Simulation Analytics for Deeper Comparisons

Graham Laidler, MMath (Hons.), MRes

Submitted for the degree of Doctor of Philosophy at Lancaster University

June 2023

Abstract

Output analysis for stochastic simulation has traditionally focused on obtaining statistical summaries of time-averaged and replication-averaged performance measures. Although providing a useful overview of expected long-run results, this focus ignores the finer behaviour and dynamic interactions that characterise a stochastic system, motivating an opening for *simulation analytics*. Data analysis efforts directed towards the detailed event logs of simulation sample paths can extend the analytical toolkit of simulation beyond static summaries of long-run behaviour. This thesis contributes novel methodologies to the field of simulation analytics. Through a careful mining of sample path data and application of appropriate machine learning techniques, we unlock new opportunities for understanding and improving the performance of stochastic systems.

Our first area of focus is on the real-time prediction of dynamic performance measures, and we demonstrate a k-nearest neighbours model on the multivariate state of a simulation. In conjunction with this, metric learning is employed to refine a system-specific distance measure that operates between simulation states. The involvement of metric learning is found not only to enhance prediction accuracy, but also to offer insight into the driving factors behind a system's stochastic performance. Our main contribution within this approach is the adaptation of a metric learning formulation to accommodate the type of data that is typical of simulation sample paths.

Secondly, we explore the continuous-time trajectories of simulation variables. Shapelets are found to identify the patterns that characterise and distinguish the trajectories of competing systems. Tailoring to the structure of discrete-event sample paths, we probe a deeper understanding and comparison of the dynamic behaviours of stochastic simulation.

Acknowledgements

It is difficult to summarise everything that belongs here. My life has been dealt numerous privileges, of which the undertaking of this PhD stands as both a consequence and an example. This does not get forgotten.

Nonetheless, the most direct influence over the contents of these pages lies with my supervisors, Dr. Lucy Morgan, Prof. Barry Nelson, and Dr. Nicos Pavlidis. Allowing me the freedom to develop independence, while choosing the right moments to intervene, their understanding, guidance, and expertise have been invaluable. In particular, Lucy, thank you for your cheerful and willing engagement, and for initiating my contact with Simul8. Barry, your deep knowledge of simulation was shared helpfully and encouragingly, and your hospitality towards me at Northwestern was appreciated. And Nicos, thank you for your valuable suggestions and feedback, which were always well-timed, appropriate, and thorough. Collectively, your consistent input and agreement always simplified my task.

Most importantly, I am grateful to the Engineering and Physical Sciences Research Council (EPSRC) for the provision of funding, and the STOR-i Centre for Doctoral Training for creating the framework for this degree. The value of being surrounded by the friendly and vibrant research community at STOR-i cannot be overstated. So much of my laughter and enjoyment of the past few years rests with the individuals who have shaped my experience and brought it to life. I am fortunate to have found real friends in Lancaster, including the perfect housemates with whom to share the months of lockdown.

I am thankful for family, friends, teachers, and every influence that led me to pursuing a PhD. The decision was refreshingly simple; the door was opened and inviting, and I have no regrets about stepping through. It spanned a period that I will associate with the fondest of

memories: a blend of *Bon Iver* and barefoot shoes, squash, sauna, and pan-fried porridge. The mixture was rich. Learning is a gift, and one of the greatest gifts that we can give, to ourselves and to others, is an open mind..

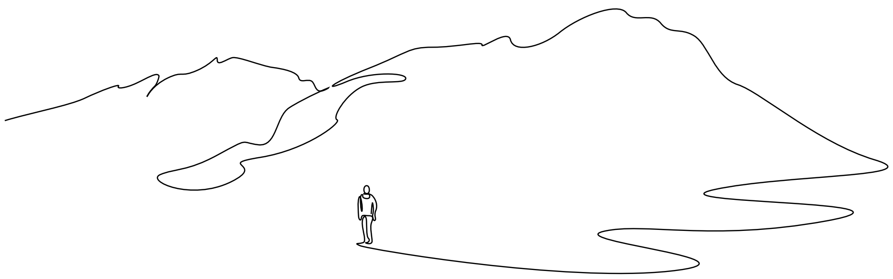
Wherein *there lies a scission* across the path to peace.

To *stand up with the vision*, with *feet out from the crease.*

I hear a simulation that plays a changing tune.

But *as I'm standing at the station*

It might be over s∞n.


[1]

[1] s1mple life/Shutterstock.com

Declaration

I declare that the work presented in this thesis has been produced by myself and has not been submitted elsewhere for the award of any other degree.

The thesis is constructed as a series of papers:

> Chapter 3 has been published as Laidler, G., Morgan, L. E., Nelson, B. L., and Pavlidis, N. G. (2020). Metric Learning for Simulation Analytics. In *Proceedings of the 2020 Winter Simulation Conference*, pages 349-360. IEEE.
>
> Chapter 4 has been submitted for publication and is currently under review.
>
> Chapter 5 has been submitted for publication and is currently under review.

These three chapters are presented with only minor modifications from their appearance in publication or review. In particular, some notational and orthographic revisions have been made to develop consistency throughout the thesis.

The word count in the main body of the thesis is approximately 36,000.

Graham Laidler

Contents

List of Figures

List of Abbreviations

kNN	k-nearest neighbours
PSD	Positive semidefinite
CV	Cross-validation
ROC	Receiver Operating Characteristic
DES	Discrete-Event Simulation
DML	Distance Metric Learning
NCA	Neighbourhood Components Analysis
SNCA	Stochastic Neighbourhood Components Analysis
KL	Kullback-Leibler
DTW	Dynamic Time Warping
LRW	Least Remaining Work
MRW	Most Remaining Work
i.i.d.	independent and identically distributed
s.t.	such that
a.s.	almost surely
fab	fabrication facility
M/M/1	A single-server queueing model with a Poisson arrival process and exponential service times

List of Symbols

$\mathbb{R}$	The set of real numbers
d	The dimension of an object
n	The size of a dataset
$\mathrm{tr}(\,\cdot\,)$	The trace of a matrix
$H(\,\cdot\,)$	The Shannon entropy function
$O(\,\cdot\,)$	Big O notation for runtime complexity

Chapters 3 and 4

A	A transformation matrix
M	A Mahalanobis matrix
$d_M(\,\cdot\,,\,\cdot\,)$	A Mahalanobis distance function
$\boldsymbol{x}^\top, A^\top$	The transpose of a vector or matrix
$\mathbb{S}_+^d$	The set of d-dimensional PSD matrices
$\mathscr{S}$	A set of similarity constraints
$\mathscr{D}$	A set of dissimilarity constraints
$\mathscr{R}$	A set of relative similarity constraints
I_d	The d-dimensional identity matrix
$\|\cdot\|_2$	The Euclidean norm of a vector
$\mathcal{X}$	A state space
$\mathcal{Y}$	A set of class values

$\boldsymbol{X}$	The random state vector
Y	The random class variable
$\boldsymbol{b}_l$	An observed state vector
y	An observed class value
$\mathcal{D}_n$	A dataset of size n
c_l	The observed count of $\boldsymbol{b}_l$ in $\mathcal{D}_n$
c_l^y	The observed count of $(\boldsymbol{b}_l, y)$ in $\mathcal{D}_n$
q	A discrete data-generating distribution
$\hat{q}$	The maximum likelihood estimates of q
p^A	The distribution imposed by A under SNCA
$D_{\mathrm{KL}}(q\|p)$	The KL divergence of p from q
$\mathbb{E}_{\hat{q}}[\,\cdot\,]$	The expectation with respect to $\hat{q}$

Chapter 5

Z	A discrete-time series
S	A discrete-time shapelet
y	A continuous-time series
s	A continuous-time shapelet
s^c	A continuous-time shapelet with a vertical shift
T	The length of a continuous-time series
ℓ	The length of a shapelet
$\boldsymbol{y}$	A multivariate continuous-time series
$\boldsymbol{s}$	A multivariate continuous-time shapelet
D	A classified time series dataset
$d(\,\cdot\,,\,\cdot\,)$	The distance between equal-length discrete-time sequences
$\|\cdot\|_1$	The L_1 norm of a continuous-time function

$\mathrm{dist}(\,\cdot\,,\,\cdot\,)$	The distance between a shapelet and a series
$\widetilde{\mathrm{dist}}(\,\cdot\,,\,\cdot\,)$	The location-invariant distance between a continuous-time shapelet and series
γ	A distance threshold
$\mathcal{I}$	A set of lengths for shapelet candidates
τ	A sampling frequency for extracting continuous-time shapelet candidates

Chapter 1

Introduction

Many systems in the physical world are driven by the interaction of random processes. To support decision making in such systems, stochastic simulation becomes an indispensable modelling tool. A simulation model creates a secure and cost-effective environment in which to explore the consequences of prospective real-world actions, and as such provides vital assistance across a range of industries, from healthcare (Brailsford, 2007) to manufacturing (McGinnis and Rose, 2017) and supply chain logistics (Ingalls, 1998).

Our specific focus lies in the stochastic simulation of discrete-event systems. In this modelling paradigm, events occur at distinct points in time and trigger immediate changes to the system's state. We use the term "*simulation sample path*" to refer to the multivariate trajectory of system state variables. Alternative paradigms such as agent-based modelling can also be understood through the evolution of a system state, and we acknowledge the wider applicability to some of the methods developed herein. Nevertheless, our primary motivation centres around discrete-event modelling, and each example we present belongs to this paradigm.

For decades, the focus of research and practice in simulation analysis has centered around the precise and efficient estimation of system performance measures. These are typically long-run averages of output quantities, such as a mean waiting time or expected profit. However, this habit of time-averaging overlooks the dynamic behaviour that occurs throughout the simulation. At the same time, the accomplishments of data analytics and machine learning in today's data-rich world suggest that deeper insights into simulation behaviour may be obtained by directing analysis efforts towards the simulation sample path. It is the recognition of this opportunity that inspires the field of *simulation analytics* (Nelson, 2016).

We begin by establishing further motivation for the field of simulation analytics, which underpins our efforts throughout this thesis. The contributions to this field made in the following chapters are outlined in Section 1.2.

1.1 Motivation

Simulation analytics is the term coined by Nelson (2016) to refer to the field of opportunities arising from the deliberate retention of sample path data. These opportunities surround the search for deeper insights into simulation behaviour. Over finer timescales, it may be revealed, for example, how a system responds to certain events or input behaviours, or which conditions lead to periods of poor performance. Such capabilities have the potential to move simulation towards a more adaptive style of system design and control, removing the need to comprehensively anticipate and simulate scenarios ahead of time.

At its most detailed level, the sample path of a discrete-event simulation can describe the time-stamped occurrence of every event and system state change. Traditionally, the trajectories of simulation variables have been averaged across both time and replications, with estimation of long-run performance the prevailing mode of output analysis. Illustrated in Figure 1.1.1, this practice naturally masks the time-dependent, dynamic effects which characterise the behaviour of a working system.

Simulation optimisation, for instance, constitutes a significant area of simulation application, wherein system designs are compared based on estimations of their long-run performance (Fu, 2015). Research efforts in this field continue to produce algorithms and procedures which, with increasing efficiency and confidence, can identify near-optimal solutions with regards to pre-specified, long-run objectives. The question of *which* system design

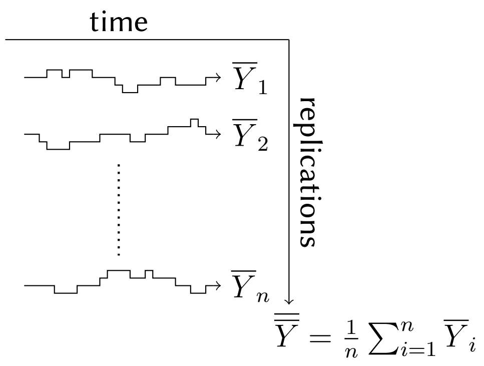

time
$\overline{Y}_1$
$\overline{Y}_2$
$\overline{Y}_n$
replications
$\overline{\overline{Y}} = \frac{1}{n}\sum_{i=1}^{n}\overline{Y}_i$

Figure 1.1.1: A traditional simulation output, $\overline{\overline{Y}}$, is typically an average of dynamic simulation behaviour over time and replications.

is best is being routinely addressed, while the questions of *why* remain largely overlooked. Simulation analytics pursues this shift of perspective. For example, can we identify the key drivers of system performance and reveal the important factors in the system design?

The aims of simulation analytics are facilitated by the fact that data storage today is cheap and, for most intents and purposes, unlimited. There is no reason to discard sample path data. Moreover, the simulation context affords us complete control of the data that we generate. Simulation data is tidy, complete, and free from measurement error, and we can stipulate exactly the information to record. This represents a significant opportunity, which is not yet widely exploited. Complete details of a system's sample path provide a rich opportunity for exploratory analysis, whilst we may also manipulate the simulation and data generation to target specific insight goals.

The intention of simulation analytics to unlock a more comprehensive view of system behaviour may also produce secondary benefits. Simulation has long been appreciated as a valuable decision making tool, although gaining the confidence of human users represents a critical facilitating step. Naturally, observing the movement of entities within a visual simulation model can instill confidence in the model's representation and operational fidelity. Simulation analytics, meanwhile, aims to instill confidence and offer explainability to the model's results. While model verification and validation can help to encourage confidence in model outputs (Sargent, 2010), an implicit level of trust is always required for simulationinformed decisions to be accepted and implemented in a real system. Reporting only the high-level summaries of simulation output without insight into the time-dynamic process leaves simulation with a distinctly *black box* feeling, and retaining trust in the simulation model can be difficult. We argue that a foundation of trust provides a vital connection between the model and the user, and without which the full benefits of simulation are unobtainable. From this perspective, inspiring and developing trust should be a constant aim throughout the simulation process, from model conception through to output analysis. The aim of simulation analytics to deliver insights into the working behaviours of a system may serve towards removing the layer of mystery between the computer model and its outputs, and inspire greater confidence in the simulation process.

As an example of pre-existing work in the spirit of simulation analytics, Brady and Yellig (2005) analysed the frequency of keyword elements relating to system entities from a single sample path. A correlation matrix was used to reveal the extent of interactions among system components and suggest those most critical to the system performance. Thus, the recognition of sample path data and its possible use for system analysis is not new. However, it is increasing (for examples, see Wu and Barton (2016); Ouyang and Nelson (2017); Jiang et al. (2020)). Modern appreciation for big data analytics and machine learning solutions provides simulation analytics with the encouragement to impact and transform our understanding of simulation output.

1.2 Outline and Contributions of the Thesis

The chapters of this thesis and their contributions to the field of simulation analytics are as follows.

Chapters 3, 4, and 5 are intended to be read as self-contained articles, with each including background material sufficient to its understanding. Nevertheless, Chapter 2 is included to offer additional context and relevant literature pertaining to the topics appearing in these subsequent chapters. In particular, we discuss the emergence of simulation analytics, and provide overviews of the topics of distance metric learning and time series shapelets. Chapter 2, therefore, serves as an auxiliary reference, and may indulge a deeper appreciation and understanding of the central concepts appearing in this thesis.

Chapter 3 approaches the task of making real-time predictions of a dynamic performance measure. We demonstrate a k-nearest neighbours (kNN) model on system state information, establishing a non-parametric framework to extract practical insights from sample path history. However, the question arises of how to define similarity and identify neighbours in the context of a high-dimensional simulation state. To address this challenge, we invoke a system-specific distance function via metric learning. This chapter acts as a proof-of-concept for the application of metric learning for kNN on simulation sample path data. We demonstrate the benefits that metric learning can bring to kNN predictions of simulation performance, both in terms of improving prediction accuracy and delivering insights into the key drivers of this performance. In supporting real-time decision-making in a live system, the applicability of this framework alongside digital twin technology is discussed in Section 6.2.1.

Chapter 4 extends the work of Chapter 3 by developing a metric learning method tailored to the context of simulation data. In contrast to the typical assumptions that underlie metric learning, the data obtained from a simulation model is inherently stochastic: we can observe identical observations of the system state leading to different simulation outcomes. This chapter adapts a method of probabilistic metric learning to this context. We establish theoretical properties of the method, and demonstrate its superior performance on simulation problems.

Chapter 5 approaches a different aspect of simulation analytics to the previous chapters. Studying the individual trajectories of simulation outputs, we uncover insights and draw comparisons among competing system designs based on their dynamic behaviour. This uses the concept of shapelets, which represent locally characteristic patterns used for time series classification. We adapt the methodology of time series shapelets to the continuous-time, piecewise constant trajectories typically encountered in discrete-event simulation. This presents a versatile methodology through which we can learn about the dynamic behaviours of different systems. An implementation of the method is supported by mathematical results, while the proof-of-concept is established through illustrative simulation applications.

Chapter 6 concludes the thesis by providing a summary of the contributions made throughout, and suggestions for further research in extension to the presented work.

Chapter 2

Background and Literature Review

This chapter serves as an introduction and background to the ideas presented in the subsequent chapters. Section 2.1 sets the foundation for the entire thesis, commencing with a brief background in simulation modelling and arriving at the present recognition of simulation analytics. Section 2.2 covers the subject of distance metric learning, providing context and background relevant to Chapters 3 and 4. In Section 2.3, we provide an overview of time series shapelets, establishing a background to the methodology presented in Chapter 5.

2.1 The Path to Simulation Analytics

The field of simulation analytics emerges out of a long history of modelling and simulation. "*Modelling*" abstractly refers to the activity of creating a simplified representation of reality, principally for the purpose of learning something of that reality (Fishwick, 2017). Whilst this broadly describes a habitual behaviour throughout human history, the advent of computers rapidly transformed the modelling capacity available to us, and facilitates the modern use of computer simulation which pervades today. The power, scale, and interactivity of a modern simulation model ensures a potent tool by which we may learn about our world and perceive the outcomes of our decisions.

Conducting a simulation study necessarily involves several steps, including, for instance, data collection, formulation and validation of a conceptual model, construction and execution of a computer program, operational validation, experimental design, and output data analysis (Nance and Sargent, 2002). Our interest in this thesis lies predominantly in the direction of output analysis. This section aims to chronicle the progress and directions of research in output analysis, arriving at the recognition of simulation as a generator of dynamic sample path data and the reasonable goals of simulation analytics.

2.1.1 Traditional Means of Output Analysis

A pervading goal of simulation modelling has been to obtain information regarding some performance value. In stochastic models, the presence of random input components precipitates random output behaviour, and simulation can only provide an estimate of the true value of interest. In the early days of computing, stochastic simulation experiments involved the running of multiple replications of a model with different random inputs to arrive at an estimation of the expected value. To acknowledge the ever-present uncertainty, statistical theory of the sample mean can equip a simulation estimate with variance characteristics and confidence bounds (Law and Kelton, 2007).

Estimating quantities such as expected customer delays illustrates a problem that has dominated output analysis for decades. Namely, that observations from within a single replication are likely to be correlated, as well as being sensitive to the system's initial conditions. Addressing these complications, Conway (1963) presented ideas critical to the tactical execution of simulation experiments, introducing the problem of *steady-state* simulation and related considerations of sample size and variance estimation. These remain central considerations to the evaluation of steady-state simulation output and continue to inspire new research (Pasupathy et al., 2022). Seeking to place simulation outputs in a statistical framework has long occupied researchers, with autoregressive time series models, for example, being fitted to assist steady-state mean and variance estimation (Schriber and Andrews, 1984).

Beyond its theoretical analysis, the awareness of variability in simulation output has influenced practice by inspiring variance reduction techniques. Established techniques include the use of common random numbers (CRN) and control variates (Nelson and Pei, 2021). CRN, for example, refers to the practice of synchronizing the random number streams used to simulate a system under alternative scenarios. In this way, output variance arising from differences in the input randomness can be mitigated, and the variance of the estimated difference between the outputs is reduced without the need for additional simulation effort. Comparing a system under alternative scenarios represents a common objective from a simulation study, and motivates the extensive field of simulation optimisation (Fu, 2015).

The ambition of simulation users is often, by altering some controllable decision variables, to discover a system design which will yield good, or near-optimal performance. When the decision variables are continuous, response surface methodology and gradient-based methods such as stochastic approximation have proved useful to optimisation (Fu, 1994). Alternatively, in a discrete decision space with a finite set of feasible solutions, the focus turns to providing statistical guarantees and results of convergence towards optimality. Ranking and selection procedures describe a popular family of methods for use when the number of feasible solutions is small enough that reasonable simulation effort may be allocated to each. The aim of such procedures is to identify the best solution under some criterion, such as meeting a pre-selected probability of correct selection (Kim and Nelson, 2006).

Accounting for the variation in simulation outputs remains a key concern for simulation optimisation procedures, and for output analysis in general. Seeking a fuller understanding of the origins and influences of this variation has led to research into topics such as sensitivity analysis and input uncertainty. Sensitivity analysis typically investigates the impact of changes in model input parameters on simulation outputs (Kleijnen, 1995). This can provide quantification to the relative influence of different inputs and assess the robustness of the simulation model. Input uncertainty, meanwhile, refers to the inherent variability or error in the fitted input models themselves, arising from factors in the data collection process such as measurement error, incomplete data, and natural variability. Input modelling errors subsequently propagate through the simulation model and contribute variability to the outputs. Quantifying and minimising these contributions continues to attract efforts in simulation research (Song and Nelson, 2015; Parmar et al., 2021).

With these developments, we see output analysis moving beyond simple output estimation. Interest has grown in exploring relationships *through* the simulation: how input features and variability propagate through to outputs; how outputs arise from inputs. Beyond stochastic input models, we also acknowledge the structural inputs, or decision variables, of a system design. Simulation metamodelling comprises another active area of research which aims to simplify and interpret the relationship of such inputs to the outputs, and provide a computationally efficient approximation to the simulation response (Barton, 2020). The evolution of simulation research shows a clear trend towards interpretable analysis. We see an overarching ambition to alleviate the black box feel of the inner simulation, and focus turns to an understanding of the *how* and *why* behind the outputs rather than simply the *what*. The field of simulation analytics finds its place at this point. Before exploring the main ambitions of simulation analytics, we turn our attention to the data generated by discrete-event simulation which will facilitate these ambitions.

2.1.2 Dynamic Sample Path Data

A discrete-event simulation models a system as a random sequence of events occurring at random times. Events trigger changes to the system state, whilst during the intervals between consecutive events, the system state remains unchanged. In general, the term *system state* may refer to some subset of the information generated by the simulation up to the current time. Glynn (1989) referred to the *internal state* of a system and portrayed its evolution in the mathematical framework of a stochastic process. For a more practical intuition, Law and Kelton (2007) define the state of a system to be the "*collection of variables necessary to describe a system at a particular time, relative to the objectives of a study*". We find this to be a useful working definition. Keeping in mind our goal to extract interpretable insights from simulation data, we might add to this that we imagine the system state to comprise variables representing those components of the simulation model which represent observable components of the real system. For example, a server in a queueing system may be attributed with a zero-one variable indicating its fluctuating state between idle and busy, while a queue may be assigned an integer variable reporting the number of its contents.

With this event-state view of simulation, we understand that full details of a replication may be captured by a time-stamped trace recording the events that occur and the corresponding change to the variables of the system state. The facility for this kind of output is not missing from simulation languages. Although primarily for the purpose of error-catching and debugging, simulation software will routinely record everything that occurs as a simulation steps through time. The ambition of simulation analytics is to recommission this detailed event log for the purpose of exploring and understanding system behaviour.

We refer to the detailed state information through time as a simulation sample path. To give some intuition for its properties, we first note that since many state variables will often be required to fully characterise a simulation model, the sample path is in general high-dimensional. In addition, it may be highly dependent. Due to the complex interactions among components of a simulation model, the paths of different state variables may be highly correlated. We can also acknowledge non-stationary behaviour. Perhaps driven by time-varying input models, the output sample path behaviour may be strongly time-dependent.

Data storage is effectively no longer a restriction, and we can readily aim to store and exploit sample paths from multiple replications of a system, and also from across system alternatives. We might point out that across replications of the same system, sample paths will be independent and identically distributed (i.i.d.). Therefore, although observations are dependent within each replication, they are independent across separate replications. This observation may prove useful to the application of analysis methods which require an assumption of i.i.d. data points. Across replications of different system alternatives, however, the sample paths will not be identically distributed, while the use of CRN can also remove independence. For the aim of comparison, simulation analytics can hope to uncover insights related to these distributional differences.

In summary, discrete-event simulation can equip us with multiple time-dynamic, high dimensional and highly dependent sample paths across different system alternatives, where the sample paths from replications within each alternative are i.i.d. It is the recognition of simulation analytics that this data is readily available, and brings the opportunity for deeper insights and comparisons of simulation models. We proceed to a discussion of these potential opportunities.

2.1.3 Objectives for Simulation Analytics

Broadly, the aim of simulation analytics is to develop a deeper understanding of simulated systems. In practical terms, the expectation is to apply statistical and machine learning techniques to the data contained in simulation sample paths. While the particular aims and possibilities rightly remain open, Nelson (2016), in introducing the field of simulation analytics, suggested the five general objectives which are discussed in the following paragraphs.

The first objective relates to generating *dynamic conditional statements*. This includes the task of making predictions conditional on the current state and time, an initial attempt of which was made by Ouyang and Nelson (2017). Their aim is to estimate a probability of the form $P(h(\boldsymbol{X}(t_0+t)) \in \mathcal{A} \mid \boldsymbol{X}(t_0) = \boldsymbol{x}_0)$. Assuming the current time of t_0, this denotes the probability that some function, h, of the system state vector, $\boldsymbol{X}$, at the future time, $t_0 + t$, will belong to the set, $\mathcal{A}$, given that the current state is $\boldsymbol{x}_0$. In a queueing network, for example, the state variable, X_i, may denote the number of customers at node i, such that taking $h(\boldsymbol{X}(t)) = X_i(t)$ and $\mathcal{A} = (l, \infty)$ allows a prediction of the probability that the population of node i will exceed a threshold, l, at the future time, $t_0 + t$.

To estimate this probability, Ouyang and Nelson (2017) propose a two-stage modelling approach in which the state and time aspects are treated separately. The first stage considers fixed prediction horizons. For each horizon, a logistic regression model is fitted on basis functions of the state vector. This requires n independent sample paths, from which the observations of $(t_0, \boldsymbol{x}_0)$ are identified, and, for the prediction horizon of $t_0 + t_j$, the binary response is defined by whether or not the event $\{h(\boldsymbol{X}(t_0 + t_j)) \in \mathcal{A}\}$ occurred. This first stage provides an estimate of the required probability at fixed prediction horizons. The second stage then accommodates any prediction time by fitting, for a given initial state, $\boldsymbol{x}_0$, a linear model on the logit-estimated probabilities based on basis functions of their prediction horizons.

This approach generates a probability estimate of the general form stated above, allowing prediction for any future time conditional on any current time and state. This is a useful objective with clear practical value in the context of system control, and the results presented suggest the approach to be effective for the relatively simple time-homogeneous queueing models considered. However, the set-up of the model remains somewhat rigid in terms of the choice of basis functions and prediction horizons, while the use of basis functions makes interpretation difficult. Although a promising first step, the approach leaves scope for further efforts towards dynamic predictions. We revisit this objective with our work in Chapter 3.

The second objective proposed in Nelson (2016) is to generate *dynamic distributional statements*. This might refer to characterising the distribution of some dynamically varying output. Smith and Nelson (2015) introduced an example of work under this objective by exploring characteristics of the *virtual waiting time*. This is considered as the waiting time that a customer would experience arriving to a queueing system at the particular time, t. Since the event of a customer arriving precisely at time t may never occur in multiple sample paths, the waiting times of customers arriving in pre-defined time buckets are used to estimate summary statistics of the virtual waiting time. Whilst identifying a performance measure of significant interest to customers, the approach taken is traditional in that the experimental design is fixed and the performance measures are anticipated ahead of time. The time buckets are also pre-determined, giving little flexibility to the method. Simulation analytics can aim to move away from rigid design and support a more adaptive approach to learning. For example, the size of time buckets might be varied dynamically to reflect the intensity of arrivals or the similarity of observed waiting times.

Additional work under the heading of virtual statistics is demonstrated by Lin et al. (2019). This work more naturally falls into the machine learning spirit of simulation analytics. A k-nearest neighbours (kNN) estimator for the mean of virtual performance is proposed, with insight provided into its asymptotic properties. As a machine learning regression method, the kNN algorithm estimates a response value based on the mean response of the k nearest training observations. In the context of virtual statistics, the k nearest neighbours of the virtual waiting time at t consist of the simulated observations from the k arrivals closest to time t.

The concept of virtual statistics identifies an important time-dependent performance measure. While the existing work provides a useful distributional analysis, only time has been considered as the conditional variable, leaving open a state-dependent equivalent to virtual performance.

The third suggested objective for simulation analytics is to generate statements on *multiple time scales*. This is a general capability which the simulation analytics philosophy of retaining fully detailed sample paths can facilitate. While events and state evolution may take place over fine-grained time increments such as seconds or minutes, an entire sample path may span a time frame of weeks, months, or years. The ability to adapt statements relating to the hourly, daily, or weekly behaviour of a system brings additional flexibility to the sphere of output analysis.

The fourth objective relates to making comprehensive *system comparisons*. While simulation has long been used to make comparisons among alternative system designs, traditional practice, such as in the context of simulation optimisation, has seen comparisons based on pre-defined, time-averaged performance measures. This, of course, masks the time-dynamic behaviour and lacks the detailed insights into how the systems will perform throughout the simulated period. From a simulation analytics perspective, we can expect to uncover comparative statements on the more detailed level of systems' dynamic behaviour. Our work in Chapter 5 falls under this objective.

The final objective suggested by Nelson (2016) is to generate *inverse conditional statements*. This might explore the relationship of outputs to inputs or the system state. Making inverse predictions can help us to identify the input behaviours and system configurations which lead to desirable output behaviour.

Across each of the objectives stated above, the overarching aim for simulation analytics is to promote a deeper understanding of system behaviour. While the primary focus for machine learning algorithms is often on achieving high prediction accuracy, applications to simulation analytics may place their focus towards interpretability. Accuracy provides an important quality measure and breeds confidence in a model, although often we will crave the insights and human learning into simulation behaviour that come with interpretable models.

The ambition to study and understand system behaviour through time represents a new aspect for output analysis, and one to which the simulation analytics perspective holds opens the door. In developing successful methods, the challenges lie in accommodating and exploiting the unique structure of sample path data. Appropriate tools and tailored procedures can be expected to unlock useful insights. The methodology presented in this thesis borrows the machine learning concepts which we discuss in the remaining sections of this chapter.

2.2 Mahalanobis Metric Learning

Chapters 3 and 4 of this thesis refer to the task of distance metric learning, or metric learning for short. This is a branch of machine learning which concerns the search for a distance function to suit a particular task and data. It is often used to improve the performance of distance-based algorithms, such as clustering and nearest neighbour classification (Kulis, 2013).

Metric learning operates in the context of some real-valued feature space, $\mathcal{X} \subseteq \mathbb{R}^d$. Its aim is to define a pairwise distance function, $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, that reflects and informs the similarity of the input data vectors, $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. For example, $d(\boldsymbol{x}_i, \boldsymbol{x}_j) < d(\boldsymbol{x}_i, \boldsymbol{x}_k)$ should imply that $\boldsymbol{x}_i$ is more similar to $\boldsymbol{x}_j$ than it is to $\boldsymbol{x}_k$.

The Euclidean distance, $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = [(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)]^{1/2}$, represents a natural choice of distance metric in real-valued vector spaces. The task of metric learning can then be understood as finding a mapping, f, of the input data such that $d(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j))$ provides preferable distance values. This represents the view of *global* metric learning, since a single mapping is applied to all data points. Alternatively, *local* metric learning, which learns separate mappings over different regions of the input space, has proved beneficial when data shows aspects of heterogeneity (see for example the extension given in Weinberger and Saul (2009)).

Placing some control on the mapping function, f, a popular paradigm is that of *Mahalanobis*, or linear metric learning. This corresponds to finding a linear mapping of the Euclidean input space. That is, we can express $f(\boldsymbol{x}) = A\boldsymbol{x}$ with some matrix A, and write the Mahalanobis distance as $d(A\boldsymbol{x}_i, A\boldsymbol{x}_j) = [(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top A^\top A(\boldsymbol{x}_i - \boldsymbol{x}_j)]^{1/2}$. This is often parameterised via the symmetric, positive semidefinite (PSD) Mahalanobis matrix, $M = A^\top A$, and the distance function written as

$$d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = [(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top M(\boldsymbol{x}_i - \boldsymbol{x}_j)]^{1/2}.$$

Our approaches in Chapters 3 and 4 fall into the category of global Mahalanobis metric learning, and this becomes our focus for the remainder of the section.

Any linear transformation of the data can be seen as inducing a Mahalanobis metric, and from this perspective, classical techniques such as principal component analysis (PCA, Pearson (1901)) deserve to be mentioned. PCA finds a linear transformation that reduces the dimension of a dataset while preserving the maximum amount of information, and is often applied as a pre-processing step. However, PCA is an unsupervised algorithm, whereas Mahalanobis metric learning is typically supervised. The data input, $\boldsymbol{x}_i$, carries a class label, y_i, and the task is to tune the Mahalanobis matrix, M, such that nearby points under d_M are likely to belong to the same class. This commonly involves the optimisation of an objective function guided by the supervision brought by the training data.

This general aim of Mahalanobis metric learning has invited numerous formulations built for various contexts, differing in key features such as their objective functions, encoding of supervision constraints, and choice of regularisation. While a number of surveys exist to provide systematic reviews of the metric learning literature (for example, Kulis (2013); Bellet et al. (2014)), this section offers an overview of some notable methods via a discussion of the common themes.

2.2.1 Supervision Sets

Since the object of our attention is a pairwise distance function, the supervision brought by the data is commonly conveyed through pairwise relationships. These are typically collected into the following sets:

$$\begin{aligned}
\mathscr{S} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \colon \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar}\},\\
\mathscr{D} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \colon \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are dissimilar}\},\\
\mathscr{R} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \colon \boldsymbol{x}_i \text{ is more similar to } \boldsymbol{x}_j \text{ than to } \boldsymbol{x}_k\}.
\end{aligned}$$

The relationships will typically be derived from the available class labels. For example, $\mathscr{S}$ and $\mathscr{D}$ can be populated with pairs of similarly and differently labelled instances, respectively, while triplets $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ with $y_i = y_j \neq y_k$ may populate $\mathscr{R}$. Many metric learning methods will utilise two out of three of these sets, with the specific data and application often guiding the choice and construction of appropriate sets, and hence methods. The weaker form of supervision represented by these sets accommodates problems for which full class labels are not available, and providing relative similarity relationships is often more natural in this context. On the other hand, fully labelled data allows the possible size of $\mathscr{R}$ to grow cubically with the number of data points, and in this case it may be preferable to choose a method which only uses $\mathscr{S}$ and $\mathscr{D}$, or else to populate $\mathscr{R}$ more selectively.

In general, metric learning algorithms assume the required supervision sets to be known, and to remain fixed throughout the learning process. The relationships that they contain are typically represented in the objective function or constraints of an optimisation problem.

2.2.2 Constrained Convex Optimisation

The family of Mahalanobis metrics can be parameterised either through the PSD matrix, M, or the unconstrained transformation matrix, A. Formulating the problem as an optimisation therefore brings a choice of perspective, both of which have advantages. Optimising the transformation matrix avoids a PSD constraint, and allows dimensionality reduction to be directly enforced by reducing the number of rows in A. However, an objective function which is convex in M will usually be non-convex in A. Convexity is an attractive feature for optimisation problems, and many methods consequently learn M in a convex formulation. The challenge for this approach is in maintaining the PSD condition, which requires the matrix to have only non-negative eigenvalues. This relates to the field of semidefinite programming, in which linear programs incorporate an additional PSD matrix constraint (Boyd and Vandenberghe, 2004). As a convex constraint, the overall convexity of the optimisation problem is maintained.

An initial example of Mahalanobis metric learning via semidefinite programming was provided by Xing et al. (2002). Using $\mathbb{S}_+^d$ to denote the set of d-dimensional symmetric PSD matrices, the method takes the supervision sets, $\mathscr{S}$ and $\mathscr{D}$, and solves the following intuitive formulation:

$$\begin{aligned} \min_{M \in \mathbb{S}_+^d} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} & d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{D}} & d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1. \end{aligned} \tag{2.2.1}$$

The optimisation of (2.2.1) used a projected gradient descent algorithm, iterating between gradient descent steps followed by projections onto the positive semidefinite cone achieved by setting negative eigenvalues to zero. This work paved the way for Mahalanobis metric learning to be treated as a convex optimisation problem, and several formulations have since emerged in this direction.

Large Margin Nearest Neighbors (LMNN) is one of the most widely used methods to date. LMNN was introduced by Weinberger and Saul (2009), and is motivated by an application to kNN. The kNN algorithm for classification makes assignments based on the labels of the k nearest training points, and thus only requires data to display clusterings of local similarity in order to perform well. With this motivation, the constraint sets used by LMNN are defined locally, by using the concept of k-neighbourhoods. These contain a point's k nearest neighbours, identified, for example, using Euclidean distance. In particular, the LMNN constraint sets can be written as

$$\begin{aligned} \mathscr{S} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \colon \boldsymbol{x}_j \text{ belongs to the } k\text{-neighbourhood of } \boldsymbol{x}_j, \text{ and } y_i = y_j\}, \\ \mathscr{R} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \colon (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}, \text{ and } y_k \neq y_i\}. \end{aligned}$$

The goal is to solve the following optimisation:

$$\min_{M \in \mathbb{S}_+^d} (1-\mu) \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) + \mu \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \in \mathscr{R}} \max\{0, 1 + d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_k)\}.$$

This objective function seeks to reduce the number of differently-labelled instances within the k-neighbourhoods of the training points. It accommodates a large number of constraints, with pair or triplet-wise relationships in general representing a combinatorial growth on the number of training points. LMNN handles the problem size with a purpose-built solver based on sub-gradient descent (Shor et al., 1985). LMNN has proved an effective formulation, and has inspired a number of related methods (for example, see Torresani and Lee (2006); Nguyen and Guo (2008)).

Optimisation of a semidefinite program often uses projection steps to maintain the PSD condition. This requires a full eigenvalue decomposition, the computational complexity of which is cubic in the dimensionality of the data. Therefore, this approach does not scale well with dimensionality, and a number of formulations have found alternative methods to satisfy the PSD constraint while retaining the problem convexity. An early method proposed by Schultz and Joachims (2003) uses the parameterisation of $M = M_0^\top D M_0$, where M_0 is fixed and known, and D is diagonal. This construction ensures that M remains PSD, while avoiding an explicit constraint by optimising over the diagonal matrix, D. This scales well with dimensionality, although sacrifices flexibility of the solution metric. It amounts to learning a weighting on the features of the pre-specified linear transformation given by M_0. In the absence of prior knowledge, using the identity matrix for M_0 will lead to a diagonal Mahalanobis matrix which is unable to accommodate interactions among dimensions.

In another notable method known as Information-Theoretic Metric Learning (ITML), Davis et al. (2007) avoid a PSD constraint by making use of the *LogDet divergence* in their objective function. The LogDet divergence between d-dimensional square matrices, M and M_0, is defined as $D_{ld}(M, M_0) = \mathrm{tr}(MM_0^{-1}) - \log\det(MM_0^{-1}) - d$. Conveniently, $D_{ld}(M, M_0)$ is convex in M, while choosing M_0 to be PSD ensures that $D_{ld}(M, M_0)$ is finite if and only if M is also PSD. In this way, the PSD condition can be easily preserved. Minimising $D_{ld}(M, M_0)$ is equivalent to minimising the Kullback-Leibler divergence between two multivariate Gaussian distributions parameterised by M and M_0, which reveals its information-theoretic interpretation. The objective of ITML is to minimise $D_{ld}(M, M_0)$, subject to threshold constraints on the distances in the supervision sets, $\mathscr{S}$ and $\mathscr{D}$. The large number of constraints are dealt with efficiently using Bregman projections (Censor et al., 1997).

Several other methods have also made use of the LogDet divergence (see for example Jain et al. (2008); Qi et al. (2009)). These approaches, as with that of Schultz and Joachims (2003), have the effect of constraining M to remain similar to an initial matrix, M_0. While this may limit the scope of the solution metric, it provides control over the learning process and serves as a form of regularisation.

2.2.3 Regularisation

Regularisation describes a common technique in machine learning used to provide control over a solution, and is an important element of many metric learning formulations. The inclusion of a regularisation term can help to avoid overfitting, a common pitfall when handling large numbers of constraints. Regularisation is also used to encourage low-rank solutions. This is a common goal, since a lower-dimensional projection can aid interpretability and allow faster computation.

A popular regularisation technique uses the squared Frobenius matrix norm, $\|M\|_F^2 = \mathrm{tr}(M^\top M)$. This is the sum of the squared elements of M, and it brings similar behaviour and advantages as the l_2 penalty used in ridge regression, such as strong convexity. In addition to constraining via the matrix, M_0, Schultz and Joachims (2003) also use Frobenius norm regularisation. Recalling $M = M_0^\top D M_0$ with diagonal D, the formulation uses relative distance constraints, and can be expressed as follows:

$$\min_D \|M_0^\top D M_0\|_F^2 + \lambda \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \in \mathscr{R}} \max\{0, 1 + d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_k)\}.$$

For further examples, Frobenius norm regularisation has also been used in an online metric learning algorithm (Shalev-Shwartz et al., 2004), and in a method accommodating noisy training constraints (Huang et al., 2010).

An alternative matrix regulariser is the trace-norm, or nuclear norm (Fazel et al., 2001). This is the sum of the singular values, and is known to be the tightest convex relaxation to the rank function (Recht et al., 2010). For a PSD matrix, the singular values are equivalent to the eigenvalues, the sum of which equals the matrix trace. To aid intuition, $\mathrm{tr}(M)$ can be viewed as the l_1 norm of the diagonal elements of M, since these are necessarily non-negative for PSD M. This provides analogy to the l_1 lasso penalty, and thus injects sparsity into the diagonal of M. By the construction of a PSD matrix, a zero diagonal element removes its entire row and column, which necessarily reduces the rank. The trace function is therefore a common choice of regulariser to target low-dimensional solution metrics.

By adding trace-norm regularisation to convex metric learning objectives, Huang et al. (2009) suggest a general framework for sparse metric learning, several examples of which have emerged (Jain et al., 2010; Yao et al., 2014). Adapting the trace-norm regulariser, Law et al. (2014) introduced the idea of Fantope regularisation, which corresponds to a sum of the $d - r$ smallest eigenvalues. This allows the control to seek a solution with a target rank of at most $r < d$. Extending this further, Huo et al. (2016) introduce the capped trace-norm penalising only the eigenvalues which are smaller than an adaptive threshold. These extensions aim to give greater control over the rank regularisation, by filtering out the non-relevant information without risking the important dimensions also becoming compressed. They provide a more stable regularisation term than the trace-norm, which fluctuates with changes to the largest eigenvalues. However, the trace-norm, as a simple sum of diagonal elements, is easy to apply. Regularisation on the eigenvalues, on the other hand, requires the computational burden of an eigendecomposition within each iteration of the optimisation.

Beyond the framework of regularisation, dimensionality reduction can be straightforwardly enforced by optimising over the transformation matrix, A. Choosing $A \in \mathbb{R}^{r \times d}$

ensures a metric of rank at most $r < d$. This approach, however, generates a low-dimensional transformed feature space (output sparsity), as opposed to performing dimensionality reduction in the original feature space (input sparsity). To retain interpretability with regards to the input features, input sparsity is often preferable in addition to output sparsity, with regularisation on the Mahalanobis matrix, M, the most obvious solution. Nevertheless, notable methods exist which optimise over A. This perspective does not exploit convexity, and thus is often applied when an objective function is already non-convex. Formulations based on probabilistic models often generate non-convex objectives and suggest an optimisation over A.

2.2.4 Probabilistic Models

Formulating a probabilistic model can provide a natural framework for the metric learning objective. Initially, the classical technique of linear discriminant analysis (LDA) may be seen through this lens (Fisher, 1936). Although not traditionally thought of as metric learning, LDA learns a linear transformation which maximises the ratio of between-class variance to within-class variance, and is commonly used for supervised dimensionality reduction. It is built on the assumption that data belonging to each class are normally distributed with equal covariance matrices. However, LDA does not require optimisation; instead, it admits a closed-form solution involving covariance matrices and eigendecomposition.

Sitting more closely in our metric learning context, Peltonen and Kaski (2005) provide a likelihood-based approach that attempts to generalise LDA by removing the restrictive probabilistic assumptions around class distributions. They construct a probabilistic model for the class value, y_i, conditional on the projected data, $A\boldsymbol{x}_i$, and maximise its log-likelihood:

$$\max_A \sum_{(\boldsymbol{x}_i, y_i)} \log \hat{p}(y_i \mid A\boldsymbol{x}_i). \tag{2.2.2}$$

They suggest using Gaussian kernel estimation for $\hat{p}(y \mid A\boldsymbol{x})$. This imposes a non-parametric probabilistic structure over the data space, allowing the restrictive assumptions of LDA to be discarded.

An influential method known as Neighbourhood Components Analysis (NCA) was introduced by Goldberger et al. (2004) and imposes a softmax probability over the nearest neighbour distribution. The point, $\boldsymbol{x}_i$, identifies $\boldsymbol{x}_j$ as its nearest neighbour with the probability p_{ij}:

$$p_{ij} = \frac{\exp\{-d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j)\}}{\sum_{k \neq i} \exp\{-d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_k)\}}, \; p_{ii} = 0.$$

Under this framework, $p_i = \sum_{j\,:\, y_j = y_i} p_{ij}$ represents the probability that $\boldsymbol{x}_i$ is correctly classified by the nearest neighbour classifier, and the objective to maximise $\sum_i p_i$ represents a maximisation of the expected leave-one-out error rate of this classifier on the training data. Alternatively, p_i can be viewed as estimating the probability that $\boldsymbol{x}_i$ belongs to class y_i, and can be seen as a special case of the kernel estimators, $\hat{p}(y_i \mid A\boldsymbol{x}_i)$, of Peltonen and Kaski (2005). The authors of NCA also acknowledge a log-likelihood objective function (2.2.2).

The objective functions of NCA are non-convex, and optimisation is performed over the transformation matrix, A. Globerson and Roweis (2005) provide a related convex formulation that uses the same probabilistic framework as NCA, known as Maximally Collapsing Metric Learning (MCML). They introduce the target distributions, p_{ij}^0, where $p_{ij}^0 \propto 1$ if $y_j = y_i$ and 0 otherwise, and minimise the Kullback-Leibler divergence between p_{ij} and p_{ij}^0 over all training points. While receiving the benefit of convexity, the formulation of MCML uses a particularly inflexible target distribution that is unsuited to represent class distributions which may not be naturally unimodal.

Further extensions to NCA have been made, such as accommodating the context of noisy labels (Wang and Tan, 2014), and adapting the objective function to reflect the expected accuracy of the kNN classifier for any choice of k (Tarlow et al., 2013). The probabilistic framework of NCA has proved popular for its natural interpretation and ease of differentiability in A. We revisit this method with our work in Chapter 4.

2.3 Time Series Shapelets

Chapter 5 of this thesis involves the concept of shapelets (Ye and Keogh, 2011), a technique used for time series classification. To establish the context, we consider a time series, $Z = z_{1:m}$, consisting of the ordered real values, $z_1, z_2, \ldots, z_m$. We assume a temporal ordering with equally spaced time intervals, although this is not necessary. For example, two-dimensional shapes can be represented as time series for classification (Keogh et al., 2006). A dataset, $D = \{(Z_1, c_1), (Z_2, c_2), \ldots, (Z_n, c_n)\}$, contains n time series, with the series, Z_i, possessing the class label, $c_i \in C$. The task of time series classification is to find a mapping from the space of possible time series to the space of possible class values.

A well-established literature exists for univariate time series classification (for example, see Bagnall et al. (2017)). Feature-based methods convert time series into feature vectors to proceed with conventional classification algorithms, with features commonly including statistical properties of the series, either globally or over short intervals (Deng et al., 2013). Features of the frequency domain obtained using spectral methods such as the discrete Fourier transform (Faloutsos et al., 1994) are also common. Alternatively, model-based methods fit a generative model to each class of series, and assign new series according to the class of model with the best fit. Under this framework, autoregressive models (Bagnall and Janacek, 2014), hidden Markov models (Smyth, 1996), and kernel models (Chen et al., 2013) have all been employed. These approaches rely on statistical characterisations of time series. On the other hand, distance-based methods such as the kNN algorithm have proved popular. Adapting this to time series that may not align exactly in time, speed, or length, the common Euclidean distance is often replaced with an elastic measure such as dynamic time warping (DTW, Berndt and Clifford (1994)).

Applying kNN with DTW is simple and robust, and is often used as a benchmark for time series classification algorithms. However, it relies on shape-based similarity over the entire series, and is hence not well suited when the shape-based characteristics of a class appear only locally. In practice, class-discriminating shapes are often short, and may appear at any point along a series. This motivates the method of shapelets, introduced by Ye and Keogh (2011). Shapelets are short subsequences extracted from a time series dataset which maximally discriminate among classes. In this thesis, we consider shapelets extracted from the time series of simulation sample paths, to discriminate the dynamic behaviour of competing systems. The remainder of this section outlines the original methodology for identifying and classifying with time series shapelets, and discusses the main extensions and emerging directions of research.

2.3.1 Original Methodology

Given the time series dataset, $D = \{(Z_1, c_1), (Z_2, c_2), \ldots, (Z_n, c_n)\}$, where Z_i is a series of length m_i, a shapelet may be defined as $S = s_{1:\ell}$, where $2 \leq \ell \leq \min m_i$. The process of discovering discriminative shapelets for a time series dataset involves a search over shapelet candidates with an evaluation of their discriminative quality. This uses a measure of distance between a shapelet and a series, with quality then defined in terms of the distance separation of the classes in the dataset. Once discovered, optimal shapelets offer interpretation towards the local shape characteristics of a class, and can be used for fast classification of new series. We describe these main components of the methodology as introduced by Ye and Keogh (2011).

Shapelet-Series Distance

The inspiration for shapelet methodology is to recognise the similarity of small common shapes (shapelets) appearing at any point in a series. For this reason, the distance between a shapelet and a series measures the distance of the shapelet to the most similar equal-length segment of the series. Therefore, we begin with a definition of the distance between $S = s_{1:\ell}$ and $S' = s'_{1:\ell}$, two length ℓ sequences. In general, this requires a real-valued, non-negative, and symmetric distance function, $d(S, S')$. The squared Euclidean distance is the suggested choice:

$$d(S, S') = \sum_{i=1}^{\ell} (s_i - s'_i)^2. \tag{2.3.3}$$

The distance between the series, $Z = z_{1:m}$, and a shapelet, $S = s_{1:\ell}$, is then given by

$$\mathrm{dist}(S, Z) = \min_{p \in \{1,2,\ldots,m-\ell+1\}} d(S, z_{p:p+\ell-1}). \tag{2.3.4}$$

Calculating the distance of S to every possible subsequence in Z requires $m - \ell + 1$ calculations. However, since $\mathrm{dist}(S, Z)$ is the minimum of these distances, Ye and Keogh (2011) advocate a practical speed-up technique by the early abandoning of redundant calculations. Recognising that the squared Euclidean distance is a sum of non-negative values, its summation can safely be abandoned as soon as it reaches or exceeds the minimum distance calculated so far. Empirical results suggested that this approach typically brings a reduction by around a factor of two to the time complexity of a shapelet search.

Several subsequent papers have suggested a z-normalisation of subsequences prior to the distance calculation (2.3.3) in order to solely reflect localised shape similarities and remain invariant to scale and offset differences among the time series (see, for example, Mueen et al. (2011)). The appropriateness of normalisation is largely problem dependent. The original work of Ye and Keogh (2011) included applications to the classification of shape outlines converted into time series using an angle-based method (Keogh et al., 2006), in which scale and offset differences were informative, and no suggestion of normalisation was made.

Shapelet Quality

The distance from a shapelet, S, to each time series in D generates a set of n distances, $D_S = \{\mathrm{dist}(S, Z_1), \mathrm{dist}(S, Z_2), \ldots, \mathrm{dist}(S, Z_n)\}$. A measure of shapelet quality is based on the ordering of the class values $\{c_1, c_2, \ldots, c_n\}$ under this set of distances. A discriminative shapelet should lead to strong separation of the available classes in C.

Ye and Keogh (2011) use the concept of information gain to provide the measure of shapelet quality. This is based on the entropy (Shannon, 1948) of the dataset before and after splitting into two subsets via a distance threshold on D_S. If D contains n_c series of class $c \in C$, its entropy is given as

$$H(D) = -\sum_{c \in C} \frac{n_c}{n} \log \left(\frac{n_c}{n} \right).$$

The shapelet, S, and distance threshold, $\gamma \in \mathbb{R}$, partition D into two subsets: $D^{\text{near}} = \{(Z_i, c_i) \in D \colon \operatorname{dist}(S, Z_i) \leq \gamma\}$ and $D^{\text{far}} = \{(Z_i, c_i) \in D \colon \operatorname{dist}(S, Z_i) > \gamma\}$. The information gain provided by S and γ on D is then defined as

$$I(D, S, \gamma) = H(D) - \left(\frac{|D^{\text{near}}|}{n} H(D^{\text{near}}) + \frac{|D^{\text{far}}|}{n} H(D^{\text{far}}) \right).$$

The quality of S is measured by its maximum information gain on D under all possible distance thresholds:

$$IG_{D,S} = \max_{\gamma \in \mathbb{R}} I(D, S, \gamma).$$

To compute $IG_{D,S}$ requires the evaluation of $I(D, S, \gamma)$ for all distinct splitting thresholds, which we can understand as the unique elements of $D_S \backslash \{\max D_S\}$. This contributes a time complexity of $O(n \log n)$, although this is generally small in relation to the computation of D_S, which is $O(n\overline{m}\ell)$, where $\overline{m} = 1/n \sum_{i=1}^{n} m_i$.

Given the dataset, D, the aim of a shapelet search is to find a shapelet, S, that maximises $IG_{D,S}$. Particularly for small datasets, many different shapelets may be found to maximise this quantity. Various tie-breaking options have been considered, including favouring the longest or the shortest shapelet. In the binary class problem, the recommended approach is to choose the shapelet which achieves the maximum separation between the two classes, defined by the difference between the mean distances to the series of each class.

The Search: Candidate Generation and Entropy Pruning

Identifying an optimal shapelet for the dataset, D, involves evaluation of $IG_{D,S}$ for each shapelet, S, in a set of candidates. While, in theory, there are no restrictions on S, in practice, a finite pool of candidates is generated by extracting subsequences from the series in D. For example, of a particular length, ℓ, we can extract $n(\overline{m} - \ell + 1)$ subsequences from D. Then, considering all subsequences of permissible shapelet lengths, $2 \leq \ell \leq \min m_i$, creates a candidate pool of size $n(\min m_i - 1)(\overline{m} - \min m_i/2)$. Calculating the set of distances, D_S, for each of these candidates leads to an overall time complexity of $O(n^2\overline{m}^4)$, and the computational challenge of an exhaustive shapelet search is clear. In practice, the size of the candidate set can be immediately reduced by restricting the range of permissible shapelet lengths, with domain knowledge often suggesting a reasonable range within which meaningful shapelets can be expected.

To mitigate the cost of a shapelet search, a further computation-saving technique was identified by Ye and Keogh (2011) and referred to as *admissible entropy pruning*. By alternating between adding distance values to D_S and updating an upper bound for $IG_{D,S}$, the candidate, S, can be safely abandoned (pruned) as soon as the upper bound falls below the value achieved by the current best shapelet. The upper bound for $IG_{D,S}$ is calculated by assuming the most optimistic scenario for the ordering of the remaining distance calculations in D_S, and can only decrease as the true distances become known. In the binary class problem, the most optimistic scenario is to assume that all distances to the remaining series of one class will lie at one extreme of the range of D_S, and all distances to the remaining series of the other class at the other extreme. In this case, there are two configurations for the ordering of D_S that we need to consider to find the upper bound. The extra cost of maintaining this upper bound is small in comparison with the potential speed-up brought by abandoning non-optimal shapelets and thereby avoiding expensive distance computations. The maximum benefit from this pruning strategy is obtained by adding distances to D_S in an alternating order between the series of each class, since this has the potential to reduce the upper bound more quickly.

Although unable to improve on the worst-case complexity, admissible entropy pruning is particularly effective for binary class problems, and shown to deliver practical speed-up over an order of magnitude to the shapelet search. For multi-class problems, however, we are faced with a combinatorial increase to the number of possible configurations of the ordering of D_S that may represent the most optimistic scenario, and the overhead for maintaining a correct upper bound can dramatically increase.

Decision Tree Classification

Once concluded, a shapelet search allows fast classification of a new time series. A single distance calculation between the new series and a shapelet can be informative, while nearest neighbour methods, for example, require distance calculations to an entire training set of time series. However, the binary framework of a single shapelet and its splitting threshold does not properly accommodate multi-class problems, and Ye and Keogh (2011) therefore imagine a general shapelet classifier as a decision tree (Breiman et al., 1984). The optimal shapelet with its optimal splitting threshold create a root node, and subsequent shapelet searches in the new subsets of the training data recursively add nodes and branches to a tree until a chosen stopping criteria is reached.

A decision tree classifier preserves the aspect of interpretability that shapelets provide. Ye and Keogh (2011) refer to the *dictionary* of shapelets used by a tree, and derive interpretations in contexts including the shape classification of arrowheads and heraldic shields, and the spectographs of varieties of wheat and coffee. The reported accuracy results also show shapelet decision trees outperforming kNN classification with both Euclidean distance and DTW.

2.3.2 Research Directions and Extensions

Since their conception, shapelets have provided useful insights and classification accuracy across a variety of applications, including gait analysis and motion capture (Shajina and Sivakumar, 2012), health monitoring (Ghalwash and Obradovic, 2012; Zorko et al., 2020), and detection of wind, wave, and seismic events (Arul and Kareem, 2021). From a methodological point of view, the research has continued in the following directions.

Speed-up Techniques

The original brute-force algorithm for a shapelet search (Ye and Keogh, 2011) requires $O(n^2\overline{m}^4)$ calculations, making it infeasible for most real applications. Although providing significant practical speed-up, computation-saving procedures such as the early abandon of distance calculations and admissible entropy pruning are unable to improve on the worst-case complexity. For this reason, much of the subsequent shapelet literature has focused on further pruning and speed-up techniques.

A further technique for candidate pruning was introduced by Mueen et al. (2011). Recognising, by the triangle inequality, that $\mathrm{dist}(S', Z) - d(S, S') \leq \mathrm{dist}(S, Z) \leq \mathrm{dist}(S', Z) + d(S, S')$, an upper bound for $IG_{D,S}$ can be found with no knowledge of D_S by reusing distance calculations from $D_{S'}$. In particular, an upper bound on $I(D, S, \gamma)$ is found by selectively moving distances in $D_{S'} \cap (\gamma - d(S, S'), \gamma + d(S, S'))$ to the other side of γ if doing so would improve the information gain, $I(D, S', \gamma)$. Finding this upper bound for $I(D, S, \gamma)$ for each γ implies an upper bound for $IG_{D,S}$, with S being pruned if this falls below the information gain of the optimal shapelet found so far. In practice, several upper bounds can be computed by using the cached distance calculations from several different shapelets, S'. Using shapelets with the *lowest* values of $IG_{D,S'}$ gives the best chance of candidate pruning.

Alternatively, He et al. (2012) recognise that the most discriminating shapelets tend to be distinctive and occur infrequently throughout a dataset, and thus prune the candidate pool via a heuristic approach of retaining only the most *infrequent* shapelets. Relinquishing the exhaustiveness of the search in this way seems to have little impact on the classification accuracy. Karlsson et al. (2016) also prune the search space by using randomised subsets of the training data and candidate pool in building a set of shapelet decision trees to use in a random forest ensemble. Another notable speed-up attempt was made by Rakthanmanon and Keogh (2013) by using symbolic piecewise aggregate approximations to the time series. Furthermore, hardware-based optimisation has assisted the discovery of shapelets by using parallel computing (Chang et al., 2012).

To avoid the high complexity of search-based methods, Grabocka et al. (2014) proposed to *learn* shapelets via gradient-based optimisation of a (non-convex) classification loss function. Differentiability of the objective is achieved by approximating the minimum function (2.3.4) with the *soft minimum*:

$$\operatorname{dist}(S, Z) \approx \frac{\sum_{p=1}^{m-\ell+1} d(S, z_{p:p+\ell-1}) \exp\{\alpha d(S, z_{p:p+\ell-1})\}}{\sum_{k=1}^{m-\ell+1} \exp\{\alpha d(S, z_{k:k+\ell-1})\}}.$$

This approaches the true minimum as $\alpha \to -\infty$. Although introducing new decisions with regards to hyper-parameters and shapelet initialisation, the approach is attractive for its speed and trainability via modern optimisation frameworks. Learning over all permissible shapelet lengths yields a runtime complexity of $O(n\overline{m}^3 \cdot \text{maxIter})$, where the parameter, maxIter, is the maximum number of iterations of the gradient step used by the optimiser.

Alternative Distance and Quality Measures and Classifiers

Shapelets provide a useful concept for time series data mining. However, the methodology for their discovery and their use for classification presents a number of decisions. Firstly, there is a choice of distance function (2.3.3). Euclidean distance represents the popular choice, although alternatives have also been explored. Notably, Shah et al. (2016) employed DTW in the context of learning shapelets (Grabocka et al., 2014). This provides greater flexibility for shapelets to identify characteristics which may be warped along the time axis, although comes with an added computational cost.

Shapelet discovery also requires a choice of quality measure. In the original methodology, the use of information gain aligns well with the decision tree classifier. However, it requires evaluation across many possible split points, contributing an overhead which increases further with multi-class problems. Alternative shapelet quality measures that do not require an explicit split point to be found were suggested by Hills et al. (2014). In particular, they consider test statistics including Kruskal-Wallis (Kruskal, 1952), Mood's Median (Mood et al., 1974), and the analysis of variance F-statistic (Scheffe, 1999). These statistics assess the separation of classes obtained by the distances in D_S based on a rank ordering, the average within-class separation by the median, and the ratio of between-class to within-class variance, respectively. Upper bounds analogous to those used for admissible entropy pruning, although not mentioned by Hills et al. (2014), may also be devised for these test statistics. Since these statistics are independent of binary splitting, their upper bounds would more easily extend to multi-class problems, for which the utility of admissible entropy pruning quickly degrades.

We also come to a choice in the classification procedure. Mueen et al. (2011) extended single shapelet classification to consider conjunctions and disjunctions of shapelets. While rule-based decision trees maintain the interpretability of individual shapelets, the most significant development for classification is referred to as the *shapelet transform*. Introduced by Lines et al. (2012) and developed by Hills et al. (2014), this uses shapelets for a feature-based method of time series classification. Under the shapelet transform, a time series dataset is transformed into an $n \times k$ matrix containing the distances of each of the n time series from each of a set of k shapelets. These distances comprise features which can be used in conjunction with a range of standard classifiers. This perspective disconnects the process of shapelet discovery from the classification algorithm, and so does not require a splitting threshold in conjunction with a shapelet. Therefore, the alternative quality measures to the information gain were suggested, with the F-statistic being preferred for its speed of computation and empirical classification results.

The shapelet transform introduces additional considerations. Firstly, the number of shapelets, k, must be set in advance. Choosing k too small limits the information available to the classifier, while setting it too large can result in overfitting. Zalewski et al. (2016) address the choice of k as a feature selection task in its own right. Moreover, the k shapelets with the highest quality measures are not necessarily the collective k best features for classification. In particular, the k best shapelets often contain much similarity, which reduces their collective value as a set of classification features. To address this, the shapelet transform initially excludes *self-similar* shapelets from the k best, defined as those extracted from the same series and having overlapping indices. However, as is the nature of shapelets, we still expect to extract similar characteristic shapes from separate series, and thus the collection of shapelets can still contain significant repetition. Therefore, a shapelet clustering procedure is also proposed to subsequently remove similar shapelets. This can benefit the classification algorithm and improve interpretability, although it introduces the risk of discarding relevant discriminatory features.

The shapelet transform provides a flexible framework through which the performance of shapelet-based classification may be enhanced. Hills et al. (2014) demonstrate its application across a variety of datasets and classification algorithms, and it has since become the widely adopted approach. For example, Arul and Kareem (2021) build random forests on shapelet transformed data to identify earthquake and thunderstorm events from ground motion and wind speed measurements. Using shapelets for a data transformation prior to classifier construction establishes them as a useful and flexible feature in time series data mining.

Multivariate Shapelets

Given the widespread presence and accessibility of multivariate time series data across various domains, interest in the field of multivariate time series classification is growing (Ruiz et al., 2021). This introduces an added layer of complexity, as discriminative features can arise from the interactions among dimensions. Moreover, it brings increased computational requirements. Despite the persistent computational challenge of a univariate shapelet search, its extension to the multivariate setting has been tackled by a number of authors. We denote a multivariate time series of dimension d by $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_d)$, where each dimension, $Z_j = z_{j,1:m}$ is a univariate series of length m.

When a classification algorithm for univariate time series is unable to handle multivariate data, a straightforward approach to adapting it for the multivariate setting is to ensemble it across the dimensions. In this manner, an adaptation of shapelet classification was made by Cetin et al. (2015). Indeed, in terms of classification performance, they conclude that an ensemble of decision trees with univariate shapelets obtained by treating the dimensions independently outperformed attempts to discover multivariate shapelets. Similarly, the random shapelet forest of Karlsson et al. (2016) admits multivariate series and builds a tree-based ensemble where the dimension, training subset, and shapelet candidates are randomised for each tree. Alternatively, a multivariate series may be converted into a univariate representation by concatenating dimensions. Patri et al. (2015) take this approach, and attempt to retain some interaction effects by interleaving segments extracted from each dimension.

Although multivariate approaches that revert to univariate shapelets may offer interpretable univariate patterns and insights into the importance of individual dimensions, they are generally unable to capture interactions among the dimensions or reveal the typical joint behaviour of the series. The earliest example of obtaining multivariate shapelets was provided by Ghalwash and Obradovic (2012) for applications to health informatics. In this approach, a multivariate shapelet, $\boldsymbol{S} = (S_1, S_2, \ldots, S_d)$, is extracted over the equivalent indices of each dimension, and its distance to a multivariate series is represented by the vector of the independent distances in each dimension:

$$\mathrm{dist}(\boldsymbol{S}, \boldsymbol{Z}) = (\mathrm{dist}(S_1, Z_1), \mathrm{dist}(S_2, Z_2), \ldots, \mathrm{dist}(S_d, Z_d))\,.$$

To assess the shapelet quality, a distance threshold is assigned to each dimension, and the subset membership of a series is decided by the proportion of dimensions in which the distance threshold is exceeded, relative to a percentage parameter. For a given shapelet and percentage parameter, the optimal set of distance thresholds is found which maximises the information gain under this splitting criteria. This approach helps to prevent shapelet quality from being influenced by irrelevant dimensions. However, it introduces additional computation and parameter optimisation. Also, distances are calculated in each dimension independently. From an interpretation perspective, this means that the discovered shapelet may reflect the behaviours that are typical to each dimension individually and not necessarily the typical joint behaviour, despite the shapelet candidates being extracted in parallel.

An alternative and natural approach to calculating a multivariate distance is to aggregate the distances from each dimension by summation. Defining a single distance value between a multivariate shapelet and a multivariate series allows the remainder of the search method to behave identically to the univariate case, by evaluating a shapelet quality measure such as the information gain. Bostrom and Bagnall (2017) demonstrate this approach, and consider the distances in each dimension to be either dependent or independent. In the former case, the best matching location of the length ℓ shapelet on the series is found with the shapelet components remaining in parallel, whereas the latter allows independent movement in each dimension in the same way as Ghalwash and Obradovic (2012):

$$\operatorname{dist}(\boldsymbol{S}, \boldsymbol{Z}) = \begin{cases} \displaystyle\min_{p \in \{1,2,\ldots,m-\ell+1\}} \sum_{j=1}^{d} d(S_j, z_{j,p:p+\ell-1}) & \text{for dependent dimensions,} \\ \displaystyle\sum_{j=1}^{d} \min_{p \in \{1,2,\ldots,m-\ell+1\}} d(S_j, z_{j,p:p+\ell-1}) & \text{for independent dimensions.} \end{cases}$$

While independence offers greater flexibility and may help to mitigate the impact of irrelevant dimensions, the dependent paradigm may be more useful to contexts such as motion capture and gesture recognition in which a movement may affect three spatial coordinates simultaneously.

When searching for dependent multivariate shapelets using a simple distance summation, there is a risk that the discovery of shapelets may be compromised by dimensions that have unequal or zero relevance to the discrimination between classes. Addressing this, Raychaudhuri et al. (2017) introduce an idea referred to as *channel masking* to discount noisy channels in the framework of learning shapelets (Grabocka et al., 2014). This involves learning positive weights to multiply the dimensions, a technique also employed by Zhang and Sun (2022). Similarly, Kidger et al. (2020) present a generalised framework for learning shapelets which is admissible of multivariate series and includes learned matrix parameters to multiply a multivariate distance function. To handle the increased computational cost, the paradigm of learning shapelets is well-suited to the multivariate extension, and recent works have explored the utility of neural networks in this task. Medico et al. (2021) present an architecture for learning multivariate shapelets based on embedding them as trainable weights in a multi-layer neural network. Meanwhile, Li et al. (2021) use a convolutional neural network to embed shapelet candidates of different lengths and dimensions into a unified space prior to candidate clustering and pruning for a multivariate shapelet transform.

The concept of shapelets has been well received in the field of time series classification, with numerous applications and extensions continuing to emerge. We explore their application to simulation sample paths in Chapter 5.

Chapter 3

Metric Learning for Simulation Analytics

The sample path generated by a stochastic simulation often exhibits significant variability within each replication, revealing periods of good and poor performance alike. As such, traditional summaries of aggregate performance measures overlook the more fine-grained insights into the operational system behaviour. In this chapter, we take a simulation analytics view of output analysis, turning to machine learning methods to uncover key insights from the dynamic sample path. We present a k-nearest neighbours model on system state information to facilitate real-time predictions of a stochastic performance measure. This model is built on the premise of a system-specific measure of similarity between observations of the state, which we inform via metric learning. An evaluation of our approach is provided on a stochastic activity network and a wafer fabrication facility, both of which give us confidence in the ability of metric learning to provide interpretation and improved predictive performance.

3.1 Introduction

Analysis of stochastic simulation has long been centered around the evaluation of aggregate performance measures. This chapter is motivated by the belief that static summaries such as these can represent a limited view of a highly dynamic and possibly non-stationary process. Dynamic performance measures such as waiting times or congestion levels are seen to fluctuate throughout simulated replications. Revealing the factors that drive these fluctuations is key to a deeper understanding of the represented system. With this aim, we propose a dual-purpose methodology for output analysis designed to support real-time predictions, whilst simultaneously revealing insight into the key drivers of dynamic performance.

Our work falls into the newly developing area of simulation analytics, first suggested by Nelson (2016). In this area, simulation is regarded as a generator of dynamic sample paths, from which data analytics and machine learning tools can glean insights into the conditional relationships and dependencies that characterise the system behaviour. We are aided in this approach by recent advances in data storage to enable cheap and effectively unlimited retention of sample path data. Indeed, many commercial simulation products currently retain a record of the events and state transitions that occur throughout a replication, albeit primarily for the purpose of debugging. Crucially, the capability is there, and whilst we are not dealing with the technicalities of how to post-process and store system traces, we are considering ways to exploit the opportunity for deeper analysis that such a detailed transaction log presents. To add further motivation and plausibility, we note that the size of datasets generated by simulation sample paths will typically not approach the volumes associated with *big data* in modern-day analytics.

To motivate the scope of our work, consider for example a doctors' surgery seeking to evaluate the performance of different staffing schedules. For this system, performance indicators such as patient waiting times or staff utilisation may be used to rank alternatives

(Brailsford, 2007). Whilst the daily averages of these indicators can allow an initial screening of solutions, they provide an incomplete picture of a system that is likely to exhibit significant variability throughout the day. A more probing analysis may relate to the systems' robustness to certain conditions such as a higher than usual demand for prescription medicines. Moreover, when our simulated systems exhibit periods of poor performance, we want to understand the cause. If we can reveal, for example, that the number of patients awaiting a blood test constitutes a driving influence behind variable waiting times, this is a useful insight which might suggest a more efficient allocation of resources. To enable such analyses, we build a predictive model for a dynamic system response, the structure of which is designed to expose the key factors which drive this response.

Specifically, our predictive model takes the form of a non-parametric k-nearest neighbour (kNN) classifier on the system state. Our choice here is motivated by the understanding that components of the state in a simulation model interact jointly in a way that is difficult to capture with parametric functions. Deferring a definition of system state to the later sections and appealing to the example above, we might include variables representing the number of patients undergoing or awaiting different treatments, or the number of nurses on duty. We let $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ denote the system state at a given time, and construct a measure of similarity over the space of $\mathcal{X}$. Each instance $\boldsymbol{x} \in \mathcal{X}$ carries with it an observed system response, y, which we measure as a categorical variable. For example, a patient entering the surgery in state $\boldsymbol{x}_i$ experiences the waiting time y_i, classed as above or below average. Our stored sample paths provide many observations of $(\boldsymbol{x}_i, y_i)$, which create our training data. Taking a kNN approach, we can classify the waiting time of a patient arriving to the system in state $\boldsymbol{x}^*$ according to the observed waiting time categories of the k nearest instances to $\boldsymbol{x}^*$ among the training data.

A key aspect of our methodology is in defining an appropriate measure of distance, or similarity, between observations in $\mathcal{X}$. For this, we delve into a rich literature on metric learning. The process of tuning a system-specific distance function is helpful in revealing interactions among the state variables and their relative contributions towards the system response. Thus, combining metric learning with kNN classification allows us to join interpretability with predictive performance. This dual benefit lends optimism to the scope of our work, which we believe extends to researchers and practitioners alike. In particular, whilst the application of metric learning serves to highlight the performance-dictating components and interactions within a system, the ability to make real-time predictions provides useful support to system control, and may complement the aims of a digital twin.

The structure of the chapter is as follows. In Section 3.2 we discuss related work and provide a background for kNN classification and metric learning. Section 3.3 presents results to motivate and demonstrate our methodology in the context of simulation, before we conclude with a brief summary in Section 3.4.

3.2 Background

In this section, we establish a background for our work, drawing on related work in the area of simulation analytics and describing our proposed application of kNN. We also introduce the field of metric learning, providing a focused review of the relevant literature.

3.2.1 Related Work in Simulation Analytics

The possibility of using simulation to inform real-time decision problems has recently begun to draw attention. A number of papers have emerged in which simulation sample paths are stored and used to build metamodels for making dynamic predictions. In the context of a queueing network, Ouyang and Nelson (2017) proposed a two-stage logistic regression modelling approach in which the state and time aspects of the sample path are treated separately, while Jiang et al. (2020) use a logistic regression model to dynamically predict the risk of financial portfolios. Wu and Barton (2016), meanwhile, show that Fourier analysis can successfully detect changes in the dynamic trajectories of system state variables to discriminate between congested and uncongested systems.

The approaches outlined above represent attempts to model sample path behaviour within a parametric framework. In reality, we understand simulation to be a complex stochastic process in which the dynamic and possibly non-stationary behaviour is difficult to capture in a parametric model. Accordingly, Lin et al. (2019) suggest a kNN approach to provide predictions for time-dependent mean performance measures. They describe this as "*virtual performance*", and further consider the behaviour of its higher order moments (Lin and Nelson, 2018). An example of virtual performance is given by the waiting time of a customer in a service system conditional on arriving to the system at time t. In our own take on virtual performance, the conditioning event relates to the state of the system; modifying this example, we consider the waiting time of a customer conditional on arriving to the system in state $\boldsymbol{x}$. In the case of Lin et al. (2019), the kNN estimator of virtual performance is one-dimensional in the sense that neighbours are determined by a single variable: simulation time. Our own kNN model, meanwhile, takes a more comprehensive view of 'neighbours', accommodating multiple predictors in an attempt to fully characterise the state of a system. Importantly, we note that a multi-dimensional state description is likely to contain variables which are not immediately comparable in terms of scale or interpretation. On these grounds, we recognise that identifying neighbours based on simple Euclidean distance, whilst effective in the one-dimensional setting of Lin et al. (2019), will not be appropriate for us.

3.2.2 kNN Classification

A kNN classification model provides a simple rule whereby instances are classified according to the labels of their k nearest neighbours (Hastie et al., 2009). In this chapter, we consider a binary classification task. Our training data, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, are obtainable directly from the stored sample paths. These observations are functions of the simulation time in a given replication; $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the system state at time t_i, and $y_i \in \{0, 1\}$ denotes an associated system performance measure, which in general may be observable at a later time. We use the term "*system state*" at time t to refer to some subset of the information generated by the simulation up to time t.

Using kNN classification to classify an instance, $\boldsymbol{x}^*$, we identify its k nearest training instances, denoted by $\boldsymbol{x}^{*(1)}, \ldots, \boldsymbol{x}^{*(k)}$, and their corresponding labels, $y^{*(1)}, \ldots, y^{*(k)}$. Our classification rule is defined as a function of $c \in [0, \infty)$:

$$\hat{y}^* = \begin{cases} 1 \text{ if } \frac{1}{k}\sum_{i=1}^{k} y^{*(i)} \geq c, \\ 0 \text{ if } \frac{1}{k}\sum_{i=1}^{k} y^{*(i)} < c. \end{cases} \tag{3.2.1}$$

The classification threshold of $c = 1/2$ corresponds to a typical majority rule, although in general we can choose c to minimise some error criterion such as the mean squared error (MSE) on a test set, or to represent a desired trade-off between the two types of misclassification.

A nearest neighbour classifier relies upon the assumption that instances which are similar to one another in the input space will yield a similar classification in the output space. Intuitively, the truth of this assumption requires that the distance measure used to identify neighbours reflects some system-specific notions of similarity between input instances. Selecting a relevant distance measure to use with kNN classification therefore requires careful consideration of the problem domain. Whilst Euclidean distance is often viewed as a default, significant advantage can be gained by using a more tailored metric (Kulis, 2013). This leads us to the topic of metric learning, which provides a data-driven way to automate the process of defining a suitable distance metric.

3.2.3 Mahalanobis Metric Learning

We recall our data of the form $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i$ is a d-dimensional vector of predictor variables and y_i its associated class label. The aim of metric learning is to adapt a distance function over the space of predictor vectors. As common in the metric learning literature, we consider the family of Mahalanobis distance functions. These take the form

$$d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = [(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top M(\boldsymbol{x}_i - \boldsymbol{x}_j)]^{1/2}, \tag{3.2.2}$$

parameterised by $M \in \mathbb{S}_+^d$, where $\mathbb{S}_+^d$ denotes the set of d-dimensional symmetric positive semidefinite matrices. This condition ensures that d_M satisfies the properties of a pseudometric (Bellet et al., 2014). We note here that the identity matrix, $M = I_d$, recovers the standard Euclidean distance.

A positive semidefinite matrix, M, always permits the decomposition, $M = A^\top A$. This allows us to write the Mahalanobis distance (3.2.2) as $d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = [(A\boldsymbol{x}_i - A\boldsymbol{x}_j)^\top (A\boldsymbol{x}_i - A\boldsymbol{x}_j)]^{1/2}$. Hence, we can understand the metric, d_M, to be equivalent to the Euclidean metric after a linear transformation of the data defined by A. This is a useful relationship which suggests two different parameterisations for the metric learning problem. The optimisation task can be performed with respect to the Mahalanobis matrix, $M \in \mathbb{S}_+^d$, or the linear transformation, A. Metric learning methods have been proposed from both perspectives, with each offering their own advantages. Optimisation over M is generally favoured as it leads to convex formulations which can be solved more efficiently. However, learning the transformation matrix allows for rank constraints to be directly imposed. In general, $A \in \mathbb{R}^{r \times d}$, where $r \leq d$ is the imposed rank of A and M. Learning a low rank matrix brings the data into a transformed space of fewer dimensions, which offers advantages when the original dimension, d, is large. For the purpose of simulation analytics, interpretation may primarily be available through M, with diagonal elements in particular reflecting the relevance of the state variable input features. However, understanding that this is obtainable as $M = A^\top A$, we may readily explore formulations from either perspective. In the experiments in this chapter, we optimise a Mahalanobis matrix, $M \in \mathbb{S}_+^d$.

The metric learning task is typically supervised by a collection of constraints summarising our prior intuition about the relative distances that we wish to emerge between training instances. Often, this supervision comes in the form of the following sets:

$$\mathscr{S} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \ \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ should be close}\} \text{ (similarity constraints)},$$

$$\mathscr{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \ \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ should not be close}\} \text{ (dissimilarity constraints)},$$

$$\mathscr{R} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_l) : \ \boldsymbol{x}_i \text{ should be closer to } \boldsymbol{x}_j \text{ than it is to } \boldsymbol{x}_l\} \text{ (relative similarity constraints)}.$$

In the absence of particular intuition, these sets can be derived from the class labels of the training instances. For example, pairs of similarly labelled instances should populate $\mathscr{S}$, while pairs of differently labelled instances can populate $\mathscr{D}$. Intuitively, triplets $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_l)$ with $y_i = y_j \neq y_l$ might populate the set $\mathscr{R}$. Whilst these sets are assumed to be given, their specific construction should be relevant to the data and the application at hand. The task of selecting appropriate constraint sets is treated as a learning task itself by Wang et al. (2012), although in general these sets are assumed to be given, and remain fixed throughout the metric learning procedure.

In general, the task of Mahalanobis metric learning may be expressed as an optimisation of the form $\min_{M \in \mathbb{S}^d_+} \ \ell(M, \mathscr{S}, \mathscr{D}, \mathscr{R}) + \lambda r(M)$. Here, ℓ is a loss function to penalise violations of the training constraints under the metric, d_M, and $r(M)$ describes some regularisation on the values of M, with $\lambda \geq 0$ the regularisation parameter. The main distinctions among different metric learning methods arise from their choices of loss function and regularisation.

The earliest method for Mahalanobis metric learning is attributed to Xing et al. (2002). The intuitive formulation seeks to minimise the sum of squared distances in $\mathscr{S}$ whilst keeping the sum of distances in $\mathscr{D}$ above a threshold:

$$\min_{M \in \mathbb{S}^d_+} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{s.t.} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{D}} d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \gamma.$$

To solve this optimisation, the authors proposed a gradient based algorithm with iterative projections onto $\mathbb{S}_+^d$ maintaining the positive semidefinite constraint. Whilst the primary motivation for this work was an application to clustering, the method has proved useful to many machine learning algorithms, paving the way for metric learning to be viewed as a convex optimisation.

Following this formulation, several more tailored methods have emerged. With an objective function inspired by the task of nearest neighbour classification, Goldberger et al. (2004) proposed a method referred to as Neighbourhood Components Analysis (NCA). They compute a softmax version of the probability that $\boldsymbol{x}_i$'s nearest neighbour is from the same class, $p_i = \sum_{j:y_j=y_i} \exp(-d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j)) / \sum_{l \neq i} \exp(-d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_l))$. Their objective function, which is to maximise the sum of these probabilities over all training points, equates to learning the distance metric which minimises the expected leave-one-out error rate of the nearest neighbour classifier. To aid the gradient calculation, the optimisation was presented in terms of the transformation matrix, A. The objective function of NCA is therefore non-convex, making the method susceptible to finding only local maxima. However, a convex extension to NCA was proposed by Globerson and Roweis (2005).

A second approach tailored to the task of nearest neighbour classification was provided by the Large Margin Nearest Neighbor (LMNN) algorithm, developed by Weinberger and Saul (2009). The construction of the constraint sets for LMNN is motivated by the fact that success of kNN only relies on local clusterings of similarly labelled points, rather than a global clustering. Specifically, the concept of target neighbours is introduced, which, in the absence of prior knowledge, are defined for each training point as the k nearest points in Euclidean distance which share the same class label. Taking $\eta_{ij} \in \{0, 1\}$ to indicate whether $\boldsymbol{x}_j$ is a target neighbour of $\boldsymbol{x}_i$, the sets $\mathscr{S} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \eta_{ij} = 1\}$, and $\mathscr{R} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_l) : \eta_{ij} = 1 \text{ and } y_l \neq y_i\}$ are defined. The objective function of LMNN then takes the following form:

$$\min_{M \in \mathbb{S}_+^d} (1-\mu) \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) + \mu \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_l) \in \mathscr{R}} \max\{0, 1 + d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_l)\}.$$

This objective function seeks a local neighbourhood of each instance that is populated with other instances of the same class, while those of a different class are repelled by a '*large*' margin. The parameter, $\mu \in [0, 1]$, controls the trade-off between attracting target neighbours and repelling oppositely labelled instances. A number of extensions to LMNN have been proposed. Torresani and Lee (2006) explore kernel methods to combine LMNN with dimensionality reduction of the feature space, while Kedem et al. (2012) suggest extensions to a non-linear metric.

Whilst NCA and LMNN remain perhaps the most notable contributions in the direction of metric learning for nearest neighbour classification, many metric learning methods exist in the wider literature, offering different perspectives on the fundamental task. We refer the interested reader to the work of Kulis (2013) and Bellet et al. (2014) for a thorough review of established methods.

In this chapter, we limit ourselves to considering a classification task. For this reason, the metric learning formulation and methods discussed above apply specifically to data in which the response variable is categorical. However, it should be recognised that kNN represents a versatile rule that readily extends to regression problems as well (see, for instance, Weinberger and Tesauro (2007)).

3.3 Metric Learning for Simulation

In this section, we demonstrate the proposed methodology for metric learning and kNN classification on simulation models. Whilst many metric learning formulations exist, the results presented here used CVXR (Fu et al., 2018), an open-source convex optimisation solver, to employ the method of Xing et al. (2002). This early and intuitive formulation suffices to provide proof-of-concept results which support the use of metric learning for simulation analytics.

To specify the constraint sets, $\mathscr{S}$ and $\mathscr{D}$, we take a local neighbourhood approach inspired in part by LMNN. We denote by $\mathcal{N}^{(q)}(\boldsymbol{x}_i)$ the set containing the q nearest points to

$\boldsymbol{x}_i$ in Euclidean distance. Then, for all $\boldsymbol{x}_i$, and for all $\boldsymbol{x}_j \in \mathcal{N}^{(q)}(\boldsymbol{x}_i)$, we make the following assignments:

$$(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \begin{cases} \mathscr{S} & \text{if } y_j = y_i, \\ \mathscr{D} & \text{if } y_j \neq y_i. \end{cases}$$

As noted by Weinberger and Saul (2009), the success of kNN requires only that the local neighbourhood of each instance be populated by others of the same classification. Particularly in our simulation context, which allows a high-dimensional representation of the system state, this will be relevant; a global clustering of each class may be inappropriate. Euclidean distance provides an initial view of the local neighbourhoods of our training points. It stands to reason that pairs of nearby points in Euclidean distance with equivalent class labels should be encouraged to remain as neighbours. These points naturally fall close in the predictor space and they share the same classification; we can confidently describe them as *similar*. On the other hand, nearby pairs with different classifications need to be forced apart, since they will impede the performance of our classifier. Moreover, owing to their different classifications, we assume that there is something fundamentally *dissimilar* about these pairs of instances which is not detected by Euclidean distance. By placing these pairs in $\mathscr{D}$, we aim to learn a distance metric that is sensitive to these more subtle dissimilarities.

We recall the formulation of Xing et al. (2002):

$$\min_{M \in \mathbb{S}_+^d} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{s.t.} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{D}} d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \gamma.$$

For the results presented in this section, we set the constraint constant to $\gamma = |\mathscr{D}|$. In other words, we aim to minimise the sum of squared distances in $\mathscr{S}$ whilst keeping the average distance in $\mathscr{D}$ from falling below 1. In reality, the choice of $\gamma > 0$ is unimportant, and, provided numerical stability of the optimisation algorithm is maintained, a different choice results only in a scaling of the solution matrix by a constant factor. To generate the sets $\mathscr{S}$

and $\mathscr{D}$, we take a local neighbourhood size of $q = 20$, and we evaluate the performance of kNN classification using $k = 50$ nearest neighbours. Whilst these values prove sufficient here to provide results supportive of our methodology, the optimal setting of the parameters q and k will in practice vary across systems, and can be selected via cross-validation (Hastie et al., 2009).

We first consider a simple motivating example to illustrate the advantages of metric learning and kNN classification in a simulation context. We then evaluate a realistic application of our methodology on a more complex simulation of a wafer fabrication facility.

3.3.1 A Stochastic Activity Network

We consider the simple stochastic activity network represented in Figure 3.3.1. We denote the five activity times by $X_1, X_2, \ldots, X_5$, modeling each as an i.i.d. random variable, $X_i \sim \text{Exp}(1)$ for $i = 1, 2, \ldots, 5$. There are three possible paths through the network, such that the total time taken for completion is given by $T = \max\{X_1 + X_4, X_1 + X_3 + X_5, X_2 + X_5\}$. To bring ourselves into a classification setting, we consider the binary response, $Y = 1$ if $T > 5$, and 0 otherwise.

Given the structure of this network, we understand that the path $X_1 \to X_3 \to X_5$, requiring three activities, will often represent the longest path through the network and hence define the value of Y. Of these three activity times, X_1 and X_5 appear in a second path also, giving them an edge over X_3 in terms of contribution towards the response. These two alternative paths suggest that some interaction effects will also exist between X_1 and X_4 and between X_2 and X_5; in each case, high values of both variables will encourage the

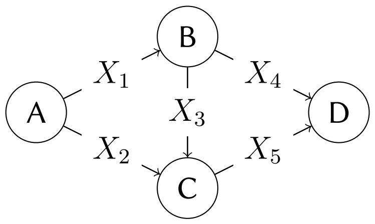


Figure 3.3.1: A small stochastic activity network.

response $Y = 1$.

We ran $n = 10,000$ replications of the network, recording the five activity times, $X_1, X_2, \ldots, X_5$, and the classification response, Y, for which the proportion of class 1 was around 16%. Knowing the exact mechanism by which the activity times generate the response gives us an understanding against which we can evaluate the output matrix of the metric learning. This output, M, provides a visual guide to the comparative relevance of the five activity times towards the response, and is shown in Figure 3.3.2. When used as the Mahalanobis distance matrix in (3.2.2), the diagonal elements in M indicate the weights given to differences in individual variables in defining the overall distance between two instances, whilst the off-diagonal elements additionally reflect relationships among the variables. We see immediately from Figure 3.3.2 that the largest diagonal elements correspond to the variables X_1, X_3, and X_5. Thus, under the metric d_M, instances with similar values of these three variables will be deemed more similar overall, with less importance given to their proximity in terms of X_2 and X_4. This reflects our understanding that $X_1 \rightarrow X_3 \rightarrow X_5$ is the most relevant path through the network. We can further note that the diagonal terms for X_1 and X_5 are slightly higher than for X_3, reflecting the additional contributions that these two variables make towards Y.

We turn our attention to the off-diagonal terms of M, and note that the positive semidefinite constraint imposes no restrictions on their sign. For the purpose of interpretation, we define $\boldsymbol{z}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$, and note from (3.2.2) that the off-diagonal element, $M(k,l)$, appears in the contribution of the term, $2\boldsymbol{z}_{ij}(k)\boldsymbol{z}_{ij}(l)M(k,l)$, to the squared Mahalanobis distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The magnitude of $M(k,l)$ indicates the impact of the product term, $\boldsymbol{z}_{ij}(k)\boldsymbol{z}_{ij}(l)$, on $d_M(\boldsymbol{x}_i, \boldsymbol{x}_j)$, whilst its sign indicates the way in which these variables interact. Specifically, a positive value of $M(k,l)$ indicates that $d_M(\boldsymbol{x}_i, \boldsymbol{x}_j)$ will increase when $\boldsymbol{z}_{ij}(k)$ and $\boldsymbol{z}_{ij}(l)$ have the same sign, and decrease otherwise, whilst the opposite is true for negative $M(k,l)$. Applying this interpretation to Figure 3.3.2, we can understand the positive off-diagonal terms in connection with the additive relationship through which the variables generate the response, whilst the greatest strengths are understandably attributed to the relationships among the dominant variables, X_1, X_3, and X_5. As such, we are encouraged to see that the matrix obtained by metric learning aligns with our intuitive understanding of this system. In realistic simulation models for which such intuition is less accessible, we suggest that metric learning can be effective in revealing relationships among system components, and their relative contributions towards driving the system performance.

The effectiveness of kNN classification using a learned distance metric is illustrated in Figure 3.3.3. The receiver operating characteristic (ROC) curves (Hastie et al., 2009) show the performance of the kNN classifier following 2-5-fold cross-validation (CV). J-K-fold CV, consisting of J independent K-fold cross-validations, is understood to represent a more robust procedure than traditional K-fold CV (Moss et al., 2018). Specifically in our experiments, each CV iteration takes four fifths of the data to comprise the training set on which the distance metric is learned, and the points in the remaining test set are classified by finding their k nearest neighbours from the training set, with respect to the learned distance metric. The ROC curves display the trade-off between the true positive rate and the false positive rate as the classification threshold varies. Here, the true positive rate (sensitivity) refers to the proportion of class 1 points correctly classified, whilst the false

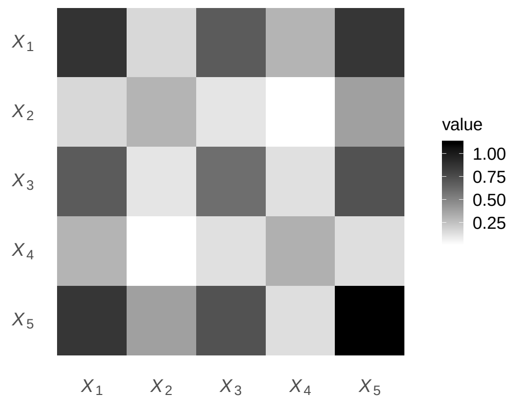


Figure 3.3.2: A visualisation of the learned matrix, M, for the stochastic activity network.

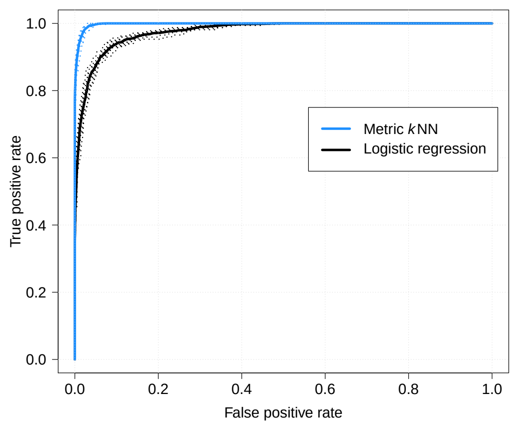


Figure 3.3.3: ROC curves for classification on the stochastic activity network.

positive rate ($1-$ specificity) refers to the proportion of class 0 points incorrectly classified as class 1. Thus, in practice, the ROC curves can be used to select the classification threshold, c, in (3.2.1), based on a desired sensitivity-specificity trade-off. In Figure 3.3.3, the ten dashed CV curves are averaged at each threshold to produce the solid curves. Using the same CV partitions, we also show the ROC curves from logistic regression, a standard classification technique.

The comparison in Figure 3.3.3 reveals the value of the kNN approach. Logistic regression represents a parametric attempt to model the response as a function of the predictor variables. However, owing to the nature of the network, measuring similarity with regards to the key activity times proves a more successful foundation for prediction. The connected dependence structure among the predictors in this example is not atypical of the relationships among state variables in many simulation models.

Further, we realise that a multi-dimensional characterisation of a simulation state is likely to include a number of variables which provide little or no contribution to the system response. To demonstrate the capability of metric learning in handling data of this nature, we augmented the five activity times with a further fifteen i.i.d. random variables, $X_i \sim \text{Exp}(1)$ for $i = 6, 7, \ldots, 20$. The response, Y, depends only on $X_1, X_2, \ldots, X_5$ in the same manner as before. These fifteen additional variables represent noise dimensions which have no bearing on Y. Metric learning on this augmented data yields the matrix shown in Figure 3.3.4. We see that metric learning is able to successfully filter out the noise dimensions and recover the important structure among the five activity times. Figure 3.3.5 shows the ROC curves for this example, and also shows the performance of the kNN classifier with Euclidean distance, which is comparable to that of logistic regression.

We are encouraged to see in Figures 3.3.4 and 3.3.5 the capability of metric learning in the presence of noise variables, both in terms of retaining interpretability and bringing improvement to Euclidean kNN classification. This lends optimism to our proposed application, since we recognise that irrelevant variables will be a common feature of the large

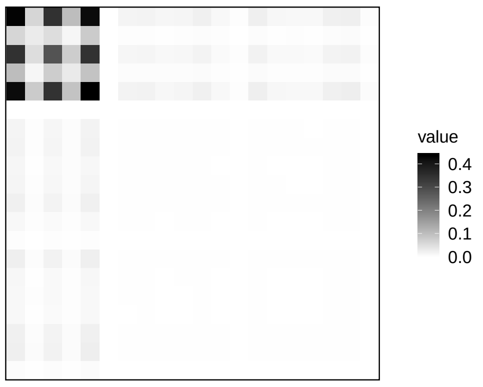


Figure 3.3.4: The learned matrix, M, after the data from the stochastic activity network is augmented with fifteen noise variables.

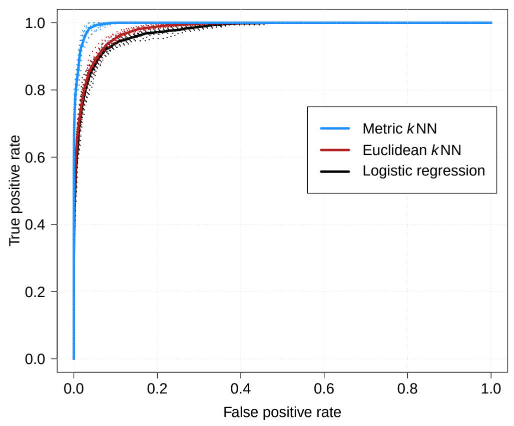


Figure 3.3.5: ROC curves for classification on the noise-augmented stochastic activity network.

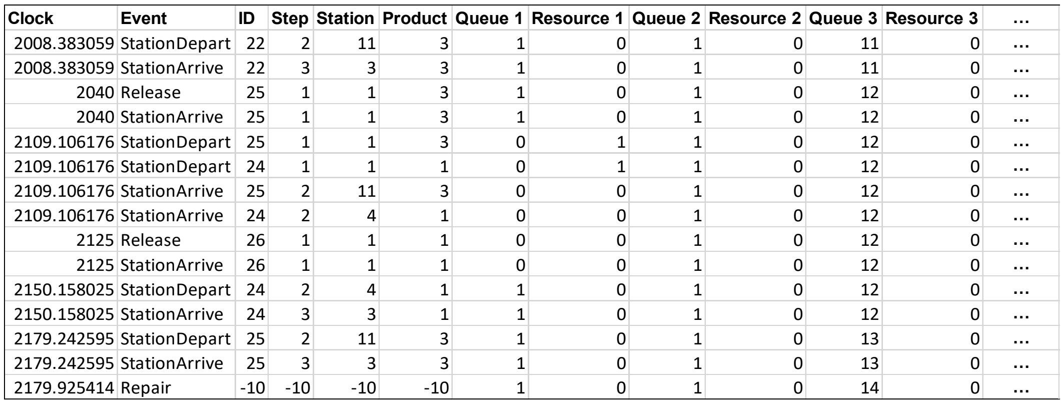
state descriptions of realistic simulation models. We progress now to one such model, in which we demonstrate further the capability of metric learning and its application to kNN classification.

3.3.2 A Wafer Fab Model

To evaluate our approach with a more realistic simulation, we employ the model of a wafer fabrication facility (fab) described by Kayton et al. (1996). The manufacturing process of semiconductor wafers involves several processing steps at a number of stations. Machines with different processing capacities and unpredictable breakdown patterns present a challenge to the management of product flow through these facilities. Moreover, the layered nature of their circuitry design requires wafers to make multiple visits to particular stations, introducing an aspect of re-entrant flow that further complicates our view of the system.

Briefly, the simulation model is comprised of 11 stations with lognormal processing times. Three product types are produced by the facility, each requiring a specific routing sequence through the 11 stations. Notable stations include station 3, characterised by an unreliable machine, and station 4, which represents the bottleneck station to which products make repeated visits. Together with the varying processing behaviour of the different stations, including batch processing at stations 1 and 2, these features establish a system with a level of complexity approaching that of typical simulation models.

On release into the system, each wafer is assigned a due date based on its expected processing time through an empty system. Therefore, we can consider observed completion times relative to due dates as a dynamic indicator of system performance. To describe the system state, we take a Markovian view in assuming that all information relevant to the future evolution of the system can be captured in the currently observable system conditions. Namely, we focus on the current values of the 22 integer variables describing the queue size and the number of resources in use at each station. We recognise that this state description is incomplete, given the different product types and processing stages of individual wafers. However, having tested numerous additional state descriptors, we find that a basic physical view of the system state is sufficient here to provide compelling support for our methodology. As such, our data, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, are as follows. We record the 22-dimensional system state, $\boldsymbol{x}_i$, at the moment a wafer is released into the system, and observe its associated binary response, y_i, indicating whether this wafer was completed early or late with respect to its due date. Our simulation model was coded using Visual Basic for Applications (VBA) in Microsoft Excel. To aid an understanding of the data, Figure 3.3.6 shows an excerpt of the



Clock	Event	ID	Step	Station	Product	Queue 1	Resource 1	Queue 2	Resource 2	Queue 3	Resource 3	...
2008.383059	StationDepart	22	2	11	3	1	0	1	0	11	0	...
2008.383059	StationArrive	22	3	3	3	1	0	1	0	11	0	...
2040	Release	25	1	1	3	1	0	1	0	12	0	...
2040	StationArrive	25	1	1	3	1	0	1	0	12	0	...
2109.106176	StationDepart	25	1	1	3	0	1	1	0	12	0	...
2109.106176	StationDepart	24	1	1	1	0	1	1	0	12	0	...
2109.106176	StationArrive	25	2	11	3	0	0	1	0	12	0	...
2109.106176	StationArrive	24	2	4	1	0	0	1	0	12	0	...
2125	Release	26	1	1	1	0	0	1	0	12	0	...
2125	StationArrive	26	1	1	1	0	0	1	0	12	0	...
2150.158025	StationDepart	24	2	4	1	1	0	1	0	12	0	...
2150.158025	StationArrive	24	3	3	1	1	0	1	0	12	0	...
2179.242595	StationDepart	25	2	11	3	1	0	1	0	13	0	...
2179.242595	StationArrive	25	3	3	3	1	0	1	0	13	0	...
2179.925414	Repair	-10	-10	-10	-10	1	0	1	0	14	0	...

Figure 3.3.6: An example of the trace from the wafer fab simulation, which we use to build a state description.

system trace information recorded by this simulation, from which we extract our system state description. In general, rather than trying to select a parsimonious state description, we suggest including all the observable state information that may prove relevant, and then allowing metric learning to discover what is actually relevant.

To avoid confusing the behaviours of the different product types, our data contains only observations from a single product type. In the results that follow, we combine observations from multiple replications of the system, discarding a warm-up period from each to leave us with insight into the operational steady-state behaviour. We are left with a data set of size $n = 2391$, with the proportion of *late* responses around 72%. The simulation can be performed with a choice of dispatching rules, relating to the order in which queueing wafers are processed (Kayton et al., 1996). The results displayed in Figures 3.3.7 and 3.3.8 used a *Least Remaining Work* rule, in which priority is given to wafers which are nearer completion.

The metric learning procedure resulted in the matrix visualised in Figure 3.3.7. The stand-out elements in this matrix correspond to the queue sizes at stations 3 and 4, which we recall to coincide with the unreliable machine and the system bottleneck, respectively. The metric learning result highlights the significance of these two queue sizes in defining the overall system performance.

Performing kNN classification on this data, with 2-5-fold CV as in the previous example, yields the ROC curves shown in Figure 3.3.8. Compared to a Euclidean kNN classifier, the curves reveal the benefit to classification performance that metric learning brings. Essentially, as the size of the state space increases, we expect the benefit of metric learning over Euclidean distance to become even more pronounced.

As a statistical learning technique, kNN is best suited to low-dimensional data (Beyer et al., 1999). To visualise the effect of metric learning with respect to dimensionality reduction, it is convenient to consider the transformation matrix, A, given by the decomposition, $M = A^\top A$. We recall that the metric, d_M, can be viewed as the Euclidean metric in the space transformed by A. Taking the eigen-decomposition of M, the i^{th} row of A is given

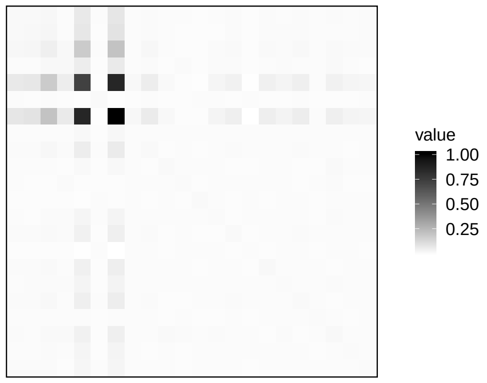


Figure 3.3.7: A visualisation of the learned matrix, M, for the wafer fab problem.

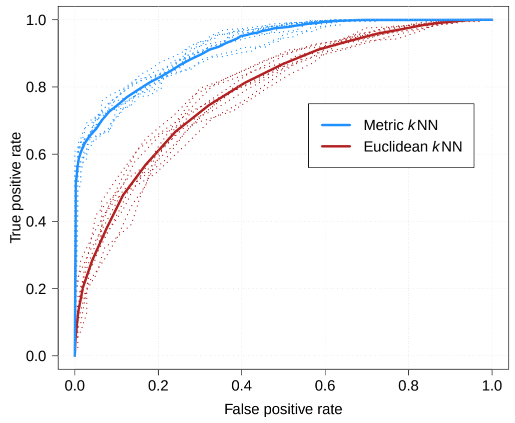


Figure 3.3.8: ROC curves for kNN classification on the wafer fab problem, comparing Euclidean distance with the learned metric.

by $\lambda_i^{1/2}\boldsymbol{v}_i^\top$, where λ_i and $\boldsymbol{v}_i$ denote, respectively, the i^{th} largest eigenvalue of M and its corresponding eigenvector. Thus, the eigenvalues of M directly impact the spread of our data in its transformed dimensions. Figure 3.3.9 shows the eigenvalues of the original data covariance matrix, in order of decreasing size, whilst Figure 3.3.10 shows the eigenvalues of M. We see that metric learning in this example results in much of the variability of the data being compressed into the first dimension. Hence, we can acknowledge the effective dimensionality reduction brought upon our data by metric learning. Projecting the transformed data against its first and second dimensions results in the plot shown in Figure 3.3.11. We can see that this dominant first dimension is effective in distributing our two response classes.

In formulations such as NCA, which directly seek a transformation of the feature space, dimensionality reduction can be straightforwardly enforced. However, even when not directly sought, the by-product of dimensionality reduction is almost inherent in the nature of the metric learning task. Since a nearest neighbour rule is understood to suffer the curse of dimensionality, and we aim to accommodate simulations with a high-dimensional state space, the dimensionality reduction encouraged by metric learning represents an important

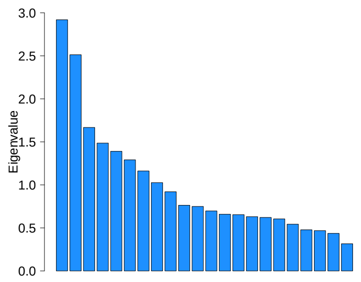


Figure 3.3.9: The eigenvalues of the original data covariance matrix.

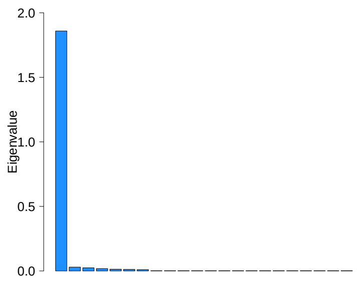


Figure 3.3.10: The eigenvalues of the learned matrix shown in Figure 3.3.7.

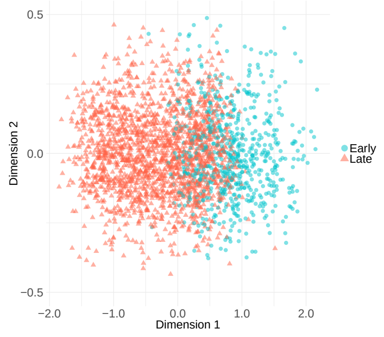


Figure 3.3.11: The data in the first two dimensions of the transformed feature space.

aspect of our methodology; it allows us to apply kNN without the need for user-intervention in trimming the state space.

3.4 Conclusion

In this chapter, we have presented a novel methodology in the field of simulation analytics. The basis of our methodology, kNN classification, provides a non-parametric framework in which to model the behaviour of a system, whilst the addition of metric learning is shown to bring both interpretability and improved prediction performance. We have demonstrated the merits of our approach for its intended application to simulation, showing that the typical features of sample path data, such as interacting and irrelevant variables, are well-handled by metric learning. Although we only present results from a single metric learning method in this chapter, the extensive research and accomplishment in this field gives optimism to the scope of metric learning for simulation. In short, we propose that a kNN approach combined with metric learning can have wide-reaching benefits, as simulation users begin to look beyond aggregate performance measures and seek a more fine-grained analysis from their simulation models.

Chapter 4

Stochastic Neighbourhood Components Analysis

Distance metric learning is a fundamental task in data mining, and is known to enhance the performance of various distance-based algorithms. In this paper, we consider stochastic training data in which repeated feature vectors can belong to different classes. Our primary motivation for this arises from the field of stochastic simulation. Storing the dynamic trajectory of the system state within a simulation model can support real-time predictions of stochastic performance measures. However, the inherent randomness within the system combined with the recurring nature of the system state leads to data of the type considered, on which existing methods of metric learning are known to struggle. Here, we present a stochastic version of the popular Neighbourhood Components Analysis. We demonstrate its behaviour using simulation examples, and reveal improvements over Neighbourhood Components Analysis when used for nearest neighbour classification of stochastic data.

4.1 Introduction

Stochastic computer simulations are widely used modelling and decision-making tools, and applications of machine learning to support their output analysis are becoming increasingly common (Giabbanelli, 2019). Discrete-event simulation (DES) provides a popular paradigm, representing the operation of a real system as a sequence of events which occur at discrete points in time. Each event, such as a customer arrival or a machine failure, triggers a change in a set of variables describing the system state, including for instance a queue size or a machine status. The trajectory of state variables obtained from a DES model hides a wealth of insight into the system behaviour, and is often referred to as the simulation sample path. Nelson (2016) recognised modern capacity for the storage of sample path data as paving the way for machine learning solutions, and coined the term *simulation analytics*. As an accessible source of measurement-error-free data, sample paths provide a rich environment for machine learning. However, to achieve the full potential of learning in this context, it is necessary to account for the distinctive properties of sample path data. This presents an opportunity for machine learning research, and in this work we take a step in this direction.

We consider the task of classifying a future system state. The ability to anticipate stochastic behaviour provides valuable assistance to system planning and control, and is well supported by the emergence of digital twin simulation (dos Santos et al., 2022). Section 4.4.2 describes a manufacturing context in which product completion times are classified as early or late at the start of a product cycle, based on simulation-generated completion times from the nearest neighbours of the current system state. At its highest level of detail, the system state is completely descriptive of the system at any moment in time, and so identifying nearest neighbours on system state provides a logical basis for predictions (Laidler et al., 2020). For this reason, we propose a method of distance metric learning (DML) tailored to the characteristics of sample path data.

DML has been shown to improve the performance of several distance-based tasks, including nearest neighbour predictions (Weinberger and Saul, 2009), clustering (Xing et al., 2002), and information retrieval (McFee and Lanckriet, 2010), with extensive research efforts resulting in a wealth of notable formulations (Kulis, 2013). However, current DML algorithms are built with an implicit assumption of having unique feature vectors for training, an assumption which conflicts with the nature of sample path data. The typically discrete representation and recurrent nature of the system state in a DES model provides repeated feature vector observations, and calculating objective functions as a sum over individual points becomes inefficient. More significantly, inherent simulation randomness results in stochastic classifications. Repeated feature vectors belonging to different classes will often introduce conflicting constraints when existing DML methods are applied, and can lead to k-nearest neighbours (kNN) committing "*substantial errors*" (Suárez et al., 2021). In the current work, we address this shortcoming by proposing a method which we refer to as Stochastic Neighbourhood Components Analysis (SNCA). This is based on the Neighbourhood Components Analysis (NCA) of Goldberger et al. (2004), which builds a probabilistic model of class assignment. We modify NCA to the context of repeated data points and stochastic classifications.

Although providing a large family of problems, the simulation domain is not the beginning and the end of our application. DES sample paths identify a characteristic type of data which arises in numerous contexts. Crowdsourced datasets, for example, in which responses are gathered from a *crowd* of people, can often result in multiple annotators providing various responses to the same labelling task (Vaughan, 2017). Tasks such as sentiment analysis or media ratings are subjective in nature and certainly result in stochastic labelling. Beyond this, however, many real-life situations experience naturally stochastic behaviour. Medical prognoses (Suo et al., 2018), financial forecasts (Cao and Tay, 2001), and sporting events (Horvat and Job, 2020), for example, are often modelled with discrete features and encounter natural variability in their observed outcomes. In summary, repeated feature vectors and stochastic labelling characterise a large body of classification problems. While methods for the statistical analysis of such data exist (Hastie et al., 2009), it is unaccommodated by methods of DML, and so we see a formulation tailored to this context as a useful contribution.

The remainder of the paper is organised as follows. Section 4.2 establishes a relevant background in simulation analytics and DML. In Section 4.3, we give a characterisation of the data and the proposed methodology for SNCA, including attention to the theoretical convergence of the optimal solution. We present experimental results to assess the performance of SNCA on sample path data in Section 4.4, before the paper concludes with a brief summary in Section 4.5.

4.2 Related Work

Numerous texts provide a comprehensive study on the subject of stochastic simulation (Nelson and Pei, 2021; Law and Kelton, 2007). In this section, we focus attention on the emerging topic of simulation analytics (Nelson, 2016), which provides a motivation for the current work. We also introduce the task of DML, and present a focused review of the existing literature.

4.2.1 Simulation Analytics

Analysis of simulation has traditionally centred around static summaries of long-run performance, with average performance *over* time being prioritised above dynamic performance *through* time. More recently, however, simulation analytics aims to prioritise the latter, recognising that a far more complete picture of system behaviour can be unlocked by directing machine learning efforts to the simulation sample path. We summarise some notable publications on this theme.

A number of papers have emerged in which sample path data are used to build metamodels for dynamic predictions. In the context of a queueing network, Ouyang and Nelson

(2017) proposed a two-stage logistic regression modelling approach in which the state and time aspects of the sample path are treated separately, while Jiang et al. (2020) also use a logistic regression model to dynamically predict the risk of financial portfolios. Morgan and Barton (2022), meanwhile, show that Fourier analysis can successfully detect changes in the trajectories of individual state variables to discriminate between congested and uncongested systems. Moving to distance-based methods, Lin et al. (2019) suggest a kNN approach to provide performance predictions based on simulation time. A first investigation of the potential for DML in simulation analytics was provided in Laidler et al. (2020), with the independent variable being extended to a multidimensional description of the system state. However, a tailored DML approach was not offered, and this is instead the main contribution of the current work.

Beyond the sphere of research, applications of machine learning in simulation are also filtering through to practitioners. Commercial providers of simulation software, including for example Simul8[1], Simio[2], and AnyLogic[3], have recognised its benefits and provide easy integration with machine learning models. Crucially, the support is already in place for research advances in simulation analytics to quickly deliver impact to simulation users.

4.2.2 Distance Metric Learning

Distance calculations among data points comprise a fundamental aspect of many machine learning tasks, and DML is therefore treated as an important objective in its own right (Kulis, 2013). In a fully supervised setting, real-valued, multivariate feature vectors, $\boldsymbol{x}_i \in \mathbb{R}^d$, are supplied with a class label, y_i, and the goal is to adapt a pairwise distance function over the feature vectors, such that nearby points are more likely to belong to the same class. A common approach to DML is to learn a generalised Mahalanobis distance. Our work in this chapter conforms to this framework, and this becomes the focus of our review.

[1]https://www.simul8.com
[2]https://www.simio.com
[3]https://www.anylogic.com

Defining the squared distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as $d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top M(\boldsymbol{x}_i - \boldsymbol{x}_j)$, the family of Mahalanobis metrics is parameterised by restricting M to the set of $d \times d$ positive semidefinite matrices, which we denote by $\mathbb{S}_+^d$. The task of learning a suitable M is generally expressed as an optimisation problem of the following form:

$$\min_{M \in \mathbb{S}_+^d} \ell(M, \mathcal{D}_n) + \lambda r(M).$$

Here, ℓ is a loss function relating the suitability of M to the supervision brought by the data, $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and $r(M)$ describes some regularisation on the values of M, with the regularisation parameter, $\lambda \geq 0$. Many formulations have been proposed, with differences arising in their choice of loss functions and regularisation. For a comprehensive review of established methods, refer to surveys provided by Kulis (2013), Bellet et al. (2014), and, more recently, Li and Tian (2018). Here, we limit our discussion to the main directions of the research, and establish a relevant background for the current work.

A common approach converts supervision into the form of similarity judgments among tuples of training points. For instance, sets $\mathscr{S}$ and $\mathscr{D}$ are often used to contain pairs of training points deemed to be similar and dissimilar, respectively. Under this framework, the loss function, $\ell(M, \mathcal{D}_n)$, is used to encode violations of the desired relationships under the metric induced by M. Construction of a linear loss function leads to convex formulations and allows the problem to be treated as a semidefinite program. The earliest example of this is Xing et al. (2002), with the following intuitive formulation:

$$\min_{M \in \mathbb{S}_+^d} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{s.t.} \quad \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{D}} d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \gamma.$$

A succession of notable semidefinite programming formulations have since emerged, including the widely used Large Margin Nearest Neighbors (LMNN, Weinberger and Saul (2009)). Motivated by kNN, LMNN seeks local neighbourhoods of each instance to be populated with other instances of the same class, while those of different class are repelled by a '*large*' margin. LMNN has been the subject of a number of extensions. Torresani and Lee (2006) combine LMNN with dimensionality reduction of the feature space, and explore non-linear kernel methods, while Kedem et al. (2012) suggest alternative extensions to a non-linear metric. Further notable works, also making use of supervision sets, $\mathscr{S}$ and $\mathscr{D}$, and offering an information-theoretic perspective, include Bar-Hillel et al. (2003) and Davis et al. (2007).

The approach of populating supervision sets with pairwise relationships is less natural in our context of repeated feature vectors and stochastic classifications. In particular, we might expect the same pair to appear a number of times in *both* $\mathscr{S}$ and $\mathscr{D}$, leading to either inefficient or infeasible formulations. Instead, the stochastic nature of the class labels leads us to prefer a probabilistic formulation, and we hence turn our attention to existing probabilistic methods.

An influential formulation known as Neighbourhood Components Analysis (NCA, Goldberger et al. (2004)) proceeds to model the nearest neighbour probabilities with a softmax normalisation over distances. A point, $\boldsymbol{x}_i$, identifies its nearest neighbour as $\boldsymbol{x}_j$ with probability p_{ij}:

$$p_{ij} = \frac{\exp\{-\|A\boldsymbol{x}_i - A\boldsymbol{x}_j\|_2^2\}}{\sum_{k\neq i}\exp\{-\|A\boldsymbol{x}_i - A\boldsymbol{x}_k\|_2^2\}}, \quad p_{ii} = 0. \tag{4.2.1}$$

This imposed distribution provides a continuous relaxation to the deterministic point mass distribution on the nearest neighbour. With $\boldsymbol{x}_i$ inheriting its classification from a neighbour selected from this distribution, the quantity, $p_i = \sum_{j\,:\,y_j=y_i} p_{ij}$, represents the probability of $\boldsymbol{x}_i$ having the class assignment of y_i. The suggested objectives are to maximise the expected leave-one-out accuracy of the nearest neighbour classifier under this distribution, or the log-likelihood function:

$$\max_A \sum_{i=1}^n p_i \quad \text{or} \quad \max_A \sum_{i=1}^n \log\left(p_i\right).$$

The optimisation of NCA is performed over an unconstrained matrix, A, which reveals a Mahalanobis matrix, M, via the relation $A^\top A = M$. Any $M \in \mathbb{S}_+^d$ permits this decomposition, so a Mahalanobis metric can equivalently be viewed as a projection of Euclidean space under a linear transformation, $\boldsymbol{x} \to A\boldsymbol{x}$. With this perspective, optimising the transformation matrix, A, provides an alternative perspective to the task of Mahalanobis metric learning. Dimensionality reduction can be directly sought by restricting the number of rows in A, whilst costly projections onto $\mathbb{S}_+^d$ are avoided. However, an optimisation of A often comes at the expense of convexity: a loss function which is convex in M is typically non-convex in A. This is no concern in the case of NCA, since the objective function is non-convex in either parameterisation.

Underneath NCA lies a predictive model of class distributions: p_i can be seen as a kernel density estimator for the conditional probability of y_i given $\boldsymbol{x}_i$ (Devroye and Wagner, 1980). The form of the softmax transformation (4.2.1) disguises a Gaussian kernel function, with the common bandwidth parameter being absorbed into the scale of A. Weinberger and Tesauro (2007) more explicitly present a metric learning method in this context, albeit for a continuous target variable. In particular, kernel regression estimates for the target variable are constructed in the same form as above, and a metric is sought to minimise the mean squared error.

A similar probabilistic model was proposed by Peltonen and Kaski (2005), in which an expression of the data log-likelihood is maximised by evaluating conditional class probability estimators of a similar form as those of NCA. Indeed, the estimates introduced by NCA establish a popular framework which has been adopted by a number of other authors, and also provides inspiration for our own approach. Globerson and Roweis (2005) obtain a convex objective function, while Tarlow et al. (2013) explicitly extend the NCA objective to reflect the expected kNN accuracy for any choice of k. Further related extensions of NCA include Wang and Tan (2014), who consider the context of noisy labels arising from measurement error. Our own context involves true noise arising from a probabilistic labelling mechanism, and in this direction, Yang (2020) adapts NCA to the context of probabilistic labels, in which a full probabilistic class distribution is assumed to be known for each instance. Our own approach assumes no such knowledge.

The NCA framework provides a useful foundation for probabilistic metric learning, although it falls short of addressing the *repeating* characteristic of our input observations. We proceed in the next section to introduce a multinomial framework for our data setting, and propose an NCA-inspired metric learning formulation to serve it.

4.3 Methodology

We begin by describing a data framework in which repeated feature vectors and stochastic labelling characterise a set of classification problems. To establish a mechanism generating observation pairs, we assume a distribution, $q_{\boldsymbol{X},Y}$, over a pair of discrete random variables, $\boldsymbol{X}$ and Y. The input variable, $\boldsymbol{X}$, takes values from a set, $\mathcal{X} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_m\}$, containing a finite number of vectors in $\mathbb{R}^d$, which we refer to as states, whilst Y represents a class label drawn from a set, $\mathcal{Y}$, containing at least two unordered values. Our task is to find a pairwise distance function acting on the states in $\mathcal{X}$, which in some way reflects similarity of the conditional class distributions of $Y \mid \boldsymbol{X}$. We receive supervision from a finite training sample, $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, drawn independently from $q_{\boldsymbol{X},Y}$.

In many applications, the training sample will not represent the entirety of $\mathcal{X}$. Whilst we expect repeats of *some* states, many others may be observed only once or not at all. Therefore, in classifying a new input, $\boldsymbol{x}^\star$, we are not expecting to rely solely on observed training repetitions of $\boldsymbol{x}^\star$ and an empirical class distribution. Instead, we require a distance metric to point us to neighbouring states which, through a similarity of class distributions, will make good predictors. As the basis for this, we seek a model which can learn from the more frequently attended states to extend a conditional structure for $Y \mid \boldsymbol{X}$ across the whole set of $\mathcal{X}$.

The discrete nature of $\mathcal{X} \times \mathcal{Y}$ allows us to view $\mathcal{D}_n$ as a multinomial sample. We introduce a random variable, $C_l^y(n) = |\{i \in \{1, 2, \dots, n\} \colon \boldsymbol{X}_i = \boldsymbol{b}_l, Y_i = y\}|$, to represent the observed frequency of the pair $(\boldsymbol{b}_l, y)$ in a data sample of size n. The random vector,

$\{C_l^y(n)\}_{l=1,\ldots,m}^{y\in\mathcal{Y}}$, follows a multinomial distribution with parameters n and $\{q(\boldsymbol{b}_l, y)\}_{l=1,\ldots,m}^{y\in\mathcal{Y}}$, and we can represent $\mathcal{D}_n = \{c_l^y\}_{l=1,\ldots,m}^{y\in\mathcal{Y}}$ as a realisation of this random vector. Conditional on $\mathcal{D}_n$, the maximum likelihood estimates of the multinomial parameters are provided by $\hat{q}(\boldsymbol{b}_l, y) = c_l^y/n$. Using c_l to represent $\sum_{y\in\mathcal{Y}} c_l^y$, we also introduce marginal and conditional data distributions with $\hat{q}(\boldsymbol{b}_l) = c_l/n$ and $\hat{q}(y \mid \boldsymbol{b}_l) = c_l^y/c_l$ provided $c_l > 0$, respectively. Although these distributions depend on $\mathcal{D}_n$, we omit this to simplify the notation.

The remainder of this section is organised as follows. We introduce a probabilistic framework for SNCA in Section 4.3.1, through which a conditional distribution of $Y \mid \boldsymbol{X}$ is modelled. We present an objective function and a proposed scheme for its optimisation in Section 4.3.2. Section 4.3.3 discusses the distinction between SNCA and NCA in the data context described above, and Section 4.3.4 presents a convergence result relating to the asymptotic behaviour of the optimal solution.

4.3.1 A Probabilistic Framework

In the style of NCA, we impose a probability distribution for neighbour assignment based on a softmax function over distances. In contrast to NCA, however, this is applied over multinomial cells as opposed to over individual $\boldsymbol{x}_i$. Under a transformation matrix, A, a cell with $\boldsymbol{X} = \boldsymbol{b}_l$ identifies the cell $(\boldsymbol{b}_h, y)$ as its nearest neighbour with probability $p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l)$:

$$p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l) = \frac{c_h^y \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{\sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_k\|_2^2\}}\ , \quad p^A((\boldsymbol{b}_l, y) \mid \boldsymbol{b}_l) = 0.$$

Again, we omit the conditioning on $\mathcal{D}_n$ from the notation. Assuming classifications to be inherited from a cell selected by this distribution, the probability of classifying $\boldsymbol{X} = \boldsymbol{b}_l$ as class y is given by

$$p^A(y \mid \boldsymbol{b}_l) = \sum_{h=1}^{m} p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l).$$

This provides us with a complete conditional distribution, $p^A_{Y|\boldsymbol{X}}$, for any matrix, A, constructed as a weighted average over observed class distributions, $\hat{q}_{Y|\boldsymbol{X}}$. That is, we can write

$$p^A(y \mid \boldsymbol{b}_l) = \sum_{h=1}^{m} p_{lh} \hat{q}(y \mid \boldsymbol{b}_h),$$

where the weight, $p_{lh} = \sum_{y \in \mathcal{Y}} p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l)$, given to the observed class distribution of the neighbour, $\boldsymbol{b}_h$, decreases exponentially with the distance under A of $\boldsymbol{b}_h$ from $\boldsymbol{b}_l$. With this perspective, we treat our task as an optimisation problem to find a transformation, A, to align the model distribution, $p^A_{Y|\boldsymbol{X}}$, with the data distribution, $\hat{q}_{Y|\boldsymbol{X}}$.

4.3.2 Objective Function and Optimisation

The primary NCA objective proposed by Goldberger et al. (2004) is to maximise the expected number of correct classifications on the training data under the modelled class probability estimates. The same intention leads us to the following objective function for SNCA:

$$f_n(A) = \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) p^A(y \mid \boldsymbol{b}_l).$$

Whilst a maximisation of $f_n(A)$ constitutes a natural objective, it does not directly target the alignment of $p^A_{Y|\boldsymbol{X}}$ with $\hat{q}_{Y|\boldsymbol{X}}$. This alignment is, however, represented with a Kullback-Leibler (KL) divergence via a log-likelihood maximisation. Obtaining the modelled joint distribution as $p^A(\boldsymbol{b}_l, y) = p^A(y \mid \boldsymbol{b}_l)\hat{q}(\boldsymbol{b}_l)$, the multinomial likelihood and log-likelihood of $\mathcal{D}_n$ can be expressed as the following functions of A, respectively:

$$L(A; \mathcal{D}_n) = \frac{n!}{\prod_l \prod_y c_l^y!} \prod_{l=1}^{m} \prod_{y \in \mathcal{Y}} p^A(\boldsymbol{b}_l, y)^{c_l^y},$$

$$\log L(A; \mathcal{D}_n) = \log(n!) + \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} c_l^y \log(p^A(\boldsymbol{b}_l, y)) - \log(c_l^y!).$$

Recalling that $\hat{q}(\boldsymbol{b}_l, y) = c_l^y/n$, we see that a maximisation of the log-likelihood corresponds to a maximisation of the function, $g_n(A)$, and observe the connection to a KL divergence:

$$\begin{aligned} g_n(A) &= \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y)\log(p^A(\boldsymbol{b}_l, y)) \\ &= \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y)\log\left(\frac{p^A(\boldsymbol{b}_l, y)}{\hat{q}(\boldsymbol{b}_l, y)}\right) + \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y)\log(\hat{q}(\boldsymbol{b}_l, y)) \\ &= -D_{\mathrm{KL}}(\hat{q}_{\boldsymbol{X},Y}\|p^A_{\boldsymbol{X},Y}) - H(\hat{q}_{\boldsymbol{X},Y}), \end{aligned}$$

where H represents the Shannon entropy (Shannon, 1948). Since $H(\hat{q}_{\boldsymbol{X},Y})$ is independent of A, the maximisation of $g_n(A)$ corresponds to a minimisation of the KL divergence of $p^A_{\boldsymbol{X},Y}$ from $\hat{q}_{\boldsymbol{X},Y}$. We have a similar interpretation in relation to the conditional distributions:

$$\begin{aligned} g_n(A) &= \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l)\hat{q}(y \mid \boldsymbol{b}_l)\log(\hat{q}(\boldsymbol{b}_l)p^A(y \mid \boldsymbol{b}_l)) \\ &= \sum_{l=1}^{m}\hat{q}(\boldsymbol{b}_l)\sum_{y\in\mathcal{Y}} \hat{q}(y \mid \boldsymbol{b}_l)\log p^A(y \mid \boldsymbol{b}_l) + \sum_{l=1}^{m}\hat{q}(\boldsymbol{b}_l)\log\hat{q}(\boldsymbol{b}_l) \\ &= -\mathbb{E}_{\hat{q}_{\boldsymbol{X}}}[D_{\mathrm{KL}}(\hat{q}_{Y|\boldsymbol{X}}\|p^A_{Y|\boldsymbol{X}})] - \mathbb{E}_{\hat{q}_{\boldsymbol{X}}}[H(\hat{q}_{Y|\boldsymbol{X}})] - H(\hat{q}_{\boldsymbol{X}}). \end{aligned}$$

Therefore, maximisation of $g_n(A)$ minimises the expected KL divergence of the class conditional distributions of the model from the data, under $\hat{q}_{\boldsymbol{X}}$. This perspective perhaps most convincingly aligns with our intuition for a desirable metric, whilst making use of a common probability-based divergence measure.

The differentiability of $g_n(A)$ allows us to apply a gradient-based optimiser. Recalling that $p_{lh} = \sum_{y\in\mathcal{Y}} p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l)$, and denoting $\boldsymbol{b}_{lh} = \boldsymbol{b}_l - \boldsymbol{b}_h$, we have the following gradient:

$$\frac{\partial g_n(A)}{\partial A} = 2A\sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y)\sum_{h\neq l}\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^{\top}p_{lh}\left(1 - \frac{\hat{q}(y \mid \boldsymbol{b}_h)}{p^A(y \mid \boldsymbol{b}_l)}\right).$$

Details for deriving this expression can be found in Appendix A.1, along with a similar expression for $\partial f_n(A)/\partial A$.

In the metric learning literature, imposing some regularisation on the parameters of the metric is a common and accepted practice (Kulis, 2013). In this work, we choose to require mutual orthogonality among the rows of A. Encouraging orthogonality has recently been recognised as an effective regularisation for metric learning (Dutta et al., 2020), and besides leading to a smaller set of projection vectors which bear less redundancy, benefits in mitigating the effects of class imbalance and avoiding overfitting have also been recognised (Xie et al., 2018).

To introduce orthogonality, we propose a row-wise construction of the solution, optimising each row in the null space of the previously determined rows. Performing this null space projection is a computationally inexpensive step, and leads to orthogonal rows of A without the need for optimisation constraints. The proposed optimisation procedure is summarised in Algorithm 1. We choose not to require unit length vectors, since a flexible scaling allows the predictive relevance of the various orthogonal directions to be reflected. In practice, we observe diminishing row norms. Correspondingly, the diminishing improvements to the

Algorithm 1 SNCA

```
**Input:** $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
$\boldsymbol{a}_1^\star \leftarrow \arg\max_{\boldsymbol{a}\in\mathbb{R}^{1\times d}} g_n(\boldsymbol{a})$
$A^\star \leftarrow \boldsymbol{a}_1^\star$
$\hat{\boldsymbol{a}}_1^\star \leftarrow \boldsymbol{a}_1^\star / \|\boldsymbol{a}_1^\star\|_2$
$P \leftarrow I_d - \hat{\boldsymbol{a}}_1^{\star\top} \hat{\boldsymbol{a}}_1^\star$
$\boldsymbol{a}_2^\star \leftarrow \arg\max_{\boldsymbol{a}\in\mathbb{R}^{1\times d}} g_n\left(\begin{bmatrix} A^\star \\ \boldsymbol{a}P \end{bmatrix}\right)$
$j \leftarrow 2$
**while** $g_n\left(\begin{bmatrix} A^\star \\ \boldsymbol{a}_j^\star \end{bmatrix}\right) > g_n(A^\star)$ **do**
    $A^\star \leftarrow \begin{bmatrix} A^\star \\ \boldsymbol{a}_j^\star \end{bmatrix} \in \mathbb{R}^{j\times d}$
    $\hat{\boldsymbol{a}}_j^\star \leftarrow \boldsymbol{a}_j^\star / \|\boldsymbol{a}_j^\star\|_2$
    $P \leftarrow (I_d - \hat{\boldsymbol{a}}_j^{\star\top} \hat{\boldsymbol{a}}_j^\star) P$
    $\boldsymbol{a}_{j+1}^\star \leftarrow \arg\max_{\boldsymbol{a}\in\mathbb{R}^{1\times d}} g_n\left(\begin{bmatrix} A^\star \\ \boldsymbol{a}P \end{bmatrix}\right)$
    $j \leftarrow j + 1$
**end while**
**Return:** $A^\star$
```

objective value suggest a natural stopping rule, bringing explicit dimensionality reduction to the solution.

Whilst we optimise a non-convex objective function, falling into local optima remains a concern. In the experiments presented in Section 4.4, we make multiple restarts for each optimisation step. We choose a Latin Hypercube space-filling design using software provided by Urquhart et al. (2020) to generate a set of initialisations.

4.3.3 Relationship to NCA

The formulations of NCA and SNCA exactly coincide when applied to training data without repeated observations, i.e. training data for which $\sum_{y\in\mathcal{Y}} c_l^y \leq 1$ holds for all $l = 1, 2, \ldots, m$. When this condition does not hold, a difference arises. Expressing both formulations as

$$\max_A \sum_l \sum_y \hat{q}(\boldsymbol{b}_l, y) \log p^A(y \mid \boldsymbol{b}_l),$$

the difference occurs in the model estimates of the conditional probabilities, $p^A(y \mid \boldsymbol{b}_l)$. Whenever $\hat{q}(\boldsymbol{b}_l, y) > 0$, these estimates are as follows:

$$\text{NCA: } p^A_{(N)}(y \mid \boldsymbol{b}_l) = \frac{c_l^y - 1 + \sum_{h\neq l} c_h^y \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{c_l - 1 + \sum_{h\neq l} c_h \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}, \tag{4.3.2}$$

$$\text{SNCA: } p^A_{(S)}(y \mid \boldsymbol{b}_l) = \frac{\sum_{h\neq l} c_h^y \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{\sum_{h\neq l} c_h \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}. \tag{4.3.3}$$

The difference between (4.3.2) and (4.3.3) stems from the inclusion and exclusion, respectively, of a point's repeats in the softmax construction of its neighbour distribution. Since identical points will always have a distance of zero, the terms $c_l^y - 1$ and $c_l - 1$ appear regardless of A and effectively act as bias to $p^A_{(N)}(y \mid \boldsymbol{b}_l)$. As a result, the influence of A on the objective function of NCA is reduced. The adjustment made by SNCA is to exclude identical neighbours from consideration in the softmax functions, which allows the SNCA objective to solely reflect the quality of distance relationships among non-identical points.

We also recognise inconsistencies arising in the NCA estimates. Specifically, we may not have $\sum_{y\in\mathcal{Y}} p^A_{(N)}(y \mid \boldsymbol{b}_l) = 1$. As n increases, the discrepancy reduces, and we can in fact identify an asymptotically optimal solution. In particular, a solution, A, with infinite column norms, such that non-identical points are infinitely far apart, will yield $p^A_{(N)}(y \mid \boldsymbol{b}_l) = (c^y_l - 1)/(c_l - 1) \to \hat{q}(y \mid \boldsymbol{b}_l)$ as $n \to \infty$. This identifies a non-informative solution to which NCA becomes susceptible, under its log-likelihood objective function, as the data size increases. This essentially represents the situation of fully overfitting to the data, a hazard which SNCA avoids by introducing implicit leave-one-out validation during learning.

4.3.4 Convergence of Optimal Solutions

In Section 4.3.2, we understood $g_n(A)$ in relation to the observed distribution, $\hat{q}_{\boldsymbol{X},Y}$, of a finite sample. In this section, we seek to establish convergence guarantees with respect to the underlying true distribution, $q_{\boldsymbol{X},Y}$.

The strong law of large numbers (Kolmogorov and Širjaev, 1992) ensures consistency of $\hat{q}_{\boldsymbol{X},Y} \overset{a.s.}{\to} q_{\boldsymbol{X},Y}$ as $n \to \infty$, and by the continuous mapping theorem we obtain the pointwise limits over fixed $A \in \mathbb{R}^{d\times d}$ as $n \to \infty$ of $g_n(A)$:

$$g_n(A) \overset{a.s.}{\to} \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} q(\boldsymbol{b}_l, y) \log\left(q(\boldsymbol{b}_l) \frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y) \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k) \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_k\|_2^2\}} \right).$$

We use $g(A)$ to denote the limit of $g_n(A)$ given above, and consider the (set of) maximisers of $g(A)$, denoted $A^\star_g$, to be the true system-optimal solutions. To address the asymptotic behaviour of a finite sample solution, $\hat{A}_g$, we refer to a result relating to the consistency of Sample Average Approximation (SAA) estimators (Theorem 5.3, Shapiro et al. (2021)). Adopting the notation introduced here, the result is as follows:

Theorem 4.3.1. *Suppose there exists a compact subset $C \subset \mathbb{R}^{d\times d}$ such that:*

(i) $A^\star_g$ is nonempty and contained in C,

(ii) for sufficiently large n, with probability 1, $\hat{A}_g$ is nonempty and contained in C,

(iii) $g(A)$ is finite-valued and continuous on C, and

(iv) $g_n(A) \overset{a.s.}{\to} g(A)$ as $n \to \infty$, uniformly on C.

Then $g_n(\hat{A}_g) \overset{a.s.}{\to} g(A_g^\star)$ and $\mathbb{D}(\hat{A}_g, A_g^\star) \overset{a.s.}{\to} 0$ as $n \to \infty$.

Here, $\mathbb{D}(B, C)$ describes the deviation of the set B from the set C, defined as $\mathbb{D}(B, C) = \sup_{x \in B}\{\inf_{x' \in C}\|x - x'\|\}$. Central to this theorem is the assumption of a compact subset C in which $A_g^\star$ is contained. This excludes the possibility of infinite-valued solutions, which, although theoretically possible, we have not encountered in practice. For simplicity, we choose to solve an unconstrained optimisation problem, and assume that *(i)* and *(ii)* can be satisfied.

To address condition *(iii)*, we consider the argument of the logarithms to ensure that $g(A)$ is finite-valued. These arguments represent the limits of $p^A(\boldsymbol{b}_l, y)$ as $n \to \infty$ for all $\boldsymbol{b}_l$ and y. Whilst these naturally reside in $[0, 1]$, we require them in the slightly stricter domain of $[a, 1]$, where $a > 0$. That is, we seek to exclude the possibility of $p^A(\boldsymbol{b}_l, y)$ converging to zero when $q(\boldsymbol{b}_l, y) > 0$. This can be ensured with the following mild condition on $q_{\boldsymbol{X},Y}$:

$$\sum_{h \neq l} q(\boldsymbol{b}_h, y) > 0 \;\; \forall \boldsymbol{b}_l \in \mathcal{X}, y \in \mathcal{Y} \text{ with } q(\boldsymbol{b}_l, y) > 0. \tag{4.3.4}$$

In other words, each class, $y \in \mathcal{Y}$, must be observable in conjunction with at least two states in $\mathcal{X}$. Alongside the assumption that $A \subset C$, condition (4.3.4) ensures that $g(A)$ is finite-valued on C, and since also continuous, condition *(iii)* is satisfied. A justification of the uniform convergence required by condition *(iv)* can be found in Appendix A.2.

Whilst we understand $\hat{A}_g$ as minimising an expected KL divergence of the class distributions of the model from the data, we consider $A_g^\star$ as relating to this expectation under the true data-generating distribution. Theorem 4.3.1 provides a helpful assurance that as the training sample increases and the empirical distribution, $\hat{q}_{\boldsymbol{X},Y}$, approaches its truth, our metric solutions should also approach their truth.

4.4 Experiments on Simulated Data

In this section, we explore the performance of SNCA using data generated by stochastic simulation. We introduce a simple queueing model with a small and comprehensible state space, before presenting a more realistic application in the simulation of a semiconductor fabrication plant. Accompanying data and code used for the analysis in this section are available on GitHub (Laidler, 2022).

4.4.1 A Tandem Queueing System

Consider the simple queueing system represented in Figure 4.4.1. Two stations are arranged in tandem, with Station $I \in \{A, B\}$ containing s_I identical servers and c_I total capacity. Customers of type $k \in \{1, 2\}$ arrive to Station A according to a Poisson process with rate λ_k, and experience service times at Station I which are exponentially distributed with rate μ_{Ik}. The parameter q is used to denote the probability of rework.

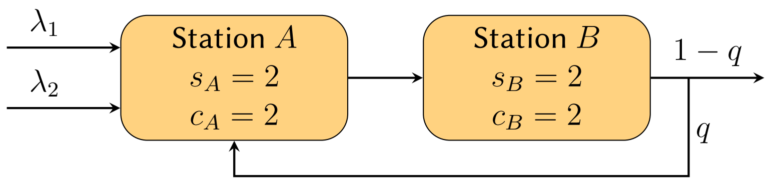


Figure 4.4.1: A tandem queueing system with feedback.

This two-station, two-type set-up suggests a four-dimensional system state, $\boldsymbol{X} = (X_{A1}, X_{A2}, X_{B1}, X_{B2})$, where X_{Ik} represents the number of type k customers at Station I. The memoryless property of the exponential distribution makes elapsed service times irrelevant, while setting $s_I = c_I$ for $I \in \{A, B\}$ removes queueing space from both stations and ensures that $\boldsymbol{X}$ fully characterises the observable system state. We set $s_I = c_I = 2$ for $I \in \{A, B\}$, which allows $\boldsymbol{X}$ to take 36 distinct states.

We refer to Station $I \in \{A, B\}$ as being blocked whenever $X_{I1} + X_{I2} = c_I$, since no customers can enter the station in this state. In the logic of the system, customers completing their service in Station A will occupy space there while Station B is blocked. More significantly, customers arriving to a blocked Station A (including those returning from Station B) are immediately lost from the system. As such, blocking in Station A represents a critical state for the system, and our interest in this example is in predicting the future probability of this event based on the currently observed state. In particular, we observe the system state, $\boldsymbol{X}(t)$, at time t, and predict whether Station A will be blocked at time $t + T$. Let

$$Y(t) = \begin{cases} 1 & \text{if } X_{A1}(t+T) + X_{A2}(t+T) = c_A, \\ 0 & \text{otherwise.} \end{cases}$$

Simulating the system provides the data, $\mathcal{D}_n = \{(\boldsymbol{X}(t_i), Y(t_i))\}_{i=1}^{n}$, and we choose the recording times such that $t_{i+1} = t_i + T$ in order to obtain non-overlapping observations. To provide a roughly even distribution for Y, we simulated the system with the parameters $\lambda_1 = \lambda_2 = 2, \mu_{A1} = 2, \mu_{A2} = 3, \mu_{B1} = 1, \mu_{B2} = 2$, and $q = 0.2$, and we set the prediction horizon to $T = 0.25$. Choosing a smaller value of T would move towards a more deterministic problem, while a longer horizon would reduce the influence of $\boldsymbol{X}(t)$ and converge each state's conditional probability of $Y = 1$ towards the system's long-run blocking probability for Station A. With the specified system parameters, setting $T = 0.25$ provides a suitable middle ground in which the set of system states exhibit a range of varying class probabilities.

Metric Embeddings

On a dataset of size $n = 100{,}000$, we obtained the following SNCA solution, after 10 initialisations for each row's optimisation:

$$A = \begin{bmatrix} 7.39 & 6.87 & 3.06 & 1.77 \\ 0.31 & -0.71 & 0.25 & 1.01 \\ -0.15 & 0.03 & 0.31 & -0.01 \end{bmatrix}.$$

The transformation matrix consists of orthogonal rows representing data directions which are informative with respect to the class distributions. These rows are appended to the solution sequentially, and as such their relevance decreases and their magnitudes shrink, pointing us to an appropriate lower-dimensional solution. In this example, a dominant first dimension provides most of the projection variation, with a second and third dimension making smaller contributions. A fourth dimension was not found to provide any further improvements to the objective value.

We consider the first row of the solution as the most informative, and we can derive interpretation from the magnitudes in this row acting on the original data dimensions. X_{A1} is deemed slightly more relevant than X_{A2}, which we can understand since the system was simulated with $\mu_{A1} < \mu_{A2}$. This means that customers represented by X_{A1} tend to have longer service times and so are more likely to still occupy Station A when Y is observed. Customers in Station B can indirectly increase the risk of Station A becoming blocked, and in the same way $\mu_{B1} < \mu_{B2}$ leads to a first dimension to which X_{B1} makes a greater contribution than X_{B2}.

To visualise the effect of SNCA, we plot the 36 unique states projected into the solution space, showing the first two dimensions which capture the majority of the variation (Figure 4.4.2, top). Recall that our distance metric is represented by Euclidean distance in this transformed space. The projection plot shows the SNCA metric arranging the states

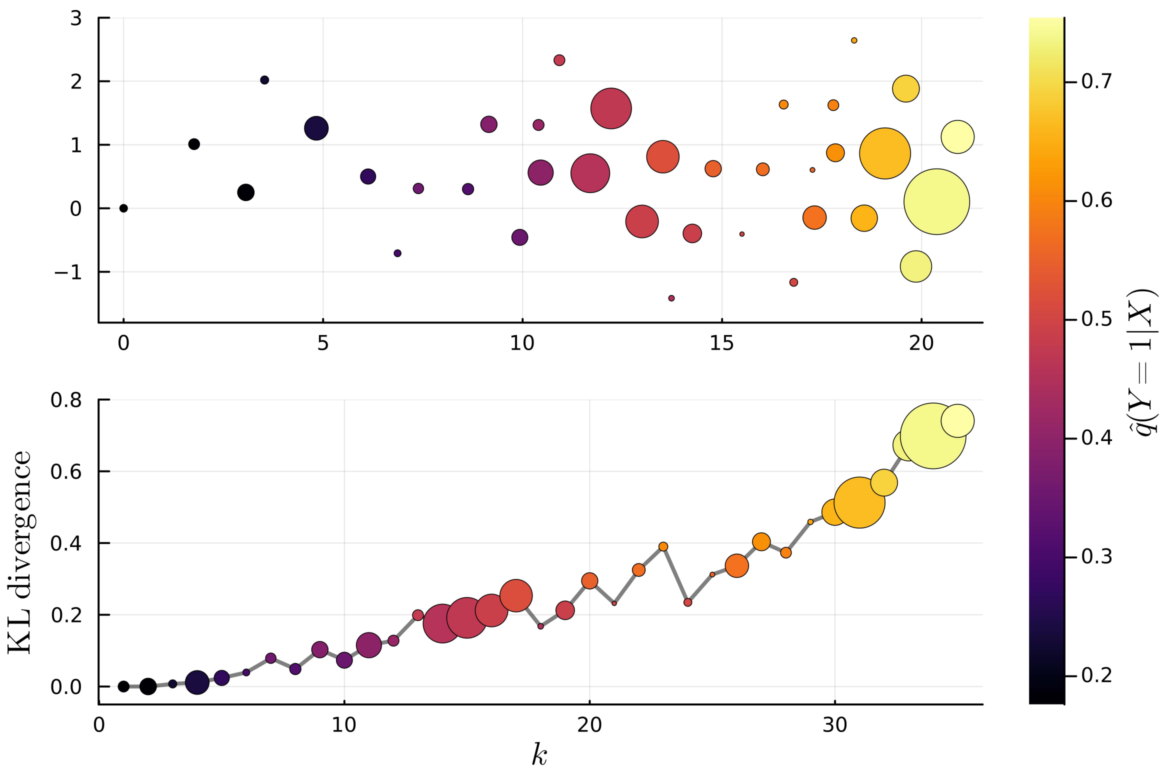


Figure 4.4.2: *Top*: the 36 unique states projected in the first two dimensions of the SNCA solution matrix. The colour of the points relates to their empirical class distribution, and the size is proportional to their observed frequency. *Bottom*: the KL divergence from the class distribution of the state $(0,0,0,0)$ to that of its k^{th} nearest neighbour.

with regards to the similarity of their stochastic class distributions, which are depicted by the colour scale. The state $\boldsymbol{X} = (0,0,0,0)$ represents the empty system and is leftmost in the projection plot. The lower plot explicitly lays out the states in the order of their distance from $(0,0,0,0)$, and shows that the KL divergence of a neighbouring state's class distribution from that of $(0,0,0,0)$ tends to increase as the neighbour moves further away.

To allow an appropriate comparison for plotting, we obtain the following two-dimensional NCA solution on the same dataset:

$$A = \begin{bmatrix} 341.23 & 234.95 & 317.98 & -78.4 \\ -15.27 & -10.11 & -14.54 & 3.8 \end{bmatrix}.$$

The projection and KL divergence plots corresponding to Figure 4.4.2 for this NCA solution

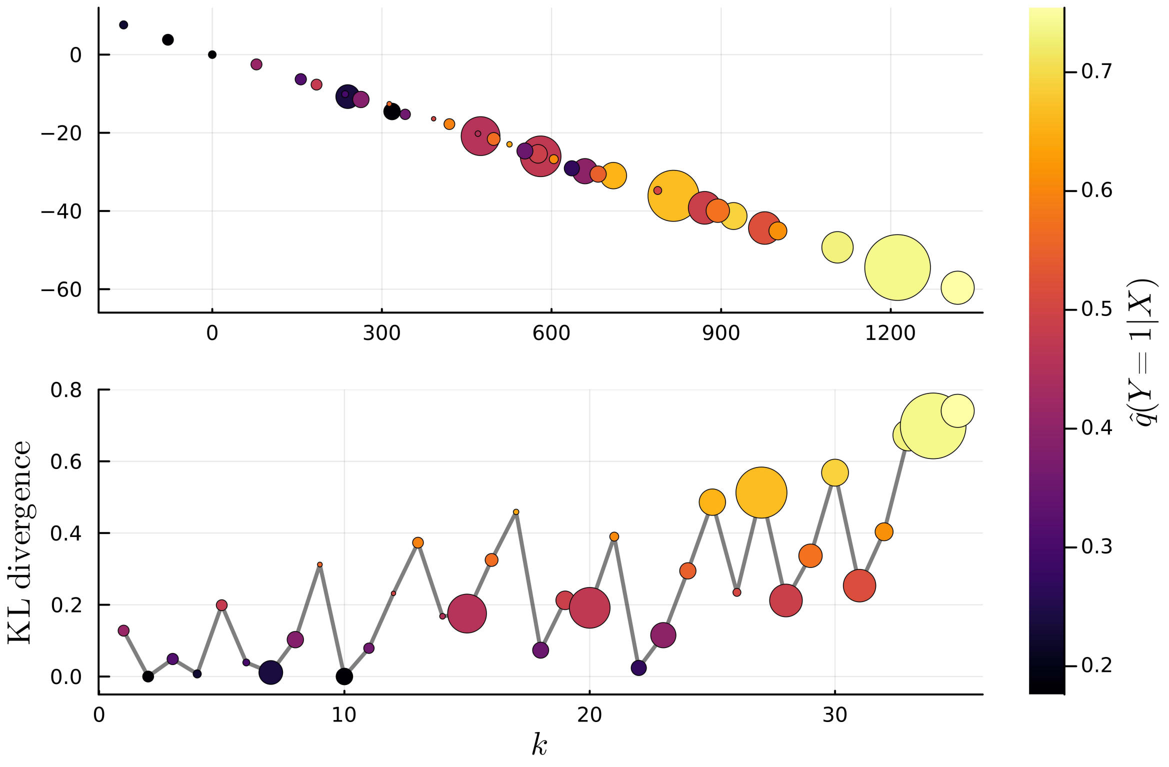


Figure 4.4.3: The equivalent plots from Figure 4.4.2 for a two-dimensional NCA solution. *Top*: the 36 unique states projected by the solution matrix. *Bottom*: the KL divergence from the class distribution of the state $(0, 0, 0, 0)$ to that of its k^{th} nearest neighbour.

are shown in Figure 4.4.3. The comparison between Figure 4.4.2 and Figure 4.4.3 highlights the deficiency of NCA when applied to discrete stochastic data, and the suitable corrections made by SNCA. Without orthogonality constraints, the two rows of the NCA solution each represent a very similar direction, and one which does not convincingly discriminate the class distributions. The adjustment made by SNCA to exclude identical points from the stochastic neighbour distributions, as described in Section 4.3.3, is effective as it forces the algorithm to find good quality neighbours. NCA, on the other hand, does not have this incentive, since identical $\boldsymbol{X}$ observations dominate the neighbour distributions and the influence of non-identical neighbours on the objective function is diminished.

Figure 4.4.4 replicates the lower plots of Figure 4.4.2 and Figure 4.4.3, averaged with respect to $\hat{q}_{\boldsymbol{X}}$ over all 36 states. Across the whole input space, SNCA has consistently arranged states' neighbours with broadly increasing divergence of their class distributions,

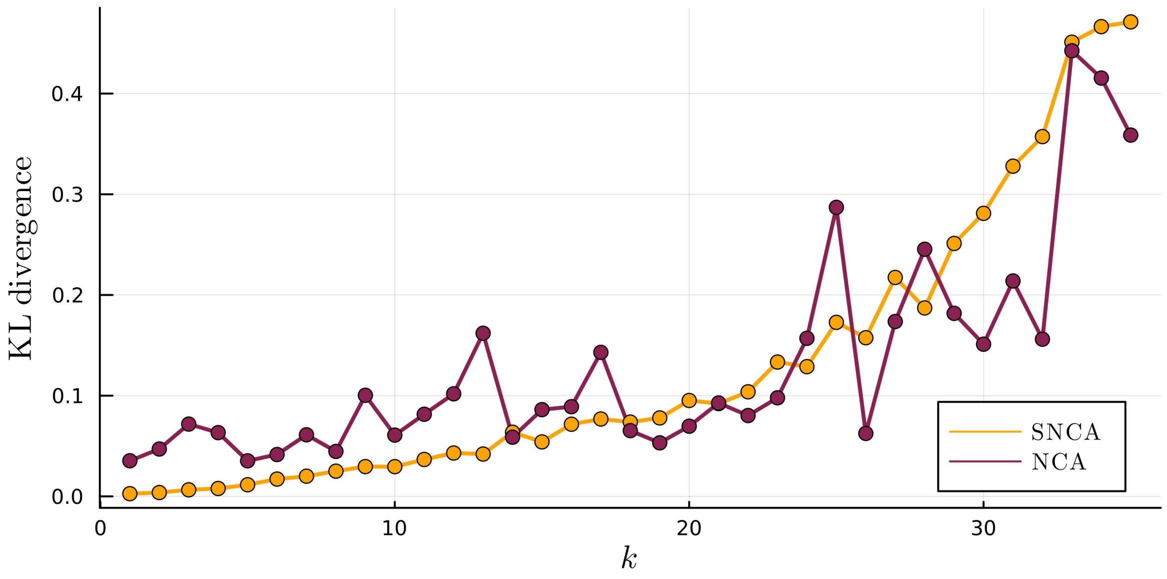


Figure 4.4.4: The average KL divergence from a point's class distribution to the class distribution of its k^{th} nearest neighbour, under the SNCA and NCA embeddings. This is equivalent to the lower plots of Figures 4.4.2 and 4.4.3 averaged with respect to $\hat{q}_{\boldsymbol{X}}$ over all states.

whilst NCA is less reliable in this regard. For the purpose of nearest neighbour predictions, the closer neighbourhoods are the most relevant, and we see SNCA bringing a desirable structure in terms of the similarity of class distributions within the closer neighbourhoods.

kNN Classification

As a further verification of metric performance, we display results of kNN classification under the learned distance metrics. To have a clearer interpretation of k in our data context, we consider k as the number of distinct neighbours, and include all of their repeats. In other words, we classify an individual point, $\boldsymbol{x}_i$, by the majority class of its k nearest neighbour states, and additionally include any states which are tied in distance with the k^{th}. Further, we exclude the other observations of $\boldsymbol{x}_i$'s own state from the neighbour search. This would be the nearest neighbour state under any metric, so its inclusion does not help a comparison across metrics. Further details of the kNN procedures used in this chapter, and the generation of the figures, are found in Appendix A.3.

Since this example carries only a small number of distinct $\boldsymbol{X}$ values, we only show results for $k = 1$. Figure 4.4.5 concerns metrics trained on datasets of size n, and shows on the left the leave-one-out correct classification rates on a large ($n = 100{,}000$) test dataset. This is to approximate the expected classification accuracy on data drawn randomly from $q_{\boldsymbol{X},Y}$. As benchmarks for comparison, we include the 1NN classification rate achieved using a Euclidean distance metric, and a 1NN classifier which omnisciently selects the optimal neighbour, defined as the state whose class distribution has minimum KL divergence from that of the query point. We are encouraged to see SNCA still performing well with small training datasets, and achieving classification rates close to optimal given the level of stochasticity in the system.

In practical problems with large input spaces, we do not expect to exhaust $\mathcal{X}$ in our training sample. For this reason, we are particularly interested in classification performance on states which are unrepresented in the metric training. Figure 4.4.5 (right) uses 6-fold cross-validation (CV) on the 36 unique states, through which the metrics were trained on data covering 30 states, and the classification accuracy recorded on the remaining 6. We are encouraged to see strong 1NN classification performance by SNCA on unseen states. For each n marked in the plots in Figure 4.4.5, the metric training was repeated with 10 independent datasets of size n, with the ribbons showing pointwise ± 1.96 standard errors

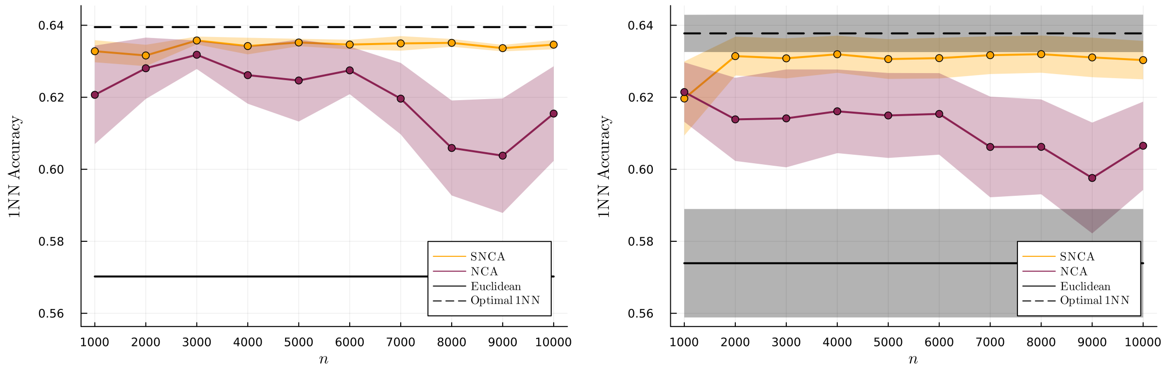


Figure 4.4.5: *Left*: the leave-one-out 1NN correct classification rate on a dataset of size 100,000, under metrics learnt from a training set of size n. *Right*: the 1NN classification performance on test states which were withheld from the metric training following 6-fold cross-validation.

around the average classification rate. We see much greater variability in the NCA solutions as compared with SNCA.

The kNN performance of SNCA does not noticeably benefit by increasing the training set beyond a size of $n = 3000$. We can be confident that this is enough training data and the metric solution is not overfitted. In practice, we advise increasing the training set size for SNCA until the classification accuracy on test data, and the interpretations of the learned metric, appear consistent.

Execution Time

To demonstrate the computational benefit of SNCA with discrete stochastic data, Figure 4.4.6 shows the mean execution time on a 1.6 GHz Intel Core i5 processor, for the metric training of SNCA and NCA across the 10 training sets of size n used in Figure 4.4.5 (left), ± 1.96 standard errors. The reported times include 10 initialisations for each optimisation. With the small state space in this example, increasing the value of n does not increase the burden on SNCA. In contrast, a pointwise metric learning method such as NCA, which involves summation over n individual points, understandably suffers as n increases. It should be noted that when training NCA on the dataset of size $n = 100,000$ to illustrate the metric embedding in Section 4.4.1, the pointwise formulation was not feasible. Instead, we employed a modified algorithm in which identical points were grouped together in the calculations of the objective

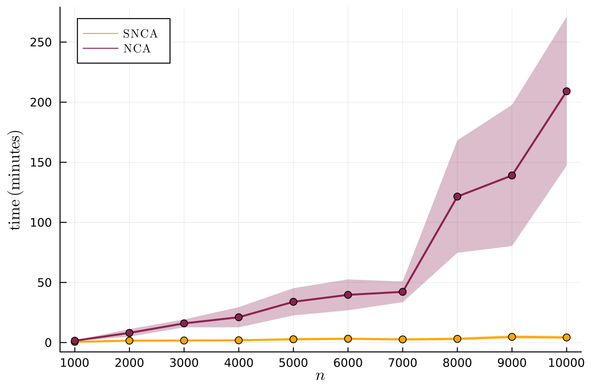


Figure 4.4.6: Metric learning execution time on a training set of size n.

function and the gradient.

4.4.2 A Wafer Fab Model

As a more realistic example for discrete stochastic data, we explore the simulation model of a wafer fabrication facility (fab) described by Kayton et al. (1996). Wafers are released into this system at fixed intervals and follow a sequence involving several processing steps at a number of stations. Aspects of re-entrant flow, and machines offering different processing capacities and unpredictable breakdown patterns complicate the management of product flow through wafer fabs, and a simulation model becomes an indispensable tool. For this example, we have used Simul8 software to imitate the model described by (Kayton et al., 1996).

To create a classification problem, we consider the practical task of predicting an early or late completion based on the system state on a wafer's release. To restrict the feasible state space and encourage repeated observations, we take an incomplete view of the system state, with $\boldsymbol{X}$ containing simply the queue sizes at each of the facility's 11 stations. The size of $\mathcal{X}$ is further limited by a total capacity constraint on the system, whereby new wafers cannot be released if the total work-in-process exceeds a threshold. The classification, Y, is observed as the wafer leaves the facility either before or after its due date, where the due date is pre-assigned by a constant inflation factor on its pure processing time. In this problem, we use a classification of $Y = 0$ to indicate an early or on-time wafer completion, while $Y = 1$ implies a late completion. Running the simulation provides an observation, $(\boldsymbol{X}, Y)$, for every wafer which passes through the system. A classification model from this simulated data can be applied on the actual wafer fab to alert planners to possible late deliveries.

The system design yields around $21{,}000$ feasible combinations of the 11 queue sizes, although the distribution over this state space is highly imbalanced. Figure 4.4.7 (left) shows a dataset projected in the first two dimensions of an SNCA solution, where size is proportional to the observed frequency of the states. There is a small number of commonly visited states, while many more are observed very few times, and overall only a small percentage of the feasible space has been represented in the sample. To aid understanding of this plot, consider the point lying on the origin, labelled as state 1. In the original context, this represents the system state in which all queues are empty. This is a commonly observed state for the system. It is also a difficult state to classify; its colour indicates the $Y = 1$ class to have an observed proportion of around 0.55. In other words, a wafer released while the system has empty queues is marginally more likely to be completed late than on time. Although counter-intuitive, this is because the first station is a batch station which requires at least two wafers to be processed simultaneously. Therefore, a wafer released when the first queue is empty must invariably wait at least until the following wafer is released. Other states shown by the colour scale to have lower proportions of late completions represent states in which the first queue is non-empty.

The labelled state 2 in Figure 4.4.7 (left) represents the state in which six wafers occupy the queue at station 3 and all other queues are empty. In the system logic, the machine at station 3 is prone to breakdowns, which leads to the occasional build up of this station queue. This represents a rare yet critical system condition and accounts for the branch of lesser-observed states extending away from the main cluster. The class distributions of these states are heavily skewed in favour of $Y = 1$, as wafers entering a blocked system are highly likely to finish late. SNCA rightly identifies this third queue size as an effective indicator of the system performance and lets this variable dominate the first projection dimension.

Of primary interest is to understand metric performance with regards to states which are underrepresented in the training sample, since these are the states for which we are more reliant on neighbours to make a prediction. Therefore, as a more targeted approach, we show the kNN accuracy on the rarely seen states. In particular, we consider states observed in the dataset with fewer than 25 repetitions. Figure 4.4.7 (right) shows the average kNN accuracy on these states ± 1.96 standard errors, after repeating the experiment with 10 simulated

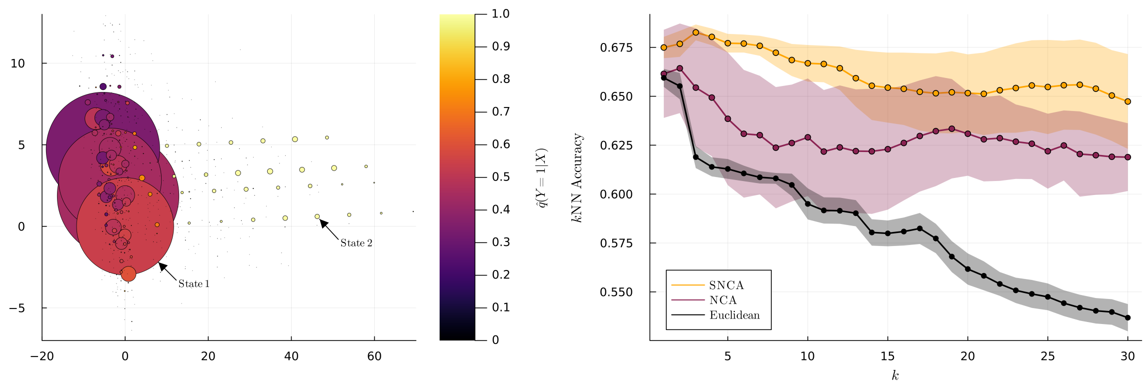


Figure 4.4.7: *Left*: using a dataset of size $n = 18{,}402$ from the wafer fab model, the states are projected by the first two rows of an SNCA solution. The colour of the points relates to their observed class proportions, and the size is proportional to their observed frequency. *Right*: the kNN correct classification rate on rarely seen states.

datasets. Additional details on the generation of this plot are given in Appendix A.3.

To provide context to the kNN accuracy results, the system was simulated to give roughly equal proportions of early and late wafers. Employing kNN on the simplified state description used in this example provides additional predictive power, and the benefit of a trained distance metric is evident from the comparison with Euclidean distance. Reaching above 65% accuracy following metric learning represents a marked and useful achievement considering the stochasticity of the system. This example is broadly representative of the type of data we aim to cater to with SNCA, and the further performance improvement brought over NCA is encouraging. Compared with NCA across all examples we have looked at offering discrete stochastic data, SNCA was found to give equivalent or improved kNN performance.

4.5 Conclusion

In this work, we proposed a method of DML which we refer to as Stochastic Neighbourhood Components Analysis (SNCA). The aim of SNCA is to extend the reach of metric learning to scenarios involving repeated feature vectors with stochastic labelling. This is a type of data which is characteristic of classification problems arising from the sample paths of stochastic simulation. Interest in probing the dynamic behaviour of simulation is growing, and to this purpose we demonstrated SNCA for nearest neighbour classification on sample path data. This is a useful application for SNCA; classification models on sample path data can translate to real-time planning and control in a live system, whilst an SNCA solution matrix can reveal the influential drivers of a system's stochastic behaviour. Experimental results show SNCA to be effective, and to bring further improvement to kNN performance over NCA when applied to this type of data.

We recognise numerous contexts to which the model of discrete stochastic data extends itself. For example, crowdsourcing represents an increasingly attractive solution to data generation, and will often result in an instance being assigned to various classes. Although we focus our motivation on simulation analytics, the scope of SNCA stretches further and we see possibility for a wider impact.

Chapter 5

Simulation Shapelets: Comparing Characteristics of Time-Dynamic Trajectories

Shapelets are short, interpretable patterns in temporal data which can be characteristic of a class. In this chapter, we identify shapelets from the trajectories of discrete-event simulation to indicate the characteristic dynamic behaviours of competing system alternatives. This deviates from traditional simulation output analysis, in which estimations of time-averaged performance measures overlook the more fine-grained time-dynamic features that shape the evolution of a system. We propose a shapelet methodology tailored towards simulation trajectories, and provide mathematical observations to support its implementation. To illustrate the potential of this methodology, we demonstrate its application to three examples. In particular, we reveal disruption recovery behaviour in a manufacturing simulation, provide a means for dynamic model validation, and expose the typical joint behaviour of a multivariate system state. By offering a visual characterisation of trajectories, we find that simulation shapelets can promote a deeper understanding of the dynamic behaviour and performance of simulated models.

5.1 Introduction

A stochastic simulation is a representation of a time-dynamic process. In spite of this, long-run average performance remains the prevailing mode of output analysis (Nelson and Pei, 2021). Chasing precise estimations of long-run behaviour leads to dynamic simulation outputs being averaged over both time and replications. Although an important view of simulation performance, this neglects insights into the dynamic behaviours that characterise a working system.

In this chapter, we present a methodology for investigating the dynamic behaviour of stochastic simulation. We borrow the concept of shapelets (Ye and Keogh, 2011) from the literature on time series classification. Shapelets are local segments of a series that represent the characteristic patterns of behaviour by which we can discriminate among classes. Applied to trajectories of simulation variables, we find that shapelets can identify the typical dynamic features which characterise and distinguish competing system alternatives.

We use the term "*trajectory*" to refer to a measurable quantity in a simulation model recorded as a continuous-time function. For example, the number-in-system in a discrete-event queueing model constitutes an integer variable that changes value only at the time of an arrival or departure event, and thus follows a piecewise constant trajectory. We specifically consider piecewise constant trajectories generated by discrete-event simulation over a finite time horizon. This affords us an enviable position with regards to data generation. We possess a ready source of clean, measurement-error-free data, with control over the form and quantity of output trajectories. Consequently, our controlled simulation environment provides advantages to a shapelet analysis which most time series applications do not. While our primary interest is in the benefits of shapelets to simulation, we are thus encouraged by reciprocal benefits of simulation to shapelets.

Our contributions in this chapter are threefold. Firstly, we adapt a shapelet methodology from the setting of discretely sampled time series to the setting of piecewise constant simulation trajectories. In doing so, we offer a general-purpose methodology for output analysis. However, the computational demands of a shapelet search can be significant. Therefore, our second contribution exploits the piecewise constant structure to provide mathematical results which ease this computational aspect and simplify an implementation. Thirdly, we provide proof-of-concept illustrations of some possible applications for simulation shapelets. Specifically, we demonstrate the use of shapelets in identifying the differing dynamic response to disruption experienced by a manufacturing simulation under two design alternatives. We also show their potential for assisting operational model validation, as well as for exploring a multivariate state process. These examples demonstrate the value in moving an analysis beyond average performance, and the various benefits that shapelets can bring to an understanding of simulation behaviour.

The chapter is structured as follows. Section 5.2 establishes our motivation and background, including an overview of time series shapelets. Section 5.3 describes the proposed extension to simulation shapelets, and establishes mathematical results to facilitate its implementation. We illustrate some potential applications in Section 5.4, showing the versatility of the approach in providing insight for various purposes. The chapter concludes with a summary in Section 5.5.

5.2 Background

This section provides context for our work. Section 5.2.1 contains a brief discussion of simulation trajectories, while Section 5.2.2 introduces the concept of time series shapelets, and reviews existing literature.

5.2.1 Simulation Trajectories

Discrete-event simulation describes a modelling paradigm in which the operation of a real system is represented by a random sequence of events occurring at discrete points in time. The state of the system can be described by a collection of variables which undergo instantaneous changes at event times, and otherwise remain constant. Consequently, the trajectories of simulation state variables take the form of piecewise constant functions of simulation time. We can also consider output variables such as waiting times or number of completions. Figure 5.2.1 provides some examples of trajectories from the simulation of an M/M/1 queue. From left to right, we show the number-in-system, the throughput, and the latest sojourn time as functions of simulation time.

Over the years, time series methods have been routinely applied to dynamic simulation output. However, the overarching purpose for such work has been for the study of steady-state behaviour. Fishman and Kiviat (1967) introduced a spectral analysis toolbox for simulation generated time series, with several other authors developing similar analyses to serve specific models (Heidelberger and Welch, 1983; Lada et al., 2007). Autoregressive models have been applied to dynamic simulation outputs to assist steady-state mean and variance estimation (Schriber and Andrews, 1984; Yuan and Nelson, 1993). In fitting such models, we can imagine interest in the interpretations of the autoregressive coefficients and residual variance. However, these models rely on a weak stationarity assumption, and an autocovariance depending only on the time lag. We consider these restrictions limiting for the admission of general non-stationary trajectories.

Recently, Nelson (2016) highlighted the need and opportunity for output analysis to move towards time-dependent characterisations. This prompted the emerging field of *simulation*

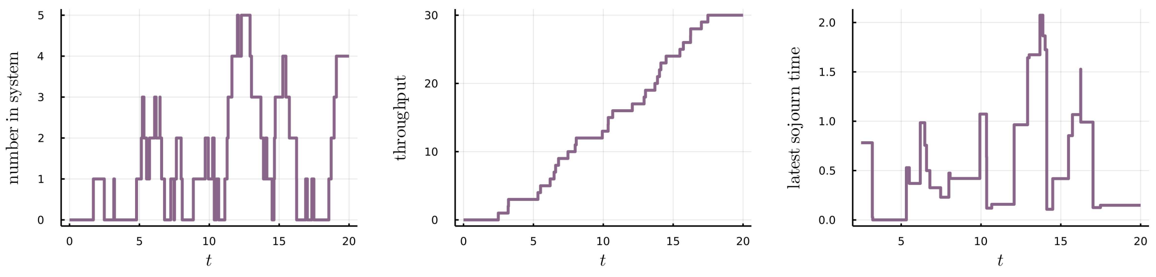


Figure 5.2.1: Examples of piecewise constant, continuous-time trajectories from a simulation of an M/M/1 queue.

analytics, which has since produced dynamic metamodelling for performance predictions conditional on time or state information (Laidler et al., 2020; Lin et al., 2019). In a direct effort to characterise dynamic simulation behaviour, Fourier analysis has been successfully applied to the time series of queue length and number-in-system trajectories, with coefficient magnitudes providing interpretation with regards to system congestion (Morgan and Barton, 2022). This approach provides a statistical characterisation of dynamic behaviour. Our own approach provides a *visual* characterisation, while retaining the continuous-time form of simulation trajectories. We base our approach on a methodology appearing in time series classification, which we discuss subsequently.

5.2.2 Time Series Shapelets

Time series shapelets were introduced by Ye and Keogh (2011) as a fast (once trained) and interpretable means of time series classification. A shapelet refers to a short subsequence whose appearance or absence in a time series can be informative of its class. This exploits the idea that the characteristic behaviour of a class of series tends to reveal itself over short patterns and local variations rather than on the global structure. Furthermore, the natural visualisation of shapelets can offer valuable interpretation. Shapelets have been shown to provide useful insights and classification accuracy across a range of applications, including motion capture (Shajina and Sivakumar, 2012), health monitoring data (Zorko et al., 2020), and detection of wind, wave, and seismic events (Arul and Kareem, 2021).

To formalise the key concepts used in time series shapelets, we introduce a time series $Z = z_{1:m}$ as a sequence of m real-valued variables, and similarly a shapelet $S = s_{1:\ell}$ of length $\ell \leq m$. The methodology relies on a notion of distance between a shapelet and a series. Initially, we define a symmetric, real-valued distance function, d, between $S = s_{1:\ell}$ and $S' = s'_{1:\ell}$, two length ℓ sequences. The squared Euclidean distance,

$$d(S, S') = \sum_{i=1}^{\ell} (s_i - s'_i)^2,$$

represents a common choice, although extensions such as dynamic time warping (Shah et al., 2016) have since been used. The distance between the shapelet, S, and the series, Z, is then defined by the closest appearance of S along the length of Z, as measured by d:

$$\operatorname{dist}(S, Z) = \min_{t \in \{1,2,\ldots,m-\ell+1\}} d(S, z_{t:t+\ell-1}).$$

With this distance function, we can assess the quality of a shapelet in providing class discrimination. We are given a dataset, D, containing the classified time series, $Z_1, Z_2, \ldots, Z_n$. Over this dataset, a shapelet, S, generates the set of distances, $\{\operatorname{dist}(S, Z_1), \operatorname{dist}(S, Z_2), \ldots, \operatorname{dist}(S, Z_n)\}$. Therefore, together with a distance threshold, γ, S divides D into two subsets: $D^{\text{near}} = \{Z_i \in D \colon \operatorname{dist}(S, Z_i) \leq \gamma\}$ and $D^{\text{far}} = \{Z_i \in D \colon \operatorname{dist}(S, Z_i) > \gamma\}$. The ability of S to discriminate classes is based on how well, for some γ, the classifications in D can be separated between D^{near} and D^{far}. This is measured by information gain, which begins with a definition of the entropy of a set of classified objects. For the time series dataset D containing n_c series of class $c \in C$ and $n = \sum_{c \in C} n_c$ total series, the entropy of D is defined as

$$H(D) = -\sum_{c \in C} \frac{n_c}{n} \log\left(\frac{n_c}{n}\right).$$

After splitting D with the shapelet, S, and distance threshold, γ, this entropy becomes

$$\hat{H}(D, S, \gamma) = \frac{|D^{\text{near}}|}{n} H(D^{\text{near}}) + \frac{|D^{\text{far}}|}{n} H(D^{\text{far}}),$$

and the information gain of this splitting strategy can be written as

$$I(D, S, \gamma) = H(D) - \hat{H}(D, S, \gamma).$$

An optimal distance threshold for S on D can be defined as a value, $\gamma^{\star}_{S,D}$, such that $I(D, S, \gamma^{\star}_{S,D}) \geq I(D, S, \gamma)$ for all other $\gamma \in \mathbb{R}$. The goal of a shapelet search on D is to find a shapelet, S, that maximises $I(D, S, \gamma^{\star}_{S,D})$. While in theory there are no restrictions on the values of S, typical practice is to extract a finite pool of candidates by considering all subsequences of permissible length occurring in D. Assuming Z_i has length m_i, the permissible shapelet lengths might be considered as $\mathcal{I} = \{2, 3, \ldots, \min m_i\}$, although typically domain knowledge can be used to reduce the size of $\mathcal{I}$ and expedite the search without sacrificing meaningful shapelets.

Alternative quality measures to the information gain, such as the F-statistic and Kruskal-Wallis and Mood's Median tests (Hills et al., 2014), have also been explored, with claims of faster computation and similar discrimination. However, the information theory framework naturally lends itself to the original decision tree classifier implemented by Ye and Keogh (2011). Framing the shapelet classifier as a decision tree retains the advantages of interpretability, while readily extending the binary architecture of shapelet splitting to multi-class problems. Beyond decision trees, however, Hills et al. (2014) proposed a shapelet transformation by taking the distances from a series to each of a collection of shapelets as classification features to be used in conjunction with a range of standard classifiers.

The computational demands of an exhaustive shapelet search are unfortunately prohibitive for many problems, and much of the subsequent literature has focused on speed-up techniques. Logical computation-saving steps such as an early abandon of unfruitful distance and entropy calculations were introduced (Ye and Keogh, 2011), and although unable to improve on the worst-case complexity, bring significant speed-up in practice. Mueen et al. (2011) introduced a further admissible pruning technique on the candidate pool based on an application of the triangle inequality with previously cached distance computations. However, the most significant speed-ups have been achieved by pruning via heuristic approaches (for example, see He et al. (2012)), claiming the sacrifice of exactness for speed to have little impact on classification accuracy. Alternatively, Karlsson et al. (2016) consider using randomised subsets of the training set and candidate pool in building a random forest ensemble, while Rakthanmanon and Keogh (2013) reduce computation by using piecewise aggregate approximations of the time series.

Whilst the computational challenges surrounding shapelet methodology continue to be addressed by other researchers, our interest lies in the application and benefits of shapelets for an analysis of simulation trajectories.

5.3 Simulation Shapelets

The interpretability offered by shapelets is a valuable feature among methods for time series classification. Common alternatives such as nearest neighbour methods provide no insight into why an object is assigned to a particular class, whereas shapelets can reveal the characteristic patterns by which classes are distinguished. It is this aspect of interpretability that attracts us as we look to characterise and understand dynamic simulation behaviour, rather than a need for classification. Our ambition is for shapelets to identify the characteristic dynamics which vary across competing system designs.

Our scope is to consider simulation over a finite time horizon. Independent replications of the same system provide us with multiple trajectories of constant length, representing, in the context of shapelet methodology, a collection of series of one particular class. The nature of simulation can lead to replications of the same system with vastly different overall trajectories. However, we expect the underlying dynamics to consistently produce its short-term characteristic patterns, encouraging the idea that shapelets can be well-suited to an analysis of simulation trajectories. This section describes the proposed methodology and its implementation.

5.3.1 Continuous-Time Series

The classical shapelet method accommodates time series sampled at discrete, regularly spaced time points, whereas simulation generates discrete-valued series measured in continuous time. This is the main point of difference to which we adapt.

In comparable work, continuous-time simulation trajectories have been converted into regularly sampled series to proceed with standard time series methods (Morgan and Barton, 2022). This approach introduces the decision of sampling frequency and inevitably incurs a loss of information. In a shapelet search, we also realise that results can be sensitive to this decision. For example, Figure 5.3.2 (bottom) shows the distance of a shapelet to two continuous-time trajectories from an M/M/1 queue over a range of sampling frequencies. The ordering of the series with regards to their distance to the shapelet is seen to depend heavily on the sampling frequency. Since shapelet selection relies on such distance orderings, we cannot guarantee robust results with this approach. Instead, we choose to preserve simulation trajectories in their original continuous-time format, and adapt a distance function to this context.

We denote a continuous-time simulation trajectory by $y\colon [0,T] \to \mathbb{R}$. As a piecewise constant function, we can write $t_0 = 0$, $t_m = T$, and use t_i for $i = 1, 2, \ldots, m-1$ to

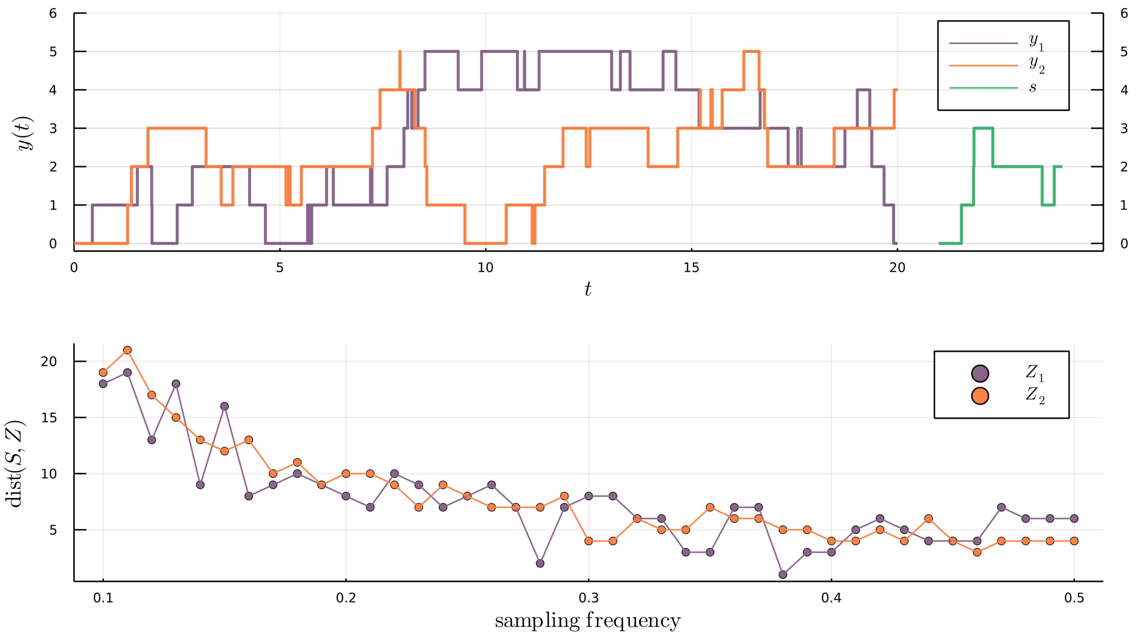


Figure 5.3.2: *Top*: from a capacitated M/M/1 queueing simulation, we show two continuous-time queue length trajectories, y_1 and y_2, and an example shapelet, s. We convert these into regularly sampled series, Z_1, Z_2, and S, respectively, and calculate $\mathrm{dist}(S, Z)$ as described in Section 5.2.2. *Bottom*: the ordering of the distance of the shapelet to each series is heavily dependent on the sampling frequency.

represent the time of the i^{th} change of y. We have $y(t) = y(t_i)$ for $t_i \leq t < t_{i+1}$. We represent the similarly piecewise constant shapelet of length $\ell \leq T$ by $s\colon [0, \ell] \to \mathbb{R}$.

To calculate a distance between two equal-length segments, we use the L_1 distance. That is, for $s, s'\colon [0, \ell] \to \mathbb{R}$, we write

$$\|s - s'\|_1 = \int_0^\ell |s(u) - s'(u)| du.$$

For piecewise constant functions, this is easily calculable as a sum of rectangular areas. Figure 5.3.3 illustrates the alteration made to the distance function as compared with the standard time series setting. Now, we define the distance between s and y as

$$\operatorname{dist}(s, y) = \min_{t \in [0, T-\ell]} \|s - y([t, t + \ell])\|_1, \tag{5.3.1}$$

where we allow shifts of the time axis to align segment domains. In particular, for

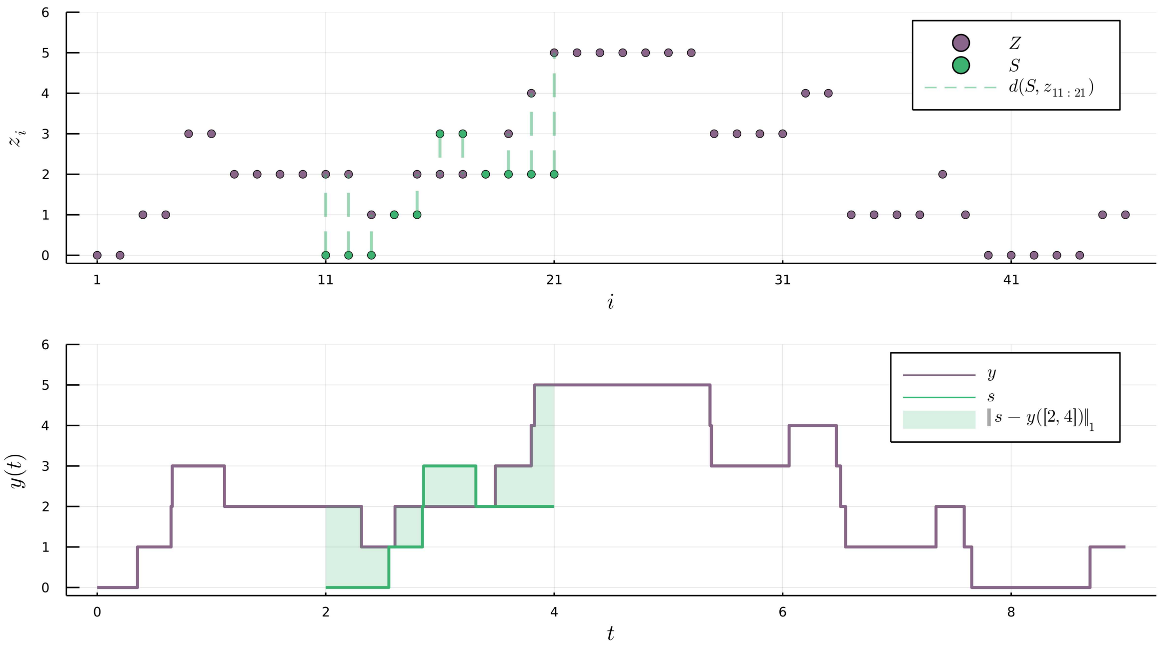


Figure 5.3.3: Illustrating the proposed alteration to the distance function in moving from the discrete-time (*top*) to the continuous-time (*bottom*) context.

$t \in [0, T-\ell]$, we define

$$\|s - y([t, t+\ell])\|_1 = \int_t^{t+\ell} |s(u-t) - y(u)| du.$$

Using the shapelet-series distance function (5.3.1), the remaining methodology surrounding the evaluation of shapelet candidates via splitting thresholds and information gain can proceed in the same way as described in Section 5.2.2.

In selecting candidates for the shapelet search, we recognise that the continuous-time format allows an infinite collection of shapelet candidates to be extracted from a dataset. To restrict this, we might extract candidates beginning every τ time units along our training trajectories, with the choice of τ being a consideration of computational budget as well as the granularity of the trajectories. Alternatively, to remove the need for a decision, we might extract candidates from a trajectory beginning at times at which the trajectory changes value. However, this approach may in general lead to needlessly large candidate sets. Therefore, we prefer the former approach of selecting τ, which allows us easy control over the size of the candidate set.

We also prescribe a finite selection, $\mathcal{I}$, of shapelet lengths, relying on intuition for the potential shapelet interpretations in the problem context. Our choices for τ and $\mathcal{I}$ directly control the size of the search, establishing a compromise between speed and quality. We note also that our adaptations to the continuous-time context remain compatible with existing approaches to candidate pruning (for example, see Mueen et al. (2011)).

Having outlined our methodology for continuous-time shapelets, we turn attention to its practical implementation. Calculation of the distance between a shapelet and a series (5.3.1) forms the workhorse of the method, so we continue with an observation towards its efficient computation.

5.3.2 Efficient Distance Computation

In calculating $\operatorname{dist}(s, y)$, we look for the segment of y with minimum L_1 distance to s. Intuitively, we are sliding the shapelet along the series to find its best matching location, with $\|s - y([t, t+\ell])\|_1$ providing a continuous function of $t \in [0, T-\ell]$ which we seek to minimise. However, rather than requiring numerical optimisation, we notice a unique structure to this function arising from the fact that s and y represent piecewise constant trajectories. This decomposes $\|s - y([t, t+\ell])\|_1$ into piecewise linear segments in t, such that its minimum must exist at a finite collection of points. Specifically, we make use of the following result:

Theorem 5.3.1. *Let $y\colon [0, T] \to \mathbb{R}$ be a piecewise constant function with change times in the set $\mathcal{T} = \{0 = t_0, t_1, \dots, t_m = T\}$ and $s\colon [0, \ell] \to \mathbb{R}$ with $\ell \leq T$ a piecewise constant function with change times in the set $\mathcal{U} = \{0 = u_0, u_1, \dots, u_{m'} = \ell\}$. Let $U_j = \{t_i - u_j : i = 0, 1, \dots, m\} \cap [0, T-\ell]$ for $j = 0, 1, \dots, m'$, and let $\mathcal{V} = \bigcup_{j=0}^{m'} U_j$.*

Then $\|s - y([t, t+\ell])\|_1$ is linear in t for $t \in [v, v']$, where $v, v' \in \mathcal{V}$ and $(v, v') \cap \mathcal{V} = \emptyset$.

Proof. See Appendix B.1. □

Theorem 5.3.1 states that $\|s - y([t, t+\ell])\|_1$ is a piecewise linear function of t which can only change its slope at the times contained in the finite set, $\mathcal{V}$. The consequence of this for the calculation of $\operatorname{dist}(s, y)$ is that we only need to evaluate $\|s - y([t, t+\ell])\|_1$ at points $t \in \mathcal{V}$ to find its minimum. Furthermore, calculating these distances as the sum of individual area components, as implied in Appendix B.1, allows for early abandon pruning as soon as the current minimum is exceeded. These observations allow substantial speed-up of the computation of $\operatorname{dist}(s, y)$.

5.3.3 Location Invariance

We notice that $\|s - s'\|_1$ is heavily dependent on the vertical displacements of the segments, s and s'. In particular, two segments with identical shape but separated by a vertical shift will yield a positive distance value, while a smaller distance may be obtained between two segments which show vastly different shapes yet have similar vertical locations. In other words, similarities in shape can become swamped by the location factor. Depending on the purpose for examining simulation trajectories, however, it may be the shape rather than the location which is of greater interest in characterising the short-term system dynamics. For example, a queue size may show similar fluctuations regardless of its underlying level. Traditional output analysis can summarise information about the level of a series, so our primary aim with a shapelet analysis is to reveal dynamic characteristics of the shape. For this reason, we look to amend the distance function (5.3.1) such that it only reflects differences in shape.

In standard time series shapelets, z-normalisation on the individual segments is often performed prior to the distance calculation to achieve scale and offset invariance (Cetin et al., 2015). However, in our context, scale contains relevant information of the dynamics. We are comparing trajectories which all measure the same simulation quantity, such that differences in scale represent relevant dynamical differences that we need to preserve. To remove only the location factor while retaining the original shape and scale, we optimise a vertical displacement in conjunction with the horizontally sliding window. In other words, we allow a shapelet free movement vertically as well as horizontally to find its best matching location on a series. We denote the transformed shapelet by $s^c\colon [0,\ell] \to \mathbb{R}$ with $c \in \mathbb{R}$ such that $s^c(t) = s(t) + c$, and define the location-invariant distance between the shapelet, s, and the series, y, to be

$$\widetilde{\operatorname{dist}}(s, y) = \min_{t \in [0, T-\ell], c \in \mathbb{R}} \|s^c - y([t, t+\ell])\|_1.$$

This modification ensures that the distance between a shapelet and a series depends only on resemblances to the shapelet's shape within the series. Conveniently, this contributes only a minor computational extension to the calculations implied in Section 5.3.2. Firstly, we retain the result of Theorem 5.3.1 which states that the starting position of the best matching location on the time axis can only belong to a finite set of possibilities. This arises from the following corollary:

Corollary 5.3.2. *Let* $y\colon [0,T] \to \mathbb{R}$ *and* $s\colon [0,\ell] \to \mathbb{R}$ *be as in Theorem 5.3.1. Let* $c \in \mathbb{R}$ *and define* $s^c\colon [0,\ell] \to \mathbb{R}$ *such that* $s^c(t) = s(t) + c$. *Let* $f(t) = \min_{c\in\mathbb{R}} \|s^c - y([t, t+\ell])\|_1$.

Then $\min_{t\in[0,T-\ell]} f(t)$ *is attained by an element of the finite set* $\mathcal{V}$ *described in Theorem 5.3.1.*

Proof. Assume $t^\star \notin \mathcal{V}$ minimises $f(t)$. Let $c^\star = \arg\min_{c\in\mathbb{R}} \|s^c - y([t^\star, t^\star + \ell])\|_1$. Then $\|s^{c^\star} - y([t^\star, t^\star + \ell])\|_1 \le \|s^c - y([t, t+\ell])\|_1$ for all other $t \in [0, T-\ell]$ and $c \in \mathbb{R}$. But by Theorem 5.3.1, $\min_{t\in[0,T-\ell]} \|s^{c^\star} - y([t, t+\ell])\|_1$ is attained by an element of $\mathcal{V}$. This either provides a contradiction, or we have that $\|s^{c^\star} - y([t, t+\ell])\|_1 = \|s^{c^\star} - y([t^\star, t^\star + \ell])\|_1$ for some $t \in \mathcal{V}$. □

Corollary 5.3.2 implies that to find $\widetilde{\mathrm{dist}}(s, y)$, we only need to calculate $\|s^c - y([t, t+\ell])\|_1$ for $t \in \mathcal{V}$. Therefore, for a given point $t \in \mathcal{V}$, we now address the task of finding $\min_{c\in\mathbb{R}} \|s^c - y([t, t+\ell])\|_1$. We observe that the piecewise constant behaviour of s and y allows $\|s^c - y([t, t+\ell])\|_1$ to be expressed as a sum of rectangular areas with known dimensions. This regular structure allows us to write down its minimiser via the following theorem.

Theorem 5.3.3. *Let* $y\colon [0,T] \to \mathbb{R}$, $s\colon [0,\ell] \to \mathbb{R}$, *and* $s^c\colon [0,\ell] \to \mathbb{R}$ *be as in Corollary 5.3.2. Let* $\mathcal{W}^t = \{\mathcal{T} \cap (t, t+\ell)\} \cup \{\mathcal{U} + t\}$, *with ordered elements denoted by* w_i^t *such that* $y(w_i^t) - s(w_i^t - t) \le y(w_{i+1}^t) - s(w_{i+1}^t - t)$ *for* $i = 1, 2, \ldots, m_t$. *Let*

$$\lambda_i = \begin{cases} \min\{w_j^t - w_i^t \colon w_j^t \in \mathcal{W}^t, w_j^t > w_i^t\} & \textit{if } w_i^t \in \mathcal{W}^t \backslash \{t+\ell\}, \\ 0 & \textit{if } w_i^t = t + \ell, \end{cases}$$

and let $k^\star = \min\left\{k \in \{1, \ldots, m_t\} \colon \sum_{i=1}^{k} \lambda_i \ge \frac{\ell}{2}\right\}$.

Then $\min_{c\in\mathbb{R}} \|s^c - y([t, t+\ell])\|_1$ *is attained by* $c^t = y(w_{k^\star}^t) - s(w_{k^\star}^t - t)$.

Proof. See Appendix B.2. □

In Theorem 5.3.3, $\{\mathcal{U} + t\}$ denotes the set $\{u_j + t \colon u_j \in \mathcal{U}\}$. Corollary 5.3.2 and Theorem 5.3.3 combine to provide an efficient computation procedure for $\widetilde{\operatorname{dist}}(s, y)$: we evaluate $\|s^c - y([t, t+\ell])\|_1$ at the points (t, c^t) for $t \in \mathcal{V}$ and c^t as given by Theorem 5.3.3, with early abandoning again providing further practical speed-up.

5.3.4 Multivariate Shapelets

We also consider extending the application of shapelets to multivariate simulation trajectories. This would accommodate multivariate output trajectories, as well as multivariate trajectories of the system state. Changes in state variables often represent the physical movements of entities in a system, and the multivariate state can thus be highly correlated. Multivariate shapelets extracted over each dimension simultaneously may provide an insight into typical system-wide behaviours.

Bostrom and Bagnall (2017) consider options for the formulation of multivariate time series shapelets, after extracting candidates over the same indices of each dimension. To find the best matching location of a multivariate shapelet on a multivariate series, two paradigms were considered: dependent shapelets, in which the shapelet components remain in parallel, and independent shapelets, in which the components are free to move in each dimension independently. The latter approach will not necessarily reveal the common joint behaviour, and so we choose to explore the former paradigm.

We calculate a multivariate shapelet-series distance value by summing the distance contributions from each dimension. Specifically, in our continuous-time context, we calculate the location-invariant distance between a d-dimensional, length ℓ shapelet $\boldsymbol{s} = (s_1, s_2, \ldots, s_d)$ and a d-dimensional, length T series $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)$ as

$$\widetilde{\operatorname{dist}}(\boldsymbol{s}, \boldsymbol{y}) = \min_{t \in [0, T-\ell], \boldsymbol{c} \in \mathbb{R}^d} \sum_{j=1}^{d} \|s_j^{c_j} - y_j([t, t+\ell])\|_1. \tag{5.3.2}$$

The vector, $\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$, contains the independent vertical displacements in each dimension. Interpreting this distance function, the best matching location of $\boldsymbol{s}$ on $\boldsymbol{y}$ is found by the shapelet components s_j moving dependently with one another across time, but independently of one another in their vertical shifts.

Calculation of $\widetilde{\operatorname{dist}}(\boldsymbol{s},\boldsymbol{y})$ follows straightforwardly from the univariate case, by application of the following corollary:

Corollary 5.3.4 (to Theorem 5.3.1). *Let $\boldsymbol{y}\colon [0,T]\to\mathbb{R}^d$ and $\boldsymbol{s}\colon [0,\ell]\to\mathbb{R}^d$ be d-dimensional piecewise constant functions, with $\ell\le T$. For $j=1,2,\dots,d$, we have the univariate piecewise constant functions $y_j\colon [0,T]\to\mathbb{R}$ and $s_j\colon [0,\ell]\to\mathbb{R}$. Let $\mathcal{V}_j$ be the set described in Theorem 5.3.1 for the j^{th} dimension. Let $f(t)=\min_{\boldsymbol{c}\in\mathbb{R}^d}\sum_{j=1}^d\|s_j^{c_j}-y_j([t,t+\ell])\|_1$.*

Then $\min_{t\in[0,T-\ell]} f(t)$ is attained by an element of the finite set $\mathcal{V}=\bigcup_{j=1}^d\mathcal{V}_j$.

Proof. For each $j=1,2,\dots,d$, Theorem 5.3.1 states that $\|s_j-y_j([t,t+\ell])\|_1$ is linear in t for t between successive elements of $\mathcal{V}_j$. Now consider successive elements $v,v'\in\mathcal{V}$ with $(v,v')\cap\mathcal{V}=\emptyset$. Since $\mathcal{V}_j\subseteq\mathcal{V}$ for all j, we must also have $(v,v')\cap\mathcal{V}_j=\emptyset$, implying that $\|s_j-y_j([t,t+\ell])\|_1$ is linear in t for $t\in[v,v']$ for all j. Therefore, $\sum_{j=1}^d\|s_j-y_j([t,t+\ell])\|_1$ must also be linear in t for $t\in[v,v']$. We conclude that $\sum_{j=1}^d\|s_j-y_j([t,t+\ell])\|_1$ is minimised by an element of $\mathcal{V}$.

Now assume $t^\star\notin\mathcal{V}$ minimises $f(t)$. Let $\boldsymbol{c}^\star=\arg\min_{\boldsymbol{c}\in\mathbb{R}^d}\sum_{j=1}^d\|s_j^{c_j}-y_j([t^\star,t^\star+\ell])\|_1$. Then $\sum_{j=1}^d\|s_j^{c_j^\star}-y_j([t^\star,t^\star+\ell])\|_1\le\sum_{j=1}^d\|s_j^{c_j}-y_j([t,t+\ell])\|_1$ for all other $t\in[0,T-\ell]$ and $\boldsymbol{c}=(c_1,c_2,\dots,c_d)\in\mathbb{R}^d$. But we know that $\sum_{j=1}^d\|s_j^{c_j^\star}-y_j([t,t+\ell])\|_1$ is minimised by an element of $\mathcal{V}$. This either provides a contradiction, or we have that $\sum_{j=1}^d\|s_j^{c_j^\star}-y_j([t,t+\ell])\|_1=\sum_{j=1}^d\|s_j^{c_j^\star}-y_j([t^\star,t^\star+\ell])\|_1$ for some $t\in\mathcal{V}$. $\square$

By Corollary 5.3.4 and Theorem 5.3.3, we see that the calculation of $\widetilde{\operatorname{dist}}(\boldsymbol{s},\boldsymbol{y})$ only requires us to evaluate $\sum_{j=1}^d\|s_j^{c_j}-y_j([t,t+\ell])\|_1$ at points $(t,\boldsymbol{c}^t)$ where $t\in\mathcal{V}$ and $\boldsymbol{c}^t=(c_1^t,c_2^t,\dots,c_d^t)$. Establishing a calculation of $\widetilde{\operatorname{dist}}(\boldsymbol{s},\boldsymbol{y})$ allows the shapelet search to proceed as in the univariate case, by calculating for each shapelet candidate the information gain achieved by the optimal splitting threshold.

In general, for variables measuring different quantities and with different scales, we might consider applying the summation (5.3.2) with unequal weightings. We note that this adjustment would not affect the result of Corollary 5.3.4. In the example presented in Section 5.4.3, however, we calculate the multivariate distance using equally weighted dimensions.

5.4 Applications of Simulation Shapelets

The proposed methodology offers insight into short-term simulation behaviour, which can be helpful and informative for various objectives. Some exemplar applications of simulation shapelets are illustrated in this section.

In each example, our aim is to reveal the characteristic dynamics of two alternative systems. For this purpose, we identify shapelets as follows. We use s_1 to denote a shapelet extracted from the trajectories of system 1 that maximises the information gain, with the added condition that the proportion of system 1 trajectories in the *near* subset, defined by s_1's optimal distance threshold, $\gamma^{\star}_{s_1,D}$, must exceed the proportion in the *far* subset. In particular, letting D denote the complete set of training trajectories, we write $D^{\text{near}} = \{y_i \in D\colon \widetilde{\text{dist}}(s_1, y_i) \leq \gamma^{\star}_{s_1,D}\}$ and $D^{\text{far}} = \{y_i \in D\colon \widetilde{\text{dist}}(s_1, y_i) > \gamma^{\star}_{s_1,D}\}$. Then, letting $D_1 \subset D$ denote the set of system 1 trajectories, we require that $|D_1 \cap D^{\text{near}}|/|D^{\text{near}}| > |D_1 \cap D^{\text{far}}|/|D^{\text{far}}|$. Without this extra condition, s_1 may instead identify characteristic behaviour of system 2 which on occasion appears in a trajectory of system 1. In the same way, we denote by s_2 an optimal shapelet which is characteristic of system 2. In the event that multiple characteristic shapelets are found that maximise the information gain, we choose the one that maximises the margin between the two systems, which is defined as the difference between the mean distance to system 1 trajectories and the mean distance to system 2 trajectories. This is a tie-breaking strategy suggested by Ye and Keogh (2011).

In searching for the optimal shapelets, we generate candidates using the strategy of appointing τ and $\mathcal{I}$ as described in Section 5.3.1. However, we realise that multiple shapelets of similar lengths extracted from similar places in a trajectory will often have large overlaps, and therefore show similar ability to discriminate classes. On this basis, we see an opportunity to avoid spending computation on shapelet candidates which are unlikely to be successful. By applying the triangle inequality, Mueen et al. (2011) introduce a mathematical strategy for candidate pruning which is based on this same idea. However, in the examples here, we take a heuristic approach to candidate selection. We extract candidates of an initial length, $\ell \in \mathcal{I}$, at spacings of τ time units along each trajectory in the training set, calculating for each candidate the optimal information gain, and using admissible entropy pruning as described by Ye and Keogh (2011). Whenever a new candidate, $s^{\star}$, improves the current best information gain, we conduct a *local* search by extracting additional candidates from the trajectory around the starting position of $s^{\star}$, over the full set of shapelet lengths in $\mathcal{I}$. This focuses a search to a selection of promising candidates. This approach has proved useful in our experiments, although we leave scope for further work in the problem of effective candidate selection. In the presented examples, ℓ was taken to be the median value from the reported sets of $\mathcal{I}$.

Each example demonstrates the value in looking beyond a simulation's long-run average performance to its dynamic behaviour. Section 5.4.1 shows the effectiveness of shapelets in targeting a comparison of a manufacturing system's dynamic response to disruption under two design alternatives. In this example, while the long-run average behaviour of the target performance indicator is similar across the two alternatives, shapelets reveal that the observed behaviour while operating is significantly different. Section 5.4.2 uses a simple tandem queueing simulation to demonstrate the potential for shapelets to assist operational model validation. Assuming the availability of trajectories from a real-world system, we look for shapelets to assess the capability of two simulation models in mimicking the real system's dynamic behaviour. Using the same simulation, Section 5.4.3 explores the possibility for multivariate shapelets to provide insights into the joint behaviour of system state trajectories. Again, we find a deeper comparison and understanding of system behaviour provided by the dynamic characteristics.

The simulation models behind these examples were implemented in Simul8. In running replications of competing systems, we use the established practice of common random numbers (Nelson and Pei, 2021), which ensures that replications of each system experience similar random input behaviour. This highlights an aspect of simulation control which can be helpful towards a shapelet analysis. By minimising the differences of input randomness affecting the trajectories of each system, we expect the characteristic differences that shapelets identify to more reliably reflect the underlying dynamics of system behaviours.

The computational complexity of the shapelet search depends on several factors, including the number of training trajectories, their length, T, the shapelet length, ℓ, and the parameters, τ and $|\mathcal{I}|$. In the multivariate case, the dimension, d, also becomes relevant. We report these values for each example, and provide an indication of the shapelet search times, performed using a 1.6 GHz Intel Core i5 processor.

The simulation data and code used for the analyses and presentation of figures in this section is made available through a GitHub repository (Laidler, 2023).

5.4.1 Dynamic Response to System Disruption

Systems in the supply chain and manufacturing sectors often experience disruption as a result of unexpected shortages or breakdowns of unreliable components. A system's resilience and recovery from such events is an important consideration, and reactive decisions and schedules are often sought to mitigate their impact (Mehta and Uzsoy, 1999). While traditional summary measures may capture long-term impacts, we expect the practical repercussions of a disruption to appear in a system's temporary dynamic behaviour. In this example, we look at the dynamic throughput sequence of a manufacturing system, and exploit shapelets to reveal the short-term impacts of machine failures.

Kayton et al. (1996) introduced a simplified model of a wafer fabrication facility (fab), in which semiconductor wafers undergo various processing steps at a number of stations.

Their simulation experiment explored the effects of machine breakdowns and dispatching rules on time-averaged performance measures such as the average work-in-process and the bottleneck utilisation. Dispatching rules refer to the rules for priority ordering by which queueing jobs are sequenced to work centers. Borrowing the wafer fab model of Kayton et al. (1996), our intention is for shapelets to compare the impact of machine failures on the system's dynamic throughput under different dispatching rules.

We consider the rules *Least Remaining Work* (LRW) and *Most Remaining Work* (MRW), which prioritise wafers at each station according to the number of their remaining processing steps. LRW is commonly used to accelerate throughput and system de-congestion, while MRW can target specific performance goals such as minimising the maximum lateness (Kaban et al., 2012). We simulated week-long replications featuring machine breakdowns, with the LRW and MRW dispatching rules yielding 95% confidence intervals for the weekly average throughput of $[84.1, 84.4]$ and $[82.1, 82.9]$, respectively. The similarity of these long-run statistics suggests we may find value in a more detailed dynamic comparison.

In the original work of Kayton et al. (1996), a machine which is visited once and early in the processing sequence of each wafer is suggested to have low reliability. To specifically uncover the breakdown and recovery behaviour, we consider imposing a fixed structure to the breakdowns of this machine. Specifically, we inject a breakdown lasting for a fixed duration of 16 hours at a set time in each replication. This promotes consistency among the trajectories; it ensures that each trajectory displays a breakdown and recovery period of similar proportions and at a similar time. In this way, we can isolate the breakdown effect and focus the shapelet search accordingly. This demonstrates an advantage that the simulation context provides. To uncover specific behaviour surrounding a known system event, our control over the simulation translates to control over the trajectories and allows us to manipulate a simulation and shapelet search to target the desired insights.

We take 100 training replications of the system under each dispatching rule. To generate the candidate pool for a shapelet search, the consistent structure surrounding breakdown times allows us to shrink the window for the start times of candidate shapelets. We extracted candidates every $\tau = 2$ hours from within a suitable range over which the breakdown effects begin to show in each replication. To allow the shapelets to display breakdown and recovery behaviour considering a breakdown length of 16 hours, we searched for shapelets with lengths of $\mathcal{I} = \{40, 45, 50\}$ (hours). Each shapelet search lasted approximately 30 minutes.

The optimal shapelets discovered are shown in Figure 5.4.4 (left). Since the shapelets are location-invariant (see Section 5.3.3), we in fact plot $s(t) - \min_u s(u)$. This ensures that s_1 and s_2 can easily display on the same axes. The shapelets suggest distinctly different dynamic behaviour resulting from machine breakdowns. The shapelet, s_1, which is characteristic of systems operating the LRW rule, shows relatively brief disruptions to the throughput sequence. In contrast, s_2 represents the system under the MRW rule and shows a lengthy period without any throughput. However, once s_2 resumes throughput, its rate is very quickly recovered. We realise that the MRW rule restrains throughput, with the volume of jobs nearing completion accumulating until the completion process can resume. This results in a longer period of throughput starvation following system disruption, whereas the LRW rule prioritises re-establishing the throughput at the expense of a more gradual recovery of its rate.

The scatter plots show the distance of each trajectory to each shapelet, and reveal strong

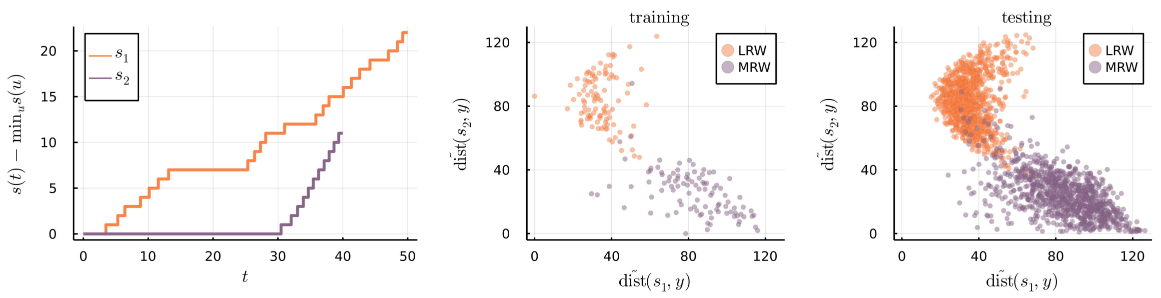


Figure 5.4.4: Comparing the impact of machine breakdowns on throughput behaviour for systems operating under the LRW and MRW dispatching rules. The characteristic shapelets are shown on the left. The scatter plots show the distribution of the 200 training trajectories (*middle*) and 2000 testing trajectories (*right*) with regards to their distances from the two shapelets.

separation. A set of 1000 unseen test replications of each system were used for the right hand scatter plot. These follow the same behaviour as the training replications, giving us confidence that the discovered shapelets are reliably characteristic of the two dispatching rules and not merely a result of overfitting to the training trajectories.

As stated, targeting specific behaviour allows us to impose structure to the simulation and focus our shapelet search accordingly. To test the power of shapelets without such structure, perhaps to uncover insights in the absence of prior knowledge or expectations, we repeat the experiment with random breakdown behaviour. We use distributions for the mean time to failure and mean time to repair as implied by Kayton et al. (1996), such that that replications contain breakdowns at random times and of random durations. The average breakdown duration is now 30 hours as opposed to a fixed 16 hours. Some replications contain no breakdown. However, we retain some consistency across the corresponding replications of each dispatching rule by simulating with common random numbers. As such, the differences across the dispatching rules remain in the impact and recovery of breakdowns rather than in the breakdowns themselves.

Figure 5.4.5 shows the results from the unstructured simulation. Without the computational benefits of a targeted shapelet search, we used $\tau = 10$ and $\mathcal{I} = \{70, 75, 80, 85, 90\}$ (hours), and each search lasted approximately 60 minutes. Despite the inconsistent breakdown behaviour across replications, the shapelets still manage to separate the systems by their dispatching rule. Breakdowns under the MRW rule are again seen to produce the most

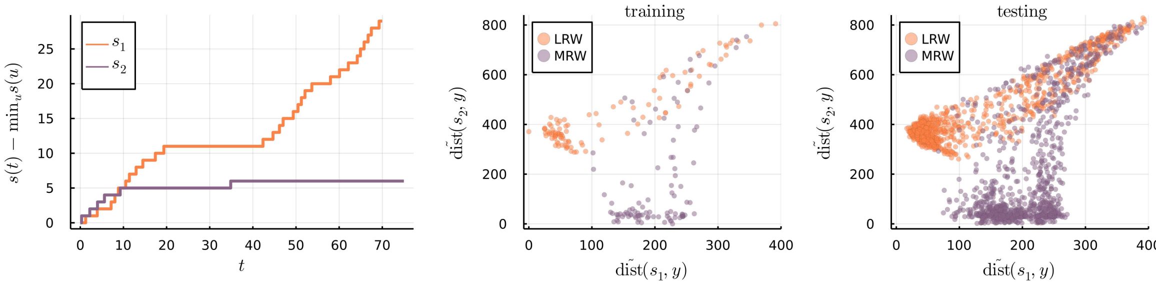


Figure 5.4.5: Repeating the results shown in Figure 5.4.4 using trajectories in which breakdowns occur randomly and with random recovery durations.

disruption to throughput. The scatter plots show a number of replications with large distances to each shapelet, mainly identifying the replications in which a complete breakdown and recovery period was not observed. Revealing the characteristic behaviour surrounding breakdown impact from replications without any targeted structure gives us confidence in the ability of shapelet methodology to discover hidden and unexpected insights into dynamic simulation behaviour.

5.4.2 Dynamic Model Validation

Operational validation is an important aspect of simulation modelling. For effective decisions to be taken in a real system based on the output of a simulation model, it is important for these outputs to bear sufficient resemblance to those of the real system. Operational model validation often amounts to a statistical or graphical comparison between the distributional properties of output variables, such as comparing histograms of waiting times, or conducting hypothesis tests around their means (Sargent et al., 2016). Thus, model validation is often restricted to the level of output summaries and grouped behaviours.

Shapelets may provide an additional aspect to model validation by discerning differences in the dynamic behaviour of simulation trajectories. In a related context, the emergence of digital twins has increased interest in bringing model validation into an online setting. Digital twins are models which aim to synchronise with a live system, thus providing support for real-time monitoring and control. As such, validating digital twins requires an online validation of the model behaviour in comparison to dynamic data received from the real system. Hua et al. (2022) provide an overview of opportunities and challenges in the validation of digital twins. Crucially, interest is growing in performing model validation via dynamic behaviour, and this is an area to which shapelets may be applicable.

To construct an example, we consider the three-station tandem queue depicted in Figure 5.4.6. Station I contains k_I identical servers, and we set $k_I = 1$ for $I \in \{A, B, C\}$. Each station admits infinite queueing space and generates exponential service times with

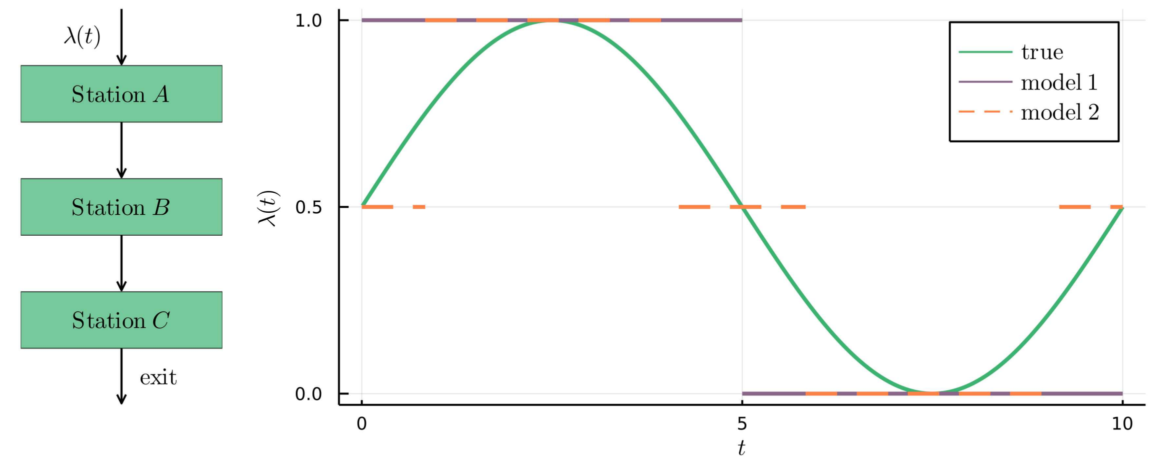


Figure 5.4.6: *Left*: a diagram of a tandem queue. *Right*: the arrival rates, $\lambda(t)$, of the true system and the two models. These repeat with period $t = 10$.

rate $\mu = 0.6$. We simulate customer arrivals following a Poisson process, and let the *true* system have the smooth arrival rate function, $\lambda(t) = (\sin(2\pi t/10) + 1)/2$. In a real system, $\lambda(t)$ would be unknown, and therefore might be modelled using a piecewise constant rate function estimated from data. Two options for modelling $\lambda(t)$ with piecewise constant segments are chosen, and shown in Figure 5.4.6. Model 1 uses only two constant levels, rounding $\lambda(t)$ to the nearest integer, while model 2 uses three levels, rounding to the nearest 0.5.

We consider model validation for the dynamic number-in-system behaviour, using replications of length $T = 60$ minutes after discarding a warm-up period of length 20 minutes. From 10,000 replications, the mean number-in-system for the true system is found to be 8.28. Model 1 gives a 95% confidence interval for the mean number-in-system to be $[8.38, 8.52]$, while for model 2 this confidence interval is $[8.24, 8.39]$. Based on this long-run behaviour, both models would be deemed operationally valid to within 0.25 customers. We look for a shapelet analysis to go beyond this and discern if there exist differences in the dynamic behaviour. Intuitively, we should struggle to find characteristic dynamics that distinguish a good model from the true system.

Figure 5.4.7 compares model 1 with the true system. We took 200 training replications of

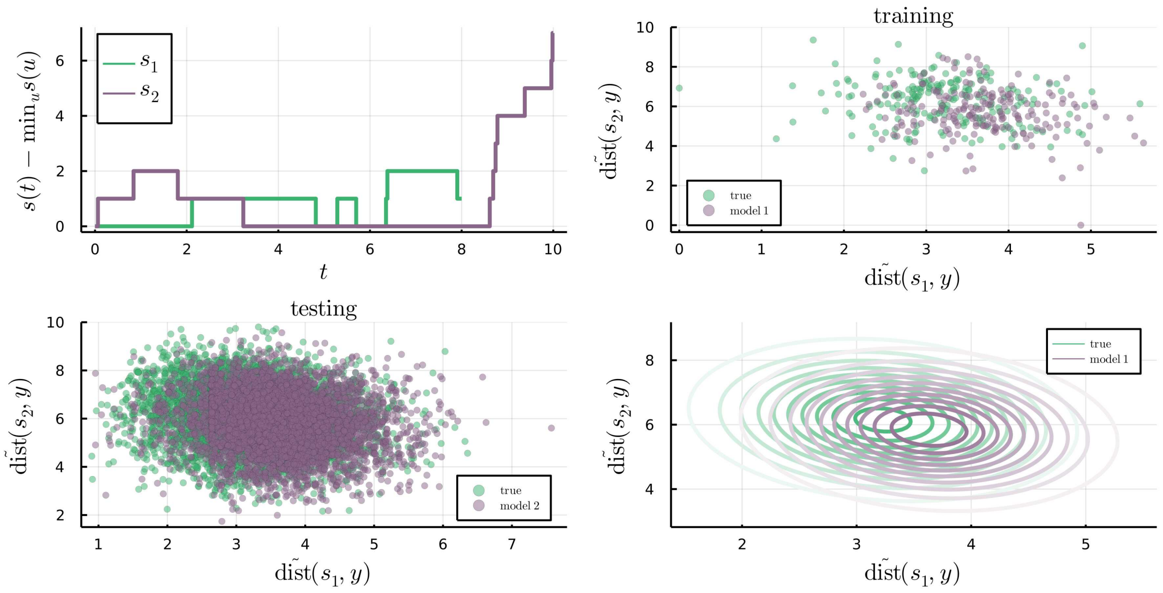


Figure 5.4.7: Comparing the dynamic number-in-system behaviour of the true system with model 1. The characteristic shapelets are shown in the top left. Gaussian distributions (*bottom right*) are fitted to the collection of testing trajectories (*bottom left*).

both the model and the true system, and searched for shapelets from the number-in-system trajectories using $\tau = 2$ and $\mathcal{I} = \mathbb{Z} \cap [8, 12]$ (minutes), each search lasting around 2.5 hours. The optimal shapelets are shown in the top left of Figure 5.4.7. The shapelet, s_1, is characteristic of the true system, and depicts a stable behaviour with arrivals and exits both occurring intermittently. On the other hand, s_2 is characteristic of model 1, and shows a long period of no arrivals followed by a sudden burst. The scatter plot in the bottom left of Figure 5.4.7 shows the distances of the trajectories from 5000 test replications of both the true system and the model to each of the shapelets. We see significant overlap, yet definite inclinations of the true system and the model to display behaviours closer to s_1 and s_2, respectively.

Fitting Gaussian distributions to the clouds of points in the test scatter plot results in the contours shown in the bottom right of Figure 5.4.7. To quantify the results of the shapelet analysis for model validation, we might consider the classification accuracy. An ideal model may generate indistinguishable behaviour from the true system and yield a classification accuracy of 0.5. Classifying the test trajectories based on their distances to s_1 and s_2 and the probability densities of the fitted distributions yielded a classification accuracy of 0.61.

Figure 5.4.8 shows the equivalent plots comparing model 2 with the true system. The discovered shapelets are less distinct from one another, while the scatter plot of test replications does not clearly show a disposition of either system towards either shapelet. The classification accuracy was found to be 0.51. The greater separation in the training scatter plot as compared to the testing scatter plot suggests that the discovered shapelets may result from overfitting to the training trajectories rather than being more broadly characteristic of the two systems. A comparison between the scatter plots of the training and testing replications is useful to indicate whether the identified shapelets are reliably characteristic of the system behaviour, or merely overfitted to the training replications. In practice, we might search for shapelets using an increasing number of training replications until the distributions of each class displayed in each scatter plot appear equivalent.

In summary, our shapelet analysis identified different characteristic behaviour of the number-in-system trajectories between the true system and model 1, but not between the true system and model 2. We conclude that model 2 is a more valid model of the true system.

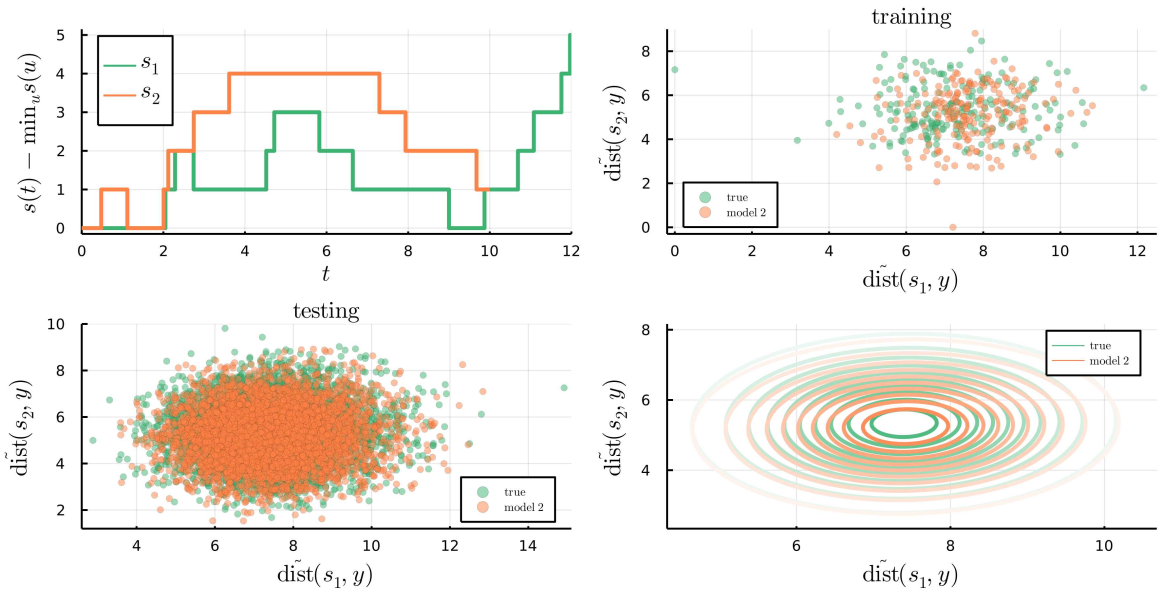


Figure 5.4.8: Shapelets are not able to discriminate between the dynamic number-in-system behaviours of the true system and model 2.

Referring to the model approximations shown in Figure 5.4.6, this is the conclusion we should expect. Shapelets appear able to present a comparison between systems' dynamic behaviours, and may represent a viable avenue for online model validation.

5.4.3 Multivariate Shapelets of the System State

To test the potential for multivariate shapelets in simulation, we remain with the tandem queueing model used in Section 5.4.2. We simulate Poisson arrivals with the true arrival rate function, $\lambda(t)$, seen in Figure 5.4.6, and exponential service times with rate $\mu = 0.6$ at each station. Suppose that we now have a total capacity for four servers, and consider two system alternatives:

1. system 1: $k_A = 1, k_B = 2, k_C = 1$,
2. system 2: $k_A = 1, k_B = 1, k_C = 2$.

We again use replications of length $T = 60$ minutes after discarding a warm-up period of length 20 minutes. Both systems yield an overall mean number-in-system of 2.29. The number-in-system trajectories themselves also show similar behaviour, and so we go further by breaking this down into the separate stations. We consider a three-dimensional system state $\boldsymbol{y}(t) = (y_A(t), y_B(t), y_C(t))$, where $y_I(t)$ represents the number of customers in station I at time t (queueing and in service).

Initially, we performed a univariate shapelet search in the three dimensions of $\boldsymbol{y}(t)$ individually. We used 100 training replications of each system, and searched for shapelets using $\tau = 2$ and $\mathcal{I} = \{10, 11, 12\}$ (minutes). Each search lasted around 30 minutes. The resulting shapelets are displayed in Figure 5.4.9, where s_{Ij} denotes the shapelet in the dimension y_I that is characteristic of system j. Common random numbers ensured identical trajectories of $y_A(t)$ across the replications of both systems, causing a shapelet search in this dimension to be futile. The differences emerge in the trajectories of $y_B(t)$ and $y_C(t)$, and as we might expect, system 1 is characterised by greater stability of y_B and volatility of

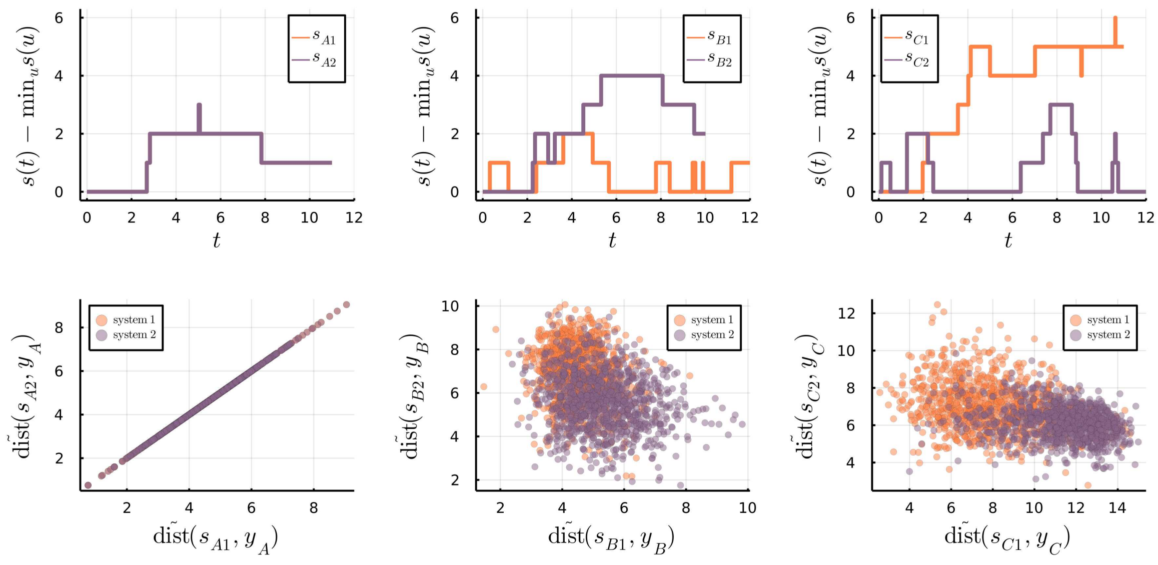


Figure 5.4.9: Independently finding univariate shapelets in the three dimensions of $\boldsymbol{y}(t)$.

y_C, while the reverse is true of system 2. The bottleneck at station C in system 1 appears more significant than the bottleneck at station B in system 2. The scatter plots show the distributions of 1000 test trajectories of each system with respect to their distances from s_{I1} and s_{I2}. The ability to discriminate the systems increases the further downstream we move.

We also performed a multivariate shapelet search, using the summation distance function (5.3.2). For this, we used $\tau = 10$, and each search lasted around 60 minutes. The results are shown in Figure 5.4.10, where $\boldsymbol{s}_1$ is characteristic of system 1 behaviour, and $\boldsymbol{s}_2$ of system 2. Retaining the multidimensional shapelet shows the dimensions moving concurrently. For example, we see that as y_A decreases, y_B increases, and as y_B decreases, y_C increases, reflecting the movement of customers through the system. We receive similar interpretations regarding the shifting of the bottleneck station between the two systems. Further, the multivariate shapelets support the conclusion of a more severe bottleneck at station C in system 1 than at station B in system 2. This may be understood from the differing server capacities at their preceding stations. At times when $\lambda(t)$ exceeds the service rate of $\mu = 0.6$, the departure rate from these preceding stations will be governed by their number of servers. As such, we expect a greater workload on station C in system 1 than on station B in system 2.

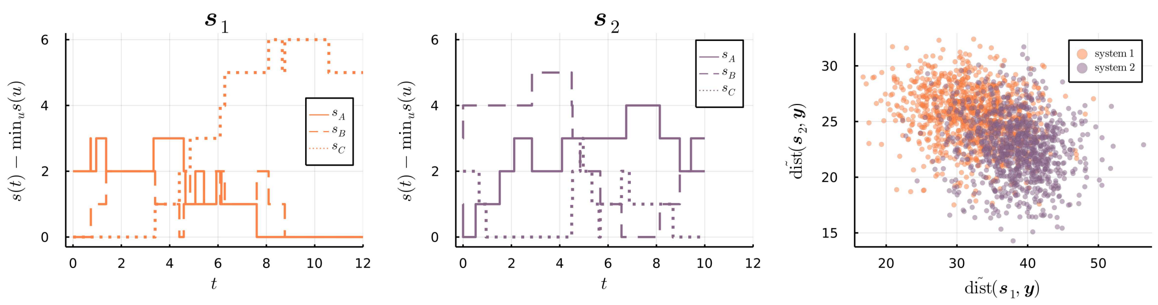


Figure 5.4.10: Finding multivariate shapelets reveals the typical joint behaviour of the station capacities for each system.

The scatter plot of test trajectories shows good separation of the two systems.

In the simple example presented here, the known server allocations represent the obvious difference between systems 1 and 2, and the multivariate shapelets identify the behaviours that match our expectations for the dynamic effect of this difference on the station populations. However, the differences among systems may not always be so clear, and the results in this simple case give us confidence that a multivariate shapelet search can extract meaningful and unanticipated behaviour when competing systems feature more nuanced differences. In summary, we see promise in the application of multivariate shapelets to a simulation state process, and potential for developing a more tailored methodology for this purpose.

5.5 Conclusion

We present a methodology for simulation output analysis which places focus on local characteristics of the underlying system dynamics. This methodology is based on the use of shapelets, which describe locally interpretable patterns capable of discriminating among time series classes. Applied to simulation trajectories, shapelets provide a means of characterising dynamic behaviour and performing deeper comparisons across competing system alternatives. Comparison across systems represents a common objective in the use of simulation, and when alternatives exhibit similar long-run performance, we conceive value in making deeper comparisons on the basis of dynamic behaviour.

To propose a shapelet methodology adapted to the structure of simulation trajectories, we present a reframing of the original discrete-time setting to the context of continuous-time functions. We also propose a natural approach to achieving location invariance, such that we solely represent the local shape dynamics appearing in the trajectories. These adaptations introduce theory through which the piecewise constant construction of simulation trajectories lends convenient structure to the shapelet-series distance function and relieves the computational aspect of its minimisation. Through this, we establish an efficient and practical methodology for piecewise constant shapelet discovery, and demonstrate its promise in application to simulation problems. For example, we see effective results in uncovering dynamic behaviour of interest, alongside potential applications to dynamic model validation and furthering an understanding of the dynamic behaviour of a multivariate system state.

Although the computational complexity of a shapelet search poses a challenge for many real-scale problems, the progress and focus of ongoing research in addressing this aspect gives practical encouragement to the proof-of-concept provided here. Additionally, the control over data generation afforded to us by simulation should not be ignored. Targeting specific behaviour and employing control procedures such as common random numbers allow us to focus a search and improve our training set to the task of meaningful shapelet extractions.

Simulation trajectories provide a rich environment for analysis, and a shapelet-based methodology shows potential to provide valuable insight into the often overlooked dynamics of system behaviour. An increasing demand for machine learning solutions and data-driven insights across system operations gives us confidence in the scope and impact of this approach.

Chapter 6

Conclusion

This thesis has developed methodology to encourage a deeper analysis and understanding of stochastic simulation models. Responding to the call for simulation analytics, we have looked to exploit the opportunities provided by the data-rich, time-dynamic record of the simulation sample path. The main implications of the work, and the particular methodological contributions of the preceding chapters are summarised in Section 6.1. Limitations of the presented work are acknowledged and potential developments suggested in Section 6.2, before we conclude the thesis with some closing remarks.

6.1 Summary of Contributions

Our first contribution was to propose an application of k-nearest neighbours (kNN) in Chapter 3. This provides a framework for making dynamic predictions conditional on a system state. With this, we can reveal critical system conditions, and provide support to system control by anticipating poor performance in a live system. Making use of historical simulation allows a fast, data-driven approach to real-time prediction, rather than spending further simulation on a model initialised from the current system state. Additionally, the use of metric learning in this context is able to provide interpretations as to the driving factors of system performance, as well as enhancing the kNN prediction accuracy. Understanding the relevant components of the state can be useful for purposes of system design and improvement.

In Chapter 4, we developed this methodology further by proposing a metric learning method tailored to the particular structure of sample path data. We refer to this method as Stochastic Neighbourhood Components Analysis (SNCA). Recognising that existing methods of metric learning are ill-suited to accommodating the repeated state vectors and stochastic classifications which are typical of simulation problems, SNCA adapts a probabilistic model of metric learning to this setting. Catering to a type of data which is not uncommon in real-world applications, we recognise the potential contribution of SNCA to extend beyond the sphere of simulation.

A general-purpose methodology for simulation trajectory analysis was contributed in Chapter 5. Borrowing the concept of shapelets from time series classification, we tailored a shapelet-based approach to the continuous-time, piecewise constant trajectories that arise as the dynamic output of discrete-event simulation. The discovery of simulation shapelets allows a visualisation of the patterns that characterise and distinguish the dynamic behaviour of competing system alternatives. To support the methodology, we provided mathematical results to suggest an efficient implementation procedure, and demonstrated its useful application in various simulation contexts.

We proceed in this chapter by considering the possible further development of these methodologies.

6.2 Prospective Use and Developments

Our primary objective throughout this thesis has been to use sample path data to gain insights into a system's dynamic behaviour. The feasibility and success of this objective is naturally reliant on an underlying assumption that the simulation model accurately imitates this dynamic behaviour. Therefore, our aims perhaps call for a new avenue of model validation. Traditionally focused on average outputs, we find it necessary that model validation also considers the dynamic behaviour of the simulation. This should become an important aspect of model development, and appeals to the potential application of simulation shapelets demonstrated in Section 5.4.2.

The remainder of this section offers a discussion on the applicability of the specific methodologies introduced, and their possible extensions. Section 6.2.1 discusses the use of the kNN model on simulation states, which was introduced in Chapter 3. While an adaptation of metric learning to the simulation context was presented in Chapter 4, we discuss potential further adaptations in Section 6.2.2. Chapter 5 presented methodology for simulation shapelets, and we discuss possible extensions in method and application in Section 6.2.3.

6.2.1 kNN on Simulation State

Chapter 3 introduced the idea of a kNN model on simulation states, with metric learning providing an interpretable distance function. An aim of this methodology is to allow real-time prediction of a system's dynamic performance, which can be a useful aid to system control. For this purpose, the emergence of digital twins provides convincing support. A digital twin aligns a simulation model with live data from a real-world system, and can therefore provide the current simulation state in the digital format required by the prediction model. We see this as a potential context in which the kNN methodology may be implemented. Real-time prediction for system control, via repeated forward simulation from the present state, currently exists as an aim and advantage of digital twins. However, in addition to the interpretability benefits of the metric learning, using a kNN classification model can provide faster results than expending simulation in the digital twin. The predictions obtained via forward simulation can nevertheless provide a useful accuracy benchmark for the kNN model. Therefore, we recognise digital twins as providing a useful environment for both the training and implementation of the kNN methodology.

In the practical use of this methodology, we encounter decisions regarding the level of detail included in the system state. Beginning at the building of the simulation model itself, decisions arise after which certain features of the physical system will be represented and certain others ignored. This defines the available complexity of the simulation state description used by the kNN model. We can then choose to take a subset of these variables into the model. Although these practical considerations have not been our focus, we can acknowledge various trade-offs relevant to the computational aspect. The more state variables included in the kNN model, the more data points (which typically translates to more simulation replications) will be needed to sufficiently cover the state space. A larger state space brings increased computational demands to the metric learning optimisation. Conversely, including fewer state variables, although risking the loss of useful information and potentially sacrificing prediction accuracy, enables state repetitions to be observed with greater frequency among fewer data points, which reduces computation times and accentuates the advantage provided by SNCA over alternative methods of metric learning. In our experiments, we generally constructed the state description used by the kNN model from aspects of the simulation state representing observable components of the real system. If the dimension of this is prohibitively large, some initial reduction via existing feature selection algorithms may be helpful.

To place in perspective the performance of the proposed methodology for kNN with metric learning, some assessment of its achieved predictive capability in the context of the stochastic response would be helpful. We reiterate that a digital twin environment would allow the kNN prediction accuracy to be compared with that obtained by repeated forward simulation. This would represent the *optimal* benchmark for prediction accuracy, reflecting the inherent stochasticity in the response. Additionally, a fuller comparison with the performance of alternative machine learning prediction algorithms would also provide valuable context.

We now direct our focus towards potential extensions to the metric learning formulation, which constitutes a significant component of the kNN methodology.

6.2.2 Metric Learning for Simulation

Simulation is an inherently time-dependent process. Non-stationary input models naturally precipitate non-stationary outputs. Moreover, we expect relationships among system state variables and dynamic outputs to also exhibit non-stationarity. For example, when service rates increase, a high queue size becomes less detrimental to a customer's waiting time. The kNN and metric learning methodology in Chapters 3 and 4 does not address this; the similarity between system state observations was defined without consideration of the simulation time at which the observations were recorded. In the notation of Chapter 4, we recognise that the data-generating distribution, $q(\boldsymbol{x}, y)$, may in fact vary with time, and as such, a single metric may not be globally optimal. We begin in this section by considering various approaches to accommodating non-stationarity in the metric learning.

Initially, we might simply include time as an additional dimension of the state description. However, this would be a naive approach, with a linear Mahalanobis metric unable to reflect the possible periodic nature of the time effect. In models for which the non-stationary characteristics are well-understood, for example with input behaviours repeating over an hourly or daily cycle, then an appropriate periodic conversion of the time variable may prove sufficient. In general, however, simulation models induce non-stationarity of unknown characteristics. In addition to the time variable, we can envisage other state variables also having potentially non-linear relationships with performance. For example, a queue length being very low or very high might in different ways lead to high costs. Therefore, an exploration of *non-linear* metric learning for simulation problems may prove a useful exercise in a wider sense.

A common approach to non-linear metric learning is through a kernelisation of linear methods. To understand this approach, we recognise that the Mahalanobis metric framework may sometimes be represented in terms of the inner products, $\boldsymbol{x}_i^\top \boldsymbol{x}_j$, between data points. For example, the Euclidean distance can be written as $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = [\boldsymbol{x}_i^\top \boldsymbol{x}_i - 2\boldsymbol{x}_i^\top \boldsymbol{x}_j + \boldsymbol{x}_j^\top \boldsymbol{x}_j]^{1/2}$. Inner products can then be replaced with a *kernel* function,

$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$, using a non-linear function, ϕ. Applying Mahalanobis metric learning with such a kernel function corresponds to learning a distance of the form $\|A\phi(\boldsymbol{x}_i) - A\phi(\boldsymbol{x}_j)\|_2$. In other words, we learn a linear transformation in the space of ϕ. However, we only need to access the inner products through κ, thereby avoiding the computational cost of working in the potentially high dimensional space of ϕ. As an example of this approach, Torresani and Lee (2006) propose a kernalised version of the Large Margin Nearest Neighbors method (Weinberger and Saul, 2009).

Kernelisation is a technique used more widely throughout machine learning to represent non-linear feature spaces. For example, it is often applied in conjunction with support vector machines, with common choices of kernel function including the polynomial, Gaussian, and radial basis function (RBF) kernels (Hastie et al., 2009). For our purpose, we might look for a periodic kernel function, such as introduced by MacKay et al. (1998).

While kernelisation has the potential to improve the metric learning performance on highly non-linear problems, it can alternatively lead to overfitting. Depending on the levels of non-stationarity and non-linearity present, it may not be helpful as a general approach to simulation problems. In most cases, we expect a linear relationship between state variables and performance to provide an appropriate fit. Therefore, rather than attempting to generalise every relationship with non-linearity, we might consider a gentler approach of isolating the time variable, and using the idea of *local* metric learning.

Local metric learning trains separate metrics to operate over separate regions of the input space. To accommodate non-stationary simulation behaviour, we might look to partition the state data according to the value of the time variable, and learn a collection of Mahalanobis matrices. In the practical application of classifying a new state observation, we have the advantage of knowing its time value, and hence knowing which metric and training subset to use. In typical applications of local metric learning, the region of a new test point is not assumed to be known, and distances are computed to all training points, using the appropriate metrics. Weinberger and Saul (2008) consider two options for constructing regions; using class labels, or using an unsupervised clustering approach, while Frome et al. (2007) take the extreme approach of learning a separate distance for each training point. In general, the regions used for local metric learning are assumed to be pre-defined. With non-stationary simulation behaviour of unknown characteristics, the discovery of an appropriate time-partitioning may become a learning task of its own.

Learning a separate distance function to apply to data over different time periods seems a natural approach to accommodating non-stationarity. We may, however, expect a core similarity to exist across each of the metrics. This motivates the topic of *multi-task* metric learning, in which the metrics across each region may share some composition, and thus the separate tasks can inform one another. For example, Parameswaran and Weinberger (2010) define the Mahalanobis matrix for the i^{th} task to take the form $M_0 + M_i$. With this construction, we may interpret M_0 as representing the general system-wide behaviours, while M_i collects the more subtle differences to the interactions specific to a particular time period. Alternatively, the separate matrices can be encouraged to remain similar via regularisation with the common matrix, M_0 (for example, see Yang et al. (2013)). In this case, M_0 might be initially learned by global metric learning over the state variables. As an example from the perspective of learning transformation matrices (as is our approach for SNCA), Yang et al. (2011) use the decomposition $A_i A_0$, where $A_0 \in \mathbb{R}^{r \times d}$ embeds each task into a common subspace of dimension $r < d$, and $A_i \in \mathbb{R}^{r \times r}$ describes a task-specific transformation.

Apart from separating offline metric learning tasks via a partitioning of the time variable, we also acknowledge the setting of *online* metric learning. In this framework, new training instances allow continual updates to the optimal distance metric. The first example of this was provided by Shalev-Shwartz et al. (2004), in which, at each step, the metric is updated via an orthogonal projection to satisfy a new pairwise constraint, followed by a second projection to return to the positive semidefinite cone. Extending this approach, Jin et al. (2009) present a trade-off between satisfying the new constraint and remaining close to the previous metric, allowing flexibility as to the extent of influence of new observations. An online approach to metric learning for simulation would particularly suit a digital twin environment. Digital twins are simulation models which aim to synchronise with live data from a real system, and are becoming increasingly popular for real-time monitoring and system control (dos Santos et al., 2022). These are practical scenarios to which the metric learning and kNN methodology for simulation are ideally suited.

Accommodation of non-stationary behaviour seems a desirable extension for metric learning in the simulation context. In addition to this, we might also consider accommodating continuous performance variables. Metric learning for a regression response has received little attention beyond the initial work of Weinberger and Tesauro (2007). In the framework of SNCA, we may in the simplest case construct $\hat{q}(y \mid \boldsymbol{b}_l)$ as the mean response from the observations of $\boldsymbol{b}_l$ (provided that $c_l > 0$), and use a suitable objective function such as minimising the mean squared error to the model estimates, $p^A(y \mid \boldsymbol{b}_l)$.

Simulation performance measures, such as costs and waiting times, are often naturally continuous-valued variables. However, their precise values may largely be a reflection of simulation noise, and we do not necessarily lose relevant information by discretising to a classification response. Furthermore, a discrete output more naturally aligns with system control objectives. Therefore, while an extension of SNCA to the regression context may be theoretically straightforward, it may not necessarily be advantageous in a practical sense.

6.2.3 Simulation Shapelets

The shapelet approach developed in Chapter 5 provides a versatile methodology for the analysis of dynamic simulation trajectories. This can be variously applied depending on purpose, and we demonstrated examples in which comparisons of trajectories from two competing systems were made. An alternative application which we have not shown is to provide a comparison between '*good*' and '*bad*' trajectories of the *same* system, as defined by some overall performance value. Replications of a system can often exhibit vastly different performance, and shapelets may be able to reveal the characteristic behaviours which distinguish this phenomenon. This might assist an understanding of the inherent variability of a system, and the conditions and short-term patterns of behaviour that define its performance.

Continuing on the side of application, the use of shapelets for dynamic model validation, as illustrated in Section 5.4.2, would benefit from a more rigorous framework. Quantifying the validity of a model based on classification accuracy suggests that we should make an effort to construct the most accurate classifier possible. With this aim, the idea of performing a shapelet transform and applying alternative classifiers on a collection of distance features, as proposed by Hills et al. (2014), seems sensible.

To understand the contribution of shapelets in the context of existing methodology, we might also explore a comparison with previous work. Simulation shapelets target a visual interpretation of dynamic behaviour for which we have not found directly comparable alternatives. However, simulation trajectories are analysed by Morgan and Barton (2022) from a statistical perspective of their Fourier transformations, and a comparison of interpretations with this approach may be possible and useful.

In terms of methodology, extensions to time series shapelets have progressed in a number of directions, which were outlined in Section 2.3.2. We can imagine the progression of simulation shapelets in similar directions. Importantly, we note that our main motivation for simulation shapelets is towards interpretation rather than classification. For this reason, we do not require a decision tree classifier, which makes the use of the information gain quality measure non-essential. Computing the optimal information gain requires its evaluation over $n-1$ possible splitting thresholds. Speed-up in the shapelet search could therefore be attained by using an alternative measure which is independent of a splitting threshold, such as the F-statistic suggested by Hills et al. (2014).

6.3 Closing Remarks

This thesis has presented methodology to unlock deeper insights and comparisons of the dynamic behaviour of stochastic simulation models. We have proposed a kNN view of simulation states to allow predictions of a dynamic performance measure, and tailored a method of metric learning to apply to this context. We have also developed methodology for a shapelet analysis of simulation trajectories.

We target a more nuanced understanding of simulation behaviour, providing fine-grained insights from the dynamic sample path and facilitating real-time predictions. More broadly, our contributions indicate the rich opportunities contained in sample path data. The constant progression of machine learning and data analytics has the potential to significantly expand our understanding of simulated systems. The perspective of simulation analytics presents an open field of opportunities, and it is hoped that the ideas developed in this thesis, at the very least, may serve to inspire further efforts in this emerging field.

Appendix A

Appendix to Chapter 4

A.1 Gradient Derivations

We derive the gradient expression for $\partial g_n(A)/\partial A$ given in Section 4.3.2, along with a similar expression for $\partial f_n(A)/\partial A$. We let $\boldsymbol{b}_{lh}$ denote $\boldsymbol{b}_l - \boldsymbol{b}_h$. Recalling

$$p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) = \frac{c_h^y \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}, \quad p^A((\boldsymbol{b}_l, y)|\boldsymbol{b}_l) = 0,$$
$$p_{lh} = \sum_{y\in\mathcal{Y}} p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l),$$

and using that $d\|A\boldsymbol{x}\|_2^2/dA = 2A\boldsymbol{x}\boldsymbol{x}^\top$, we have

$$\begin{aligned}\frac{\partial p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)}{\partial A} &= \frac{\begin{array}{c}-2A\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top c_h^y \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\} \sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\} \\ - c_h^y \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\} \sum_{k\neq l} -2A\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\end{array}}{\left(\sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\right)^2} \\ &= 2A\left\{-\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) + p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)\sum_{k\neq l}\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk}\right\} \\ &= 2A\left\{p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)\left(\left(\sum_{k\neq l}\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk}\right) - \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top\right)\right\}.\end{aligned}$$

Therefore, making use of the identity, $p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) = p_{lh}\hat{q}(y|\boldsymbol{b}_h)$,

$$\begin{aligned}
\frac{\partial \sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)}{\partial A} &= 2A \sum_{h\neq l} \left\{ p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) \left(\left(\sum_{k\neq l} \boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk} \right) - \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top \right) \right\} \\
&= 2A \left\{ p^A(y|\boldsymbol{b}_l) \sum_{k\neq l} \boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk} - \sum_{h\neq l} \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}\hat{q}(y|\boldsymbol{b}_h) \right\} \\
&= 2A \sum_{h\neq l} \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h)).
\end{aligned}$$

Now the gradient expression can be reached as follows:

$$\begin{aligned}
g_n(A) &= \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \log \left(\hat{q}(\boldsymbol{b}_l) \sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) \right), \\
\frac{\partial g_n(A)}{\partial A} &= 2A \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \frac{\hat{q}(\boldsymbol{b}_l) \sum_{h\neq l} \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h))}{\hat{q}(\boldsymbol{b}_l) \sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)} \\
&= 2A \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \frac{\sum_{h\neq l} \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h))}{p^A(y|\boldsymbol{b}_l)} \\
&= 2A \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \sum_{h\neq l} \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh} \left(1 - \frac{\hat{q}(y|\boldsymbol{b}_h)}{p^A(y|\boldsymbol{b}_l)} \right).
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
f_n(A) &= \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l), \\
\frac{\partial f_n(A)}{\partial A} &= 2A \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \sum_{h\neq l} \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h)).
\end{aligned}$$

A.2 Proof of Uniform Convergence in Theorem 4.3.1

Theorem 4.3.1. *Suppose there exists a compact subset $C \subset \mathbb{R}^{d\times d}$ such that:*

(i) $A_g^\star$ is nonempty and contained in C,

(ii) for sufficiently large n, with probability 1, $\hat{A}_g$ is nonempty and contained in C,

(iii) $g(A)$ *is finite-valued and continuous on* C*, and*

(iv) $g_n(A) \overset{a.s.}{\to} g(A)$ *as* $n \to \infty$*, uniformly on* C*.*

Then $g_n(\hat{A}_g) \overset{a.s.}{\to} g(A_g^\star)$ *and* $\mathbb{D}(\hat{A}_g, A_g^\star) \overset{a.s.}{\to} 0$ *as* $n \to \infty$*.*

Letting $\boldsymbol{b}_{lh}$ denote $\boldsymbol{b}_l - \boldsymbol{b}_h$, we establish the uniform convergence of

$$g_n(A) \overset{a.s.}{\to} \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} q(\boldsymbol{b}_l, y) \log \left(q(\boldsymbol{b}_l) \frac{\sum_{h \neq l} q(\boldsymbol{b}_h, y) \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k \neq l} q(\boldsymbol{b}_k) \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \right), \tag{A.2.1}$$

as required by condition *(iv)* of Theorem 4.3.1. Recall that

$$\begin{aligned} g_n(A) &= \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \log p^A(\boldsymbol{b}_l, y) \\ &= \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \log \left(\hat{q}(\boldsymbol{b}_l) \frac{\sum_{h \neq l} \hat{q}(\boldsymbol{b}_h, y) \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k \neq l} \hat{q}(\boldsymbol{b}_k) \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \right). \end{aligned}$$

We begin by establishing the convergence of

$$p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) = \frac{\hat{q}(\boldsymbol{b}_h, y) \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k \neq l} \hat{q}(\boldsymbol{b}_k) \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \to \frac{q(\boldsymbol{b}_h, y) \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k \neq l} q(\boldsymbol{b}_k) \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}. \tag{A.2.2}$$

We assume that the following condition holds for some $\delta > 0$:

$$q(\boldsymbol{b}_l) > \delta \quad \forall l = 1, 2, \ldots, m. \tag{A.2.3}$$

We have that $\hat{q}(\boldsymbol{b}_h, y) \overset{a.s.}{\to} q(\boldsymbol{b}_h, y)$ and $\hat{q}(\boldsymbol{b}_h) \overset{a.s.}{\to} q(\boldsymbol{b}_h)$ $\forall h, y$ as $n \to \infty$. This can be expressed in the following way:

Let $\epsilon > 0$. Then $\exists\, N$ s.t. $\forall h \in \{1, 2, \ldots, m\}$ and $\forall y \in \{0, 1\}$, with probability 1 $\forall n > N$,

$$\begin{aligned} |\hat{q}(\boldsymbol{b}_h, y) - q(\boldsymbol{b}_h, y)| &< \frac{\epsilon\delta^2}{m + \epsilon\delta}, \quad \text{(note that } \frac{\epsilon\delta^2}{m + \epsilon\delta} < \frac{\epsilon\delta^2}{m} < \epsilon) \\ |\hat{q}(\boldsymbol{b}_h) - q(\boldsymbol{b}_h)| &< \frac{\epsilon\delta^2}{m + \epsilon\delta}. \end{aligned}$$

To target the convergence of (A.2.2), consider the absolute difference when $n > N$:

$$\left| \frac{\hat{q}(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l}\hat{q}(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} - \frac{q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l}q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \right|$$

$$= \left| \frac{\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}\left[\sum_{k\neq l}\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))\right]}{\left(\sum_{k\neq l}q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\right)\left(\sum_{k\neq l}\hat{q}(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\right)} \right|$$

$$= \left| \frac{\begin{array}{l}\exp\left\{-\|A\boldsymbol{b}_{lh}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\cdot \\ \quad\left[\sum_{k\neq l}\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))\right]\end{array}}{\left(\sum_{k\neq l}q(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)\left(\sum_{k\neq l}\hat{q}(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)} \right|$$

where $r_l = \arg\min_{k\neq l}\|A\boldsymbol{b}_{lk}\|_2^2$

$$= \left| \frac{\begin{array}{l}\exp\left\{-\|A\boldsymbol{b}_{lh}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\cdot \\ \quad\left[\sum_{k\neq l}\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))\right]\end{array}}{\begin{array}{c}\left(q(\boldsymbol{b}_{r_l}) + \sum_{k\neq l,r_l}q(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)\cdot \\ \left(\hat{q}(\boldsymbol{b}_{r_l}) + \sum_{k\neq l,r_l}\hat{q}(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)\end{array}} \right|$$

Every exponential term is ≤ 0. The summations in the denominator are > 0.

Therefore,

$$\begin{aligned}
&\leq \left| \frac{1\cdot\sum_{k\neq l}1\cdot(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \right| \\
&\leq \frac{\sum_{k\neq l}|\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k)|}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
&= \frac{\sum_{k\neq l}|(\hat{q}(\boldsymbol{b}_h, y) - q(\boldsymbol{b}_h, y))q(\boldsymbol{b}_k) + q(\boldsymbol{b}_h, y)(q(\boldsymbol{b}_k) - \hat{q}(\boldsymbol{b}_k))|}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
&\leq \frac{\sum_{k\neq l}q(\boldsymbol{b}_k)|\hat{q}(\boldsymbol{b}_h, y) - q(\boldsymbol{b}_h, y)| + q(\boldsymbol{b}_h, y)|q(\boldsymbol{b}_k) - \hat{q}(\boldsymbol{b}_k)|}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
&\leq \frac{\left(\frac{\epsilon\delta^2}{m+\epsilon\delta}\right)\sum_{k\neq l}(q(\boldsymbol{b}_k) + q(\boldsymbol{b}_h, y))}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
&\leq \frac{\left(\frac{\epsilon\delta^2}{m+\epsilon\delta}\right)m}{\delta\left(\delta - \frac{\epsilon\delta^2}{m+\epsilon\delta}\right)} \qquad \left(\text{note that } \frac{\epsilon\delta^2}{m+\epsilon\delta} < \delta\right) \qquad \text{(A.2.4)} \\
&= \frac{\epsilon\delta^2 m}{\delta^2(m+\epsilon\delta) - \epsilon\delta^3} \\
&= \epsilon
\end{aligned}$$

This shows that the convergence (A.2.2) holds uniformly over A. (Note that r_l depends on A, but the condition (A.2.3) ensures that the bound (A.2.4) holds for all r_l, i.e. for all A.)

To establish the uniform convergence of (A.2.1), we need to give attention to the logarithm function. Following the uniform convergence of (A.2.2), simple results regarding the sums and products of convergent sequences ensure that the argument of the logarithms in $g_n(A)$ converge uniformly. We also note that the logarithm is a uniformly continuous function on $[a, \infty), a > 0$ (a bounded derivative implies uniform continuity). Combining these two facts provides uniform convergence of the logarithms:

Let $\epsilon > 0$. Then $\exists\ \xi > 0$ s.t. for every $x, y \in [a, \infty)$ with $|x - y| < \xi$, we have that $|\log(x) - \log(y)| < \epsilon$ (uniform continuity of the logarithm on $[a, \infty), a > 0$). Using this $\xi, \exists\ N$ s.t. $\left|p^A(\boldsymbol{b}_l, y) - q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right| < \xi\ \forall n > N, \forall A$ (uniform convergence of the argument of the logarithms in $g_n(A)$). Provided that $p^A(\boldsymbol{b}_l, y) \in [a, \infty)$ and $q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \in [a, \infty)$, this gives

$$\left|\log(p^A(\boldsymbol{b}_l, y)) - \log\left(q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right)\right| < \epsilon \quad \forall n > N, \forall A$$

$$\implies \log(p^A(\boldsymbol{b}_l, y)) \to \log\left(q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right) \text{ uniformly over } A.$$

Therefore, the logarithms in $g_n(A)$ will preserve uniform convergence as long as their arguments $\in [a, \infty)$. The condition given by (4.3.4) in Section 4.3.4 ensures that this holds. With this, we establish the uniform convergence of (A.2.1).

A.3 kNN Procedures

To assess the practical performance of the distance metrics considered in Chapter 4, we performed kNN classification in which k refers to the number of unique, non-identical neighbours and includes all of their repeats. Algorithm 2 outlines the procedure that was followed. We use $\mathcal{D}^{\text{test}}$ to denote the set of query points which cover the subset of states,

Algorithm 2 kNN

```
   Input: $\mathcal{X}_{\text{test}}, \mathcal{X}_{\text{train}} \subseteq \mathcal{X}$,
      $\mathcal{D}^{\text{test}} = \{c_l^y(\text{test})\}_{l:\, \boldsymbol{b}_l \in \mathcal{X}_{\text{test}}}^{y \in \mathcal{Y}}$,
      $\mathcal{D}^{\text{train}} = \{c_l^y(\text{train})\}_{l:\, \boldsymbol{b}_l \in \mathcal{X}_{\text{train}}}^{y \in \mathcal{Y}}$,
      $A$,
      $k < |\mathcal{X}_{\text{train}}|$
1: count $\leftarrow 0$
2: for $\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}$ do
3:     $\mathcal{N}_l^k \leftarrow \arg\min_{\boldsymbol{b}_h \in \mathcal{X}_{\text{train}} \setminus \{\boldsymbol{b}_l\}} \|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2$
4:     while $|\mathcal{N}_l^k| < k$ do
5:         $\mathcal{N}_l^k \leftarrow \mathcal{N}_l^k \cup \{\arg\min_{\boldsymbol{b}_h \in \mathcal{X}_{\text{train}} \setminus (\{\boldsymbol{b}_l\} \cup \mathcal{N}_l^k)} \|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2\}$
6:     end while
7:     $\hat{y} \leftarrow \arg\max_{y \in \mathcal{Y}} \{\sum_{\boldsymbol{b}_h \in \mathcal{N}_l^k} c_h^y(\text{train})\}$
8:     count $\leftarrow$ count $+ c_l^{\hat{y}}(\text{test})$
9: end for
   Return: Accuracy $\leftarrow$ count$/ \sum_{\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}} \sum_{y \in \mathcal{Y}} c_l^y(\text{test})$
```

$\mathcal{X}_{\text{test}}$. For the state $\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}$, we use $c_l^y(\text{test})$ to denote the frequency of the pair $(\boldsymbol{b}_l, y)$ in $\mathcal{D}^{\text{test}}$. Similarly, $\mathcal{D}^{\text{train}}$, $\mathcal{X}_{\text{train}}$, and $c_l^y(\text{train})$ relate to the set of training points. The following figures in Chapter 4 are specifically constructed from Algorithm 2 as follows:

1. *Figure 4.4.5 (left)*: The metrics are trained on a dataset, $\mathcal{D}_n$, where n is shown on the x-axis. For each metric, the leave-one-out 1NN accuracy is calculated on a fixed dataset, $\mathcal{D}_{100{,}000}$. That is, Algorithm 2 is applied with $\mathcal{X}_{\text{test}} = \mathcal{X}_{\text{train}} = \mathcal{X}$ and $\mathcal{D}^{\text{test}} = \mathcal{D}^{\text{train}} = \mathcal{D}_{100{,}000}$. For each n marked on the x-axis, we repeat the process for 10 independent datasets of $\mathcal{D}_n$, with the average classification accuracy and ± 1.96 standard errors plotted.

2. *Figure 4.4.5 (right)*: We begin with a dataset, $\mathcal{D}_n$, where n is shown on the x-axis, and perform 6-fold CV on the 36 states in $\mathcal{X}$. For each fold, $|\mathcal{X}_{\text{test}}| = 6$ and $|\mathcal{X}_{\text{train}}| = 30$, and the metrics are trained on the dataset constructed as $\{c_l^y \in \mathcal{D}_n \mid \boldsymbol{b}_l \in \mathcal{X}_{\text{train}}\}$. The 1NN accuracy results are again obtained from a fixed dataset $\mathcal{D}_{100{,}000}$. We construct $\mathcal{D}^{\text{test}} = \{c_l^y \in \mathcal{D}_{100{,}000} \mid \boldsymbol{b}_l \in \mathcal{X}_{\text{test}}\}$, and $\mathcal{D}^{\text{train}} = \{c_l^y \in \mathcal{D}_{100{,}000} \mid \boldsymbol{b}_l \in \mathcal{X}_{\text{train}}\}$. We plot the average 1NN accuracy and ± 1.96 standard errors over the 6 folds and after repeating the metric training over 10 independent datasets of $\mathcal{D}_n$. The specific

partitioning of the CV folds remained consistent throughout to create a fair comparison over n.

3. *Figure 4.4.7 (right)*: 10 datasets of $\mathcal{D}_{18,402} = \{c_l^y\}_{l:\, \boldsymbol{b}_l \in \mathcal{X}}^{y \in \mathcal{Y}}$ were used. For each dataset, the metric and kNN training states were taken to be those with at least 25 observations, and those with fewer than 25 observations made up the test states. Specifically, $\mathcal{X}_{\text{train}} = \{\boldsymbol{b}_l \in \mathcal{X} \mid c_l \geq 25\}$, $\mathcal{X}_{\text{test}} = \{\boldsymbol{b}_l \in \mathcal{X} \mid c_l < 25\}$, $\mathcal{D}^{\text{train}} = \{c_l^y \in \mathcal{D}_{18,402} \mid c_l \geq 25\}$, and $\mathcal{D}^{\text{test}} = \{c_l^y \in \mathcal{D}_{18,402} \mid c_l < 25\}$. From the 10 repetitions, the plot shows the average kNN accuracy and ± 1.96 standard errors over a range of k.

Appendix B

Appendix to Chapter 5

B.1 Proof of Theorem 5.3.1

Theorem 5.3.1. *Let $y\colon [0,T] \to \mathbb{R}$ be a piecewise constant function with change times in the set $\mathcal{T} = \{0 = t_0, t_1, \ldots, t_m = T\}$ and $s\colon [0,\ell] \to \mathbb{R}$ with $\ell \leq T$ a piecewise constant function with change times in the set $\mathcal{U} = \{0 = u_0, u_1, \ldots, u_{m'} = \ell\}$. Let $U_j = \{t_i - u_j : i = 0, 1, \ldots, m\} \cap [0, T-\ell]$ for $j = 0, 1, \ldots, m'$, and let $\mathcal{V} = \bigcup_{j=0}^{m'} U_j$.*

Then $\|s - y([t, t+\ell])\|_1$ is linear in t for $t \in [v, v']$, where $v, v' \in \mathcal{V}$ and $(v, v') \cap \mathcal{V} = \emptyset$.

Proof. We consider the piecewise constant segments of s.

For $j = 0, 1, \ldots, m'-1$, we have $s(t) = s(u_j)$ for $u_j \leq t < u_{j+1}$. Let $\mathcal{V}_j = U_j \cup U_{j+1} \cup \{0, T-\ell\}$, and consider two points, $x, x' \in [0, T-\ell]$, such that $(x, x') \cap \mathcal{V}_j = \emptyset$. We can write $x + \delta = x'$, with $\delta > 0$. For a graphical understanding, $\mathcal{V}_j$ contains the shapelet shifts such that an endpoint of the $(j+1)^{\text{th}}$ segment of s coincides with a change time of y.

Consider $\|s(u_j) - y([x+u_j, x+u_{j+1}])\|_1 = \int_{x+u_j}^{x+u_{j+1}} |s(u_j) - y(t)| dt$. This is the distance between the $(j+1)^{\text{th}}$ component of s when shifted by x and the corresponding portion of y. Since y is piecewise constant, this can be expressed as a sum of rectangular areas. The endpoints along the time axis of these rectangles are represented in the following set:

$$\{x + u_j\} \cup \{\mathcal{T} \cap (x + u_j, x + u_{j+1})\} \cup \{x + u_{j+1}\}. \tag{B.1.1}$$

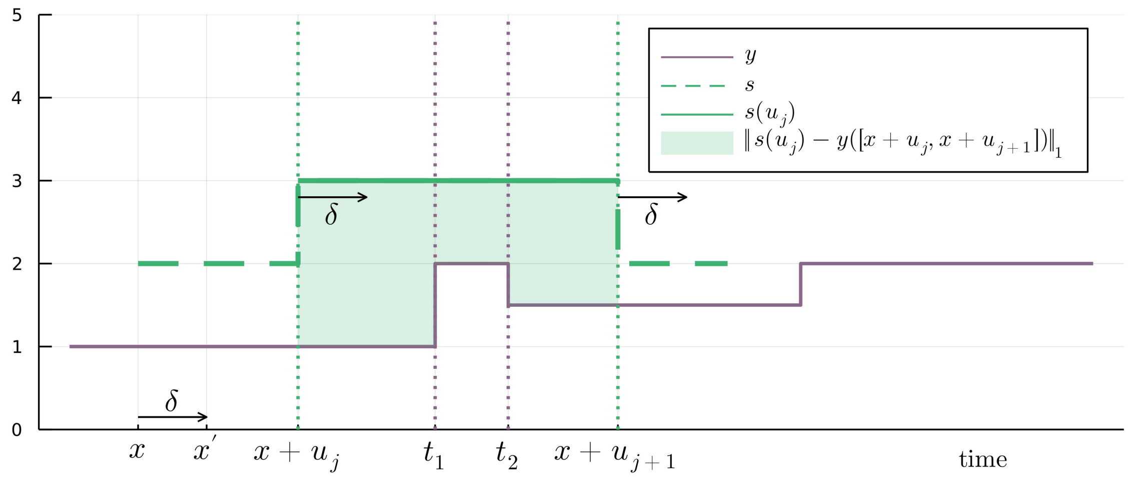


Figure B.1.1: A labelled example illustrating the notation used throughput this proof.

Figure B.1.1 provides a graphical illustration.

Similarly, consider $\|s(u_j) - y([x+h+u_j, x+h+u_{j+1}])\|_1$ for $0 < h \leq \delta$, with the corresponding set of endpoints:

$$\{x+h+u_j\} \cup \{\mathcal{T} \cap (x+h+u_j, x+h+u_{j+1})\} \cup \{x+h+u_{j+1}\}. \tag{B.1.2}$$

We can show that $(x+u_j, x+h+u_j) \cap \mathcal{T} = \emptyset$. This follows from a simple contradiction:

Assume that $t_i \in (x+u_j, x+h+u_j) \cap \mathcal{T}$. Then we can write $t_i = x+u_j+\epsilon$, with $0 < \epsilon < h$. Therefore $t_i - u_j = x+\epsilon \in (x, x')$. This ensures that $t_i - u_j \in [0, T-\ell]$, and since $t_i \in \mathcal{T}$, we must have $t_i - u_j \in U_j$. Therefore we have $t_i - u_j \in \mathcal{V}_j$ and also $t_i - u_j \in (x, x')$, but this is a contradiction since $(x, x') \cap \mathcal{V}_j = \emptyset$.

Similarly, we have that $(x+u_{j+1}, x+h+u_{j+1}) \cap \mathcal{T} = \emptyset$. This follow the same logic:

Assume that $t_i \in (x+u_{j+1}, x+h+u_{j+1}) \cap \mathcal{T}$. Then we can write $t_i = x+u_{j+1}+\epsilon$, with $0 < \epsilon < h$. Therefore $t_i - u_{j+1} = x+\epsilon \in (x, x')$. This ensures that $t_i - u_{j+1} \in [0, T-\ell]$, and since $t_i \in \mathcal{T}$, we must have $t_i - u_{j+1} \in U_{j+1}$. Therefore we have $t_i - u_{j+1} \in \mathcal{V}_j$ and also $t_i - u_{j+1} \in (x, x')$, but this is a contradiction since $(x, x') \cap \mathcal{V}_j = \emptyset$.

Therefore, we see that $\{\mathcal{T} \cap (x + u_j, x + u_{j+1})\} = \{\mathcal{T} \cap (x + h + u_j, x + h + u_{j+1})\}$. Comparing (B.1.1) and (B.1.2), this means that, in the graphical representation as a sum of rectangular areas, $\|s(u_j) - y([x+u_j, x+u_{j+1}])\|_1$ and $\|s(u_j) - y([x+h+u_j, x+h+u_{j+1}])\|_1$ can only differ in the widths of the first and last rectangles. Specifically, we have

$$\begin{aligned}\|s(u_j) - y([x + h + u_j, x + h + u_{j+1}])\|_1 = {} & \|s(u_j) - y([x + u_j, x + u_{j+1}])\|_1 \\ & -(x+h + u_j - (x + u_j))|s(u_j) - y(x + u_j)| \\ & +(x+h + u_{j+1} - (x + u_{j+1}))|s(u_j) - y(x + u_{j+1})| \\ = {} & \|s(u_j) - y([x + u_j, x + u_{j+1}])\|_1 \\ & +h(|s(u_j) - y(x + u_{j+1})| - |s(u_j) - y(x + u_j)|).\end{aligned}$$

Therefore, since $x < x + h \leq x'$, we see that $\|s(u_j) - y([t + u_j, t + u_{j+1}])\|_1$ is linear in t for $t \in [x, x']$.

Now, we can write

$$\|s - y([t, t + \ell])\|_1 = \sum_{j=0}^{m'-1} \|s(u_j) - y([t + u_j, t + u_{j+1}])\|_1.$$

Note that $\mathcal{V} = \bigcup_{j=0}^{m'} \mathcal{V}_j$, and so $\mathcal{V}_j \subseteq \mathcal{V}$ for each $j = 0, 1, \ldots, m' - 1$. This means that for any pair of successive elements $v, v' \in \mathcal{V}$ with $(v, v') \cap \mathcal{V} = \emptyset$, we must also have $(v, v') \cap \mathcal{V}_j = \emptyset$ for each j. Therefore, the component $\|s(u_j) - y([t + u_j, t + u_{j+1}])\|_1$ is linear in t for $t \in [v, v']$ for each j, and we see that $\|s - y([t, t + \ell])\|_1$, as a sum of these linear components, is also linear in t for $t \in [v, v']$.

□

B.2 Proof of Theorem 5.3.3

Theorem 5.3.3. *Let $y \colon [0, T] \to \mathbb{R}$, $s \colon [0, \ell] \to \mathbb{R}$, and $s^c \colon [0, \ell] \to \mathbb{R}$ be as in Corollary 5.3.2. Let $\mathcal{W}^t = \{\mathcal{T} \cap (t, t + \ell)\} \cup \{\mathcal{U} + t\}$, with ordered elements denoted by w_i^t such that*

$y(w_i^t) - s(w_i^t - t) \le y(w_{i+1}^t) - s(w_{i+1}^t - t)$ *for* $i = 1, 2, \dots, m_t$. *Let*

$$\lambda_i = \begin{cases} \min\{w_j^t - w_i^t \colon w_j^t \in \mathcal{W}^t, w_j^t > w_i^t\} & \text{if } w_i^t \in \mathcal{W}^t \backslash \{t+\ell\}, \\ 0 & \text{if } w_i^t = t + \ell, \end{cases}$$

and let $k^\star = \min\left\{k \in \{1, \dots, m_t\} \colon \sum_{i=1}^{k} \lambda_i \ge \frac{\ell}{2}\right\}$.

Then $\min_{c \in \mathbb{R}} \|s^c - y([t, t+\ell])\|_1$ *is attained by* $c^t = y(w_{k^\star}^t) - s(w_{k^\star}^t - t)$.

Proof. $\mathcal{W}_t$ contains the relevant time points required to express $\|s^c - y([t, t+\ell])\|_1$ as a sum of rectangular area components. The set, $\{\lambda_i\}_{i=1}^{m_t}$, comprise the widths of these components. Note that the index, i, orders the time points in $\mathcal{W}^t$ by their series value minus their shapelet value, and the widths, λ_i, are ordered equivalently. Figure B.2.2 provides an example for illustration.

We can express

$$\|s^c - y([t, t+\ell])\|_1 = \sum_{i=1}^{m_t} \lambda_i |s(w_i^t - t) + c - y(w_i^t)|.$$

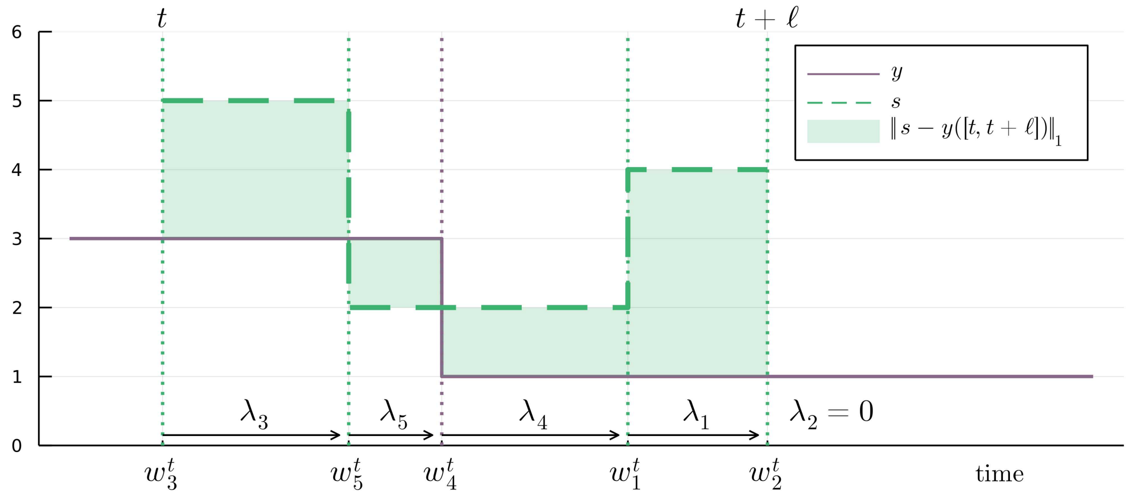


Figure B.2.2: A labelled example illustrating the notation used throughput this proof.

We have

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} = \sum_{i=1}^{m_t} \lambda_i \mathsf{sign}(c - (y(w_i^t) - s(w_i^t - t)))$$
$$\text{for } c \neq y(w_i^t) - s(w_i^t - t), i = 1, 2, \ldots, m_t.$$

Therefore, for $c \in (y(w_k^t) - s(w_k^t - t), y(w_{k+1}^t) - s(w_{k+1}^t - t))$, we can write

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} = \sum_{i=1}^{k} \lambda_i - \sum_{i=k+1}^{m_t} \lambda_i,$$

which implies

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} \begin{cases} < 0 \text{ if } \sum_{i=1}^{k} \lambda_i < \sum_{i=k+1}^{m_t} \lambda_i \\ \geq 0 \text{ if } \sum_{i=1}^{k} \lambda_i \geq \sum_{i=k+1}^{m_t} \lambda_i. \end{cases}$$

We note that since $\{\lambda_i\}_{i=1}^{m_t}$ represent the widths of a partition of $[t, t+\ell]$, we have $\sum_{i=1}^{m_t} \lambda_i = \ell$. Therefore, for $k^\star = \min\left\{k \in \{1, 2, \ldots, m_t\} \colon \sum_{i=1}^{k} \lambda_i \geq \frac{\ell}{2}\right\}$, we have

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} < 0 \text{ for } c \in (y(w_{k^\star-1}^t) - s(w_{k^\star-1}^t - t), y(w_{k^\star}^t) - s(w_{k^\star}^t - t)),$$
$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} \geq 0 \text{ for } c \in (y(w_{k^\star}^t) - s(w_{k^\star}^t - t), y(w_{k^\star+1}^t) - s(w_{k^\star+1}^t - t)).$$

More generally, we will have

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} < 0 \text{ for } c < y(w_{k^\star}^t) - s(w_{k^\star}^t - t),$$
$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} \geq 0 \text{ for } c > y(w_{k^\star}^t) - s(w_{k^\star}^t - t).$$

This implies that $\|s^c - y([t, t+\ell])\|_1$ is minimised at $c^t = y(w_{k^\star}^t) - s(w_{k^\star}^t - t)$.

$\square$

List of References

Arul, M. and Kareem, A. (2021). Applications of Shapelet Transform to Time Series Classification of Earthquake, Wind and Wave Data. *Engineering Structures*, 228:111564.

Bagnall, A. and Janacek, G. (2014). A Run Length Transformation for Discriminating between Auto Regressive Time Series. *Journal of Classification*, 31:154–178.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The Great Time Series Classification Bake Off: A Review and Experimental Evaluation of Recent Algorithmic Advances. *Data Mining and Knowledge Discovery*, 31:606–660.

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2003). Learning Distance Functions using Equivalence Relations. In *Proceedings of the 20th International Conference on Machine Learning*, pages 11–18.

Barton, R. R. (2020). Tutorial: Metamodeling for Simulation. In *Proceedings of the 2020 Winter Simulation Conference*, pages 1102–1116. Institute of Electrical and Electronics Engineers (IEEE).

Bellet, A., Habrard, A., and Sebban, M. (2014). A Survey on Metric Learning for Feature Vectors and Structured Data. `https://arxiv.org/pdf/1306.6709.pdf`, accessed 12th June 2023.

Berndt, D. J. and Clifford, J. (1994). Using Dynamic Time Warping to find Patterns in Time Series. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, volume 10, pages 359–370.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When Is 'Nearest Neighbor' Meaningful? In *International Conference on Database Theory*, pages 217–235, Berlin, Heidelberg. Springer.

Bostrom, A. and Bagnall, A. (2017). A Shapelet Transform for Multivariate Time Series Classification. https://arxiv.org/pdf/1712.06428.pdf, accessed 12th June 2023.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brady, T. F. and Yellig, E. (2005). Simulation Data Mining: A New Form of Computer Simulation Output. In *Proceedings of the 2005 Winter Simulation Conference*, pages 285–289. Institute of Electrical and Electronics Engineers (IEEE).

Brailsford, S. C. (2007). Tutorial: Advances and Challenges in Healthcare Simulation Modeling. In *Proceedings of the 2007 Winter Simulation Conference*, pages 1436–1448. Institute of Electrical and Electronics Engineers (IEEE).

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and Regression Trees. *Biometrics*, 40(3):358–361.

Cao, L. and Tay, F. E. (2001). Financial Forecasting using Support Vector Machines. *Neural Computing & Applications*, 10(2):184–192.

Censor, Y., Zenios, S. A., et al. (1997). *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

Cetin, M. S., Mueen, A., and Calhoun, V. D. (2015). Shapelet Ensemble for Multi-Dimensional Time Series. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 307–315. Society for Industrial and Applied Mathematics (SIAM).

Chang, K.-W., Deka, B., Hwu, W.-M. W., and Roth, D. (2012). Efficient Pattern-based Time Series Classification on GPU. In *IEEE 12th International Conference on Data Mining*, pages 131–140. Institute of Electrical and Electronics Engineers (IEEE).

Chen, H., Tang, F., Tino, P., and Yao, X. (2013). Model-based Kernel for Efficient Time Series Analysis. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–400.

Conway, R. W. (1963). Some Tactical Problems in Digital Simulation. *Management Science*, 10(1):47–61.

Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-Theoretic Metric Learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216.

Deng, H., Runger, G., Tuv, E., and Vladimir, M. (2013). A Time Series Forest for Classification and Feature Extraction. *Information Sciences*, 239:142–153.

Devroye, L. P. and Wagner, T. (1980). Distribution-Free Consistency Results in Nonparametric Discrimination and Regression Function Estimation. *The Annals of Statistics*, pages 231–239.

dos Santos, C. H., Montevechi, J. A. B., de Queiroz, J. A., de Carvalho Miranda, R., and Leal, F. (2022). Decision Support in Productive Processes through DES and ABS in the Digital Twin Era: A Systematic Literature Review. *International Journal of Production Research*, 60(8):2662–2681.

Dutta, U. K., Harandi, M., and Sekhar, C. C. (2020). Unsupervised Deep Metric Learning via Orthogonality Based Probabilistic Loss. *IEEE Transactions on Artificial Intelligence*, 1(1):74–84.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast Subsequence Matching in Time-Series Databases. *ACM Sigmod Record*, 23(2):419–429.

Fazel, M., Hindi, H., and Boyd, S. P. (2001). A Rank Minimization Heuristic with Application to Minimum Order System Approximation. In *Proceedings of the 2001 American Control*

Conference (Cat. No. 01CH37148), volume 6, pages 4734–4739. Institute of Electrical and Electronics Engineers (IEEE).

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188.

Fishman, G. S. and Kiviat, P. J. (1967). The Analysis of Simulation-Generated Time Series. *Management Science*, 13(7):525–557.

Fishwick, P. A. (2017). Modeling as the Practice of Representation. In *Proceedings of the 2017 Winter Simulation Conference*, pages 4276–4287. Institute of Electrical and Electronics Engineers (IEEE).

Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning Globally-Consistent Local Distance Functions for Shape-based Image Retrieval and Classification. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. Institute of Electrical and Electronics Engineers (IEEE).

Fu, A., Narasimhan, B., and Boyd, S. (2018). CVXR: An R Package for Disciplined Convex Optimization. https://arxiv.org/pdf/1711.07582.pdf, accessed 12th June 2023.

Fu, M. C. (1994). Optimization via Simulation: A Review. *Annals of Operations Research*, 53:199–247.

Fu, M. C. (2015). *Handbook of Simulation Optimization*. Springer.

Ghalwash, M. F. and Obradovic, Z. (2012). Early Classification of Multivariate Temporal Observations by Extraction of Interpretable Shapelets. *BMC Bioinformatics*, 13:1–12.

Giabbanelli, P. J. (2019). Solving Challenges at the Interface of Simulation and Big Data using Machine Learning. In *Proceedings of the 2019 Winter Simulation Conference*, pages 572–583. Institute of Electrical and Electronics Engineers (IEEE).

Globerson, A. and Roweis, S. (2005). Metric Learning by Collapsing Classes. *Advances in Neural Information Processing Systems*, 18:451–458.

Glynn, P. W. (1989). A GSMP Formalism for Discrete Event Systems. *Proceedings of the IEEE*, 77(1):14–23.

Goldberger, J., Hinton, G. E., Roweis, S. T., and Salakhutdinov, R. R. (2004). Neighbourhood Components Analysis. In *NIPS'04: Proceedings of the 17th International Conference on Neural Information Processing Systems*, pages 513–520, Cambridge, Massachusetts. MIT Press.

Grabocka, J., Schilling, N., Wistuba, M., and Schmidt-Thieme, L. (2014). Learning Time-Series Shapelets. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 392–401.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, New York, 2nd edition.

He, Q., Zhuang, F., Shang, T., Shi, Z., et al. (2012). Fast Time Series Classification Based on Infrequent Shapelets. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 215–219. Institute of Electrical and Electronics Engineers (IEEE).

Heidelberger, P. and Welch, P. D. (1983). Simulation Run Length Control in the Presence of an Initial Transient. *Operations Research*, 31(6):1109–1144.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., and Bagnall, A. (2014). Classification of Time Series by Shapelet Transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881.

Horvat, T. and Job, J. (2020). The Use of Machine Learning in Sport Outcome Prediction: A Review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(5):e1380.

Hua, E. Y., Lazarova-Molnar, S., and Francis, D. P. (2022). Validation of Digital Twins: Challenges and Opportunities. In *Proceedings of the 2022 Winter Simulation Conference*, pages 2900–2911. Institute of Electrical and Electronics Engineers (IEEE).

Huang, K., Jin, R., Xu, Z., and Liu, C.-L. (2010). Robust Metric Learning by Smooth Optimization. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 244–251.

Huang, K., Ying, Y., and Campbell, C. (2009). GSML: A Unified Framework for Sparse Metric Learning. In *2009 9th IEEE International Conference on Data Mining*, pages 189–198. Institute of Electrical and Electronics Engineers (IEEE).

Huo, Z., Nie, F., and Huang, H. (2016). Robust and Effective Metric Learning using Capped Trace Norm: Metric Learning via Capped Trace Norm. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1605–1614.

Ingalls, R. (1998). The Value of Simulation in Modeling Supply Chains. In *Proceedings of the 1998 Winter Simulation Conference*, volume 2, pages 1371–1375. Institute of Electrical and Electronics Engineers (IEEE).

Jain, P., Kulis, B., and Dhillon, I. (2010). Inductive Regularized Learning of Kernel Functions. *Advances in Neural Information Processing Systems*, 23.

Jain, P., Kulis, B., Dhillon, I., and Grauman, K. (2008). Online Metric Learning and Fast Similarity Search. *Advances in Neural Information Processing Systems*, 21.

Jiang, G., Hong, L. J., and Nelson, B. L. (2020). Online Risk Monitoring using Offline Simulation. *INFORMS Journal on Computing*, 32(2):356–375.

Jin, R., Wang, S., and Zhou, Y. (2009). Regularized Distance Metric Learning: Theory and Algorithm. *Advances in Neural Information Processing Systems*, 22.

Kaban, A., Othman, Z., and Rohmah, D. (2012). Comparison of Dispatching Rules in Job-Shop Scheduling Problem using Simulation: A Case Study. *International Journal of Simulation Modelling*, 11(3):129–140.

Karlsson, I., Papapetrou, P., and Boström, H. (2016). Generalized Random Shapelet Forests. *Data Mining and Knowledge Discovery*, 30(5):1053–1085.

Kayton, D., Teyner, T., Schwartz, C., and Uzsoy, R. (1996). Effects of Dispatching and Down Time on the Performance of Wafer Fabs Operating under Theory of Constraints. In *19th IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pages 49–56. Institute of Electrical and Electronics Engineers (IEEE).

Kedem, D., Tyree, S., Sha, F., Lanckriet, G. R., and Weinberger, K. Q. (2012). Non-linear Metric Learning. In *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems*, pages 2573–2581.

Keogh, E., Wei, L., Xi, X., Lee, S.-H., and Vlachos, M. (2006). LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures. In *Proceedings of the 32nd International Conference on Very large Data Bases*, pages 882–893.

Kidger, P., Morrill, J., and Lyons, T. (2020). Generalised Interpretable Shapelets for Irregular Time Series. https://arxiv.org/pdf/2005.13948.pdf, accessed 12th June 2023.

Kim, S.-H. and Nelson, B. L. (2006). Selecting the Best System. *Handbooks in Operations Research and Management Science*, 13:501–534.

Kleijnen, J. P. (1995). Sensitivity Analysis and Optimization in Simulation: Design of Experiments and Case Studies. In *Proceedings of the 1995 Winter Simulation Conference*, pages 133–140. Institute of Electrical and Electronics Engineers (IEEE).

Kolmogorov, A. N. and Širjaev, A. N. (1992). *Selected Works of AN Kolmogorov. Vol. 2, Probability Theory and Mathematical Statistics.* Kluwer.

Kruskal, W. H. (1952). A Nonparametric Test for the Several Sample Problem. *The Annals of Mathematical Statistics*, pages 525–540.

Kulis, B. (2013). Metric Learning: A Survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

Lada, E. K., Wilson, J. R., Steiger, N. M., and Joines, J. A. (2007). Performance of a Wavelet-based Spectral Procedure for Steady-State Simulation Analysis. *INFORMS Journal on Computing*, 19(2):150–160.

Laidler, G. (2022). GitHub repository. https://github.com/GrahamLaidler/SNCA.

Laidler, G. (2023). GitHub repository. https://github.com/GrahamLaidler/SimulationShapelets.

Laidler, G., Morgan, L. E., Nelson, B. L., and Pavlidis, N. G. (2020). Metric Learning for Simulation Analytics. In *Proceedings of the 2020 Winter Simulation Conference*, pages 349–360. Institute of Electrical and Electronics Engineers (IEEE).

Law, A. M. and Kelton, W. D. (2007). *Simulation Modeling and Analysis, 3rd edition.* Mcgraw-Hill New York.

Law, M. T., Thome, N., and Cord, M. (2014). Fantope Regularization in Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1051–1058.

Li, D. and Tian, Y. (2018). Survey and Experimental Study on Metric Learning Methods. *Neural Networks*, 105:447–462.

Li, G., Choi, B., Xu, J., Bhowmick, S. S., Chun, K.-P., and Wong, G. L.-H. (2021). Shapenet: A Shapelet-Neural Network Approach for Multivariate Time Series Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8375–8383.

Lin, Y. and Nelson, B. L. (2018). Variance and Derivative Estimation of Virtual Performance. *ACM Transactions on Modeling and Computer Simulation*, 28(3):1–20.

Lin, Y., Nelson, B. L., and Pei, L. (2019). Virtual Statistics in Simulation via k Nearest Neighbors. *INFORMS Journal on Computing*, 31(3):576–592.

Lines, J., Davis, L. M., Hills, J., and Bagnall, A. (2012). A Shapelet Transform for Time Series Classification. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–297.

MacKay, D. J. et al. (1998). Introduction to Gaussian Processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166.

McFee, B. and Lanckriet, G. R. (2010). Metric Learning to Rank. In *International Conference on Machine Learning*.

McGinnis, L. F. and Rose, O. (2017). History and Perspective of Simulation in Manufacturing. In *Proceedings of the 2017 Winter Simulation Conference*, pages 385–397. Institute of Electrical and Electronics Engineers (IEEE).

Medico, R., Ruyssinck, J., Deschrijver, D., and Dhaene, T. (2021). Learning Multivariate Shapelets with Multi-Layer Neural Networks for Interpretable Time-Series Classification. *Advances in Data Analysis and Classification*, pages 1–26.

Mehta, S. V. and Uzsoy, R. (1999). Predictable Scheduling of a Single Machine subject to Breakdowns. *International Journal of Computer Integrated Manufacturing*, 12(1):15–38.

Mood, A. M., Graybill, F. A., Boes, D. C., et al. (1974). Introduction to the Theory of Statistics. *International Editions, McGraw-Hill.*

Morgan, L. E. and Barton, R. R. (2022). Fourier Trajectory Analysis for System Discrimination. *European Journal of Operational Research*, 296(1):203–217.

Moss, H. B., Leslie, D. S., and Rayson, P. (2018). Using J-K-Fold Cross Validation to Reduce Variance when Tuning NLP Models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2978–2989. Association for Computational Linguistics.

Mueen, A., Keogh, E., and Young, N. (2011). Logical-Shapelets: An Expressive Primitive for Time Series Classification. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1154–1162.

Nance, R. E. and Sargent, R. G. (2002). Perspectives on the Evolution of Simulation. *Operations Research*, 50(1):161–172.

Nelson, B. L. (2016). 'Some Tactical Problems in Digital Simulation' for the Next 10 Years. *Journal of Simulation*, 10(1):2–11.

Nelson, B. L. and Pei, L. (2021). *Foundations and Methods of Stochastic Simulation: A First Course, 2nd edition.* Springer Nature.

Nguyen, N. and Guo, Y. (2008). Metric Learning: A Support Vector Approach. In *ECML PKDD 2008: Machine Learning and Knowledge Discovery in Databases*, pages 125–136. Springer.

Ouyang, H. and Nelson, B. L. (2017). Simulation-based Predictive Analytics for Dynamic Queueing Systems. In *Proceedings of the 2017 Winter Simulation Conference*, page 1716–1727. Institute of Electrical and Electronics Engineers (IEEE).

Parameswaran, S. and Weinberger, K. Q. (2010). Large Margin Multi-task Metric Learning. *Advances in Neural Information Processing Systems*, 23.

Parmar, D., Morgan, L., Titman, A., Williams, R., and Sanchez, S. (2021). A Two Stage Algorithm for Guiding Data Collection towards Minimising Input Uncertainty. In *Proceedings of the Operational Research Society Simulation Workshop 2021*. Operational Research Society.

Pasupathy, R., Singham, D. I., and Yeh, Y. (2022). Overlapping Batch Confidence Regions on the Steady-State Quantile Vector. In *Proceedings of the 2022 Winter Simulation Conference*, pages 25–36. Institute of Electrical and Electronics Engineers (IEEE).

Patri, O. P., Kannan, R., Panangadan, A. V., and Prasanna, V. K. (2015). Multivariate Time Series Classification using Inter-Leaved Shapelets. In *NIPS 2015 Time Series Workshop*.

Pearson, K. (1901). LIII. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Peltonen, J. and Kaski, S. (2005). Discriminative Components of Data. *IEEE Transactions on Neural Networks*, 16(1):68–83.

Qi, G.-J., Tang, J., Zha, Z.-J., Chua, T.-S., and Zhang, H.-J. (2009). An Efficient Sparse Metric Learning in High-dimensional Space via l_1-Penalized Log-determinant Regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 841–848.

Rakthanmanon, T. and Keogh, E. (2013). Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. Society for Industrial and Applied Mathematics (SIAM).

Raychaudhuri, D. S., Grabocka, J., and Schmidt-Thieme, L. (2017). Channel Masking for Multivariate Time Series Shapelets. https://arxiv.org/pdf/1711.00812.pdf, accessed 12th June 2023.

Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3):471–501.

Ruiz, A. P., Flynn, M., Large, J., Middlehurst, M., and Bagnall, A. (2021). The Great Multivariate Time Series Classification Bake Off: A Review and Experimental Evaluation of Recent Algorithmic Advances. *Data Mining and Knowledge Discovery*, 35(2):401–449.

Sargent, R. G. (2010). Verification and Validation of Simulation Models. In *Proceedings of the 2010 Winter Simulation Conference*, pages 166–183. Institute of Electrical and Electronics Engineers (IEEE).

Sargent, R. G., Goldsman, D. M., and Yaacoub, T. (2016). A Tutorial on the Operational Validation of Simulation Models. In *Proceedings of the 2016 Winter Simulation Conference*, pages 163–177. Institute of Electrical and Electronics Engineers (IEEE).

Scheffe, H. (1999). *The Analysis of Variance*, volume 72. John Wiley & Sons.

Schriber, T. J. and Andrews, R. W. (1984). ARMA-based Confidence Intervals for Simulation Output Analysis. *American Journal of Mathematical and Management Sciences*, 4(3-4):345–373.

Schultz, M. and Joachims, T. (2003). Learning a Distance Metric from Relative Comparisons. *Advances in Neural Information Processing Systems*, 16.

Shah, M., Grabocka, J., Schilling, N., Wistuba, M., and Schmidt-Thieme, L. (2016). Learning DTW-Shapelets for Time-Series Classification. In *Proceedings of the 3rd IKDD Conference on Data Science, 2016*, pages 1–8.

Shajina, T. and Sivakumar, P. B. (2012). Human Gait Recognition and Classification using Time Series Shapelets. In *2012 International Conference on Advances in Computing and Communications*, pages 31–34. Institute of Electrical and Electronics Engineers (IEEE).

Shalev-Shwartz, S., Singer, Y., and Ng, A. Y. (2004). Online and Batch Learning of Pseudo-Metrics. In *Proceedings of the 21st International Conference on Machine Learning*, page 94.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2021). *Lectures on Stochastic Programming: Modeling and Theory, 3rd edition.* Society for Industrial and Applied Mathematics (SIAM).

Shor, N., Kiwiel, K. C., and Ruszcayski, A. (1985). *Minimization Methods for Non-differentiable Functions*. Springer-Verlag.

Smith, J. S. and Nelson, B. L. (2015). Estimating and Interpreting the Waiting Time for Customers Arriving to a Non-Stationary Queueing System. In *Proceedings of the 2015 Winter Simulation Conference*, pages 2610–2621. Institute of Electrical and Electronics Engineers (IEEE).

Smyth, P. (1996). Clustering Sequences with Hidden Markov Models. *Advances in Neural Information Processing Systems*, 9.

Song, E. and Nelson, B. L. (2015). Quickly Assessing Contributions to Input Uncertainty. *IIE Transactions*, 47(9):893–909.

Suárez, J. L., García, S., and Herrera, F. (2021). A Tutorial on Distance Metric Learning: Mathematical Foundations, Algorithms, Experimental Analysis, Prospects and Challenges. *Neurocomputing*, 425:300–322.

Suo, Q., Ma, F., Yuan, Y., Huai, M., Zhong, W., Gao, J., and Zhang, A. (2018). Deep Patient Similarity Learning for Personalized Healthcare. *IEEE Transactions on NanoBioscience*, 17(3):219–227.

Tarlow, D., Swersky, K., Charlin, L., Sutskever, I., and Zemel, R. (2013). Stochastic k-Neighborhood Selection for Supervised and Unsupervised Learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 199–207. Proceedings of Machine Learning Research (PMLR).

Torresani, L. and Lee, K.-c. (2006). Large Margin Component Analysis. In *NIPS'06: Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 1385–1392. MIT Press.

Urquhart, M., Ljungskog, E., and Sebben, S. (2020). Surrogate-based Optimisation using Adaptively Scaled Radial Basis Functions. *Applied Soft Computing*, 88.

Vaughan, J. W. (2017). Making Better Use of the Crowd: How Crowdsourcing can Advance Machine Learning Research. *Journal of Machine Learning Research*, 18(1):7026–7071.

Wang, D. and Tan, X. (2014). Robust Distance Metric Learning in the Presence of Label Noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Wang, J., Woznica, A., and Kalousis, A. (2012). Learning Neighborhoods for Metric Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 223–236. Springer.

Weinberger, K. Q. and Saul, L. K. (2008). Fast Solvers and Efficient Implementations for Distance Metric Learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1160–1167.

Weinberger, K. Q. and Saul, L. K. (2009). Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*, 10(9):207–244.

Weinberger, K. Q. and Tesauro, G. (2007). Metric Learning for Kernel Regression. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pages 612–619. Proceedings of Machine Learning Research (PMLR).

Wu, X. and Barton, R. R. (2016). Fourier Trajectory Analysis for Identifying System Congestion. In *Proceedings of the 2016 Winter Simulation Conference*, pages 401–412. Institute of Electrical and Electronics Engineers (IEEE).

Xie, P., Wu, W., Zhu, Y., and Xing, E. (2018). Orthogonality-promoting Distance Metric Learning: Convex Relaxation and Theoretical Analysis. In *International Conference on Machine Learning*, pages 5403–5412. Proceedings of Machine Learning Research (PMLR).

Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2002). Distance Metric Learning, with Application to Clustering with Side-Information. *Advances in Neural Information Processing Systems*, 15:521–528.

Yang, P., Huang, K., and Liu, C.-L. (2011). Multi-task Low-rank Metric Learning based on Common Subspace. In *18th International Conference on Neural Information Processing*, pages 151–159. Springer Berlin Heidelberg.

Yang, P., Huang, K., and Liu, C.-L. (2013). Geometry Preserving Multi-task Metric Learning. *Machine learning*, 92:133–175.

Yang, X. (2020). *Essays on Distance Metric Learning*. PhD thesis, UCL (University College London).

Yao, B., Zhao, Z., and Liu, K. (2014). Metric Learning with Trace-Norm Regularization for Person Re-identification. In *2014 IEEE International Conference on Image Processing*, pages 2442–2446. Institute of Electrical and Electronics Engineers (IEEE).

Ye, L. and Keogh, E. (2011). Time Series Shapelets: A Novel Technique that allows Accurate, Interpretable and Fast Classification. *Data Mining and Knowledge Discovery*, 22(1):149–182.

Yuan, M. and Nelson, B. L. (1993). Multiple Comparisons with the Best for Steady-State Simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 3(1):66–79.

Zalewski, W., Silva, F., Maletzke, A. G., and Ferrero, C. A. (2016). Exploring Shapelet Transformation for Time Series Classification in Decision Trees. *Knowledge-Based Systems*, 112:80–91.

Zhang, N. and Sun, S. (2022). Multiview Unsupervised Shapelet Learning for Multivariate Time Series Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zorko, A., Frühwirth, M., Goswami, N., Moser, M., and Levnajić, Z. (2020). Heart Rhythm Analyzed via Shapelets Distinguishes Sleep from Awake. *Frontiers in Physiology*, 10:1554.