}^T\mathbf{KHKW})^{\dagger}\mathbf{W}^T(\mathbf{I}_n + \lambda \mathbf{KLK})\mathbf{W}, \\ &= (\mathbf{W}^T(\mathbf{I}_n + \lambda \mathbf{KLK})\mathbf{W})^{-1}\mathbf{W}^T\mathbf{KHKW}.\end{aligned} \tag{B.9}$$

The minimization of Equation B.9 is done by performing the eigenvalue decomposition of $(\mathbf{I}_n + \lambda \mathbf{KLK})^{-1}\mathbf{KHK}$. The matrix $\mathbf{W}$ is the matrix whose columns are the m eigenvectors associated with the largest m eigenvalues. As stated in [Matasci et al., 2011], once $\mathbf{W}$ has been calculated, the samples may be transformed using $\tilde{\mathbf{X}} = \mathbf{KW}$.

Geodesic Flow Kernel

In this section we provide the construction of the geodesic flow involved in the calculation of $\mathbf{G}$. In our discussion, we follow the steps presented in [Gong et al., 2012] and their supplementary material. Thus, let $\mathbf{R}_S \in \mathbb{R}^{D\times(D-d)}$ be the orthogonal complement of $\mathbf{B}_S$, that is, $\mathbf{R}_S^T\mathbf{B}_S = \mathbf{0}$. We may further express $\mathbf{\Phi}(t)$, for arbitrary t, as follows,

$$\mathbf{\Phi}(t) = \mathbf{B}_S\mathbf{U}_1\mathbf{\Gamma}(t) - \mathbf{R}_S\mathbf{U}_2\mathbf{\Sigma}(t), \tag{B.10}$$

where $\mathbf{U}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{U}_2 \in \mathbb{R}^{(D-d)\times d}$ are orthonormal matrices, given by a pair of Singular Value Decompositions (SVDs),

$$\mathbf{B}_S^T\mathbf{B}_T = \mathbf{U}_1\mathbf{\Gamma}\mathbf{V}^T, \tag{B.11}$$

$$\mathbf{R}_S^T\mathbf{B}_T = -\mathbf{U}_2\mathbf{\Sigma}\mathbf{V}^T, \tag{B.12}$$

where $\mathbf{\Gamma}$ and $\mathbf{\Sigma}$ are $d \times d$ diagonal matrices, whose elements in the diagonal γ_i and σ_i are the singular values of their respective matrices. Following [Zhu and Knyazev, 2013], there is a link between these quantities and the so-called principal angles between $\mathbf{B}_S$ and $\mathbf{B}_T$.

Following the supplementary materials of [Gong et al., 2012], the principal angles θ_i are defined as follows,

$$cos(\theta_i) = \max_{\mathbf{s}_i \in \text{span}(\mathbf{B}_S)} \max_{\mathbf{t}_i \in \text{span}(\mathbf{B}_T)} \frac{\langle \mathbf{s}_i, \mathbf{t}_i \rangle}{||\mathbf{s}_i||\ ||\mathbf{t}_i||}, \tag{B.13}$$

provided that for each $\mathbf{s}_k \in \text{span}(\mathbf{B}_S)$, and for each $\mathbf{t}_k \in \text{span}(\mathbf{B}_T)$, one has $\mathbf{s}_i \perp \mathbf{s}_k$ and $\mathbf{t}_i \perp \mathbf{t}_k$. In this case, $\mathbf{s}_i$ and $\mathbf{t}_i$ are the principal vectors associated with the angle θ_i.

In addition to Equation B.13, consider the SVD decomposition presented in Equation B.11. As stated in Theorem 2.1 of [Zhu and Knyazev, 2013], one has $cos(\theta_i) = \gamma_i$, hence the SVD decomposition can be used to calculate the principal angles. Hence, the diagonal elements of Γ and Σ are $cos(\theta_i)$ and $sin(\theta_i)$. More generally, $\mathbf{\Gamma}(t) = cos(t\theta_i)$, $\mathbf{\Sigma}(t) = sin(t\theta_i)$.

With these definitions, we are able to calculate $\mathbf{G}$. To do so, we proceed as [Gong et al., 2012] in their supplementary materials. Let $\mathbf{\Omega}$ be a matrix defined as follows,

$$\mathbf{\Omega}^T = \begin{bmatrix}\mathbf{B}_S & \mathbf{R}_S\end{bmatrix}\begin{bmatrix}\mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2\end{bmatrix},$$

the geodesic flow, expressed by Equation B.10 can be rewritten using $\mathbf{\Omega}$,

$$\mathbf{\Phi}(t) = \mathbf{\Omega}^T \begin{bmatrix} \mathbf{\Gamma}(t) \\ -\mathbf{\Sigma}(t) \end{bmatrix},$$

plugging back this latter equation into Equation 4.20, one has,

$$\begin{aligned}
\mathbf{G} &= \int_0^1 \mathbf{\Phi}(t)\mathbf{\Phi}(t)^T dt, \\
&= \int_0^1 \mathbf{\Omega}^T \begin{bmatrix} \mathbf{\Gamma}(t) \\ \mathbf{\Sigma}(t) \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma}(t) & \mathbf{\Sigma}(t) \end{bmatrix} \mathbf{\Omega}^T dt, \\
&= \mathbf{\Omega}^T \left(\int_0^1 \begin{bmatrix} \mathbf{\Gamma}(t)\mathbf{\Gamma}(t) & -\mathbf{\Gamma}(t)\mathbf{\Sigma}(t) \\ \mathbf{\Gamma}(t)\mathbf{\Sigma}(t) & -\mathbf{\Sigma}(t)\mathbf{\Sigma}(t) \end{bmatrix} dt \right) \mathbf{\Omega}, \\
&= \mathbf{\Omega}^T \begin{bmatrix} \int_0^1 \mathbf{\Gamma}(t)\mathbf{\Gamma}(t)dt & -\int_0^1 \mathbf{\Gamma}(t)\mathbf{\Sigma}(t)dt \\ \int_0^1 \mathbf{\Gamma}(t)\mathbf{\Sigma}(t)dt & -\int_0^1 \mathbf{\Sigma}(t)\mathbf{\Sigma}(t)dt \end{bmatrix} \mathbf{\Omega}, \\
&= \mathbf{\Omega}^T \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_2 & \Lambda_3 \end{bmatrix}.
\end{aligned}$$

Note that, since both $\mathbf{\Gamma}$ and $\mathbf{\Sigma}$ are diagonal matrices, their product and consequently their integrals are also diagonal matrices. We may calculate their diagonals analytically, as done in [Gong et al., 2012],

$$\begin{aligned}
\Lambda_{1i} &= \int_0^1 cos^2(t\theta_i)dt = 1 + \frac{sin(2\theta_i)}{2\theta_i}, \\
\Lambda_{2i} &= \int_0^1 cos(t\theta_i)sin(t\theta_i)dt = \frac{cos(2\theta_i) - 1}{2\theta_i}, \\
\Lambda_{3i} &= \int_0^1 sin^2(t\theta_i)dt = 1 - \frac{sin(2\theta_i)}{2\theta_i}.
\end{aligned}$$

Appendix C

Building a Dynamic System's Simulator

C.1 Linear State Space Simulation

When the dynamical system represented by $\mathbf{f}$ and $\mathbf{g}$ is linear, there exists a closed-form solution $\mathbf{x}$. The form of this solution will be later useful for analyzing the stability of the dynamical system, even when it is $\mathbf{f}$ and $\mathbf{g}$ are not linear. We begin our analysis with the homogeneous case,

Proposition C.1. *Let* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *and* $\mathbf{x}(t) \in \mathbb{R}^n$, $\forall t$. *The solution to the differential equation*

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}, \tag{C.1}$$

is given by $\mathbf{x}(t) = \mathbf{x}(0)e^{\mathbf{A}t}$, *where* $e^{\mathbf{A}t}$ *is defined as,*

$$e^{\mathbf{A}t} = \mathbf{I} + \sum_{k=1}^{\infty} \frac{\mathbf{A}^k t^k}{k!}$$

Proof. We begin by showing that $\mathbf{x}(t) = \mathbf{x}(0)e^{\mathbf{A}t}$ is a solution to Equation C.1. To that end, consider the Taylor expansion given in the definition,

$$\begin{aligned}
\frac{d}{dt}\mathbf{x} &= \mathbf{x}(0)\frac{d}{dt}\left(\mathbf{I} + \sum_{k=1}^{\infty} \frac{\mathbf{A}^k t^k}{k!}\right), \\
&= \mathbf{x}(0)\left(\sum_{k=1}^{\infty} \frac{A^k t^{k-1}}{(k-1)!}\right), \\
&= \mathbf{x}(0)\left(\mathbf{A}\sum_{m=0}^{\infty} \frac{\mathbf{A}^m t^m}{m!}\right), \\
&= \mathbf{x}(0)\mathbf{A}\left(\mathbf{I} + \sum_{m=1}^{\infty} \frac{\mathbf{A}^m t^m}{m!}\right), \\
&= \mathbf{A}\left(\mathbf{x}(0)e^{\mathbf{A}t}\right), \\
&= \mathbf{A}\mathbf{x},
\end{aligned}$$

where from the third to the fourth line we used the substitution $m = k - 1$.

□

The next step in complexity is adding inputs to the dynamic system, using the state-space representation. The solution characterization is given in the next proposition,

Proposition C.2. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{B} \in \mathbb{R}^{n\times m}$. *The linear dynamical system,*

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{C.2}$$

has as solution,

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{C.3}$$

Proof. We begin by re-writing Equation C.2 as,

$$e^{-\mathbf{A}t}\dot{\mathbf{x}} - e^{-\mathbf{A}t}\mathbf{A}\mathbf{x} = e^{-\mathbf{A}t}\mathbf{B}\mathbf{u}(t),$$

notice that the left hand side of the latter equation is equal to $\frac{d}{dt}e^{-\mathbf{A}t}\mathbf{x}(t)$. Therefore, through integration we have the following,

$$\frac{d}{dt}e^{-\mathbf{A}t}\mathbf{x}(t) = e^{-\mathbf{A}t}\mathbf{B}\mathbf{u}(t),$$
$$e^{-\mathbf{A}t}\mathbf{x}(t) - \mathbf{x}(0) = \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau,$$

therefore, by multiplying both sides of this latter equation, one has the desired formula,

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau.$$

□

The two previous propositions have shown that the solutions of a linear system can be found in closed form. Therefore, provided the state-space model given by matrices $\mathbf{A}$ and $\mathbf{B}$, the input $\mathbf{u}(t)$, and the initial condition $\mathbf{x}(0)$, one may retrieve the solution $\mathbf{x}(t)$.

C.2 Nonlinear State Space Simulation

In this Section we follow the approach of [Burden and Faires, 2011] for discussing two methods for solving nonlinear differential equations numerically: the Euler method, and the Runge-Kutta method. Note that solving an ODE numerically is equivalent to find samples $\mathbf{x}_i$ from the solution $\mathbf{x}(t)$ at time-steps $t_i = ih$, where h is the time-step. We will set our discussion in a more general scenario than [Burden and Faires, 2011], since we consider ODEs of the form,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)),$$

whereas [Burden and Faires, 2011] supposes $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$. This generalization step can be done with little effort, as the two cases are analogous.

Euler Method

The Euler method is based on the Taylor expansion of first order for the function $\mathbf{x}(t)$. Thus, let $t_i = ih$ for a fixed time-step h. The first order Taylor expansion

$$\begin{aligned}\mathbf{x}(t_{i+1}) &= \mathbf{x}(t_i + h),\\ &\approx \mathbf{x}(t_i) + h\dot{\mathbf{x}}(t_i),\\ &= \mathbf{x}(t_i) + h\mathbf{f}(\mathbf{x}(t_i), \mathbf{u}(t_i)),\end{aligned}$$

therefore $\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)$. Therefore we may calculate the state-vector $\mathbf{x}$ at time-step t_{i+1} from the values computed at time-step t_i. Assuming $\mathbf{x}_0$ given, the simulation of a non-linear state-space using Euler Method is summarized in Algorithm 4.

Algorithm 4 Numerical simulation of nonlinear dynamical system through Euler's method

Input: Initial condition $\mathbf{x}_0$, input vector at each time-step $\mathbf{u}_i$, $i = 0, \cdots, T-1$, time-step h.
  Initialize $i = 0$, $t = 0$,
  while $i < T - 1$ **do**
    $\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)$
    $i \leftarrow i + 1$
  end while
Output: Sampling instants t_i, state-vectors $\mathbf{x}_i$, for $t_i = ih$.

Runge-Kutta Method

The previous algorithm approximates $\mathbf{x}(t_i + h)$ using Taylor series expansion of order 1. Note that intuitively, the accuracy of such method is defined by how small h is, reflecting how small is the approximated neighborhood. Another way to increase accuracy is considering higher order terms. There are various ways to use these terms. In this discussion, we take the approach known as Runge-Kutta order four for doing so, due to its wide adoption [Burden and Faires, 2011]. Thus, we calculate the following quantities,

$$\begin{aligned}k_1 = h\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i), k_2 = h\mathbf{f}(\mathbf{x}_i + 0.5k_1, \mathbf{u}_i),\\ k_3 = h\mathbf{f}(\mathbf{x}_i + 0.5k_2, \mathbf{u}_i), k_4 = h\mathbf{f}(\mathbf{x}_i + k_3, \mathbf{u}_i),\\ \mathbf{x}_{i+1} = \mathbf{x}_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).\end{aligned}$$

This method has better numerical accuracy in terms of h than the Euler method. As done previously, we may also express these computations as an algorithm,

Algorithm 5 Numerical simulation of nonlinear dynamical system through fourth order Runge-Kutta method

Input: Initial condition $\mathbf{x}_0$, input vector at each time-step $\mathbf{u}_i$, $i = 0, \cdots, T-1$, time-step h.
  Initialize $i = 0$, $t = 0$,
  while $i < T - 1$ **do**
    $k_1 = h\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)$
    $k_2 = h\mathbf{f}(\mathbf{x}_i + 0.5k_1, \mathbf{u}_i)$
    $k_3 = h\mathbf{f}(\mathbf{x}_i + 0.5k_2, \mathbf{u}_i)$
    $k_4 = h\mathbf{f}(\mathbf{x}_i + k_3, \mathbf{u}_i)$
    $\mathbf{x}_{i+1} = \mathbf{x}_i + \dfrac{k_1 + 2k_2 + 2k_3 + k_4}{6}$
    $i \leftarrow i + 1$
  end while
Output: Sampling instants t_i, state-vectors $\mathbf{x}_i$, for $t_i = ih$.

Discrete PID implementation

In the previous discussion we have assumed that $\mathbf{u}_i, i = 0, \cdots, T-1$ is known for all time-steps. This is true when one wants to simulate the non-linear state-space for an impulse or step function. However, when a controller is implemented, this is usually not true. Recalling definition 2.4, a controller is a function $C(e(t))$ where $e(t)$ is the error signal. In this section we will describe how to simulate a non-linear state-space model controller by a PID controller.

We recall that in a non-linear state-space $\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))$. Let us assume that one has a single input $u(t)$, a single output $y(t)$, and that $u(t)$ does not influence $\mathbf{g}$, as it was the case in the dynamical systems treated in this thesis. Furthermore, let us assume that we can query the function $\mathbf{g}(\mathbf{x}_i)$ in order to acquire the output $y_i \in \mathbb{R}$ at time-step t_i. We similarly assume that a set-point $r(t) = y_{sp}\ \forall t$ has been defined. This implies that the error at each time-step is given by $e_i = y_i - y_{sp}$. We explore two discrete approaches for the PID controller,

$$u_{i+1} = K_p e_i + K_i h \sum_{k=0}^{i} e_k + K_d \frac{e_i - e_{i-1}}{h}, \tag{C.4}$$

$$u_{i+1} = u_i + K_p(e_i - e_{i-1}) + K_i h e_i + \frac{e_i - 2e_{i-1} + e_{i-2}}{h}, \tag{C.5}$$

we call these two approaches as direct and recursive implementations, respectively. Note that in Equation C.4, one is forced to maintain an history of previous errors for calculating the integral term, whereas the recursive approach one only needs the previous 2 errors. We thus modify the Runge-Kutta approach for handling manipulated inputs,

Algorithm 6 Numerical simulation of nonlinear dynamical system with PID controller through fourth order Runge-Kutta method.

Input: Initial condition $\mathbf{x}_0$, controller function C, input set-point y_{sp}, $i = 0, \cdots, T-1$, time-step h.
  Initialize $i = 0$, $t = 0$,
  while $i < T - 1$ **do**
    $y_i = \mathbf{g}(\mathbf{x}_i)$
    $e_i = y_i - y_{sp}$
    $u_i = C(e_i)$
    $k_1 = h\mathbf{f}(\mathbf{x}_i, u_i)$
    $k_2 = h\mathbf{f}(\mathbf{x}_i + 0.5k_1, u_i)$
    $k_3 = h\mathbf{f}(\mathbf{x}_i + 0.5k_2, u_i)$
    $k_4 = h\mathbf{f}(\mathbf{x}_i + k_3, u_i)$
    $\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$
    $i \leftarrow i + 1$
  end while
Output: Sampling instants t_i, state-vectors $\mathbf{x}_i$, inputs u_i, outputs y_i, for $t_i = ih$.

Appendix D

Omitted Proofs and Derivations

D.1 Chapter 2 Proofs

Theorem 2.1.

We will show the proof only for $\mathbf{f}$, as the case for $\mathbf{g}$ is analogous. Let $\Delta\mathbf{x}(t) = \mathbf{x}(t) - \bar{\mathbf{x}}$ and $\Delta\mathbf{u}(t) = \mathbf{u} - \bar{\mathbf{u}}$ be small perturbations. Since $\bar{\mathbf{x}}$ (resp. $\bar{\mathbf{u}}$) is fixed, the following holds,

$$\begin{aligned}\Delta\dot{\mathbf{x}}(t) &= \frac{d}{dt}(\mathbf{x}(t) - \bar{\mathbf{x}}(t)),\\ &= \dot{\mathbf{x}}(t),\\ &= \mathbf{f}(\mathbf{x}, \mathbf{u}),\\ &= \mathbf{f}(\Delta\mathbf{x} + \bar{\mathbf{x}}, \Delta\mathbf{u} + \mathbf{u}).\end{aligned}$$

Note that if $\Delta\mathbf{x}$ (resp. $\Delta\mathbf{u}$) is small enough, we may approximate the term $\mathbf{f}(\Delta\mathbf{x}+\bar{\mathbf{x}}, \Delta\mathbf{u}+\mathbf{u})$ as,

$$\begin{aligned}\mathbf{f}(\Delta\mathbf{x} + \bar{\mathbf{x}}, \Delta\mathbf{u} + \mathbf{u}) &\approx \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\Delta\mathbf{x} + \frac{\partial\mathbf{f}}{\partial\mathbf{u}}\Delta\mathbf{u},\\ &= \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\Delta\mathbf{x} + \frac{\partial\mathbf{f}}{\partial\mathbf{u}}\Delta\mathbf{u}.\end{aligned}$$

Thus, by identifying $\dfrac{\partial\mathbf{f}}{\partial\mathbf{x}}$ as $\mathbf{A}$, and $\dfrac{\partial\mathbf{f}}{\partial\mathbf{u}}$ as $\mathbf{B}$, one retrieves the first equation in the system 2.7. Repeating the process for $\mathbf{g}$ derives the second equation.

D.2 Chapter 3 Proofs

Theorem 3.1.

Let $\mathbf{X}_i$ and $\mathbf{X}_j$ be random variables, and consider the following function:

$$h(t) = \mathbb{E}[((\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j))^2]$$

We may rewrite this last expression using the linearity of the expectation, and Equation 3.2,

$$\begin{aligned}h(t) &= t^2\mathbb{E}[(\mathbf{X}_i - \mu_i)^2] + \mathbb{E}[(\mathbf{X}_j - \mu_j)^2] + 2t\mathbb{E}[(\mathbf{X}_i - \mu_i)(\mathbf{X}_j - \mu_j)],\\ &= t^2\sigma_i^2 + \sigma_j^2 + 2t\Sigma_{ij}.\end{aligned}$$

This is a quadratic equation on t, with coefficients $a = \sigma_i^2$, $b = -2\Sigma_{ij}$ and $c = \sigma_j^2$. The discriminant $\Delta = b^2 - 4ac$ is,

$$\Delta = 4\Sigma_{ij}^2 - 4\sigma_i^2\sigma_j^2.$$

Notice that $h(t)$ is the expectation of the random variable $((\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j))^2$, which is non-negative. Hence, $h(t) \geq 0$. Thus, $h(t)$ must have at most one real root, which is equivalent to $\Delta \leq 0$. This yields the following inequality,

$$-\sigma_i\sigma_j \leq \Sigma_{ij} \leq \sigma_i\sigma_i,$$

which is equivalent to $-1 \leq R_{ij} \leq 1$, proving that $R_{ij} \in [-1, 1]$. Now, consider $|R_{ij}| = 1$, which implies that $h(t)$ has only one real root. Since $((\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j))^2 \geq 0$, $\mathbb{E}[((\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j))^2] = 0$ if and only if,

$$\begin{aligned}\mathbb{P}(((\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j))^2 = 0) = 1,\\ \mathbb{P}((\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j) = 0) = 1,\end{aligned}$$

this last equation says that, with probability 1,

$$\begin{aligned}(\mathbf{X}_i - \mu_i)t + (\mathbf{X}_j - \mu_j) = 0,\\ \mathbf{X}_j = (\mu_j + t\mu_i) - t\mathbf{X}_i,\end{aligned}$$

where t is the root of $h(t)$, that is, $t = -\Sigma_{ij}/\sigma_i^2$. Notice that this last equation express a linear relationship between $\mathbf{X}_j$ and $\mathbf{X}_i$. Moreover, the angular coefficient $-t = \sigma_{ij}/\sigma_i^2$ has the same sign as Σ_{ij}, and therefore R_{ij}.

D.3 Chapter 4 Proofs

Proposition 4.1.

Start with Equation 4.7 and notice that $\log(\gamma_{ij}/K_{ij})$ is simply $\log(\gamma_{ij}) - \log(K_{ij})$. Moreover, $\log(K_{ij}) = -C_{ij}/\lambda$,

$$\begin{aligned}\mathrm{KL}(\gamma|\mathbf{K}) &= \sum_{i,j} \gamma_{ij}\log(\gamma_{ij}) + \frac{C_{ij}}{\lambda}\gamma_{ij} - \gamma_{ij} + K_{ij},\\ &= \frac{1}{\lambda}\sum_{i,j} C_{ij}\gamma_{ij} + \sum_{i,j}\gamma_{ij}(\log(\gamma_{ij}) - 1) + \sum_{i,j} K_{ij}\end{aligned}$$

Finally, notice that K_{ij} is independent of γ, so it can be excluded during optimization. Moreover, by multiplying the divergence by λ, one recovers Equation 4.5. This can be done since $\lambda > 0$.

Proof of Proposition 4.2.

Consider two Lagrange multipliers, $\mathbf{f}$ and $\mathbf{g}$ for the constraints $\gamma\mathbf{1}_n = \mathbf{a}$ and $\gamma^T\mathbf{1}_m = \mathbf{b}$. The Lagrange function for the optimization problem reads as,

$$\mathcal{L}(\gamma, \mathbf{f}, \mathbf{g}) = \sum_{i,j} C_{ij}\gamma_{ij} + \lambda\sum_{ij}\gamma_{ij}(log(\gamma_{ij}) - 1) - \sum_i f_i\sum_j \gamma_{ij} - \sum_k g_k\sum_\ell \gamma_{\ell k}.$$

Therefore, to find γ^*, it is needed to solve $\nabla_\gamma\mathcal{L} = 0$. Using the fact that

$$\nabla\Omega(\gamma) = \log(\gamma),$$

one has,

$$\nabla_\gamma\mathcal{L} = C_{ij} + \lambda log(\gamma_{ij}) - f_i - g_j = 0,$$

which implies that $\gamma^*_{ij} = e^{f_i/\lambda}e^{-C_{ij}/\lambda}e^{g_j/\lambda}$. By letting $u_i = e^{f_i/\lambda}$ and $v_j = e^{g_j/\lambda}$ concludes the proof.

Proof of Lemma 4.1.

For this proof, we will assume that $\mathcal{H}$ is a RKHS, with reproducing kernel $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}}$. For a in-depth discussion of RKHS spaces, we use [Saitoh and Sawano, 2016]. With the aforementioned assumptions,

$$\begin{aligned}
\text{MMD}(P_S, P_T) &= \sup_{f \in \mathcal{H}: ||f||_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[f(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[f(\mathbf{y})], \\
&= \sup_{f \in \mathcal{H}: ||f||_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\langle f, \phi(\mathbf{x}) \rangle_{\mathcal{H}}] - \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\langle f, \phi(\mathbf{y}) \rangle_{\mathcal{H}}], \\
&= \sup_{f \in \mathcal{H}: ||f||_{\mathcal{H}} \leq 1} \langle f, \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})] \rangle_{\mathcal{H}} - \langle f, \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})] \rangle_{\mathcal{H}}, \\
&= \sup_{f \in \mathcal{H}: ||f||_{\mathcal{H}} \leq 1} \langle f, \mathop{E}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})] - \mathop{E}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})] \rangle_{\mathcal{H}}, \\
&= || \mathop{E}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})] - \mathop{E}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})] ||_{\mathcal{H}}.
\end{aligned}$$

Proof of Lemma 4.2.

By using the definition of expectation,

$$\begin{aligned}
\mathcal{R}_S(h_1, h_2) &= \int_{\mathcal{X}} |h_1(\mathbf{x}) - h_3(\mathbf{x})| \mu_S(\mathbf{x}) d\mathbf{x}, \\
&= \int_{\mathcal{X}} |h_1(\mathbf{x}) - h_2(\mathbf{x}) + h_2(\mathbf{x}) - h_3(\mathbf{x})| \mu_S(\mathbf{x}) d\mathbf{x}, \\
&\leq \int_{\mathcal{X}} |h_1(\mathbf{x}) - h_2(\mathbf{x})| \mu_S(\mathbf{x}) d\mathbf{x} + \int_{\mathcal{X}} |h_2(\mathbf{x}) - h_3(\mathbf{x})| \mu_S(\mathbf{x}) d\mathbf{x}, \\
&= \mathcal{R}_S(h_1, h_2) + \mathcal{R}_S(h_2, h_3).
\end{aligned}$$

Proof of Theorem 4.3.

The proof relies on the use of Equations 4.30 and 4.33,

$$\begin{aligned}
\mathcal{R}_T(h) &\leq \mathcal{R}_T(h^*) + \mathcal{R}_T(h, h^*), \\
&\leq \mathcal{R}_T(h^*) + \mathcal{R}_T(h, h^*) + \mathcal{R}_S(h, h^*) - \mathcal{R}_S(h, h^*), \\
&\leq \mathcal{R}_T(h^*) + \mathcal{R}_S(h, h^*) + (\mathcal{R}_T(h, h^*) - \mathcal{R}_S(h, h^*)), \\
&\leq \mathcal{R}_T(h^*) + \mathcal{R}_S(h, h^*) + \alpha \varphi(P_S, P_T), \\
&\leq \mathcal{R}_T(h^*) + \mathcal{R}_S(h) + \mathcal{R}_S(h^*) + \alpha \varphi(P_S, P_T), \\
&= \mathcal{R}_S(h) + \alpha \varphi(P_S, P_T) + \lambda.
\end{aligned}$$

Proof of Theorem 4.4.

$\mathcal{H}$-distance: Following [Ben-David et al., 2010], the symmetric difference hypothesis space, $\mathcal{H}\Delta\mathcal{H}$ is introduced:

$$g \in \mathcal{H}\Delta\mathcal{H} \iff g(\mathbf{x}) = h(\mathbf{x}) \oplus h(\mathbf{x}),$$

where $\oplus$ is the XOR function. Intuitively, every function in $\mathcal{H}\Delta\mathcal{H}$ is the set of disagreements between hypothesis in $\mathcal{H}$. Note that the $\mathcal{H}-$distance may be rewritten in terms of hypothesis in

$\mathcal{H}\Delta\mathcal{H}$. As follows,

$$\begin{aligned} d_{\mathcal{H}}(P_S, P_T) &= 2 \sup_{g \in \mathcal{H}\Delta\mathcal{H}} |P_S(I(g)) - P_T(I(g))|, \\ &= 2 \sup_{h,h' \in \mathcal{H}} |P_S(h(\mathbf{x}) \neq h'(\mathbf{x})) - P_T(h(\mathbf{x}) \neq h'(\mathbf{x}))|, \\ &= 2 \sup_{h,h' \in \mathcal{H}} |\mathcal{R}_S(h, h') - \mathcal{R}_T(h, h')|, \\ &\geq 2|\mathcal{R}_S(h, h') - \mathcal{R}_T(h, h')|, \end{aligned}$$

which in turn implies that $\mathcal{R}_T(h, h') \leq \mathcal{R}_S(h, h') + \frac{1}{2} d_{\mathcal{H}}(P_S, P_T)$, that is, the $\mathcal{H}-$divergence fits Definition 4.12 with constant $\alpha = \frac{1}{2}$.

Wasserstein Distance: This discussion is slightly adapted from the proof presented by [Redko et al., 2017]. This latter work was the first to provide a theoretical analysis of Wasserstein-based minimization domain adaptation. Thus, let P_S, P_T be probability measures, and let $c(\mathbf{x}_S, \mathbf{x}_T) = ||\phi(\mathbf{x}_S) - \phi(\mathbf{x})_T||$ denote the cost between points $\mathbf{x}_S \sim P_S$ and $\mathbf{x}_T \sim P_T$, where ϕ are the feature mappings associated with the kernel k. Moreover, k is associated with a RKHS $\mathcal{H}_k$. This is equivalent to say that ϕ maps vectors on $\mathcal{X}$, to vectors on $\mathcal{H}_k$.

In addition, let ℓ be a convex, symmetric, bounded loss function that obeys the triangular inequality, with the parametric form

$$\ell(h(\mathbf{x}), f(\mathbf{x})) = |h(\mathbf{x}) - f(\mathbf{x})|^q,$$

for some $q > 0$ where f is the ground-truth labeling function for the learning problem. Note that, under the assumption that $h \in \mathcal{H}_k$ and $f \in \mathcal{H}_k$, by abuse of notation we may also assume $\ell(\mathbf{x}) = \ell(h(\mathbf{x}), f(\mathbf{x}))$ an element of $\mathcal{H}_k$. As follows, the risk functional on each domain may be expressed as,

$$\begin{aligned} \mathcal{R}_S(h) &= \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\ell(h(\mathbf{x}), f(\mathbf{x}))] = \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\langle \phi(\mathbf{x}), \ell \rangle_{\mathcal{H}_k}], \\ \mathcal{R}_T(h) &= \mathop{\mathbb{E}}_{\mathbf{x} \sim P_T}[\ell(h(\mathbf{y}), f(\mathbf{x}))] = \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\langle \phi(\mathbf{y}), \ell \rangle_{\mathcal{H}_k}]. \end{aligned}$$

Now, considering any two hypothesis h, h' in $\mathcal{H}_k$, one has,

$$\begin{aligned} \mathcal{R}_T(h, h') &= \mathcal{R}_T(h, h') + \mathcal{R}_S(h, h') - \mathcal{R}_S(h, h'), \\ &= \mathcal{R}_S(h, h') + \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\langle \phi(\mathbf{y}), \ell \rangle_{\mathcal{H}_k}] - \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\langle \phi(\mathbf{x}), \ell \rangle_{\mathcal{H}_k}], \\ &= \mathcal{R}_S(h, h') + \langle \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})], \ell \rangle_{\mathcal{H}_k}, \end{aligned}$$

where from the second to the third line, the linearity of the expectation was used. Now, using the Cauchy-Schwarz inequality, one has,

$$\mathcal{R}_T(h, h') \leq \mathcal{R}_S(h, h') + ||\ell||_{\mathcal{H}_k} || \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})]||_{\mathcal{H}_k}.$$

Note that in the context of classification problems, $h(\mathbf{x})$ and $f(\mathbf{x})$ take values in a finite set of possible classes, hence their difference is bounded, and so is ℓ. Thus, we may assume

$$||\ell(\mathbf{x})||_{\mathcal{H}_k} \leq 1,$$

which implies that,

$$\mathcal{R}_T(h, h') \leq \mathcal{R}_S(h, h') + || \mathop{\mathbb{E}}_{\mathbf{y} \sim P_T}[\phi(\mathbf{y})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim P_S}[\phi(\mathbf{x})]||_{\mathcal{H}_k}. \tag{D.1}$$

Notice that the right-hand-side of Equation D.1 is very similar to the expression derived for the MMD, in Equation 4.15. Working on this term,

$$\begin{aligned}
||\mathop{\mathbb{E}}_{\mathbf{y}\sim P_T}[\phi(\mathbf{y})] - \mathop{\mathbb{E}}_{\mathbf{x}\sim P_S}[\phi(\mathbf{x})]||_{\mathcal{H}_k} &= ||\int_{\mathcal{X}} \phi d(P_S - P_T)||_{\mathcal{H}_k}, \\
&= ||\int_{\mathcal{X}} (\phi(\mathbf{x}) - \phi(\mathbf{y}))d\gamma(\mathbf{x},\mathbf{y})||_{\mathcal{H}_k}, \\
&\leq \int_{\mathcal{X}} ||\phi(\mathbf{x}) - \phi(\mathbf{y})||_{\mathcal{H}_k} d\gamma(\mathbf{x},\mathbf{y}), \\
&\leq \inf_{\gamma\in\Pi(P_S,P_T)} \int_{\mathcal{X}} ||\phi(\mathbf{x}) - \phi(\mathbf{y})||_{\mathcal{H}_k} d\gamma(\mathbf{x},\mathbf{y}),
\end{aligned}$$

where from the first to the second step the joint distribution $\gamma(\mathbf{x},\mathbf{y})$ of P_S and P_T was taken. The second to the third line follows from the triangle inequality. Finally, the last inequality is obtained by taking the infimum over all possible joint distributions γ [Redko et al., 2017]. Hence,

$$\mathcal{R}_T(h,h') \leq \mathcal{R}_S(h,h') + W_1(P_S,P_T),$$

which implies that the Wasserstein distance fits Definition 4.12 for $\alpha = 1$.

Maximum Mean Discrepancy Distance: As a starting point, we will consider the right-hand-side of Equation D.1. The proof proceeds in the inverse path of the derivation of Equation 4.15,

$$\begin{aligned}
||\mathop{\mathbb{E}}_{\mathbf{y}\sim P_T}[\phi(\mathbf{y})] - \mathop{\mathbb{E}}_{\mathbf{x}\sim P_S}[\phi(\mathbf{x})]||_{\mathcal{H}_k} &= \sup_{||f||_{\mathcal{H}_k}\leq 1} \langle f, \mathop{\mathbb{E}}_{\mathbf{x}\sim P_S}[\phi(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{y}\sim P_T}[\phi(\mathbf{y})]\rangle, \\
&= \sup_{||f||_{\mathcal{H}_k}\leq 1} \mathop{\mathbb{E}}_{\mathbf{x}\sim P_S}[\langle f,\phi(\mathbf{x})\rangle] - \mathop{\mathbb{E}}_{\mathbf{y}\sim P_T}[\langle f,\phi(\mathbf{y})\rangle], \\
&= \sup_{||f||_{\mathcal{H}_k}\leq 1} \mathop{\mathbb{E}}_{\mathbf{x}\sim P_S}[f(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{y}\sim P_T}[f(\mathbf{y})], \\
&= \mathrm{MMD}(P_S,P_T).
\end{aligned}$$

Hence, one concludes that,

$$\mathcal{R}_T(h,h') \leq \mathcal{R}_S(h,h') + \mathrm{MMD}(P_S,P_T),$$

that is, the MMD distance fits into Definition 4.12 with constant $\alpha = 1$, and hence its minimization is valid for domain adaptation.

D.4 Chapter 6

CSTR Dynamical System Equations

As discussed throughout Chapter 2, the characterization of the system's dynamics is done through finding a differential equation in terms of the state variables $\mathbf{x}$. For the CSTR system, $\mathbf{x} = [C, T, T_c] \in \mathbb{R}^3$. As in Example 2.1, we adopt the first principles analysis, enunciated in Equation 2.1.

We start with mass balance. As follows, we may use the concentrations C_i and C to express the inlet and outlet of mass. Since Q is the volumetric flow-rate, the molar flow-rate is given by $F_i = QC_i$ (resp. $F = QC$), in mol/min. Moreover, since $A \to B$, one has the consumption of A given by the N^{th} order rate-law,

$$\frac{d[A]}{dt} = -akC^N, \tag{D.2}$$

where a is a constant associated with catalyst decay (which can be taken as 1, for normal operation purposes), and $k = k_0 e^{-E/RT}$ is the Arrhenius rate constant, which is independent of time but dependent on the temperature T. Therefore one has:

$$V\frac{dC}{dt} = QC_i - QC - akC^N V,$$

which gives the first equation for the system dynamics,

$$\frac{dC}{dt} = \frac{Q}{V}(C_i - C) - akC^N.$$

The next two equations (for T and T_c) are based on the application of Equation 2.1 for the energy. Hence, there is one equation associated with the reaction $A \to B$, and another one associated with the coolant temperature. The analysis is based on the fact that $E = \rho C_p T$, where ρ is the fluid density and C_p is the fluid specific heat. The input is given by the inlet energy flow $F_i = (Q/V)\rho C_p T_i$ in joules per minute. Moreover, the outlet energy flow is given by $F = (Q/V)\rho C_p T$. Since the reaction is exothermic, it generates heat, which amounts to,

$$\begin{aligned} E_r &= \Delta H_r[A], \\ &= -a\Delta H_r k C^N, \end{aligned} \tag{D.3}$$

where ΔH_r heat of the reaction, in calories per mol. Another term enters the balance, which is the heat exchange between the reactor and the jacket. This term is given by,

$$\dot{Q} = -\frac{bUA}{V}(T - T_c), \tag{D.4}$$

where U is the heat transfer coefficient, b is a constant associated with heat transfer fouling, and A is the area. The energy balance for the equation $A \to B$ is given by,

$$\rho C_p \frac{dT}{dt} = \frac{Q}{V}\rho C_p T_i - \frac{Q}{V}\rho C_p T - a\Delta H_r k C^N - bUA(T - T_c),$$

which gives the second equation for the system dynamics,

$$\frac{dT}{dt} = \frac{Q}{V}(T_i - T) - \frac{a\Delta H_r k}{\rho C_p}C^N - \frac{bUA}{\rho C_p V}(T - T_c). \tag{D.5}$$

The analysis for the coolant temperature T_c is very similar to the last one. In that case, we may exclude the generation of heat by the reaction, and invert the signal of $\dot{Q}$ (since the heat is absorbed). Moreover, the volume associated with the coolant is the jacket volume V_c instead of V, associated with the reactor. The equation is given by,

$$\frac{dT_c}{dt} = \frac{Q_c}{V_c}(T_{ci} - T_c) + \frac{bUA}{\rho C_p V}(T - T_c). \tag{D.6}$$

Thus Equation 5.1 represents the dynamics of the CSTR system. Note that, so far, the dynamical system is not controlled. Nonetheless Figure 5.1, as well as previous work [Pilario and Cao, 2018], [Li et al., 2020] suggest manipulating the coolant flow-rate Q_c to control the tank temperature T.