EURECOM
Sophia Antipolis

SORBONNE UNIVERSITÉ

PHD THESIS

In Partial Fulfilment of the Requirements for the
Degree of Doctor of Philosophy from Sorbonne University
Specialisation: Data Science

Comparing High-Dimensional Data: Interpretable Two-Sample Testing by Variable Selection

KENSUKE MITSUZAWA

Defended on 02/12/2024 before a committee composed of:

Reviewer	**Mamoru Komachi**, Hitotsubashi University, Japan
Reviewer	**Damien Garreau**, Julius-Maximilians-Universität Würzburg, Germany
Advisor	**Paolo Papotti**, EURECOM, France
Co-Advisor	**Motonobu Kanagawa**, EURECOM, France
Examiner	**Paola Goatin**, INRIA, France
Examiner (Jury President)	**Jérôme Härri**, EURECOM, France

"Errors using inadequate data are much less than those using no data at all."
Charles Babbage

Abstract

High-dimensional data is everywhere, and the amount and quality of this data keep increasing, even though analysing it remains time-consuming. Two-sample testing is a common method for comparing two datasets, but it often does not provide enough information for humans to understand, intuit, and comprehend the results.
In this thesis, we investigate variable selection for comparing a pair of high-dimensional datasets, enabling humans to gain insights without time-consuming analysis work. The variable selection is performed during two-sample testing and identifies the variables (or dimensions) responsible for the discrepancies between the two distributions.
We focus on Maximum Mean Discrepancy (MMD), which is a distance metric between probability distributions, and an optimisation problem of its estimator. This problem optimises the Automatic Relevance Detection (ARD) weights in a kernel function. The kernel function is defined for individual variables to maximise the power of the MMD-based test. We extend this optimisation problem into the variable selection task by adding sparse regularisation. Since this regularisation term requires an arbitrary parameter, we develop algorithms to find appropriate regularisation parameters.
Furthermore, we address a variable selection problem with a set of high-dimensional time-series data. Our main aim is to identify important variables that reflect the differences between two probability distributions. To accomplish this, we have devised an algorithm for selecting variables from pairs of time-series data. The algorithm divides a set of time steps into a certain number of buckets and applies a variable selection method for each bucket. We empirically show that MMD-based variable selection methods are a suitable approach for this task.
Finally, we demonstrate that a model parameter calibration task, which involves estimating a suitable parameter for a black-box model (e.g., an intractable simulation model), can be conducted in a human-in-the-loop style by introducing the MMD-based variable selection method. The calibration method employs KernelABC of which distance metric is an optimised MMD estimator.
Overall, this work provides advancements in variable selection methods, significantly improving the interpretability and efficiency of high-dimensional data analysis in various applications.

Résumé

Les données de haute dimension sont omniprésentes, et leur quantité ainsi que leur qualité continuent d'augmenter, bien que leur analyse reste chronophage. Le «Two-Sample Testing» est une méthode courante pour comparer deux ensembles de données, mais il ne fournit souvent pas suffisamment d'informations pour que les humains puissent comprendre et interpréter les résultats de la comparison par le Two-Sample Testing.
Cette thèse étudie la sélection de variables pour comparer une paire de données de haute dimension, permettant ainsi aux humains d'obtenir un aperçu sans avoir à effectuer des travaux d'analyse longs et fastidieux. La sélection de variables est réalisée lors du Two-Sample Testing et permet d'identifier les variables (ou dimensions) responsables des écarts entre les deux distributions de probabilités.
Cette thèse porte sur «Maximum Mean Discrepancy» (MMD), une métrique de distance entre deux distributions de probabilités, ainsi que sur un problème d'optimisation de MMD estimateur. Ce problème optimise les paramètres de «Automatic Relevance Detection» (ARD) dans une «Kernel fonction». La fonction objective vise à maximiser l'approximation de la «Test Power» du test basé sur la MMD. Nous étendons ce problème d'optimisation à la sélection de variables (sélection de caractéristiques) en ajoutant une «sparse régularisation». Étant donné que cette régularisation nécessite un hyperparamètre arbitraire, nous développons des algorithmes permettant de déterminer automatiquement les paramètres de régularisation optimaux.
De plus, nous abordons un problème de sélection de variables avec un ensemble de données temporelles de haute dimension. Le principal objectif est d'identifier les variables importantes dans une paire de séries temporelles, qui reflètent les différences entre deux distributions de probabilité. À cette fin, nous avons développé un algorithme de sélection de variables pour une paire de séries temporelles.
Enfin, nous démontrons qu'une calibration de paramètres de modèle, qui consiste à estimer un paramètre adapté à un modèle «Black-Box»(par exemple, un modèle de simulation intractable), peut être réalisée avec l'intervention humaine en utilisant la méthode de sélection de variables basée sur la MMD. La calibration de modèle avec l'intervention humaine est une approche efficace lorsque le modèle Black-Box nécessite des coûts computationnels élevés, que ce soit en termes de puissance de calcul ou de temps.

Acknowledgements

I would like to thank the many people who helped through comments and discussions during the writing of the various papers composing this thesis.

Firstly, I would like to sincerely thank my advisors Motonobu Kanagawa and Paolo Papotti, for having strongly wanted me to start this PhD, for having directed and guided my research, for having constantly given value and importance to my work, and for having shared with me goals and responsibilities.

I would like to thank my research collaborator, Margherita Grossi and Stefano Bortoli in the Intelligent Cloud Technologies Laboratory at Huawei Munich Research Center, for having granted me previous advices from the industrial perspective and for allowing me have had the internship opportunity.

Sophia Antipolis, Spetember 2024 ***Kensuke Mitsuzawa***

Contents

List of Figures

List of Tables

Mathematical Notation

$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
D	Dimension size of a vector
P	A probability distribution
Q	Another probability distribution
$\boldsymbol{x} \in \mathbb{R}^D$	A datapoint sampled from P
$\boldsymbol{y} \in \mathbb{R}^D$	A datapoint sampled from Q
$n, m \in \mathbb{N}$	The number of samples for $\boldsymbol{x}, \boldsymbol{y}$, respectively
$\boldsymbol{X} := \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$	A dataset consisted of i.i.d. (independent and identically distributed) sampled $\boldsymbol{x}$
$\boldsymbol{Y} := \{\boldsymbol{y}_1, ..., \boldsymbol{y}_m\}$	A dataset consisted of i.i.d. sampled $\boldsymbol{y}$
$H_0 := P = Q$	The null hypothesis of hypothesis or two-sample testing
$H_1 := P \neq Q$	The alternative hypothesis of hypothesis or two-sample testing

Acronyms and Abbreviations

TST	Two-Sample Testing
MMD	Maximum Mean Discrepancy
$i.i.d$	independent and identically distributed

Chapter 1

Introduction

Since the 1990s, high-dimensional data has been ubiquitous not only in scientific communities but also in society (Fan and Li, 2006). Data comparisons are fundamental parts of data analysis when humans need to make decisions based on the results or derive insights from them. Two-sample testing is an objective method for comparing pairs of high-dimensional data. However, its binary outcome often provides insufficient information for humans to effectively handle high-dimensional data. The main goal of this thesis is to develop such practical analysis tools that provide sufficient information for humans for their analysis works.

The central focus of this thesis is on **high-dimensional data comparison**, with a specific emphasis on human interpretability. This is achieved through Two-Sample Testing (TST) combined with variable selection using Maximum Mean Discrepancy (MMD).

1.1 High-Dimensional Data and their Comparison

Data comparisons are essential for data analysis when making decisions or deriving insights. In our society, we frequently come across situations where we compare sets of high-dimensional data[1]. For example, image data, where each pixel represents a variable[2], is a typical high-dimensional data, with $4,096$ variables when the image size in pixel is 64×64 (Deng et al., 2009). As another example, tabular data, such as time-series or transactional data, often comprises many variables, represented by columns (Dau et al., 2019).

[1] There is no clear criterion for distinguishing "high" dimensionality. For instance, Liu et al. (2017) consider dimensionality to be high when the number of variables exceeds three, while the bioinformatics field deals with data containing more than 1,000 variables and refers to it as "high dimension." High dimensionality is generally understood to occur when the *Curse of Dimensionality* impacts a target application, leading to degraded performance or efficiency in machine learning or statistical applications.

[2] The field of machine learning often uses the term "feature" instead of "variable". In this thesis, we consistently use the term "variable".

Data comparison involves checking the compatibility or equivalence of data for integrating high-dimensional data (Shi et al., 2005). Before merging two datasets, a data operator must confirm that they are the same, compatible, or similar. However, manually comparing high-dimensional data, such as DNA microarray data that can have up to 12,600 dimensions, is difficult and time-consuming (Borgwardt et al., 2006).

The comparison of interest could involve probability distributions, such as generative models (Goodfellow et al., 2020) that produce high-dimensional data. However, in practice, the shapes of probability distributions are unknown in most cases; therefore, it is impossible to compare these distributions directly. Instead, we sample high-dimensional data from these distributions and then compare the samples. For instance, we might want to determine whether a surrogate model is compatible or equivalent to the distribution it approximates. Similar to the data integration example, the manual comparison is hard and time-consuming work.

Given these challenges, there is a need for effective methods that bridge the gap between theoretical distribution comparisons and practical applications of high-dimensional data. Indeed, high-dimensional data poses challenges for humans due to several reasons. Firstly, humans struggle to manage a large number of variables (Liu et al., 2017). Secondly, when dealing with high-dimensional data, the number of variable combinations grows exponentially. This is because variables may have interdependencies, leading to a vast number of variable combinations that become difficult for humans to manage.

Therefore, our goal is to offer a solution that empowers humans to make informed decisions and gain valuable insights. To achieve this, two essential requirements must be met;

1. Helping determine whether a comparison pair is the same (similar) or not.[3]
2. Providing humans with evidence, factors, or reasons for the difference, when the comparison pair is not the same.

1.2 High-Dimensional Data Comparisons by Two Sample Testing

Two-Sample Testing (TST) is the preferred objective method for comparing pairs of high-dimensional datasets, as it requires fewer subjective decisions compared to other approaches. Graphical comparisons following dimensionality reduction methods such as PCA or t-SNE (Cutura et al., 2020) rely on human interpretation. While plotting histograms has been a long-standing method for data comparison (Thas, 2010), it is limited to univariate analysis and is not suitable for high-dimensional data.

[3]We assume the comparison pair does not necessarily have to be the same. Yet, the pair should be similar such that people, who handle data, commonly agree on the similarity.

In the context of TST, we translate "same (similar)" or "not" into *acceptance* and *rejection* of the null hypothesis H_0. The null hypothesis, in this case, asserts that the two samples are drawn from the same underlying distribution. Therefore, accepting the null hypothesis suggests that the distributions are the same or pretty similar, whereas rejecting it indicates a significant difference between them. In TST settings, there are two probability distributions P, Q of which shapes are unknown; however, we are able to sample datasets $\boldsymbol{X}, \boldsymbol{Y}$ from P, Q, respectively. The null hypothesis H_0 represents $P = Q$, and the alternative hypothesis H_1 is for $P \neq Q$. When P and Q are different, TST rejects H_0, otherwise the acceptance of the H_0.

Historically, TST of high-dimensional data has been challenging since the 1990s notably in computational biology, health studies, financial engineering, and risk management, which is driven by the rapid growth in information technology and data collection capabilities (Fan and Li, 2006). Maximum Mean Discrepancy (MMD) (Gretton et al., 2012a) has become a well-known and often-used de facto standard in recent studies (Kübler et al., 2022a; Kübler et al., 2022b). It is recognised as a robust distance metric in high-dimensional spaces, particularly effective when the available datapoints are limited (Gretton et al., 2012a).

By applying MMD to high dimensional TST, we can meet the first requirement mentioned earlier: helping determine whether a comparison pair is the same (similar) or not. However, the second requirement, providing humans with evidence, factors, or reasons for the difference, remains unmet. In the next section, we explain the limitation of the high dimensional TST.

1.3 Challenges in Two-Sample Testing and Proposed Solution

Although TST is an objective method for comparisons, the binary outcome of TST (acceptance or rejection of the H_0) is not sufficient for humans in handling high-dimensional data. When the H_0 is rejected, there is no further guidance on what to do next.

TST can provide various statistics, including the p-value. However, this information alone may not be enough for humans. The p-value simply indicates the likelihood of observing a test statistic under the assumption that the two samples come from the same distribution. As a result, it does not offer humans explanations for differences in high-dimensional data[4].

We propose a solution by manual analysis supported by variable selection. This manual analysis fulfils the second condition, which is to provide humans with evidence, factors, or reasons for the difference. Humans can review the selected variables and make further decisions. There could be several reasons for the rejection of the H_0; discrepancies may exist in

[4] Foremost, we should avoid relying on the p-value alone. The American Statistical Association warned of overusing the p-value in scientific and business fields, saying "Scientific conclusions and business or policy decisions should not be based only on whether a p-value passes a specific threshold." (Wasserstein and Lazar, 2016).

specific areas of a data domain that are not the main focus of the comparison, or the rejection could result from errors in data preprocessing.

Two-sample testing through hypothesis testing can address the first requirement of determining whether a comparison pair is similar or not. However, the binary outcome does not satisfy the second requirement of providing humans with evidence, factors, or reasons for the difference, which can be achieved through *variable selection*.

In the literature, there have been various efforts to realise variable selection based on two-sample testing. Here, we briefly introduce some of these works. Yamada et al. (2019) and Lim et al. (2020) proposed this combination by applying the test independently to each dimension. Although their methods can meet the two requirements, they are unable to account for interdependencies among variables. Thus, the selected variables may become redundant if one probability distribution, P, has correlations among variables while the counterpart, Q, does not. As discussed earlier in Section 1.1, high-dimensional data usually involves interdependencies among variables, requiring additional analysis to sharply detect relevant variables. Therefore, a preferred variable selection method should consider the interdependency structure among variables to ensure accurate selection. Another issue is that the number of selected variables is a hyperparameter, which is often unknown in practice. Wang et al. (2023) proposed another approach to variable selection based on TST. However, this method shares the same issue regarding the hyperparameter of the number of selected variables, requiring humans to have some prior knowledge about the two distributions. We will review these works later in Section 3.1.

Our variable selection method is based on TST and selects variables by solving an optimisation problem that takes into account interdependencies among variables. Crucially, it does not require a hyperparameter for the number of selected variables. As a result, our proposal meets both requirements, sharply selecting relevant variables without the need to know the exact number of selected variables in advance. The proposed method is based on optimising an MMD estimator to maximise the ability to distinguish a pair of given samples. This maximising problem optimises ARD weights in a kernel function, and the variable selection method selects a set of notably weighted variables. Our proposal consists of two parts, 1) introducing a regularisation term to the optimisation problem of the MMD estimator, 2) developing an automated method for selecting the optimal regularisation parameter.

1.4 Application Examples of Our Variable Selection

Our variable selection method is broadly applicable to a variety of scenarios requiring the comparison of high-dimensional data. We provide two application examples to illustrate its utility: data analysis and model validation for model development.

1.4.1 Variable Selection for Data Analysis

Demonstration Showcase: Image Analysis

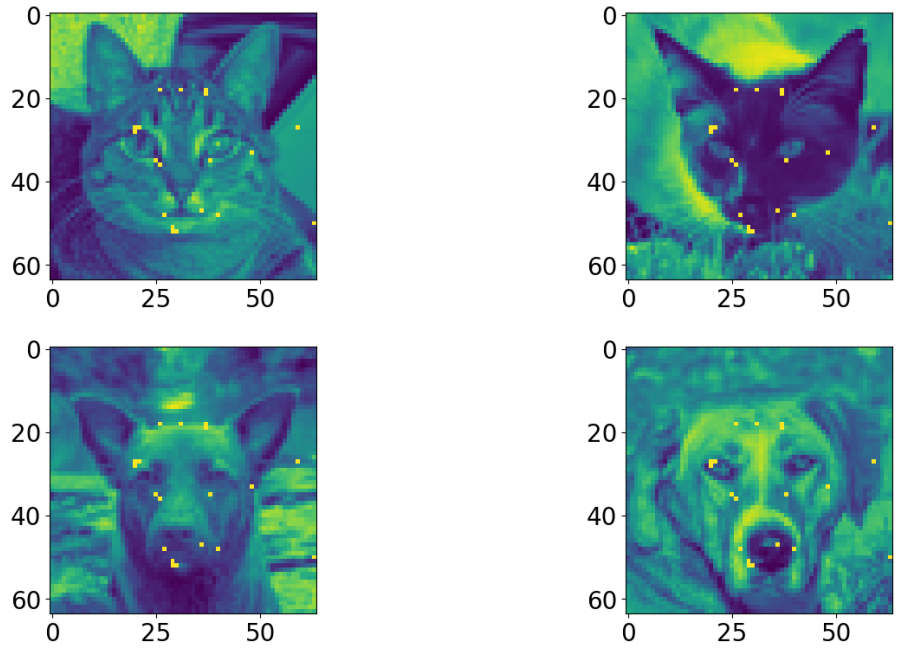


Figure 1.1: Example of discovered variables for the data analysis by comparisons. Two datasets $\boldsymbol{X}, \boldsymbol{Y}$ are sets of images of cats' and dogs' faces. The highlighted yellow dots depict the variables selected by the proposed method in this thesis work. Details are in Section 3.5.4.

The demonstration case represents the data analysis by comparisons of images. The full description is in Section 3.5.4, and we briefly introduce the demonstration here.

The AFHQ dataset (Choi et al., 2020) contains a collection of face images of cats and dogs. Comparing images of cats and dogs is a general case that does not require specific background knowledge to interpret the variable detection results. More practical applications of image comparison are, for example, medical image analysis (Litjens et al., 2017).

We assume that two probability distributions P, Q generate these images. The face images are greyscale and have a resolution of 64×64 pixels, resulting in samples of 4,096 variables. Although the differences are visually apparent, we aim to identify the variables (pixels) that contribute to these differences. This approach can be useful in other applications, such as medical imaging and anomaly detection, where pinpointing the exact pixels that differentiate between classes is crucial for diagnosis and accurate analysis.

We apply the variable selection method to the image datasets to identify the variables that influence the rejection of the H_0. The yellow dots in Figure 1.1 represent these variables, highlighting the variations in the shapes of eyes and noses.

1.4.2 Variable Selection for the Model Validation

A model is designed to represent a system or a process in the real-world (Sargent, 2010). A model may improperly approximate the real-world system if errors lie in model logic designs or inappropriately set parameters[5]. Therefore, we need to validate a model before deploying it to applications. Model validation is a procedure to confirm that model outcomes fall within the acceptable range of accuracy expected by the application (Sargent, 2010)[6]. In the context of two-sample testing, model validation involves evaluating the null hypothesis (H_0) where $\mathscr{M}_P$ represents the model and Q is the counterpart probability distribution.

In practice, tests will reject the H_0 if the sample size is large enough, even for practically insignificant differences, with some small vanishing probability of the rejection being a result of random chance alone. As an aphorism "all models are wrong" (Box, 1979), it is inherently impossible to perfectly model a real-world system. Additionally, as discussed in Section 1.3, there are numerous reasons for the rejection of the H_0, especially dealing with high-dimensional outcomes. Consequently, model validators spend a significant amount of time analysing and comparing the model outcomes with their counterpart.

A solution is manual validation supported by the variable selection. As Box (1979) said "but some (models) are useful", some models can achieve acceptable accuracy and validity for specific applications. The information derived from these useful models can be highly beneficial. Our variable selection method accurately identifies the variables contributing to the rejection of the null hypothesis (H_0), allowing humans to review these variables and make informed decisions about model validation.

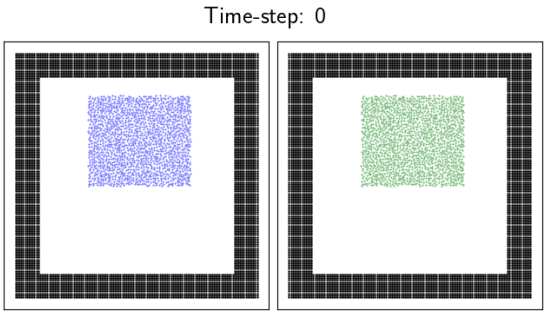
Demonstration Showcase: Model Validation of Fluid Particle Surrogate Model

The demonstration is focused on validating a surrogate model for fluid-particle simulations. The full description is in Section 4.5, and we briefly introduce the demonstration here. A physic-model based simulator (Bender et al., n.a) is prepared as $\mathscr{M}_P$, and $\mathscr{M}_Q$ is a surrogate model approximated by a deep neural network (Prantl et al., 2022). Given that the model $\mathscr{M}_P$ involves high computational costs, the surrogate model $\mathscr{M}_Q$ is expected to offer a practical solution for rapid predictions. Our goal is to establish a well-approximated surrogate model $\mathscr{M}_Q$ that is suitable for the application, and then validate $\mathscr{M}_Q$ by comparing it with $\mathscr{M}_P$.

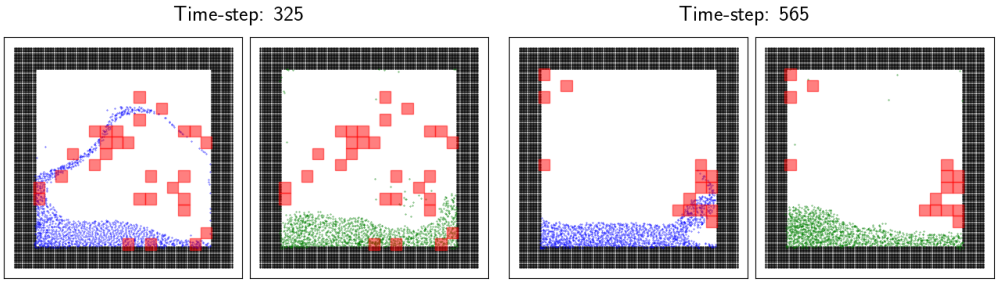
We sample a pair of model outcomes, specifically the counts of fluid particles, from $\mathscr{M}_P$ and $\mathscr{M}_Q$ and apply our variable selection method. Figure 1.2 illustrates outcomes from $\mathscr{M}_P$ (blue

[5] A certain model may be without parameters. This thesis does not distinguish between models with or without parameters.

[6] Sargent (2010) distinguishes model validation from *model verification*. The model verification handles errors in model logic designing or in implementation. The model validation aims at confirming appropriate parameters. This thesis regards model validation, including model verification.



(a) At $t = 0$.



(b) At $t = 325$.

(c) At $t = 565$.

Figure 1.2: A pair of model outcomes from $\mathscr{M}_P$ (blue particles) and $\mathscr{M}_Q$ (green particles). Two outcomes start at time $t = 0$ and end at time $t = 599$. The red boxes are detected variables by the variable selection method.

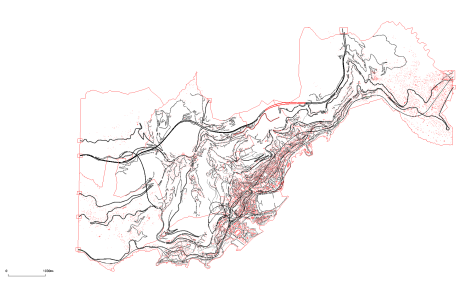
particles) and $\mathscr{M}_Q$ (green particles) at steps $t = 0$, $t = 325$ and $t = 565$. The red boxes are the variables detected by the variable selection method.

At $t = 325, 565$, the visualisation of blue and green particles reveals major differences between the two models. However, the proposed variable selection method revealed slight differences between the two models, such as particles located in the upper-left corner at time $t = 565$. A few green particles (by the surrogate model) adhere to the upper-left corner, while no corresponding blue particles (by the physic-model-based simulator) are present.

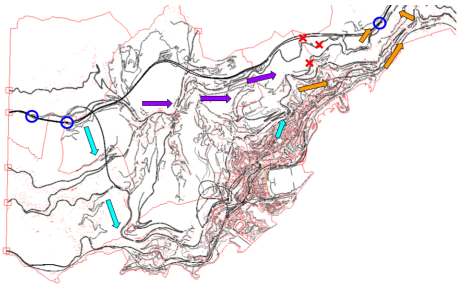
Visual representation alone cannot adequately highlight subtle differences, leading humans to potentially overlook significant implications. By combining the visual representation and variable selection results, the visualisation can offer an intuitive yet powerful interpretation for humans.

Demonstration Showcase: Model Validation of Macroscopic Traffic Simulation

This demonstration focuses on the model validation of a realistic macroscopic traffic simulation scenario in Monaco (Codeca and Härri, 2018). The full description is provided in



(a) A simulation model $\mathscr{M}_P$.



(b) The perturbed model $\mathscr{M}_Q$.

(c) Selected variables caused by the perturbation. The roads affected by the red "X" blocks in the model $\mathscr{M}_Q$.

Figure 1.3: (Left) A simulation model $\mathscr{M}_P$. (Right) The counterpart perturbed model $\mathscr{M}_Q$. Red "X" marks are the blocked roads. Colour arrows are vehicles' rerouting paths caused by the perturbation. Each colour represents a rerouting group. (Bottom) Variable selection results by the proposed method. The green-colour lines are selected variables that are reasons and factors for rejecting the H_0.

Section 4.6, but we briefly introduce the demonstration here. Two simulation models are prepared: $\mathscr{M}_P$ represents the model, while $\mathscr{M}_Q$ represents a perturbed model created by blocking some major roads in the study area. In this scenario, the model validation results in the rejection of the null hypothesis (H_0). However, our primary interest lies in identifying the variables that contribute to this rejection.

Figure 1.3 (left, right) illustrates differences between the two models. The simulation model $\mathscr{M}_Q$ represents the perturbed model, where the red "X" marks and the coloured arrows indicate the blocked roads and the rerouting paths of vehicles, respectively, caused by the perturbation (each colour represents a rerouting group)[7]. The model $\mathscr{M}_P$ is the counterpart without the perturbation. Apart from the perturbation, all configurations are the same be-

[7]The rerouting groups are identified manually, without using variable selection methods.

tween these two simulation models. We sampled traffic count values per road from both $\mathscr{M}_P$ and $\mathscr{M}_Q$ and applied our variable selection method. The bottom figure shows the result, with selected variables highlighted in green colour. Among 4,404 variables (roads), the selected green lines represent the variables that are reasons and factors contributing to the rejection of the null hypothesis (H_0). These roads are primarily impacted by the roadblocks in the model $\mathscr{M}_Q$. Without the variable selection method, manual analysis would be time-consuming and challenging. With the selected variables, a model validator can focus on variables with high priorities in comparing the two models.

1.5 Thesis Contributions and Structure

This thesis contributes to the field of comparing pairs of high-dimensional data. An effective comparison tool for humans must meet the two requirements clarified in Section 1.1. The analysis tool developed in this thesis not only meets these two requirements but also offers the following three contributions:

Automated Variable Selection. We propose an automated and data-driven variable selection method that does not require specifying the number of selected variables as a hyperparameter, addressing a limitation present in related works (Yamada et al., 2019; Lim et al., 2020; Wang et al., 2023). Our proposed method is based on a regularised optimisation problem of an MMD estimator. Since the regularised optimisation problem demands an appropriate regularisation parameter, we developed automated methods for selecting the optimal regularization parameter. Based on the optimisation result, we select variables. We empirically demonstrate that our automated methods are more accurate than existing variable selection methods. The proposed methods are detailed in Chapter 3, and these findings were previously presented in (Mitsuzawa et al., 2023).

Automated Variable Selection for High Dimensional Time-Series Data. Using the proposed methods, we develop a variable selection framework for comparing pairs of high-dimensional time-series data. In practical validation scenarios, there is a demand for rigorously understanding a model's behaviour by focusing on a single pair or a few pairs of high-dimensional time-series, often spatial-temporal data. The key question addressed is: "Which variables are significantly different at certain time ranges?". We empirically demonstrate that the proposed framework successfully addresses this question. The framework is generic and allows for the use of any variable selection algorithms; however, our empirical results show that the proposed automated variable selection method is the most preferred choice. These studies are detailed in Chapter 4, and these findings were previously presented in (Mitsuzawa et al., 2024).

Automated Variable Selection for Human-in-Loop Parameter Calibration. We focus on the *black-box model calibration (parameter estimation)* problem, where the model outcomes are high-dimensional data and the black-box model requires significant computational resources, such as computational power or time. Human-in-the-loop model calibration offers a solution where manual analysis informs decisions on whether to continue or halt the calibration process. However, analysing high-dimensional data is time-consuming and labour-intensive. Thus we propose the human-in-loop model calibration supported by the automated validation selection. These contents are detailed in Chapter 5 and are based on research collaboration with an industrial partner. The proposed algorithm is part of preliminary research[8]; therefore, additional empirical results are needed, and further empirical validation is preferred.

Thesis Structure This thesis is organised as follows. Chapter 2 provides preliminaries on Maximum Mean Discrepancy (MMD) and Two Sample Testing (TST) (Gretton et al., 2012a), as well as methods for optimising an MMD estimator in a data-driven manner (Sutherland et al., 2017). Chapter 3 goes in-depth into the variable selection based on the optimisation of the MMD estimator. In Chapter 4, we report an extension of the variable selection method to compare a pair of high-dimensional time-series data. Chapter 5 shows our work on "Human-in-loop model calibration" by integrating the variable selection method into a parameter estimation task. Empirical assessments and analysis demonstrations are found at ends of each chapter. The appendix contains supplementary materials, including theoretical analysis, additional experiments, and details of data generation procedures for empirical assessments and demonstrations.

[8] Indeed, all work was completed during the author's first year

Chapter 2

Maximum Mean Discrepancy and Optimisation of a Maximum Mean Discrepancy Estimator

In Chapter 2, we will provide the necessary background and definition of Maximum Mean Discrepancy (MMD) and the optimisation of its estimator. Before diving into the specifics of MMD, we will briefly introduce the literature on high-dimensional two-sample testing (TST) in Section 2.1 to emphasise the advantages of using MMD for TST. Section 2.2 will introduce MMD, and the optimisation problem will be discussed in Section 2.3, which is connected to variable selection in Chapter 3.

We first introduce notations used in the following sections. Let P and Q probability distributions on the D-dimensional Euclidean space $\mathbb{R}^D$. Let $\mathbf{X} := \{X^1, \ldots, X^n\} \subset \mathbb{R}^D$ be a dataset of n i.i.d. (independently and identically distributed) sample vectors from P, and $\mathbf{Y} := \{Y^1, \ldots, Y^m\} \subset \mathbb{R}^D$ be a dataset of i.i.d. sample vectors of size m from Q. The task of TST is to test the null hypothesis $H_0 : P = Q$ against the alternative hypothesis $H_1 : P \neq Q$ using the two datasets $\mathbf{X}$ and $\mathbf{Y}$. For evaluating a TST method, a "test power" is often used. The test power is the probability of the test correctly rejecting the null hypothesis H_0 when the alternative hypothesis H_1 is true, where this probability is with respect to the random generation of the i.i.d. samples $\mathbf{X}$ and $\mathbf{Y}$. Thus, we interpret that a TST method with a greater test power is preferred.

2.1 Challenges of High-Dimensional Two-Sample Testing

To the best of our knowledge, there is no clear definition or common understanding of "high-dimension" in the context of TST. As a result, the term "high-dimension" may encompass and refer to "multivariate" and "large p, small n" situations. This thesis does not clearly distinguish these three terms and regards them all as "high-dimensional" settings. "Multivariate," has been used since the early days to denote data with multiple variables (Marinković, 2008). "Large p, small n" refers to situations where the number of variables p exceeds the number of observations n (West, 2003; Kosorok and Ma, 2007).[1] Lastly, "high dimensional" is often used in a similar context to "large p, small n" implying that while the data dimension is high, the sample size is relatively small (Chen and Qin, 2010). However, it does not necessarily meet the same criteria as "large p, small n."

Multivariate hypothesis testing dates back to Hotelling's T^2 test in 1931 (Hotelling, 1992), a multivariate extension of the Student's t-test (Student, 1992). Subsequently, various approaches were proposed for multivariate two-sample testing, and research attention shifted to high-dimensional data in the late 1990s and 2000s (Fan and Li, 2006). Understanding the evolution of high-dimensional TST and comparing with MMD is crucial for appreciating the advancements that our work builds upon. Below, we briefly introduce major approaches to treating multivariate and high-dimensional data.

Parametric Tests

The parametric test, especially Hotelling's T^2 statistic, was widely used for TST as late as 2014 (Biswas and Ghosh, 2014), although Bai and Saranadasa (1996) demonstrated and pointed out its limitations that is degrading test power with higher dimensionality. The degradation is primarily caused by the assumption in the test that the two Gaussian distributions must have the same covariance. In real-world high-dimensional settings, non-Gaussian data distributions are common, which makes the Gaussian assumption often unrealistic (Bono et al., 2017). To address this problem, Bai and Saranadasa (1996) proposed a new test statistic that does not rely on the assumption of Gaussian distributions. However, this approach still necessitates the assumption of homoscedasticity between the two covariances. This assumption of covariance homoscedasticity is once again impractical when applying TST to high-dimensional data, e.g. dataset shift adaptation (Quionero-Candela et al., 2009).

The test statistic proposed by Chen and Qin (2010) addressed the issue of covariance heteroscedasticity, specifically tailored for high-dimensional gene data. Instead of relying on the covariance matrix, the test statistic compares the sum of three inner products: two self-inner

[1] Some studies use the term "High-Dimension and Low-Sample-Size (HDLSS)" (Aoshima and Yata, 2018). However, this term does not appear to be mainstream.

products of $\boldsymbol{X}$, $\boldsymbol{Y}$, and an inner product between $\boldsymbol{X}$ and $\boldsymbol{Y}$. While this test offers more flexibility than previous methods, it is not robust in high-dimensional spaces characterised by a "spike" where some eigenvalues of the covariance matrix are significantly larger than others, as pointed out in (Aoshima and Yata, 2018).

When comparing with parametric tests, MMD does not require the assumption of underlying distributions. This property, often referred to as nonparametric and distribution free (Dodge, 2008), is a strong motivation for employing MMD as the test statistic.

Ranking Based Tests

Ranking-based tests do not require the assumption of underlying distributions, unlike parametric tests; therefore it falls into a category of non-parametric tests. Despite this advantage, nonparametric testing, including the ranking-based test, has a significant drawback: it generally yields lower test power compared to appropriately set parametric tests (Dodge, 2008). As a result, more samples are required to achieve a test power comparable to that of parametric methods. This increased sample requirement often discourages the use of the ranking-based test in situations involving high-dimensional data.

A well-known ranking-based test is *Mann-Whitney U test* (Mann and Whitney, 1947)[2], which ranks combined data points from both datasets and uses these ranks for statistical analysis. *Thompson's ranking test* (Thompson, 1990) is multivariate extension of the Mann-Whitney U test. However, the test is challenging to apply to high-dimensional data. That is because a monotone transformation of assigning ranks to datapoints is inappropriate to complex correlated variable structure in high-dimensional data, and therefore test power degrades (Bathke et al., 2008; Zhou and Chen, 2023).

In (Zhou and Chen, 2023)[3], the limitation of monotone transformation was identified and a graph-based ranking test was proposed. This test combines a distance between two datasets and graph-based representations. The sophisticated graph representation enables the handling of high-dimensional data and complex correlation structures.

The combination of graph-based and ranking-based tests is powerful, but MMD has its own advantages. Firstly, MMD outperforms the test power of the proposed method in certain distribution settings, for example when a multivariate Gaussian distribution has subtle differences in its locations between two probability distributions (Zhou and Chen, 2023, Setting I(b) in Table 1 and Setting III(b) in Table 3, respectively). Considering that the kernel function is not

[2] Mann-Whitney U test is also known as the *Wilcoxon rank-sum test*. There is a similarly named test, *Wilcoxon signed-rank test*, which is a *paired difference test* where the two samples must have a paired relationship.

[3] Zhou and Chen (2023) provides a comprehensive overview of recent developments in high-dimensional ranking tests.

well-tuned in the assessment of Zhou and Chen (2023), a well-tuned kernel function may lead to better test powers[4]. Additionally, MMD is able to be flexibly customised by swapping a kernel function of Eq. (2.2) while the proposal of Zhou and Chen (2023) allows for less customisation. For example, Gao et al. (2021) proposed a Deep-Neural-Network-based kernel function for detecting adversarial attacks.

Inter-Point Distance Tests

Inter-Point Distance (Rosenbaum, 2005) employs a metric of measuring distances between two datasets for a test statistic. Regarding measuring a distance between two datasets, MMD has a connection with the inter-point distance test.

A frequency-based inter-point test was proposed by (Friedman and Rafsky, 1979), whose test statistic counts frequencies of $\boldsymbol{X}, \boldsymbol{Y}$ label pairs in a minimal spanning tree. Variations of (Friedman and Rafsky, 1979) were presented in Henze (1988); Schilling (1986), which use the *K-nearest neighbour (K-NN)* instead of the minimal spanning tree. A test statistic of *Cross-match test* (Rosenbaum, 2005) is the frequency count of data pairs composed by datapoints of both datasets $\boldsymbol{X}, \boldsymbol{Y}$. The cross-match test may be effective in high-dimensional data settings, but the computational costs are prohibitively expensive, requiring $\mathcal{O}(m+n)^3$, where m, n represent the number of datapoints from $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.[5]

Biswas and Ghosh (2014) proposed a distance-based test statistic. Roughly say, this test statistic is based on inter-point Euclidean distances and the difference in the means of the inter-point distances between the two distributions.

Comparing the methods mentioned above with MMD, it is important to note that the Euclidean distance can be a limiting factor. Since the test statistics of these methods depend on the Euclidean distance, they may not be robust when such distance is not a suitable distance measure. For instance, in cases where probability distributions have manifold-shaped structures, the Euclidean distance is not suitable (Gu and Xu, 2006), while kernel-based methods are more robust (Sedghi et al., 2020). The use of MMD for the manifold-structure distributions is further discussed in Briol et al. (2019).

[4] Zhou and Chen (2023) used a `R` package `kerTests` (Song and Chen, 2023) whose MMD estimator is with a Gaussian kernel having a length scale computed by median heuristic (Garreau et al., 2018).

[5] Since methods of Friedman and Rafsky (1979); Henze (1988); Schilling (1986); Rosenbaum (2005) construct a graph-structure composed of datapoints, we can categorise them in *graph-based approach*, as Zhou and Chen (2023) does.

2.2 Maximum Mean Discrepancy (MMD)

MMD is a *distance metric* between probability distributions (Gretton et al., 2012a), thus enabling quantifying how the two distributions P and Q differ.

To define MMD, we need to introduce a *kernel function* $k(x, x')$ on $\mathbb{R}^D$, which defines the similarity between input vectors $x, x' \in \mathbb{R}^D$. More precisely, let $k : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ be a *positive semi-definite kernel*, examples including the linear kernel $k(x, x') = x^\top x'$, the polynomial kernel $k(x, x') = (x^\top x' + c)^\ell$, where $c \geq 0$ and $\ell \in \mathbb{N}$, the Gaussian kernel $k(x, x') = \exp(-a^2 \left\| x - x' \right\|^2)$ with $a > 0$, and the Laplace kernel $k(x, x') = \exp(-a \left\| x - x' \right\|)$ (Schölkopf and Smola, 2002).

For a given kernel k, the MMD between probability distributions P and Q is then defined as

$$\begin{aligned} \mathrm{MMD}_k^2(P, Q) := \mathbb{E}_{X,X' \sim P}[k(X, X')] + \mathbb{E}_{Y,Y' \sim Q}[k(Y, Y')] \\ - 2\mathbb{E}_{X \sim P,\ Y \sim Q}[k(X, Y)], \end{aligned} \tag{2.1}$$

where $X, X' \in \mathbb{R}^D$ are independent random vectors from P and $Y, Y' \in \mathbb{R}^D$ are those of Q and the expectation $\mathbb{E}$ is taken for the random vectors in the subscript.

Intuitively, the MMD compares the average similarities between random vectors *within* each distribution, i.e., $\mathbb{E}_{X,X' \sim P}[k(X, X')]$ and $\mathbb{E}_{Y,Y' \sim Q}[k(Y, Y')]$, with the average similarities between random vectors *across* the two distributions, i.e., $2\mathbb{E}_{X \sim P,\ Y \sim Q}[k(X, Y)]$. As such, we have $\mathrm{MMD}_k^2(P, Q) = 0$ if $P = Q$. It is known that we have $\mathrm{MMD}_k^2(P, Q) \geq 0$ for any P and Q. Moreover, if k has the property called *characteristic* (Fukumizu et al., 2007), we have $\mathrm{MMD}_k^2(P, Q) = 0$ if and *only if* $P = Q$. That is, whenever $P \neq Q$ we have $\mathrm{MMD}_k^2(P, Q) > 0$. Therefore, if the kernel k is characteristic and the MMD can be estimated from data, the estimated MMD can be used as a test statistic for TST. Among the kernels mentioned above, the Gaussian and Laplace kernels are characteristic (Sriperumbudur et al., 2010).

Given i.i.d. samples $\mathbf{X} = \{X^1, \ldots, X^n\} \overset{i.i.d.}{\sim} P$ and $\mathbf{Y} = \{Y^1, \ldots, Y^m\} \overset{i.i.d.}{\sim} Q$, one can estimate the MMD by replacing the expectations in Eq. (2.1) by the corresponding empirical averages:

$$\begin{aligned} \widehat{\mathrm{MMD}}_U^2(\mathbf{X}, \mathbf{Y}) := \frac{1}{n(n-1)} \sum_{1 \leq i \neq i' \leq n} k(X^i, X^{i'}) \\ + \frac{1}{m(m-1)} \sum_{1 \leq j \neq j' \leq m} k(Y^j, Y^{j'}) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(X^i, Y^j). \end{aligned} \tag{2.2}$$

This is an unbiased estimate of the MMD Eq. (2.1), and converges at the rate $O_p(\min(n, m)^{-1/2})$ as $n, m \to \infty$ provided that $\sup_{x \in \mathbb{R}^d} k(x, x) < \infty$ (Gretton et al., 2012a, Theorem 10). As mentioned, one can use the estimate Eq. (2.2) as a test statistic for TST, as a larger value of it implies that the two distributions would be more different, and a similar value implies they would be more similar.

2.3 Automatic Relevance Detection (ARD)

We now describe the approach of Sutherland et al. (2017) to optimise an MMD estimator for TST. In the MMD estimate Eq. (2.2), it uses the so-called *ARD kernel* defined as

$$k(x,y) = \exp\left(-\frac{1}{D}\sum_{d=1}^{D}\frac{a_d^2(x_d-y_d)^2}{\gamma_d^2}\right) \tag{2.3}$$
$$x := (x_1,\dots,x_D)^\top \in \mathbb{R}^D,\ \ y := (y_1,\dots,y_D)^\top \in \mathbb{R}^D,$$

where $a_1,\dots,a_D \geq 0$ are called *ARD weights* and $\gamma_1,\dots,\gamma_D > 0$ length scales. Intuitively, each ARD weight a_d represents the importance of the d-th variable (x_d and y_d) in measuring the similarity of the input vectors x and y: If a_d is larger, the difference $x_d - y_d$ in the d-th variable has a larger effect on the kernel value, and a smaller a_d implies a smaller effect on the kernel value. On the other hand, the length scale γ_d unit-normalises the scale of the d-th variable based on the data distribution on this variable, and can be specified by the variable-wise median heuristic described in Section E.

Sutherland et al. (2017) proposed to optimise the ARD weights $a_1,\dots,a_D$ to maximise the *test power* of the MMD TST. The test power is the probability of the test correctly rejecting the null hypothesis $H_0 : P = Q$ when the alternative hypothesis $H_1 : P \neq Q$ is true, where this probability is with respect to the random generation of the i.i.d. samples $\mathbf{X}$ and $\mathbf{Y}$. While the test power cannot be directly computed (as it depends on the unknown P and Q), one can calculate its asymptotic approximation (Gretton et al., 2012b). Thus, Sutherland et al. (2017) proposed to maximise this approximation, which results in the following objective function:

$$\ell\left(a_1,\dots,a_D\right) := \frac{\widehat{\mathrm{MMD}}_U^2(\mathbf{X},\mathbf{Y})}{\sqrt{\widehat{V}_{n,m}(\mathbf{X},\mathbf{Y}) + C}} \tag{2.4}$$

where $\widehat{\mathrm{MMD}}_U^2(\mathbf{X},\mathbf{Y})$ is the unbiased MMD estimate in Eq. (2.2) and $C \geq 0$ is a small constant. The quantity $\widehat{V}_{n,m}(\mathbf{X},\mathbf{Y})$ in Eq. (2.4) is an unbiased estimate of the *variance* of $\widehat{\mathrm{MMD}}_U^2(\mathbf{X},\mathbf{Y}) \in \mathbb{R}$, where the variance is calculated for the randomness of datasets $\mathbf{X}$ and $\mathbf{Y}$. Intuitively, it measures the stability of $\widehat{\mathrm{MMD}}_U^2(\mathbf{X},\mathbf{Y})$ against a slight perturbation of $\mathbf{X}$ and $\mathbf{Y}$. For $n = m$, it is given by[6]

$$\widehat{V}_{n,m}(\mathbf{X},\mathbf{Y}) := \frac{4}{n^3}\sum_{i=1}^{n}\left(\sum_{j=1}^{n}H_{ij}\right)^2 - \frac{4}{n^4}\left(\sum_{i,j=1}^{n}H_{ij}\right)^2, \tag{2.5}$$

where $H_{i,j} := k(X^i,X^j) + k(Y^i,Y^j) - k(X^i,Y^j) - k(Y^i,X^j)$.

The objective function Eq. (2.4) can be understood as follows. First, both $\widehat{\mathrm{MMD}}_U^2(\mathbf{X},\mathbf{Y})$ in

[6]This estimator is the version of Liu et al. (2020, Eq. (5)), which is simpler than the estimator of Sutherland et al. (2017, Eq. (5)).

the numerator and $\widehat{V}_{n,m}(\mathbf{X},\mathbf{Y})$ in the denominator depend on the kernel Eq. (2.3) and thus on the ARD weights $a_1,\ldots,a_D$. The estimate $\widehat{\mathrm{MMD}}^2_U(\mathbf{X},\mathbf{Y})$ in the numerator becomes large if the ARD weights serve to separate well the two datasets $\mathbf{X}$ and $\mathbf{Y}$. On the other hand, the variance $\widehat{V}_{n,m}(\mathbf{X},\mathbf{Y})$ in the denominator becomes small if the ARD weights make the estimate $\widehat{\mathrm{MMD}}^2_U(\mathbf{X},\mathbf{Y})$ stable, i.e. if the estimate does not change substantially even if the datasets $\mathbf{X}$ and $\mathbf{Y}$ are slightly perturbed. There is a trade-off between these two requirements; good ARD weights should balance the trade-off.

The constant C in the denominator exists to prevent the objective function from being numerically unstable, which can happen when the variance $\widehat{V}_{n,m}(\mathbf{X},\mathbf{Y})$ is extremely small. Following Liu et al. (Liu et al., 2020), we set the value $C = 10^{-8}$ throughout our experiments.

Chapter 3

Variable Selection via MMD Optimisation

This chapter describes the proposed methods. Related works are introduced in Section 3.1. Section 3.2 introduces notations for the variables selection task. Section 3.3 discusses an issue in optimising the ARD weights and introduces a regularisation method to address it. Sections 3.4.1 and 3.4.2 propose two approaches to perform variable selection based on regularised ARD optimisation.

3.1 Related Works

We discuss two topics of related works separately: the "Sample-selection approach" and the "Variable-selection approach".

3.1.1 Sample-Selection Approach

The sample-selection approach selects *sample locations* $x \in \mathbb{R}^D$ on which the probability densities (or their masses) $P(x)$ and $Q(x)$ of the two probability distributions P and Q differ significantly. Although this approach selects a set of representative sample locations, humans have to conduct an analysis work to understand the differences between two datasets, which is a disadvantage we discussed in Chapter 1. The variable-selection approach is advantageous since it can select such representative sample locations once variables are selected.

Duong and Koch (2009); Duong (2013) proposed to estimate the density functions of P and Q using a non-parametric density estimation method, and then obtain the sample locations where the two density estimates differ significantly. Instead of estimating the density functions,

Lloyd and Ghahramani (2015), Jitkrittum et al. (2016) and Kim et al. (2016) proposed to estimate the kernel mean embeddings of P and Q and obtain the sample locations where the two mean embeddings differ significantly. Cazals and Lhéritier (2015) and Kim et al. (2019) formulated TST as a regression problem of predicting whether a given sample point is from P or Q. They proposed to solve this regression problem using a non-parametric method and obtain sample locations where the discrimination of P or Q is the easiest (which are the locations where the densities of P and Q differ significantly).

3.1.2 Variable-Selection Approach

The variable selection approach selects a subset $S \subset \{1, 2, \ldots, D\}$ of variables (or coordinates) out of the D variables $x = (x_1, \ldots, x_D)^\top \in \mathbb{R}^D$ on which the marginal distributions P_S and Q_S of P and Q differ significantly.

The classifier TST approach (Friedman, 2003; Hido et al., 2008; Lopez-Paz and Oquab, 2017), which is closely related to the regression approach above Cazals and Lhéritier (2015); Kim et al. (2019), first assigns positive labels to sample points from P and negative labels to those from Q, and then learns a classifier to discriminate the two samples; the resulting classification accuracy, which becomes higher if P and Q are more different, is used as a test statistic.[1] Hido et al. (2008) and Lopez-Paz and Oquab (2017) performed qualitative (but not quantitative) experiments suggesting that the learned classifier's features may be used for understanding the features (variables) responsible for the discrepancies between P and Q. Even though classifier approaches are effective for TST problems, it is not for the variable selection task mainly due to the interdependent variable structures. Their objective functions will satisfactorily discriminate two given samples even though classifiers select variables that are dependent on other variables, as pointed out by (Wang et al., 2023).

Hara et al. (2017) and Zheng et al. (2020) proposed to construct a $D \times D$ matrix where the (i, j)-th element with $i, j \in \{1, \ldots, D\}$ is the estimated distance between the bivariate marginals $P_{\{i,j\}}(x_i, x_j)$ and $Q_{\{i,j\}}(x_i, x_j)$, where the distance metric is the Kullback-Leibler divergence in (Hara et al., 2017) and the Wasserstein distance in (Zheng et al., 2020). By computing a sparse submatrix with large positive entries from this $D \times D$ matrix, they obtain variables for which the univariate or bivariate marginal distributions of P and Q differ. Mueller and Jaakkola (2015) proposed optimising the data vectors' linear projections onto the real line $\mathbb{R}$ to maximise the Wasserstein distance between these projections. The optimised weights defining the projections may be used for selecting informative variables for the discrepancies

[1] MMD-based TST, including ours, can be related to classifier TST in that the MMD (or integral probability metrics in general, such as the Wasserstein distance) can be lower-bounded by the smoothness of the optimal classifier discriminating P and Q (Sriperumbudur et al., 2012, Prop. 2.6); if the optimal classifier is smoother, the discrimination is easier, and thus P and Q are more different and the MMD becomes larger. See also (Liu et al., 2020, Section 4).

between distributions P and Q. These approaches, however, are nonconvex problems and possibly result in local optimal solutions, as noted by (Wang et al., 2023). Furthermore, these methods mainly rely on distances of univariate, bivariate, or projection; therefore, there is a possibility that these representations are improper and unable to represent the original distribution shapes correctly.

Each of the above methods has explicitly or implicitly a hyperparameter specifying the number of selected variables or the strength of regularisation. In practice, an appropriate value of such a hyperparameter is typically unknown. Our methodological contribution is to develop two methods for addressing this issue. Moreover, as variable selection in TST is a relatively new topic in the literature, the definition of the task itself, or that of the "ground-truth" variables, has not been established. We propose a mathematically rigorous definition and analyse its properties in the next section.

3.1.3 Variable Selection Approaches employing MMD

As we show already in Section 2.3, Sutherland et al. (2017, Section 4) proposed to optimise the ARD weights to maximise the test power in an MMD test; however, their work did not conduct qualification assessments of the optimised ARD weights. As we show in Sections 3.3 and 3.5.2, their optimisation problem does not realise satisfactory variable selection.

The following studies (Yamada et al., 2018; Lim et al., 2020; Wang et al., 2023) are related work in terms of employing MMD for the variable selection task; however, these works require the number of variables to be selected as a hyperparameter. Usually, the number of variables to be selected is unknown in practice, and an algorithm, ideally and preferably, could work with fewer hyperparameters. Our proposal automatically selects the number of variables to be selected and the regularisation parameter, which is a significant advantage over these works.

Yamada et al. (2019) and Lim et al. (2020) proposed MMD-based post-selection inference methods for selecting variables x_i with $i \in \{1, \ldots, D\}$ such that the *one-dimensional* marginal distributions $P_i(x_i)$ and $Q_i(x_i)$ of P and Q differ. By definition, these methods cannot detect discrepancies appearing only in multivariate marginal distributions. For example, for $i, j \in \{1, \ldots, D\}$ with $i \neq j$, suppose that P and Q have equal one-dimensional marginals $P_i(x_i) = Q_i(x_i)$ and $P_j(x_j) = Q_j(x_j)$, and that P has a correlation for x_i and x_j while Q does not have. In this case, their approaches cannot detect the discrepancy between the bivariate marginals $P_{\{i,j\}}(x_i, x_j)$ and $Q_{\{i,j\}}(x_i, x_j)$, as demonstrated in Section 5.3.

Wang et al. (2023)[2] employ similar concepts to our work in terms of designing a kernel function

[2]We did not compare our method with theirs empirically since their codebase was not public when we conducted assessments in Section 3.5.2.

with ARD weights[3], yet their proposal requires the number of variables to be selected as a hyperparameter. Indeed, this work, similar to our work, is motivated by optimising an MMD estimator (Sutherland et al., 2017). However, their optimisation approach, differently from ours, employs *Sparse Trust Region Subproblem (STRS)* which is an optimisation problem of selecting a subset. Since STRS is a non-convex and NP-hard problem, they propose a heuristic algorithm to solve it by *simulated annealing*. Their objective function is found at Eq. 7, and the heuristic algorithm for the STRS is in Algorithm 3. From a viewpoint of applications, the significant difference from us is in requiring two hyperparameters; 1. the number of variables to be selected which is d in Eq. 3; 2. a set of regularisation parameters $\mathscr{G}$ in Algorithm 3.

3.2 Problem Formulation

We first define TST. Let P and Q probability distributions on the D-dimensional Euclidean space $\mathbb{R}^D$. Let $\mathbf{X} := \{X^1, \ldots, X^n\} \subset \mathbb{R}^D$ be a dataset of n i.i.d. (independently and identically distributed) sample vectors from P, and $\mathbf{Y} := \{Y^1, \ldots, Y^m\} \subset \mathbb{R}^D$ be a dataset of i.i.d. sample vectors of size m from Q. The task of TST is to test the null hypothesis $H_0 : P = Q$ against the alternative hypothesis $H_1 : P \neq Q$ using the two datasets $\mathbf{X}$ and $\mathbf{Y}$.

We define necessary notation. Let $S \subset \{1, \ldots, D\}$ be a subset of variable indices, and $\{1, \ldots, D\} \setminus S$ be its complement. Let P_S and $P_{\{1,\ldots,D\}\setminus S}$ be the marginal distributions of P on S and $\{1, \ldots, D\} \setminus S$, respectively: $P_S(x_S) := \int P(x) dx_{\{1,\ldots,D\}\setminus S}$ and $P_{\{1,\ldots,D\}\setminus S}(x_{\{1,\ldots,D\}\setminus S}) := \int P(x) dx_S$, where $x = (x_S, x_{\{1,\ldots,D\}\setminus S}) \in \mathbb{R}^D$. Likewise, let Q_S and $Q_{\{1,\ldots,D\}\setminus S}$ be the marginal distributions of Q on S and $\{1, \ldots, D\} \setminus S$, respectively.

For any disjoint subsets $S, U \subset \{1, \ldots, D\}$, we define $P_S \otimes P_U$ as the product distribution of the marginal distributions P_S and P_U, i.e., $P_S \otimes P_U(x_S, x_U) := P_S(x_S) P_U(x_U)$ for $(x_S, x_U) \in \mathbb{R}^{|S|+|U|}$. Recall that, if $P_{S \cup U} = P_S \otimes P_U$ (i.e., the joint and product distributions are equal), then for a random vector $(X_S, X_U) \sim P_{S \cup U}$, the subvectors X_S and X_U are statistically independent.

As variable selection in TST is relatively new in the literature, no established mathematical definition exists for it.[4] We thus provide a rigorous mathematical definition below, which may be of independent interest.

Definition 1. *Variable selection in TST is defined as finding a subset $S \subset \{1, \ldots, D\}$ that satisfies the following:*

[3] Wang et al. (2023) do not use the term "ARD weights" but employ the same concept, e.g. a Gaussian kernel at (Wang et al., 2023, Eq. (6)).

[4] Most existing works in Section 3.1 do not mathematically define the task of variable selection in TST. The only exception is Hara et al. Hara et al. (2017, Problem 1), where the task is to find $S \subset \{1, \ldots, D\}$ such that $P_{\{1,\ldots,D\}\setminus S} = Q_{\{1,\ldots,D\}\setminus S}$ and $P_{\{d\}\cup\{1,\ldots,D\}\setminus S} \neq Q_{\{d\}\cup\{1,\ldots,D\}\setminus S}$ for all $d \in S$. However, this definition has the following issue. Suppose $D = 2$, $P_{\{1\}} = Q_{\{1\}}$, $P_{\{2\}} = Q_{\{2\}}$ and $P \neq Q$. In this case, according to the definition of Hara et al. (2017), one can either select $S = \{1\}$ or $S = \{2\}$, but not $S = \{1, 2\}$. However, ideally, $S = \{1, 2\}$ should be selected, because the difference between P and Q only appears when the distributions on the two variables $\{1, 2\}$ are compared.

1. *There is no subset $U \subset S$ such that $P_U = Q_U$, $P_S = P_U \otimes P_{S\setminus U}$ and $Q_S = Q_U \otimes Q_{S\setminus U}$.*

2. *S is the largest among such sets. That is, we have $S \not\subset S'$ for any $S' \subset \{1,\dots,D\}$ satisfying 1).*

Condition 1) in Definition 1 requires that S does not contain any redundant variables U such that a) the marginal distributions P_U and Q_U on U are identical, and that b) U are independent of the rest of the variables $S\setminus U$ in that $P_S = P_U \otimes P_{S\setminus U}$ and $Q_S = Q_U \otimes Q_{S\setminus U}$. On the other hand, suppose that there is a subset $A \subset S$ such that $P_A = Q_A$ holds but either $P_S = P_A \otimes P_{S\setminus A}$ or $Q_S = Q_A \otimes Q_{S\setminus A}$ does *not* hold; such A is informative for distinguishing P_S and Q_S. For example, suppose that $D = 2$, $P_{\{1\}} = Q_{\{1\}}$, $P_{\{2\}} = Q_{\{2\}}$, $P = P_{\{1\}} \otimes P_{\{2\}}$, and $Q \neq Q_{\{1\}} \otimes Q_{\{2\}}$, so that $P \neq Q$. If $S = \{1,2\}$ and $A = \{1\}$, we have $P_A = Q_A$ and $P_S = P_A \otimes P_{S\setminus A}$ but $Q_S \neq Q_A \otimes Q_{S\setminus A}$. In this case, this subset $A \subset S$ is needed to distinguish $P_S (= P)$ and $Q_S (= Q)$.

Condition 2) requires that S be the largest among subsets satisfying condition 1). The following proposition shows that such S is unique. It also provides an "explicit" expression of S and the resulting decompositions of P and Q.

Proposition 1. *Suppose $P \neq Q$, and let $S \subset \{1,\dots,D\}$ be the subset satisfying the conditions in Definition 1. Then S is unique. Moreover, let $U \subset \{1,\dots,D\}$ be the largest subset such that $P = P_U \otimes P_{\{1,\dots,D\}\setminus U}$, $Q = Q_U \otimes Q_{\{1,\dots,D\}\setminus U}$ and $P_U = Q_U$. Then we have $S = \{1,\dots,D\}\setminus U$, and thus*

$$\begin{aligned} &P = P_S \otimes P_{\{1,\dots,D\}\setminus S}, \quad Q = Q_S \otimes Q_{\{1,\dots,D\}\setminus S}, \\ &\text{where} \quad P_S \neq Q_S, \quad P_{\{1,\dots,D\}\setminus S} = Q_{\{1,\dots,D\}\setminus S}, \end{aligned} \tag{3.1}$$

Proof. In Section C.1. □

Proposition 1 shows that S in Definition 1 is given as $S = \{1,\dots,D\}\setminus U$ with U the largest "redundant" variables such that $P_U = Q_U$, $P = P_U \otimes P_{\{1,\dots,D\}\setminus U}$ and $Q = Q_U \otimes Q_{\{1,\dots,D\}\setminus U}$. Consequently, we have the decompositions $P = P_S \otimes P_{\{1,\dots,D\}\setminus S}$ and $Q = Q_S \otimes Q_{\{1,\dots,D\}\setminus S}$ in Eq. (3.1), where S is the variables where distributional changes occur, $P_S \neq Q_S$, and $\{1,\dots,D\}/S = U$ are the redundant variables.

3.3 Regularisation for Variable Selection

We propose to optimise the ARD weights in the kernel (1) of the MMD by solving the following L_1 regularised optimisation problem (Tibshirani, 1996).

$$\min_{a\in\mathbb{R}^D} -\log\big(\ell(a_1,\dots,a_D)\big) + \lambda \sum_{d=1}^{D} |a_d|, \tag{3.2}$$

where ℓ is the objective function in Eq. (2.4), and $\lambda \geq 0$ is a regularisation parameter. If $\lambda = 0$, this minimisation problem is equivalent to the maximisation of the objective function Eq. (2.4).

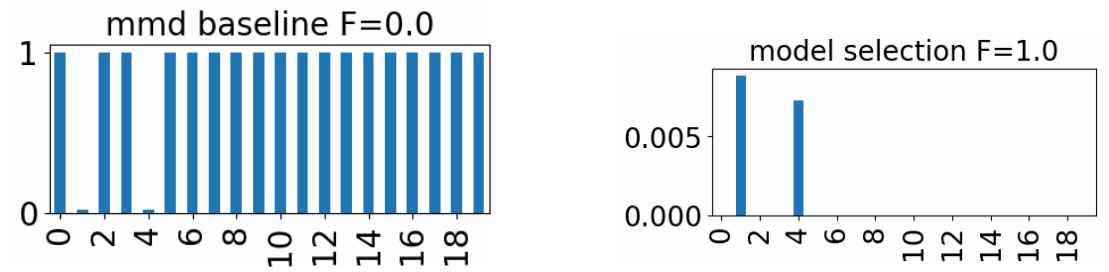


Figure 3.1: Optimised ARD weights without regularisation (Left) and with regularisation (Right). Here, $S = \{1, 4\}$ are the true variables for distinguishing P and Q, and $U = \{0, 2, 3, 5, \ldots, 19\}$ are the redundant variables with zero-variance marginal distributions. Without regularisation, the redundant variables' ARD weights do not change from their initial value 1, and bury the weights of the informative variables. With regularisation, these redundant variables are successfully eliminated. The details of the setting are described in Section 3.5.2 ("Redundant Dirac").

Regularisation works in variable selection by penalising large weights associated with redundant variables. For example, consider Example 1, where, for the d-th variable with $d \in \{1, \ldots, D\}$, P and Q have the identical marginal distribution $P_d = Q_d = \delta_\xi$ that is the Dirac distribution at $\xi \in \mathbb{R}$ (say $\xi = 1.43$). Then, for i.i.d. sample vectors $\mathbf{X} = \{X^1, \ldots, X^n\} \subset \mathbb{R}^D$ from P and those $\mathbf{Y} = \{Y^1, \ldots, Y^m\} \subset \mathbb{R}^D$ from Q, their values for the d-th variable are all identical to ξ: $X_d^1 = \cdots = X_d^n = Y_d^1 = \cdots = Y_d^m = \xi$. Therefore the d-th variable is redundant for distinguishing P and Q. However, as we have $a_d^2(X_d^i - X_d^j)^2 = 0$, $a_d^2(Y_d^i - Y_d^j)^2 = 0$, $a_d^2(X_d^i - Y_d^j)^2 = 0$ for all possible i and j, the value of the ARD weight a_d does not affect the ARD kernel (1) and thus the objective function Eq. (2.4). Therefore, without regularisation, the maximisation of the objective Eq. (2.4) (or the minimisation of (3.2) with $\lambda = 0$) does not make the ARD weight a_d small. Regularisation can fix this issue by penalising non-zero weights associated with such redundant variables. Figure 3.1 demonstrates how regularisation works.

The question is how to set the regularisation parameter λ in Eq. (3.2). If λ is too large, most optimised ARD weights may become zero, leading to false negatives. If λ is too small, the optimised ARD weights associated with redundant variables may not become zero, leading to false positives as we exemplified in Section 3.3.2. Our main methodological contributions are two approaches to the regularisation parameter selection, as described in Sections 3.4.1 and 3.4.2.

3.3.1 Variable Selection using the Optimised Weights

As preliminaries to Sections 3.4.1 and 3.4.2, we explain how to select variables using the optimised ARD weights $a^* = (a_1^*, \ldots, a_D^*)$, the solution of Eq. (3.2). One way is to set a threshold $\pi_{\text{thr}} \geq 0$ and select variables whose weights are above the threshold: $\hat{S} := \{d \in \{1, \ldots, D\} \mid a_d^* > \pi_{\text{thr}}\}$. The question is how to set the threshold. Our preliminary experiments revealed that the use of a fixed threshold, such as $\pi_{\text{thr}} = 0$ and $\pi_{\text{thr}} = 0.1$, does not work well. The reason is that the range of the ARD weights changes drastically depending on the given dataset. For example, the maximum and minimum of the ARD weights can be 10^{-3} and 10^{-7}, respectively, for one dataset, while they can be 10^2 and 10^{-1} for another dataset. In either case, however, the ARD weights distribution indicates each variable's relative importance.

Consequently, we use the following data-driven method for determining the threshold π_{thr} based on the histogram of optimised ARD weights. The idea is to set the threshold as the smallest local minimum in the histogram. For instance, if $D = 5$, $a_1^* = 0.01$, $a_2^* = 0.02$, $a_3^* = 0.02$, $a_4^* = 0.03$, $a_5^* = 0.1$, we set $\pi_{\text{thr}}^* = 0.03$, and the 5th variable is selected as $a_5^* = 0.1 > 0.03$. Algorithmically, the method first constructs a histogram of the optimised ARD weights (with 100 bins as a default setting). Subsequently, it identifies bins with zero frequency. Among these bins, the one with the smallest value is selected as threshold π_{thr}.

3.3.2 Comparison of Effects by the Regularisation Parameter λ

We study how the choice of the regularisation parameter λ in Eq. (3.2) affects the variable selection performance. To this end, we set λ as each of the candidate values from $10^{-3}, 10^{-3+0.25}, 10^{-3+0.5}, \ldots, 10^{1.25}, 10^1$, optimise the ARD weights by numerically solving Eq. (3.2) (see Section 3.5.1 for details), and perform variable selection as in Section 3.3.1. Figure 3.2 describes the F score, Precision and Recall for each value of λ and each setting, where the means and standard deviations are obtained from the results of 10 independently repeated experiments.

One can observe that the optimal value of λ varies depending on the setting of distributions P and Q. For example, the highest precision is attained with $\lambda = 10^{-1.5}$ for the "Narrower variances" setting, but it is attained with $\lambda = 1.0$ for the other settings. The dependence of the best regularisation parameter on the setting of P and Q highlights the difficulty of manually selecting an appropriate regularisation parameter. This fact motivates the proposed approaches of Algorithms 1 and 2.

3.4 Variable Selection Algorithms

We propose two algorithms for variable selection: MMD Model Selection (Algorithm 1) in Section 3.4.1, MMD CV Aggregation (Algorithm 2) in Section 3.4.2. There are two major

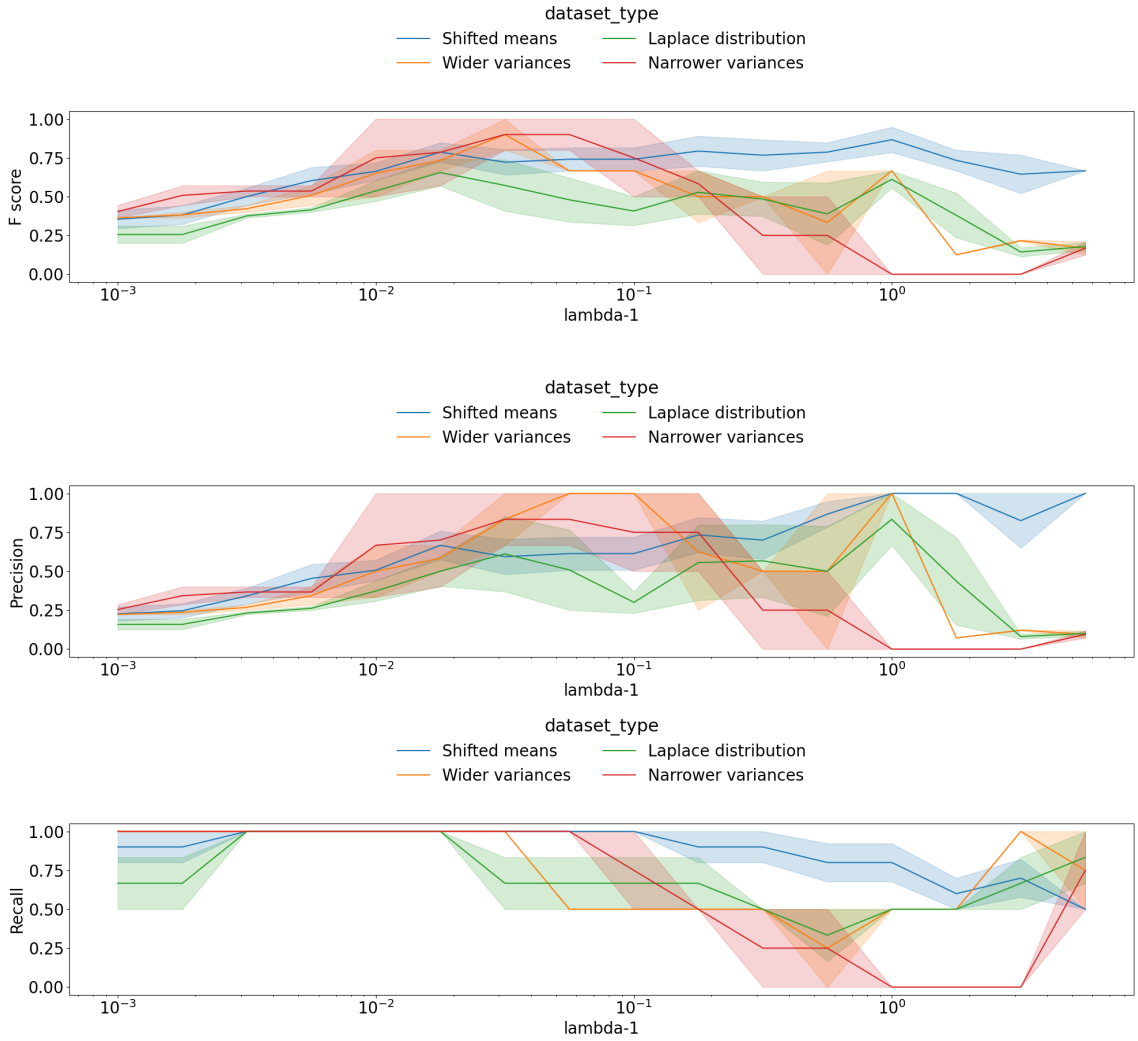


Figure 3.2: Effects of the regularisation parameter on the variable selection performance, discussed in Appendix 3.3.2. The top, middle and bottom figures show the F score, Precision and Recall for each setting of distributions P and Q, where the horizontal axis indicates the regularisation parameter λ used.

differences in these two algorithms; 1) MMD Model Selection relies on the given training datasets and validation datasets, while MMD CV Aggregation employs the cross-validation approach, 2) MMD Model Selection chooses the best λ parameter and its associated ARD weights, while MMD CV Aggregation combines a set of ARD weights obtained by several λ parameters.

Two algorithms require a set of candidate regularisation parameters Λ as an input. We introduce an automatic method of selecting Λ in Appendix D. However, one can also manually set Λ based on the preference.

3.4.1 MMD Model Selection: Data-driven Regularisation Parameter Selection

We describe a method for selecting the regularisation parameter λ in the optimisation problem Eq. (3.2). Below, let $a_\lambda \in \mathbb{R}^D$ be the (numerical) solution of Eq. (3.2), and $\hat{S}_\lambda$ be the selected variables (as done in Section 3.3.1).

We combine two criteria to select the regularisation parameter. One is the value of the objective function Eq. (2.4) evaluated for the optimised ARD weights a_λ on held-out validation data: we denote this by $\ell_{\mathrm{val}}(a_\lambda)$. The other is the P-value of a permutation TST performed on the selected variables $\hat{S}_\lambda$. We select λ with the highest $\ell_{\mathrm{val}}(a_\lambda)$ among candidates whose P-values are less than 0.05. The concrete procedure is described in Algorithm 1.

Procedure of Algorithm 1

First, we split the data $(\mathbf{X}, \mathbf{Y})$ into training data $(\mathbf{X}_{\mathrm{train}}, \mathbf{Y}_{\mathrm{train}})$ and validation data $(\mathbf{X}_{\mathrm{val}}, \mathbf{Y}_{\mathrm{val}})$. Let Λ be a set of candidate parameters (see Section D.1). We perform the following for each $\lambda \in \Lambda$ (lines 2-5). Line 2 obtains optimised ARD weights a_λ by numerically solving Eq. (3.2) using λ and $(\mathbf{X}_{\mathrm{train}}, \mathbf{Y}_{\mathrm{train}})$. Line 3 then evaluates the objective function Eq. (2.4) on $(\mathbf{X}_{\mathrm{val}}, \mathbf{Y}_{\mathrm{val}})$ using the ARD weights a_λ; let $\ell_{\mathrm{val}}(a_\lambda)$ be this value. Line 4 selects variables $\hat{S}_\lambda \subset \{1, \ldots, D\}$ using a_λ as in Section 3.3.1. Line 5 performs a permutation TST (e.g., (Efron and Tibshirani, 1994, Chapter 15)) on the validation data $\mathbf{X}_{\mathrm{val}}([:, \hat{S}_\lambda])$ and $\mathbf{Y}_{\mathrm{val}}([:, \hat{S}_\lambda])$ where only the selected variables $\hat{S}_\lambda$ are used (e.g., if $\mathbf{X}_{\mathrm{val}}$ is given as a $n_{\mathrm{val}} \times D$ matrix, then $\mathbf{X}_{\mathrm{val}}([:, \hat{S}_\lambda])$ is a $n_{\mathrm{val}} \times |\hat{S}_\lambda|$ matrix); let $0 \leq p_\lambda \leq 1$ be the resulting P-value. Once the above procedure is applied for all $\lambda \in \Lambda$, line 8 selects λ^* with maximum $\ell_{\mathrm{val}}(a_\lambda)$ among those with $p_\lambda < 0.05$, and returns the corresponding variables $\hat{S}_\lambda$ as the final selected variables. If there exists no $\lambda \in \Lambda$ with $p_\lambda < 0.05$, line 10 selects λ^* with minimal P-value p_λ, and returns the corresponding variables $\hat{S}_{\lambda^*}$.

For the permutation test on the selected variables $\hat{S}_\lambda$, one can use, in principle, any test statistic for TST applicable to multivariate data. We found in our preliminary analysis that the sliced Wasserstein distance (Bonneel et al., 2014) performs well while running faster than re-optimising the ARD weights on the selected variables $\hat{S}_\lambda$, so we use the former in our experiments.

Mechanism

We now discuss the mechanism of the proposed approach of using the two criteria. This approach is based on the following ideas:

1. A high $\ell_{\mathrm{val}}(a_\lambda)$ implies that the selected variable $\hat{S}_\lambda$ contains many of the true variables S;

2. A low p_λ implies that $\hat{S}_\lambda$ does not contain many of the redundant variables $U := \{1,\dots,D\}\backslash S$.

Point 1) is because the objective function Eq. (2.4) becomes high when the ARD weights a_λ lead to a high power of the resulting MMD TST. To see point 2), suppose that $D = 5$, the ground-truth variables are $S = \{1,3\}$, the redundant ones are $U = \{2,4,5\}$, and the selected variables are $\hat{S}_\lambda = \{1,2,3,4,5\}$. As $\hat{S}_\lambda$ contains redundant variables U, the permutation test using $\hat{S}_\lambda$ may fail to distinguish P and Q, since U adds "noises" to "signals" S. Accordingly, the P-value p_λ would not become small.

To understand the mechanism better, suppose that the objective value is high, but the P-value is not small. This happens when the selected variables $\hat{S}_\lambda$ contain many of the true variables S but also many of the redundant variables U. For instance, consider Example 1, where the ARD weights on the redundant variables $U\backslash V$ do not influence the objective value Eq. (2.4). In this case, the objective value can be large, even if the weights of the redundant variables are large so that the redundant variables are selected. Consequently, the P-value would not be small while the objective value is large.

On the other hand, suppose the objective value is not high, but the P-value is small. This happens when the selected variables $\hat{S}_\lambda$ contain *some* of the true variables S but miss some. To see this, suppose again $D = 5$, $S = \{1,3\}$, and that both variables 1 and 3 have equally different marginal distributions, i.e., P_1 and Q_1 are as different as P_3 and Q_3. In this setting, assume that the ARD weights $a_\lambda = (a_{\lambda,1},\dots,a_{\lambda,5})$ have a large weight only for variable 1 (e.g., $a_\lambda = (1.6, 0.1, 0.12, 0.2, 0.3)$), so that only variable 1 is selected: $\hat{S}_\lambda = \{1\}$. As P_1 and Q_1 are different, the permutation test using $\hat{S}_\lambda = \{1\}$ would lead to a small P-value. However, because the weight for variable 3 is not large ($a_{\lambda,3} = 0.12$), the objective value $\ell_{\mathrm{val}}(a_\lambda)$ would not become as high as alternative ARD weights where both variables 1 and 3 have large weights (e.g., $a_\lambda = (1.5, 0.2, 1.4, 0.2, 0.1)$). Therefore, the P-value is small in this example, but the objective value would not be high.

To summarise, if the objective value $\ell_{\mathrm{val}}(a_\lambda)$ is high and the P-value p_λ is small, the selected variables $\hat{S}_\lambda$ would contain many of the true variables S but do not contain many redundant variables U. Algorithm 1 selects such a regularisation parameter λ.

3.4.2 MMD CV Aggregation: Cross Validation based Aggregation

Rather than selecting variables based on one "best" regularisation parameter, our second method aggregates the outputs of different regularisation parameters. This method extends our first method by applying cross-validation and computing aggregated scores for the individual variables. It draws inspiration from the Stability Selection algorithm in high-dimensional

Algorithm 1 Data-driven Regularisation Parameter Selection

Input: Λ: a set of candidate regularisation parameters. $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$: training data. $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$: validation data.

Output: $\hat{S}_{\lambda^*} \subset \{1, \ldots, D\}$: selected variables with the best regularisation parameter $\lambda^* \in \Lambda$.

1: **for all** $\lambda \in \Lambda$ **do**
2: Obtain ARD weights $a_\lambda \in \mathbb{R}^D$ by numerically solving Eq. (3.2) using $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ and λ.
3: Compute $\ell_{\text{val}}(a_\lambda) > 0$ by evaluating Eq. (2.4) on $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$.
4: Select variables $\hat{S}_\lambda \subset \{1, \ldots, D\}$ using a_λ as in Section 3.3.1.
5: Compute P-value $0 \leq p_\lambda \leq 1$ by performing a permutation TST on $(\mathbf{X}_{\text{val}}[:, \hat{S}_\lambda], \mathbf{Y}_{\text{val}}[:, \hat{S}_\lambda])$.
6: **end for**
7: **if** there exists $\lambda \in \Lambda$ with $p_\lambda < 0.05$ **then**
8: $\lambda^* = \arg\max_{\lambda \in \Lambda} \{\ell_{\text{val}}(a_\lambda) \mid p_\lambda < 0.05\}$.
9: **else**
10: $\lambda^* = \arg\min_{\lambda \in \Lambda} p_\lambda$.
11: **end if**

statistics (Meinshausen and Bühlmann, 2010). (See Section I.5 for a systematic comparison with other candidate aggregation strategies for Algorithm 2.)

Procedure of Algorithm 2

Algorithm 2 describes the procedure of the method. For each candidate regularisation parameter $\lambda \in \Lambda$, lines 1 to 10 perform the following. Let K be the number of splits in cross-validation (e.g., $K = 10$). For each split $i = 1, \ldots, K$, line 3 randomly splits the data $(\mathbf{X}, \mathbf{Y})$ into training data $(\mathbf{X}^i_{\text{train}}, \mathbf{Y}^i_{\text{train}})$ and validation data $(\mathbf{X}^i_{\text{val}}, \mathbf{Y}^i_{\text{val}})$ by the ratio ρ_{train}: $1 - \rho_{\text{train}}$, where $0 < \rho_{\text{train}} < 1$. We set $\rho_{\text{train}} = 0.5$ as a default value. Lines 4 to 7 perform lines 2 to 5 of Algorithm 1 using $(\mathbf{X}^i_{\text{train}}, \mathbf{Y}^i_{\text{train}})$ and $(\mathbf{X}^i_{\text{val}}, \mathbf{Y}^i_{\text{val}})$; let $a^i_\lambda \in \mathbb{R}^D$ be the ARD weights, $\hat{S}^i_\lambda \subset \{1, \ldots, D\}$ be the selected variables, $\ell_{\text{val}}(a^i_\lambda)$ be the objective value, $0 \leq p^i_\lambda \leq 1$ be the P-value. (See Section 3.4.1 for the explanation.) Line 8 normalises the ARD weights $a^i_\lambda = (a^i_{\lambda,1}, \ldots, a^i_{\lambda,D})$ by dividing them by the largest weight $\max_{d \in \{1,\ldots,D\}} a^i_{\lambda,d}$ so that the largest weight of the normalised ones $\tilde{a}^i_\lambda = (\tilde{a}^i_{\lambda,1}, \ldots, \tilde{a}^i_{\lambda,D})$ becomes 1, i.e., $\max_{d \in \{1,\ldots,D\}} \tilde{a}^i_{\lambda,d} = 1$. This normalisation is needed for the aggregation performed later.

Line 10 computes the D-dimensional score vector

$$\hat{\Pi}_\lambda := \frac{1}{K} \sum_{i=1}^{K} I(p^i_\lambda < 0.05) \ell_{\text{val}}(a^i_\lambda) \tilde{a}^i_\lambda \in \mathbb{R}^D,$$

where $I(p^i_\lambda < 0.05) = 1$ if $p^i_\lambda < 0.05$ and $I(p^i_\lambda < 0.05) = 0$ if $p^i_\lambda \geq 0.05$. This vector is the average, over the K splits $i = 1, \ldots, K$, of the normalised ARD weight vector $\tilde{a}^i_\lambda$ multiplied by

the objective value $\ell_{\text{val}}(a_\lambda^i)$ such that the P-value p_λ is smaller than 0.05. Line 12 computes the average, over candidate regularisation parameters $\lambda \in \Lambda$, of the score vector $\hat{\Pi}_\lambda$ to obtain the aggregated score vector

$$\hat{\Pi} := \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \hat{\Pi}_\lambda \in \mathbb{R}^D.$$

Lastly, line 13 selects variables based on the aggregated score $\hat{\Pi}$.

Mechanism

To simplify the explanation, suppose $K = 1$, in which case the data $(\mathbf{X}, \mathbf{Y})$ are split into training data $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ and validation data $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$ only once as in Algorithm 1. As $K = 1$, we do not write the superscript i here. The final score vector $\hat{\Pi}$ in Algorithm 2 is then

$$\hat{\Pi} = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} I(p_\lambda < 0.05) \ell_{\text{val}}(a_\lambda) \tilde{a}_\lambda \in \mathbb{R}^D. \tag{3.3}$$

This score vector is the weighted average, over different regularisation parameters $\lambda \in \Lambda$, of the normalised ARD weight vector $\tilde{a}_\lambda \in \mathbb{R}^D$ weighted by the objective value $\ell_{\text{val}}(a_\lambda)$, such that the P-value p_λ is less than 0.05.

As discussed in Section 3.4.1, a large P-value suggests that the selected variables $\hat{S}_\lambda$ contain many redundant variables. Therefore, the indicator function $I(p_\lambda < 0.05)$, which becomes 0 if $p_\lambda \geq 0.05$, effectively excludes such λ leading to the selection of redundant variables from the score vector Eq. (3.3). In the score vector Eq. (3.3), a higher contribution is made by the normalised ARD weights $\tilde{a}_\lambda \in \mathbb{R}^D$ with a higher objective value $\ell_{\text{val}}(a_\lambda)$.

For example, suppose that $D = 5$, $\Lambda = \{0.01, 0.1, 1.0\}$, and

$$\begin{aligned}
&\tilde{a}_{0.01} = (0.9, 0.8, 1.0, 0.7, 0.6), \quad \ell_{\text{val}}(a_{0.01}) = 12.8, \quad p_{0.01} = 0.4 \\
&\tilde{a}_{0.1} = (0.5, 0.4, 1.0, 0.02, 0.01), \quad \ell_{\text{val}}(a_{0.1}) = 7.4, \quad p_{0.1} = 0.01 \\
&\tilde{a}_{1.0} = (0.2, 0.1, 1.0, 0.0, 0.0), \quad \ell_{\text{val}}(a_{1.0}) = 3.1, \quad p_{1.0} = 0.01.
\end{aligned}$$

Then, even if the objective value $\ell_{\text{val}}(a_{0.01}) = 12.8$ for $\lambda = 0.01$ is the highest, the normalised ARD weights $\tilde{a}_{0.01}$ for $\lambda = 0.01$ do not contribute to the score vector Eq. (3.3), because the P-value $p_{0.01} = 0.4$ is higher than the threshold 0.05. As $p_{0.1} < 0.05$ and $p_{1.0} < 0.05$, the normalised ARD weight vectors $\tilde{a}_{0.1}$ and $\tilde{a}_{1.0}$ contribute to the score vector Eq. (3.3). Because the objective value $\ell_{\text{val}}(a_{0.1}) = 7.4$ for $\lambda = 0.1$ is higher than $\ell_{\text{val}}(a_{1.0}) = 3.1$ for $\lambda = 1.0$, the normalised ARD weights $\tilde{a}_{0.1}$ for $\lambda = 0.1$ contribute to the score vector more significantly than the normalised ARD weights $\tilde{a}_{1.0}$ for $\lambda = 1.0$. The score vector then becomes $\hat{\Pi} = (4.32, 3.27, 10.5, 0.148, 0.074)$. Applying the procedure in Section 3.3.1 to $\hat{\Pi}$, the first three variables would be selected: $\hat{S} = \{1, 2, 3\}$.

A critical difference between the algorithms is that Algorithm 1 selects one "best" regularisation parameter λ maximising the objective value $\ell_{\text{val}}(a_\lambda)$ among candidates with small P-values, while Algorithm 2 aggregates the scores (as given by the normalised ARD weights) from different regularisation parameters (with small P-values) by weighting with the objective values. In the above example, Algorithm 1 would only use the ARD weights a_λ with $\lambda = 0.1$, while Algorithm 2 uses the (normalised) ARD weights $\tilde{a}_\lambda$ for $\lambda = 0.1$ and $\lambda = 1.0$ with their associated objective values. Therefore, Algorithm 2 exploits more information and thus could perform more stable variable selection. Another difference is that Algorithm 2 uses cross-validation, which can effectively improve the stability of the score vector when the sample sizes n, m are small. On the other hand, Algorithm 1 is faster to compute and thus more advantageous for large sample sizes.

Algorithm 2 Cross Validation based Aggregation

Input: Λ: a set of candidate regularisation parameters. $\mathbf{X} = \{X^1, \ldots, X^n\} \subset \mathbb{R}^D$ and $\mathbf{Y} = \{Y^1, ..., Y^m\} \subset \mathbb{R}^D$: data. K: the number of cross-validation splits. $\rho_{\text{train}} \in (0,1)$: training-validation splitting ratio (default value: 0.5).

Output: $\hat{S} \subset \{1, \ldots, D\}$: selected variables.

for all $\lambda \in \Lambda$ **do**
  for all $i \in \{1, ..., K\}$ **do**
    Randomly split $(\mathbf{X}, \mathbf{Y})$ into $(\mathbf{X}^i_{\text{train}}, \mathbf{Y}^i_{\text{train}})$ and $(\mathbf{X}^i_{\text{val}}, \mathbf{Y}^i_{\text{val}})$ by the ratio $\rho_{\text{train}} : 1 - \rho_{\text{train}}$.
    Obtain ARD weights $a^i_\lambda \in \mathbb{R}^D$ by optimising Eq. (3.2) using $(\mathbf{X}^i_{\text{train}}, \mathbf{Y}^i_{\text{train}})$ and λ.
    Compute $\ell_{\text{val}}(a^i_\lambda) > 0$ by evaluating Eq. (2.4) on $(\mathbf{X}^i_{\text{val}}, \mathbf{Y}^i_{\text{val}})$.
    Select variables $\hat{S}^i_\lambda \subset \{1, \ldots, D\}$ using a^i_λ as in Section 3.3.1.
    Compute P-value $0 \leq p^i_\lambda \leq 1$ by performing a permutation two-sample test on $(\mathbf{X}_{\text{val}}[:, \hat{S}^i_\lambda], \mathbf{Y}_{\text{val}}[:, \hat{S}^i_\lambda])$.
    Compute $\tilde{a}^i_\lambda := a^i_\lambda / \left(\max_{d \in \{1,\ldots,D\}} a^i_{\lambda,d}\right) \in \mathbb{R}^D$
  end for
  Compute $\hat{\Pi}_\lambda := \frac{1}{K} \sum_{i=1}^K I(p^i_\lambda < 0.05) \ell_{\text{val}}(a^i_\lambda) \tilde{a}^i_\lambda \in \mathbb{R}^D$.
end for
Compute $\hat{\Pi} := \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \hat{\Pi}_\lambda \in \mathbb{R}^D$.
Select variables $\hat{S} \subset \{1, \ldots, D\}$ using $\hat{\Pi}$ as in Section 3.3.1.

3.5 Empirical Assessment and Demonstration

This section describes empirical assessments of the proposed methods.

Section 3.5.1 explains common settings for different experiments. Section 3.5.2 describes experiments with synthetic datasets. Section 3.5.3 reports experiments using data generated from a traffic simulator, and Section 3.5.4 describes demonstration of the variable selection

for the image data.

3.5.1 Common Settings

We compare the proposed Algorithms 1 and 2 with three baseline methods. The first baseline is the approach by Sutherland et al. (2017, Section 5), which we call here `mmd-baseline`. It optimises the ARD weights by maximising the objective function Eq. (2.4) (or minimising Eq. (3.2) with $\lambda = 0$) and performs variable selection using the optimised ARD weights as in Section 3.3.1. The second baseline is the approach by Lim et al. (2020), referred to as `mskernel-star`; see Section 3.1 for its explanation. We use the implementation by the authors.[5] This method requires the number of selected variables as a hyper-parameter; we set it to the number of ground-truth variables. The third baseline uses the L1-regularised logistic regression, and we call it `regression-baseline`. This method performs a classifier TST using L1-regularised linear logistic regression.[6] The learned coefficients of the regression model are used for variable selection as in Section 3.3.1. We tune this method's regularisation parameter by grid search of its inverse from the range $[-4, 4]$ by performing 5-fold cross-validation.

We refer MMD Model Selection (Algorithms 1) and MMD CV Aggregation (Algorithms 2) as `model-selection` and `CV-aggregation`, respectively. In addition to these methods, we report the results for `mmd-tuning-best-F1`, which optimises the regularised objective Eq. (3.2) using the regularisation parameter tuned to maximise the F score (introduced below). In practice, this method cannot be implemented because the F score is unavailable as it requires knowing the ground-truth variables; we show its results to provide the best possible performance of Algorithm 1.

We use the Gaussian ARD kernel Eq. (2.3) for the kernel-based approaches. We set the length-scale parameters $\gamma_1, \ldots, \gamma_D$ by the variable-wise median heuristic described in Appendix E.

We use the Adam optimiser to optimise Eq. (3.2) for the ARD weights. For its learning rate, we use the `ReduceLROnPlateau`[7] learning rate scheduler of PyTorch, which starts from 0.01 and adaptively reduces the learning rate based on the changes in objective values. The initial value for each ARD weight is set to 1.

The early stopper `convergence based stopper` is configured as described in Appendix G.1[8]. we set the maximum number of epochs to 99,999.

[5] https://github.com/jenninglim/multiscale-features In `mskernel-star`, "star" indicates that the number of selected variables is known.

[6] We use the code from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html .

[7] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

[8] We did not use `variable selection based stopper` in assessments of this section since `convergence based stopper` sufficiently works.

For Algorithms 1 and 2, we obtain candidate Λ regularisation parameters by the `heuristic-lambda-range` described in Section D.1. For Algorithm 1, we set the number of cross-validation splits to $K = 10$. To compute the p-values using the sliced Wasserstein distance for Algorithms 1 and 2; We use the `POT` Python package (Flamary et al., 2021).

3.5.2 Synthetic Data Experiments

We report experiments on synthetically generated datasets where the ground-truth variables are available for evaluation. Various assessments are conducted to evaluate in Appendix I.

Below, let $\mathbb{N}(\mu, \Sigma)$ be the D-dimensional Gaussian distribution with mean vector $\mu = (\mu_1, \ldots, \mu_D)^\top \in \mathbb{R}^D$ and covariance matrix $\Sigma \in \mathbb{R}^{D \times D}$.

Data Generation Processes and Evaluation Criteria

We generate datasets $\mathbf{X} = \{X^1, \ldots, X^n\} \overset{i.i.d.}{\sim} P$ and $\mathbf{Y} = \{Y^1, \ldots, Y^n\} \overset{i.i.d.}{\sim} Q$ as i.i.d. samples from probability distributions P and Q on $\mathbb{R}^D$, where $n = 200$ and $D = 20$. Another assessment in case of higher D is found in Section I.3. We explain below how to define these probability distributions P and Q.

Let $P = \mathcal{N}(\mathbf{0}_D, I_D)$, where $\mathbf{0}_D := (0, \ldots, 0)^\top \in \mathbb{R}^D$ and $I_D \in \mathbb{R}^{D \times D}$ is the identity matrix. We define Q as follows. Let $S \subset \{1, \ldots, D\}$ be a set of ground-truth variables with the cardinality $|S| = \lfloor \rho \times D \rfloor$, where $\rho \in (0, 1)$. Here, we set $\rho = 0.1$, so $|S| = 2$ (see Appendix I.1 for experiments with $\rho = 0.8$). We define Q_S as a marginal distribution on $\mathbb{R}^{|S|}$ such that $Q_S \neq P_S$, where $P_S = \mathcal{N}(\mathbf{0}_{|S|}, I_{|S|})$ is the marginal distribution of P on S. Then we define $Q := Q_S \otimes P_{\{1,\ldots,D\} \setminus S}$, where $P_{\{1,\ldots,D\} \setminus S} = \mathcal{N}(\mathbf{0}_{D-|S|}, I_{D-|S|})$ is the marginal distribution of P on $\{1, \ldots, D\} \setminus S$. More specifically, we consider the following five different ways of defining Q_S (and thus Q):

1. Shifted means: $Q_S = \mathcal{N}(\mathbf{0.5}_{|S|}, I_{|S|})$, where $\mathbf{0.5}_{|S|} = (0.5, \ldots, 0.5)^{|S|} \in \mathbb{R}^{|S|}$.
2. Wider variances: $Q_S = \mathcal{N}(\mathbf{0}_{|S|}, 1.5 I_{|S|})$.
3. Narrower variances: $Q_S = \mathcal{N}(\mathbf{0}_{|S|}, 0.5 I_{|S|})$.
4. Laplace distribution: $Q_S = \mathcal{L}(\mathbf{0}_{|S|}, I_{|S|})$ is the Laplace distribution on $\mathbb{R}^{|S|}$ with the same mean $\mathbf{0}_{|S|}$ and covariance matrix $I_{|S|}$.
5. Correlated Gaussian: the random vector $(y_1, \ldots, y_{|S|}) \sim Q_S$ is defined as follows. Generate $y_1 \sim \mathcal{N}(0, 1)$ and set $y_2 = \cdots = y_{|S|} = y_1$.

We also consider the following setting corresponding to that discussed in Sections C and 3.3.

6. Redundant Dirac: $P_S = \mathcal{N}(\mathbf{0}_{|S|}, I_{|S|})$, $Q_S = \mathcal{N}(\mathbf{0.5}_{|S|}, I_{|S|})$ and $P_{\{1,\ldots,D\}\setminus S} = Q_{\{1,\ldots,D\}\setminus S} = \delta_{\mathbf{0}_{D-|S|}}$, where $\delta_{\mathbf{0}_{D-|S|}}$ is the Dirac distribution at $\mathbf{0}_{D-|S|} = (0,\ldots,0)^\top \in \mathbb{R}^{D-|S|}$.

For each of these settings, we generate datasets $\mathbf{X} = \{X^1,\ldots,X^{200}\} \overset{i.i.d.}{\sim} P = P_S \otimes P_{\{1,\ldots,D\}\setminus S}$ and $\mathbf{Y} = \{Y^1,\ldots,Y^{200}\} \overset{i.i.d.}{\sim} Q = Q_S \otimes P_{\{1,\ldots,D\}\setminus S}$, run each method to select variables $\hat{S} \subset \{1,\ldots,D\}$, and evaluate the Recall (Re), Precision (Pr) and the F score w.r.t. the ground-truth variables $S \subset \{1,\ldots,D\}$. These evaluation criteria are defined as

$$\mathrm{Pr} = \frac{|\hat{S} \cap S|}{|\hat{S}|}, \quad \mathrm{Re} = \frac{|\hat{S} \cap S|}{|S|}, \quad F = \frac{2 \times \mathrm{Pr} \times \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}} \tag{3.4}$$

The precision is the ratio of the true positives among the selected variables, the recall is the ratio of the true positives among the ground-truth variables, and the F score is their harmonic mean; higher values indicate better variable selection performance. We repeat the above procedure 10 times independently, and compute averages and standard deviations of the criteria.

Results

Figure 3.3 describes the results. For 1) Shifted means, 2) Wider variances, 3) Narrower variances and 4) Laplace distribution, the three baseline methods (`regression-baseline`, `mskernel-star` and `mmd-baseline`) consistently yield low F scores, indicating the difficulty of variable selection in these settings where the differences between P and Q are subtle. These low F scores are due to many false positives, as evidenced by the low precisions and high recalls.

For 5) Correlated Gaussian, `mskernel-star` fails to detect any correct variables, resulting in zero precision and zero recall. This is because `mskernel-star` only examines the differences between univariate marginal distributions (i.e., P_d v.s. Q_d for each $d = 1\ldots,D$); however, the univariate marginals distributions for 5) are all the same; the differences appear only in the correlation structures of P and Q, which cannot be detected by `mskernel-star`.

For 6) Redundant Dirac, `mmd-baseline` exhibits a significant drop in recall compared with the other settings. This is because in 6) the regularisation is necessary to eliminate the ARD weights of redundant variables; Figure 3.1 shows a visualisation of the issue.

Algorithms 1 and 2 (`model-selection` and `CV-aggregation`) yield higher F scores than the three baselines for most settings, indicating that the proposed methods address the above challenges. `CV-aggregation` performs better than or comparably with `model-selection`. Notably, `CV-aggregation` even outperforms `mmd-tuning-best-f1` for some settings. Recall that `mmd-tuning-best-f1` is not implementable in practice and is meant to provide the best possible F score of Algorithm 1.

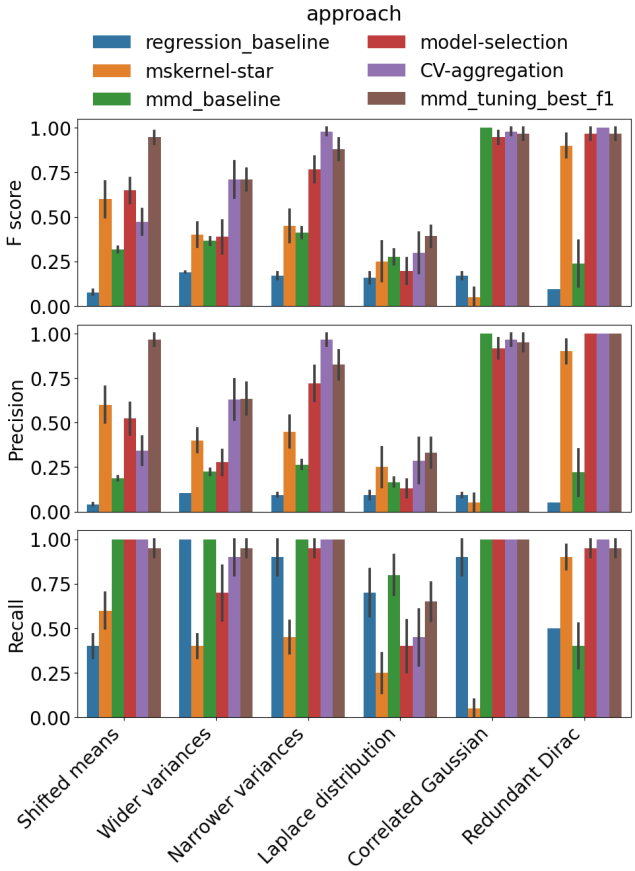


Figure 3.3: Results of the synthetic data experiments in Section 3.5.2. Top, middle and bottom plots show the F score, precision and recall, respectively. The groups correspond to the six settings. Each bar reports mean and standard deviation over 10 experiment executions.

Figure 3.4 describes how the F score of each method changes as the sample size n increases for the 4) Laplace distribution setting. The F scores of `model-selection` and `CV-aggregation` increase with higher sample size, suggesting that the methods identify ground-truth variables with subtle distributional changes with large enough samples. On the other hand, the F scores of the baseline methods do not improve for larger sample sizes and they are significantly lower than `model-selection` and `CV-aggregation`. `CV-aggregation` achieves a high F score of 0.87 with sample size 600, while `model-selection` requires sample size 1200 to reach a similar F score. This suggests that `CV-aggregation` makes a more effective use of data than `model-selection` as remarked in Section 3.4.2.

3.5.3 Demonstration: Variable Selection for Traffic Simulation Model Validation

We demonstrate how the proposed methods can be applied to exploratory data analysis. As an illustrative example, suppose we are interested in analysing datasets obtained from a city-scale

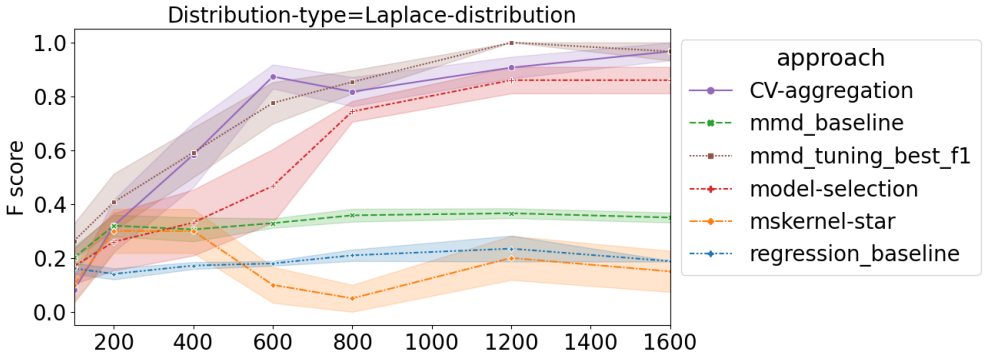


Figure 3.4: F scores for different sample sizes in the Laplace distribution setting in Section 3.5.2. The horizontal axis indicates sample size n. For each sample size and each method, the confidence interval shows the standard deviation of the F scores over 10 experiments.

traffic system. The form of data is a matrix $X \in \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$, where D_{sensor} is the number of sensors located in the road network and D_{time} is the number of time intervals. The (d,t)-th entry $X_{d,t}$ of X represents the number of vehicles detected by the d-th sensor during the t-th time interval. For example, in our experiments below, we have $D_{\text{sensor}} = 64$ sensors and $D_{\text{time}} = 12$ time intervals, where each time interval is for 5 minutes; thus, one matrix X records the traffic flows observed by the 64 sensors for the duration of $60 = 12 \times 5$ minutes.

Suppose now that we have two sets of such data matrices $\mathbf{X} = \{X^1, \ldots, X^n\} \subset \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ and $\mathbf{Y} = \{Y^1, \ldots, Y^m\} \subset \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ from two different settings. For example, one dataset $\mathbf{X}$ may be generated from a traffic simulator modelling the rush hours (8 pm - 9 pm) of the city, and the other dataset $\mathbf{Y}$ may consist of real observations from the city's rush hours. By analysing how the two datasets $\mathbf{X}$ and $\mathbf{Y}$ differ, one can understand which aspects the simulator fails to model the city's real traffic system; such insights can be used for improving the simulator.

We demonstrate how our `CV-aggregation` (Algorithm 2) can be used to analyse such datasets. To this end, we treat each data matrix $\mathbf{X} \in \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ as a $D = D_{\text{sensor}} \times D_{\text{time}}$-dimensional vector. Each variable $d \in \{1, \ldots, D\}$ corresponds to one specific pair of a sensor and a time interval. Therefore, $\mathbf{X} = \{X^1, \ldots, X^n\}$ and $\mathbf{Y} = \{Y^1, \ldots, Y^m\}$ can be regarded as two sets of D-dimensional vectors, and `CV-aggregation` can be used for selecting variables (= sensor-time pairs) where traffic flows in the two datasets differ significantly.

Experiment Setup

We briefly describe the data generation process. For details, see Appendix J. We generate both datasets $\mathbf{X}$ and $\mathbf{Y}$ from a microscopic stochastic traffic simulator[9] but from two different scenarios. We define a grid-like road network consisting of 8 intersections and $D_{\text{sensor}} = 64$

[9] We use the SUMO simulator: https://www.eclipse.org/sumo/

sensors, and define one scenario where vehicles travel to their destinations for the duration of 60 minutes; we call it scenario P. We define another scenario, which we call scenario Q, as a perturbed version of scenario P where two specific roads are blocked for the first 40 minutes. We define $D_{\text{time}} = 12$ time intervals, each 5 minutes. As mentioned, one simulation yields a matrix of the form $X \in \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$, where the (d,t)-th element $X_{d,t}$ of the matrix X is the number of vehicles detected by the d-th sensor during the t-th time interval ($d = 1,\ldots,D_{\text{sensor}}$ and $t = 1,\ldots,D_{\text{time}}$). We generate $\mathbf{X} = \{X^1,\ldots,X^n\} \subset \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ from scenario P with n random seeds, and $\mathbf{Y} = \{Y^1,\ldots,Y^n\} \subset \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ from scenario Q with n random seeds, with $n = 400$.

We use `CV-aggregation` with the same configuration as Section 3.5.1 with one modification on the choice of the length-scale parameters $\gamma_1,\ldots,\gamma_D$ with $D = D_{\text{sensor}} \times D_{\text{time}}$. A preliminary analysis showed that the variable-wise median heuristic in Section E yields zeros for some of $\gamma_1,\ldots,\gamma_D$ in this setting (which is problematic, as the length-scales should be positive by definition). The reason is each data matrix in the datasets $\mathbf{X}$ and $\mathbf{Y}$ contain many entries whose values are zero; these entries are sensor-time pairs in which no vehicle was detected. We therefore use the variable-wise *mean* instead of median for setting $\gamma_1,\ldots,\gamma_D$ here.

Analysis Demonstration with `CV-aggregation` (Algorithm 2)

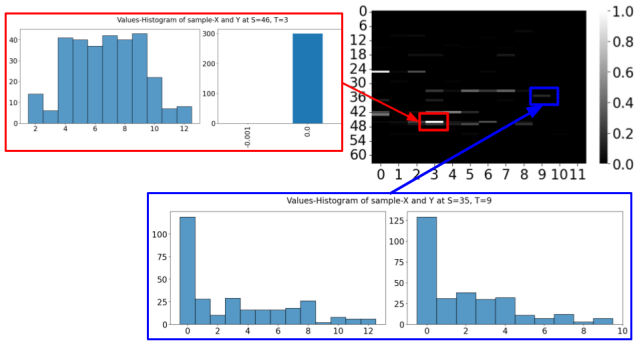


Figure 3.5: The right bottom heat map describes the score matrix $\hat{\Pi} \in \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$, which is normalised so that the highest value is 1. The vertical axis indexes $D_{\text{sensor}} = 64$ sensors, and the horizontal axis indexes the $D_{\text{time}} = 12$ time intervals. The sensor-time pair in red has the highest score, and the sensor-time pair in blue has the lowest score among the selected variables. For each sensor-time pair, the two histograms represent the empirical distributions of the vehicle counts in the datasets $\mathbf{X} = \{X^1,\ldots,X^{400}\} \subset \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ (left) and $\mathbf{Y} = \{Y^1,\ldots,Y^{400}\} \subset \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ (right) at this sensor-time pair. Note that the horizontal axis of each histogram is the number of vehicles observed in the time-interval of 5 minutes (= 300 seconds) per second; thus, for example, the value 0.02 indicates that the $0.02 \times 300 = 6$ vehicles were observed in the time interval.

Without any knowledge about the fact that two roads are blocked in the scenario Q dataset

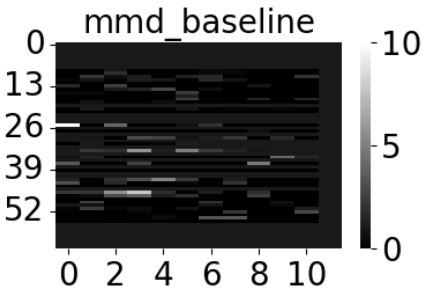
Y, we apply `CV-aggregation` to the two datasets **X** and **Y** and see whether it can identify sensor-time pairs where the traffic flows are affected by this road blocking (the results of other methods are in the next section). Note, however, that it is not straightforward to define the "ground truth" for this experiment, as the blocking of the roads would affect not only the traffic flows of the blocked roads during the 40 minutes of the road blocking, but also the traffic flows of the surrounding roads and subsequent periods. Therefore, we demonstrate explanatory data analysis using the proposed method.

As a result of applying `CV-aggregation`, we obtained 22 variables (= sensor-time pairs) selected from $D = 64 \times 12 = 768$ variables, together with the score matrix $\Pi \in \mathbb{R}^{D_{\text{sensor}} \times D_{\text{time}}}$ in Algorithm 2. Figure 3.5 describes $\hat{\Pi}$ as a heat map, along with the sensor-time pairs with the highest (highlighted in red) and lowest scores (highlighted in blue) among the selected variables. The two histograms for each sensor-time pair represent the empirical distributions of the 400 vehicle counts in the two datasets $\mathbf{X} = \{X^1, \ldots, X^{400}\}$ and $\mathbf{Y} = \{Y^1, \ldots, Y^{400}\}$ at this sensor-time pair.

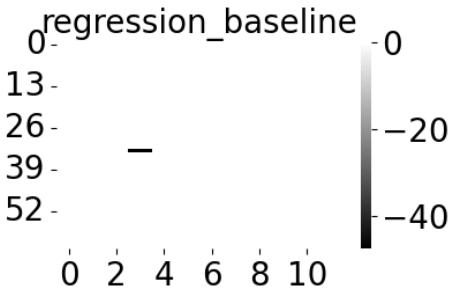
At the sensor-time pair with the highest score (red), the traffic patterns in the two datasets differ significantly. Indeed, the left histogram indicates active traffic flows in the dataset **X**, while the right histogram shows no traffic flow in the dataset **Y** at this sensor-time pair. In an actual explanatory analysis, one could hypothesise that there is a road blocking around this sensor and investigate the neighbouring sensors and time periods. The sensor-time pair associated with the lowest score among the selected variables (blue) also provides insights into the difference between the two datasets **X** and **Y**. While the two histograms look similar, there are subtle differences in the probability masses in the lower and upper limits of the histograms. This time interval is around the end of the road blocking, and thus this sensor-time pair provides a hint about the end time and final effects of the unknown incident (= road blocking) in the scenario Q.

Results of Other Approaches

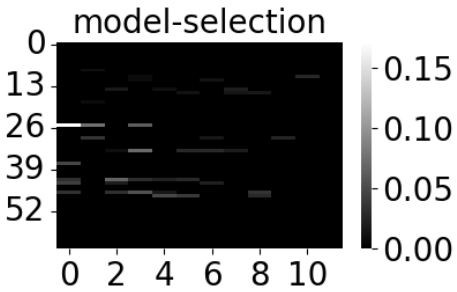
We perform the same experiments using `regression-baseline`, `mmd-baseline` and `model-selection`. Figure 3.6 visualises the score matrix obtained from each approach. The result of `regression-baseline` indicates an apparent failure, as it produces false positives and does not effectively identify relevant variables. `mmd-baseline` selects 18 variables, fewer than the 22 variables selected by `CV-aggregation`. On the other hand, `model-selection` selects 40 variables, which surpasses the other approaches.



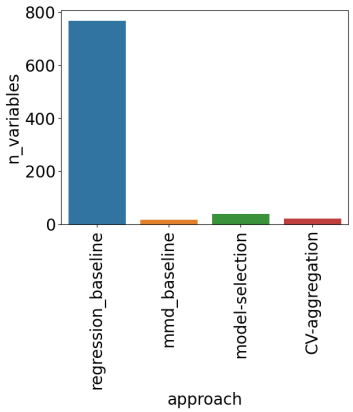
(a) Optimised ARD weights obtained without regularisation (`mmd-baseline`).



(b) Coefficients of the L1-penalised linear regression (`regression-baseline`).



(c) Optimised ARD weights obtained with Algorithm 1 (`model-selection`).



(d) The number of variables selected by each method.

Figure 3.6: Score matrices obtained by (a) `mmd-baseline`, (b) `regression-baseline` and (c) `model-selection`, represented as heatmaps, along with (d) the number of variables selected by each method.

3.5.4 Demonstration: Image Comparison Analysis

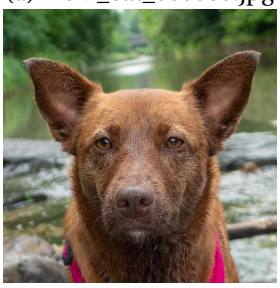
We demonstrate the application of the proposed approaches in analysing image datasets. We use a subset of the AFHQ dataset (Choi et al., 2020) consisting of high-resolution images of cats' and dogs' faces, as described in Figure 3.7. The aim here is to select variables (= pixel coordinates) that indicate differences between cats' and dogs' faces.

As preprocessing, we downscale the resolution of each image data to 64×64 pixels, resulting in a total of $4,096$ dimensions. We convert each pixel's RGB values into a greyscale value ranging from 0 to 255. For our experiments, we randomly selected 1,000 images from the AFHQ dataset containing 5,000 images. Here, we optimise the ARD weights using the Adam optimiser using 100 batches in each iteration. See Section 3.5.1 for other details.

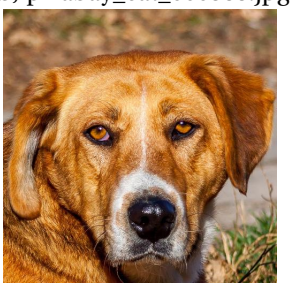
Figure 3.8 describes the score matrices obtained with `mmd-baseline`, `model-selection` and `CV-aggregation`. As `CV-aggregation` performs the best in our other assessments, we

(a) flickr_cat_000060.jpg

(b) pixabay_cat_000588.jpg



(c) flickr_dog_000039.jpg



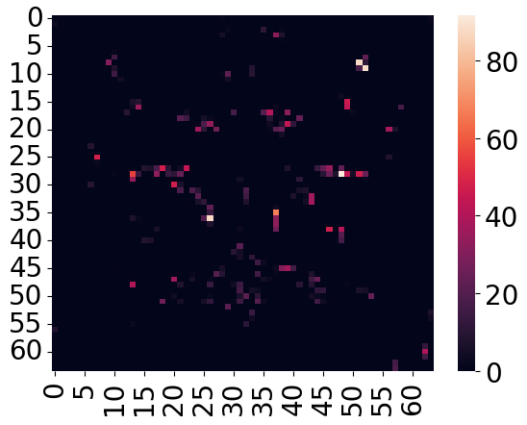
(d) pixabay_dog_000969.jpg

Figure 3.7: Examples of cats' and dogs' face images from the AFHQ dataset. The caption of each image indicates the file name in the dataset.

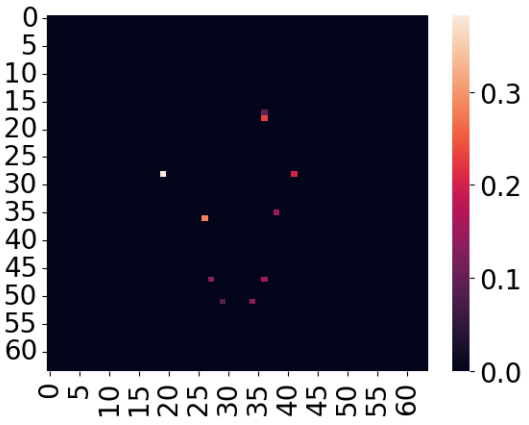
examine the variables selected by `CV-aggregation`. Figure 3.9 shows some cats' and dogs' face images on which the selected variables (pixel positions) are highlighted as yellow dots.

By examining Figure 3.9, we can observe that the selected variables capture discrepancies between cats' and dogs' faces in mainly four areas: 1) the shape of the left eye, 2) the shape of the right eye, 3) the shape from the nose to the chin, and 4) the position of the forehead. Regarding the eyes' shapes, cats' eyes are typically wide and oval-shaped, while dogs' eyes tend to be elongated ellipse-shaped. In addition to the shapes/positions of parts of a face, colour depths (i.e., greyscale values from 0 to 255) may also contribute to the discrepancies between cats' and dogs' faces. For example, cats' eyes tend to be transparent (resembling glass), while dog's eyes are darker. Differences between cats and dogs in the shape from the nose to the chin are also noticeable. A cat's nose is small and pointy, and its chin shapes a gentle curve, while a dog has a larger and rounded black-blob-shaped nose and a more acute chin.

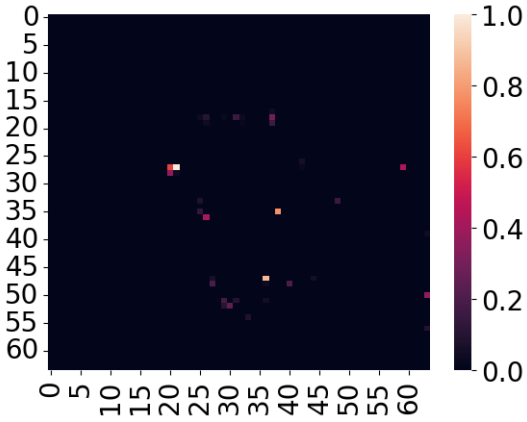
For further analysis, we compare the histograms of greyscale values in the datasets at specific pixel positions: the pixel at the 27th row and the 21st column (located on the upper left of the left eye) and at the pixel at the 47th row and the 36th column (located on the right side of the nose). These pixels have the highest and second scores given by `CV-aggregation`. Figure 3.10 shows the histograms at these pixels. Examining the histogram of the pixel at the 27th row and the 21st column, we observe that dog images have higher frequencies in low greyscale values than cat images. Since low greyscale values indicate darker colours, this may indicate



(a) Optimised ARD weights obtained by `mmd-baseline`



(b) Optimised ARD weights obtained by `model-selection`



(c) The score matrix obtained from `CV-aggregation`

Figure 3.8: Score matrices obtained with (a) `mmd-baseline`, (b) `model-selection` and (c) `CV-aggregation`.

the presence of dark iris colours in dogs' eyes. Similarly, for the pixel at the 47th row and the 36th column, dog images have higher frequencies in low greyscale values (dark colours). This pixel is often associated with a part of the nose, implying differences in the nose shapes and colours between cats and dogs.

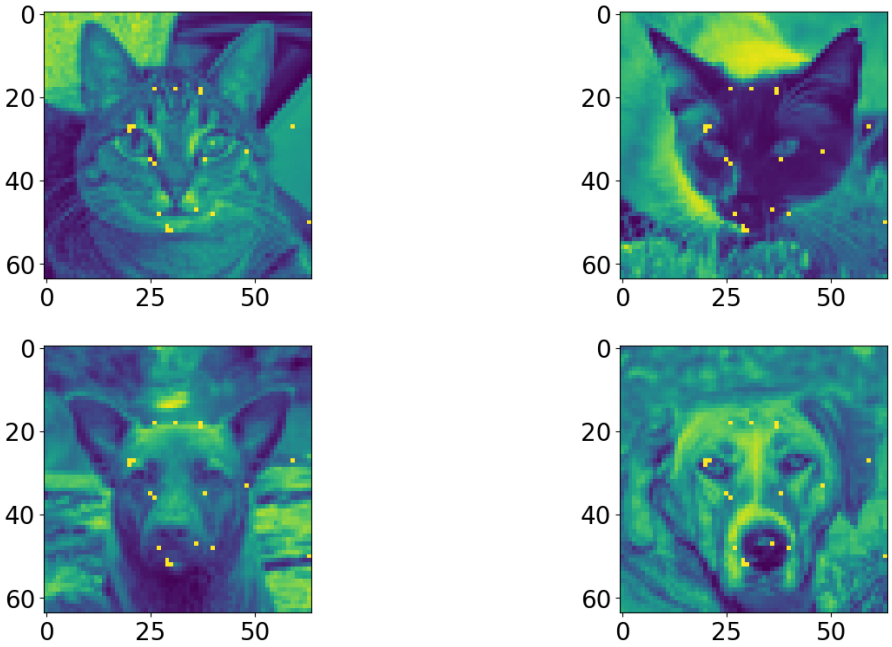


Figure 3.9: Example images of cats' and dogs' faces after preprocessing (down-scaled to 64×64 pixels, with colours transformed to greyscale values), corresponding to Figure 3.7, where the pixel positions (=variables) selected by `CV-aggregation` are highlighted in yellow.

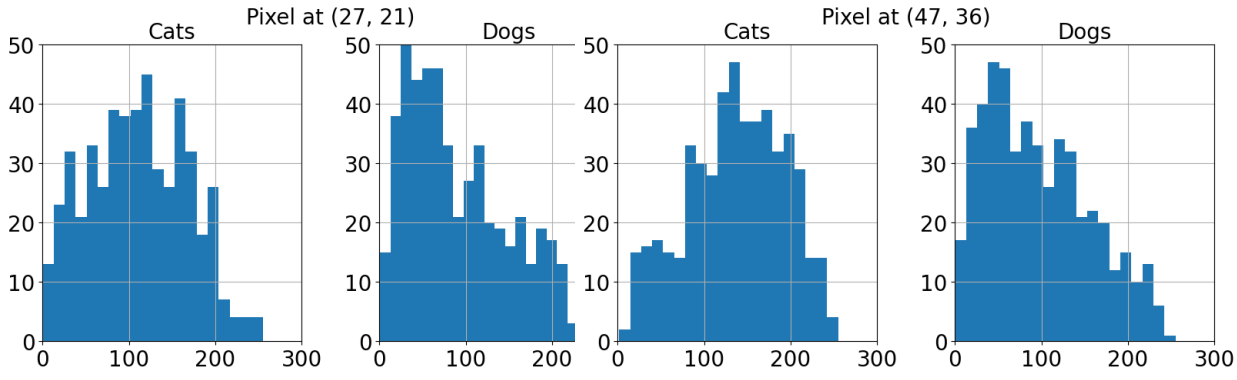


Figure 3.10: Histograms of greyscale values for the pixel at the 27th row and the 21st column (left two histograms) and those at the 47th row and 36th row (right two histograms).

Chapter 4

Variable Selection on High-Dimensional Time-Series Data

In this chapter, we introduce an extension of the proposed variable selection to high-dimensional time-series data. We first introduce related works in Section 4.1. Section 4.2 describes the proposed framework for the variable selection. We note that the proposed framework is generic, thus a choice of variable selection algorithm is left to users. Empirical assessments are in Section 4.4, and evaluations are too. Demonstrations are in Section 4.5 and Section 4.6.

We are interested in comparing a single (or a few of) pair(s) of time-series data, where each time step is high dimension. This comparison case is ubiquitous in handling time-series data; however, a common example case is in model validation of spatio-temporal models. The main interest is in identifying differences in the spatio-temporal model outcome compared with the counterpart (either the real world data or another model).

Since each time step consists of high-dimensional data and such numerous time steps exist in series; therefore, manual comparison and analysis are challenging. Manual comparison using graphical representation (e.g. animation) is probably an analysis approach (Balci, 1994). However, this approach is subjective analysis and requires a lot of time and effort.

A solution to this problem is variable selection by splitting a set of time steps into a certain number of buckets and selecting variables in each bucket, that are significantly different between the two time-series data.

There are two differences from the previous chapter, Chapter 3. First, this chapter focuses on a pair of time-series data, while Chapter 3 can handle both time-series data (e.g. the demonstration in Section 3.5.3) or others. Second, Chapter 3 assumes that a certain number of samples are available and that the comparison considers several possibilities (e.g. sampling

from a model with various random seeds). On the other hand, in this chapter, we target a few specific conditions (e.g. a few random seeds) and compare the time-series data precisely under the conditions. The question is: "Which variables are different in specific time periods". The proposed method in this chapter is a more specific and detailed comparison than the previous section. The proposed method in this chapter can answer the question, by showing the variables at a certain time.

The proposed method in this chapter has a second effect regarding the available sample size. In certain cases, it is impossible or difficult to prepare enough samples for comparison. Probably, a model, especially a simulation model, requires a long time to execute, and it is difficult to prepare a sufficient number of samples. For example, Chen et al. (2017) report that a climate simulation model which requires ten hours in a single machine of 32 CPUs needs even so four hours with a distributed computer cluster equipped with 160 CPUs[1]. Other examples of simulation in long execution time are also in (Behrens and Dias, 2015; Bharti et al., 2023; Zych et al., 2022; Najdek et al., 2021; Vijitpornkul and Marurngsith, 2015). The other case of the limited sample size is where the real world data is limited, such as the case of natural disasters. In both cases, the available pair is a few; therefore, the proposed method in this chapter is effective.

4.1 Related Work

We briefly introduce related works that focus on the variable selection task for time-series data and TST for time-series data. To our knowledge, quite a few studies focus on the intersection of TST and variable selection for high-dimensional time-series. First, we introduce a related work in the intersection. TST for time-series and variable selection for time-series are fields with long histories of research; hence, we review remarkable studies separately.

4.1.1 Two-Sample Testing and Variable Selection for High-Dimensional Temporal Data

Cortés et al. (2020) proposed TST for grid meshed time-series data (ex. grid mesh of the earth temperature) along with detecting grids that contribute to rejecting the null hypothesis H_0. The number of grids corresponds to the dimensionality D in our work. Combining all tests of the grids in a multiple-testing framework, they propose a method to control the family-wise error rate and identify which grids are significant in rejecting H_0. These processes can be regarded as a variable selection task, even if the authors do not explicitly call it so. A major

[1]The simulation model will become more lightweight in the future; therefore, the computational costs will be reduced. However, society and the scientific community would hope for more accurate and more realistic simulation models, which require more computational power. Thus, we assume that the computational costs will remain an issue.

difference from our work is in the variable treatment: Cortés et al. (2020) treat the variables as a set of independent uni-variables time-series. Since the test is uni-variate, it may not handle inter-dependent variables, and the variable detection result will be redundant. On the other hand, we treat it as a D dimensional vector over time steps and cope with the inter-dependent variable structure. Another major difference is in computational costs. The approach executes permutation tests per grid, it requires a lot of tests as the dimension (grid) increases, whereas our proposal requires a smaller number of tests. Despite the drawback in handling variables, the multiple-testing approach could serve as a powerful analytical tool when integrated with the chosen variables outlined in our proposal. By reducing the number of permutation tests, we can mitigate the computational cost drawback of this work.

4.1.2 Two-Sample Testing for Time-series Data

TST for *Functional Data Analysis (FDA)* regards an observed time-series as a function. Wynne and Duncan (2022) proposed solving the FDA TST problem using Maximum Mean Discrepancy (MMD) constructed on the functional data space. This work regards a sequence of observed values as a sequence of data points evaluated by a function, constructing a kernel function on the function space, named Squared-Exponential T kernel ("SE-T kernel"). The SE-T kernel enables the construction of an MMD on the function space using the SE-T kernel between two probability distributions on a Hilbert space of functions. A key concept of the SE-T kernel is that a linear operator "T" on the Hilbert space of the functions maps to a real space. This work explores various choices of the "T" linear operator, such as functional principle components or squaring feature expansion. They demonstrate the effectiveness of the proposed method with a small number of pairs of observations (samples), such as five pairs. Thus, this method may be suitable for testing pairs of time-series data, although it has yet to undergo empirical assessments.

Change-point detection is a task designed to detect specific points where the distribution of time-series data changes, i.e. detecting shifts in signal patterns. Sinn et al. (2012) handle the change point detection problem using MMD that compares two probability distributions before and after a certain time step t. For representing time-series as the empirical distributions, this work constructs the ordinal pattern describing increasing or decreasing observed values. For instance, if values are monotonically increasing over time intervals $(t-2, t-1, t)$, the time-series is represented by the ordinary pattern of "increase at $[t-2, t-1]$", "increase at $[t-1, t]$". This method counts frequencies of these ordinal patterns and constructs empirical distributions before and after the study point t. Scharwächter and Müller (2020) handle an event detection task, a variant of the change point detection, where the given data consists of a time-series paired with an event sequence. The whether these events goal is to determine whether the event influences the time-series or not. Thus, the null hypothesis is $H_0 : P(X_t|E_{<t}) = P(X_t)$, where X_t is a value at the time t and $E_{<t}$ are all events until the time t,

meaning there is no information shared between the time-series and the events of interest. For conducting this TST, Scharwächter and Müller (2020) proposed an algorithm consisting of multiple TST. Having a hyperparameter $K \in \mathbb{N}$ for adjusting a range of time steps after event occurrences, the algorithm executes a set of TST for all pairs of $k, k' \in \{1, ..., K\}, k \neq k'$. For representing the time-series data, this work composes two samples by collecting a set of time steps at k step after all occurred events and the same procedure for k'. Each TST does not represent time-series structure, however, the algorithm can handle the time-series by multiple testing over a pair of k, k' ranges. Both Sinn et al. (2012); Scharwächter and Müller (2020) employ unique sample representation instead of the given time-series data. Given that our proposal does not consider temporal relations directly, these representations might offer valuable alternatives.

4.1.3 Variable selection for Time-series Data

A common approach in time-series forecasting involves predicting the state at the next time step, hence features variable selection for predicting the next time step. For example, a random forest model predicts the next time step by squeezing out effective variables for the prediction (Tyralis and Papacharalampous, 2017). Time-series data are well-studied in the field of functional data analysis (FDA). Aneiros et al. (2022) reviewed variable selection and FDA, mostly focusing on regression problems. The Lasso regularisation (Tibshirani, 1996) is a well-known technique for variable selection in regression models, which can also be applied in nonlinear regression models within FDA, using techniques like B-spline expansions to approximate the non-linear functions. However, these variable selection methods do not guarantee the rejection of the null hypothesis as TST.

4.2 Proposed Framework

4.2.1 Data and Purpose

Suppose we have two D-by-T matrices $X \in \mathbb{R}^{D\times T}$ and $Y \in \mathbb{R}^{D\times T}$, each representing a time-series of D-dimensional vectors of length T:

$$X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{D\times T}, \quad \text{where} \quad \boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,D})^\top \in \mathbb{R}^D \quad (t = 1, \ldots, T),$$
$$Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T) \in \mathbb{R}^{D\times T}, \quad \text{where} \quad \boldsymbol{y}_t = (y_{t,1}, \ldots, y_{t,D})^\top \in \mathbb{R}^D \quad (t = 1, \ldots, T).$$

For example, X may be generated from a traffic simulator, and Y from the same simulator but under a different parameter setting. The comparison of X and Y amounts to analysing the effects of changing the parameter setting. Each element $x_{d,t}$ (or $y_{d,t}$) may represent the observed data, such as traffic flow, from the d-th sensor (where $d = 1, \ldots, D$) during the t-th

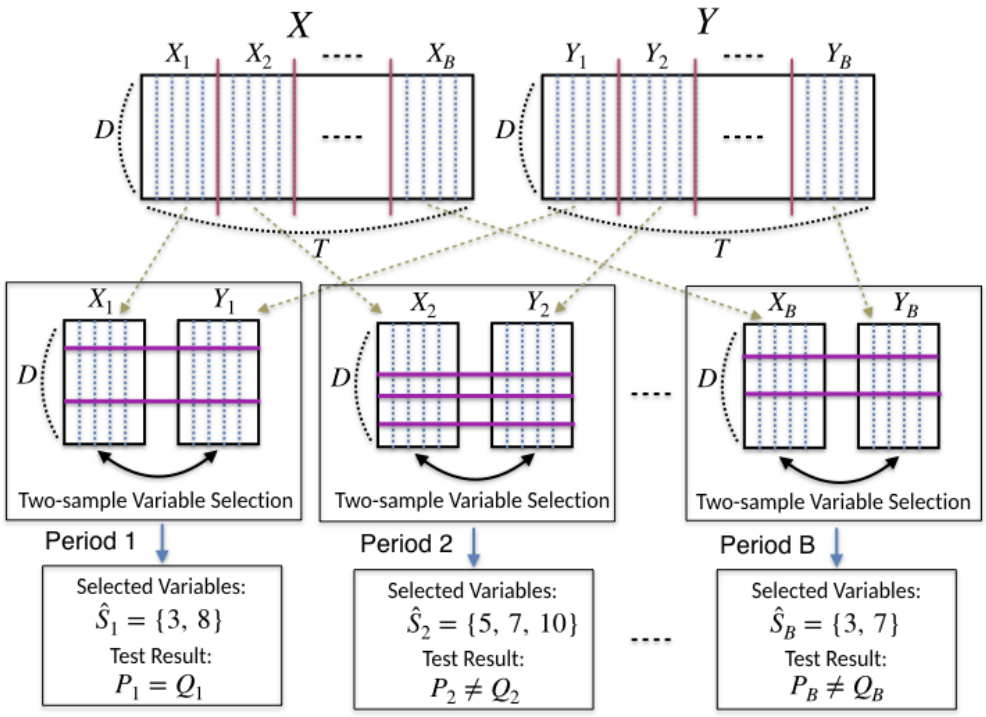


Figure 4.1: Illustration of the Proposed Framework in Chapter 4

time interval (where $t = 1, \ldots, T$) in the simulated traffic system.

We are interested in how the two time-series X and Y differ. In particular, the question is; "at which times $t = 1, \ldots, T$ and for which variables (dimensions) $d = 1, \ldots, T$, do X and Y differ significantly?" In the traffic simulation example, such times and variables are the times and locations in the traffic system where the traffic flows differ due to the change of the parameter setting. We will propose a framework for identifying such differing times and variables.

4.2.2 Procedure

We describe here the proposed procedure, which is illustrated in Figure 4.1 and summarised in Algorithm 3.

1. Time splitting. The main idea is to split the entire time interval $[1, T] := (1, 2, \ldots T)$ into B disjoint subintervals for some $B < T$:

$$[0, T] = [t_0 + 1, t_1] \cup [t_1 + 1, t_2] \cup \cdots \cup [t_{B-1} + 1, t_B], \tag{4.1}$$
$$\text{where} \quad 0 =: t_0 < t_1 < t_2 < \cdots < t_{B-1} < t_B := T.$$

These are the time points where we split the entire time interval $[1, T]$. Note that here we use the notation $[t_{b-1}+1, t_b] := (t_{b-1}+1, t_{b-1}+2, \ldots, t_b)$ for $b = 1, \ldots, B$. For example, one may split $[1, T]$ into equal-size intervals: If $T = Bm$ for some $m \in \mathbb{N}$, one sets

$$t_1 = m, \quad t_2 = 2m, \quad \ldots, \quad t_{B-1} = (B-1)m,$$

so that each interval $[t_{b-1}+1, t_b]$ consists of m time points. We use this uniform splitting in our experiments.

The idea is that we split each of X and Y according to the subintervals Eq. (4.1), and make a comparison on each subinterval. That is, we first split X into B sub time-series $X_1, \ldots, X_B$ and Y into B sub time-series $Y_1, \ldots, Y_B$:

$$\begin{aligned} &X = (X_1, \ldots, X_B), \text{ where } X_b := (\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b}) \in \mathbb{R}^{D \times (t_b - t_{b-1})} \quad (b = 1, \ldots, B), \\ &Y = (Y_1, \ldots, Y_B), \text{ where } Y_b := (\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b}) \in \mathbb{R}^{D \times (t_b - t_{b-1})} \quad (b = 1, \ldots, B). \end{aligned}$$

For the b-th subinterval ($b = 1, \ldots, B$), the sub time-series X_b is a set of D-dimensional vectors $\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b} \in \mathbb{R}^D$, and the sub time-series Y_b is a set of D-dimensional vectors $\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b} \in \mathbb{R}^D$.

2. Two-sample variable selection. For each subinterval $b = 1, \ldots, B$, we now perform two-sample variable selection and TST. Let $\rho_{\text{train}} \in (0,1)$ be a constant. We randomly split each of $X_b = (\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b})$ and $Y_b = (\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b})$ into "training" and "test" subsets in the ratio $\rho_{\text{train}} : 1 - \rho_{\text{train}}$; denote the resulting subsets by

$$\begin{aligned} &X_b^{(\text{tr})} \in \mathbb{R}^{D \times (t_b - t_{b-1})\rho_{\text{train}}}, \quad X_b^{(\text{te})} \in \mathbb{R}^{D \times (t_b - t_{b-1})(1-\rho_{\text{train}})}, \\ &Y_b^{(\text{tr})} \in \mathbb{R}^{D \times (t_b - t_{b-1})\rho_{\text{train}}}, \quad Y_b^{(\text{te})} \in \mathbb{R}^{D \times (t_b - t_{b-1})(1-\rho_{\text{train}})}. \end{aligned}$$

Here, we assume $(t_b - t_{b-1})\rho_{\text{train}}$ and $(t_b - t_{b-1})(1 - \rho_{\text{train}})$ are integers for simplicity.

We then perform two-sample variable selection on $X_b^{(\text{tr})}$ and $Y_b^{(\text{tr})}$ to select a subset of variables (or dimensions/features)

$$\hat{S}_b \subset \{1, 2, \ldots, D\}$$

on which the two datasets $X_b^{(\text{tr})}$ and $Y_b^{(\text{tr})}$ differ significantly. Here, as we propose a generic framework, the algorithm choice is left to the user. The user can choose an algorithm that is most suitable for their purpose (in terms of, e.g., available computational resources, familiarity with the algorithm, etc.).

Lastly, for each $b = 1, \ldots, B$, one performs a two-sample permutation test based on the datasets

$X_b^{(\mathrm{te})}$ and $Y_b^{(\mathrm{te})}$ using only the selected variables $\hat{S}_b$ to obtain a p-value,

$$0 \leq p_b \leq 1.$$

Here, the null hypothesis is $H_0 : P_b = Q_b$, where P_b and Q_b are the stationary distributions of $X_b = (\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b})$ and $Y_b = (\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b})$, respectively. If the p-value is small (e.g., $p_b \leq 0.05$[2]), there is an indication that these distributions are different, suggesting that there is a difference in the two time-series on the b-th interval. As for variable selection, the concrete algorithm depends on one's choice: one can use TST based on, e.g., the MMD, the sliced Wasserstein distance (Bonneel et al., 2014), etc; see Section 4.3 for details.

Remark 1 (Multiple Pairs of Time-Series)**.** *While this thesis focuses on the situation where only one pair of time-series, $X \in \mathbb{R}^{D\times T}$ and $Y \in \mathbb{R}^{D\times T}$, are available, here we describe one possible way to extend the proposed framework when there are multiple pairs $(X^{(1)}, Y^{(1)}), \ldots, (X^{(N)}, Y^{(N)})$ of time-series are available, where $X^{(i)} \in \mathbb{R}^{D\times T}$ and $Y^{(i)} \in \mathbb{R}^{D\times T}$ for $i = 1, \ldots, N$ with $N \in \mathbb{N}$.*

For each $i = 1, \ldots, N$, we apply Algorithm 3 to the pair $(X^{(i)}, Y^{(i)})$ to obtain a sequence of selected variables $\hat{S}_1^{(i)}, \ldots, \hat{S}_B^{(i)}$, where $\hat{S}_b^{(i)} \subset \{1, \ldots, D\}$ for $b = 1, \ldots, B$. Then, for each subinterval $b = 1, \ldots, B$, we aggregate the N sets of selected variables by taking their union, $\hat{\mathfrak{S}}_b = \cup_{i=1}^N \hat{S}_b^{(i)}$, or taking their intersection $\hat{\mathfrak{S}}_b = \cap_{i=1}^N \hat{S}_b^{(i)}$. We leave an investigation of such aggregation algorithms for future research.

4.2.3 Example and Discussion

Figure 4.2 shows an example of the time-splitting operation. The top tow figures describe two given time-series data $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{D\times T}$ and $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \in \mathbb{R}^{D\times T}$ with $D = 5$ variables and $T = 130$ time points. The two variables indicated by the violet and light-blue trajectories are generated from different stochastic processes for X and Y.

We split each time-series into $B = 3$ subintervals, with $t_1 = 50$, $t_2 = 100$ and $t_3 = 130$, which are shown in the bottom two figures. Here, on each subinterval $b = 1, \ldots, B$, we randomise the time order of data vectors within $X_b = (\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b})$ and those within $Y_b = (\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b})$. This is to illustrate that, by applying two-sample variable selection and TST on X_b and Y_b, *we essentially treat data vectors in X_b (and those in Y_b) as independent*, since we break the time structure within X_b and that within Y_b when applying these approaches.

Let us discuss how the proposed approach would work for this example.

- Regarding the light-blue trajectories, whose underlying stochastic processes are different for X and Y, the empirical distributions of X_b and Y_b for this variable are noticeably

[2]This is not our suggestion of using $p = 0.05$ for decision-making. As Wasserstein and Lazar (2016) recommends, the p-value should not be a supportive criterion of any scientific or business decisions.

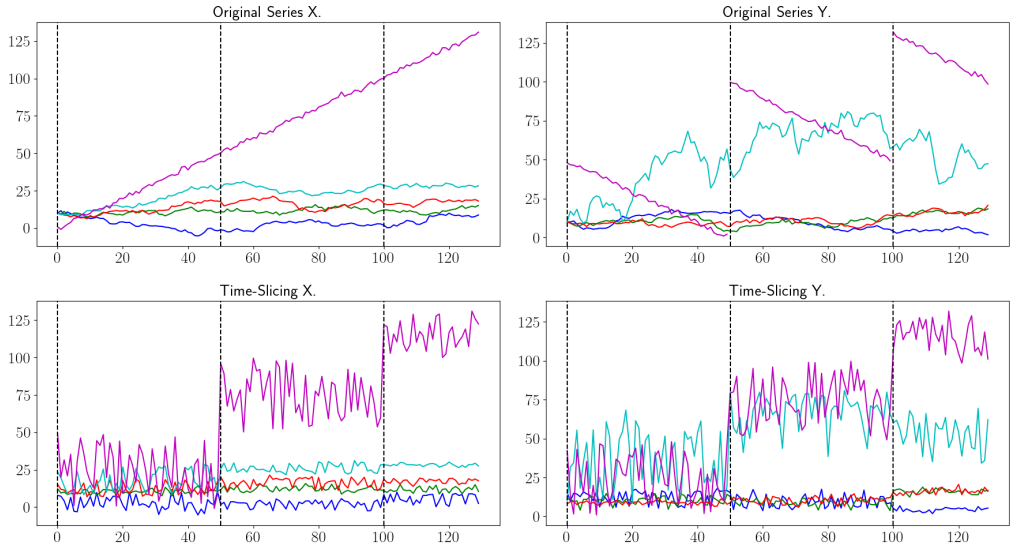


Figure 4.2: An example of time-splitting applied to two time-series data $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{D \times T}$ and $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \in \mathbb{R}^{D \times T}$ with $D = 5$ and $T = 130$, with $t_1 = 50$, $t_2 = 100$ and $t_3 = 130 = T$. The top plots show the two-time-series X and Y, where 5 different colours correspond to the 5 variables (or dimensions). In each plot, the horizontal axis represents time points, and the vertical axis the values of each variable. The two variables represented by the violet and light-blue colours follow different stochastic processes for X and Y. The bottom figures show the results of applying the time-splitting and randomisation in each subinterval.

different for each subinterval $b = 1,2,3$. Therefore, a two-sample variable selection would be able to identify this variable as differing for each of $b = 1,2,3$.

- The violet trajectories, which clearly differ for X and Y, are deliberately constructed to illustrate an "unfortunate" situation where time-slicing makes the two time-series indistinguishable. After the time-slicing, the empirical distributions of X_b and Y_b for this variable become very similar (actually, here, we constructed them so that they become exactly the same) for each interval $b = 1,2,3$. In this case, the proposed approach would not be able to detect the difference between the two time-series for this variable.

The issue with the violet trajectories could have been avoided if there were many more time-splicing points (i.e., B is larger) so that each subinterval was shorter and thus could capture the difference in the two trajectories. This is demonstrated in the experiments in Section 4.4.

This example suggests that the proposed framework would work well if

1. the time length of each subinterval $[t_{b-1}+1, t_b]$ is short enough so that the $X_b = (\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b})$ and $Y_b = (\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b})$ are (approximately) stationary, while

Algorithm 3 Proposed Framework
▷ **Input:** Two time-series data $X := (\mathbf{x}_1, ..., \mathbf{x}_T) \in \mathbb{R}^{D\times T}$ and $Y := (\boldsymbol{y}_1, ..., \boldsymbol{y}_T) \in \mathbb{R}^{D\times T}$. Time-splitting points $0 =: t_0 < t_1 < \cdots < t_{B-1} < t_B := T$. Ratio $0 < \rho_{\text{train}} < 1$ of training-test splitting on each subinterval.
▷ **Output:** Selected variables $\hat{S}_b \subset \{1, 2, \ldots, D\}$ for each $b = 1, \ldots, B$. P-value $p_b \in [0, 1]$ of TST for each $b = 1, \ldots, B$.

1: **for all** $b = 1, \ldots, B$ **do**
2: Set $X_b = (\boldsymbol{x}_{t_{b-1}+1}, \ldots, \boldsymbol{x}_{t_b}) \in \mathbb{R}^{D\times(t_b - t_{b-1})}$ and $Y_b = (\boldsymbol{y}_{t_{b-1}+1}, \ldots, \boldsymbol{y}_{t_b}) \in \mathbb{R}^{D\times(t_b - t_{b-1})}$.
3: Randomly split X_b into $X_b^{(\text{tr})} \in \mathbb{R}^{D\times(t_b - t_{b-1})\rho_{\text{train}}}$ and $X_b^{(\text{te})} \in \mathbb{R}^{D\times(t_b - t_{b-1})(1-\rho_{\text{train}})}$.
4: Randomly split Y_b into $Y_b^{(\text{tr})} \in \mathbb{R}^{D\times(t_b - t_{b-1})\rho_{\text{train}}}$ and $Y_b^{(\text{te})} \in \mathbb{R}^{D\times(t_b - t_{b-1})(1-\rho_{\text{train}})}$.
5: Perform two-sample variable selection on $X_b^{(\text{tr})}$ and $Y_b^{(\text{tr})}$ to obtain $\hat{S}_b \subset \{1, 2, \ldots, D\}$.
6: Perform a permutation TST on $X_b^{(\text{te})}$ and $Y_b^{(\text{te})}$ using the selected variables $\hat{S}_b$ to obtain a p-value $p_b \in [0, 1]$ for the null hypothesis $P_b = Q_b$.
7: **end for**

2. the number of sample vectors $t_b - t_{b-1}$ in X_b and Y_b is large enough so that the X_b and Y_b are statistically distinguishable.

In point 1, the "time length" means that in the physics sense; recall that $t_b - t_{b-1}$ is the *number* of time steps within the subinterval, not the time length. For example, suppose we consider a traffic simulation from 8 am to 9 am and set the total number of time steps to $T = 1,000$. Then the total time length is 3,600 seconds, and the time length of one time step is $3,600/1,000 = 3.6$ seconds. If the number of subintervals is $B = 4$, and the slicing points are $t_1 = 250$, $t_2 = 500$, $t_3 = 750$ and $t_4 = 1,000$, then the time length of each subinterval is $3,600/4 = 900$ seconds, and the number of time steps within the subinterval is 250.

There is a trade-off between points 1 and 2 above. That is, if we make the time length of each interval shorter (or longer), then the time steps within one interval become smaller (or larger). For example, in the above example, if we change the number of subintervals to $B = 40$, then the time length of each subinterval is 90 seconds, while the number of time steps within the subinterval is 25. In this case, the traffic flows within each subinterval would remain unchanged (and thus can be regarded as stationary), while we only have a small number of observations, making statistical estimation more challenging.

4.2.4 Practical Settings of the Time-splitting Points

The proposed algorithm requires the time-splitting points to disjoint the given time interval $[1, T]$ into B subintervals. Although we set these splitting-points uniformly in our assessments, one may consider other strategies. In practice, there will be two strategies for setting the splitting points.

One strategy is utilising the domain knowledge or pre-observable information. For example, in the traffic simulation example, one may be aware of time-changing points where the traffic flow is expected to change significantly. In this case, one can set the splitting points around these time points. Even when the domain knowledge is not available, one may know tendencies of the time-changing points by a summary statistic (e.g. the average) of the time-series data.

A more sophisticated strategy is to use a change-point detection algorithm to determine the splitting points. Sinn et al. (2012); Scharwächter and Müller (2020) proposed a change-point detection algorithm. The algorithm detects the time points where the distribution of time-series data changes, i.e. detecting shifts in signal patterns. Once the change-point detection algorithm tells us the time points where the distribution changes, we can set the splitting points around these time points.

4.3 Two-Sample Variable Selection Algorithms

The proposed framework in Section 4.2 requires a variable selection algorithm that can identify variables on which two datasets differ significantly. Since the proposed framework is generic, the choice of the variable selection algorithm is left to the user. The first choice is the MMD optimisation based variable selection that we introduce in Chapter 3. For comparison, we also consider other variable selection algorithms that we try in our experiments. We shortly describe them.

4.3.1 Variable Selection by Marginal Distribution Comparisons

One approach to two-sample variable selection is based on the comparisons of one-dimensional marginal distributions. To describe this, let P and Q be probability distributions on $\mathbb{R}^D$. For $d = 1, \ldots, D$, let P_d and Q_d be the marginal distributions of P and Q on the d-th variable. Let $\mathrm{dist}(\mu, \nu)$ be a distance metric between two probability distributions μ and ν on $\mathbb{R}$, such as the MMD and the Wasserstein distance (e.g., Villani, 2009). We then define a weight w_d for each variable $d = 1, \ldots, D$ as the distance of each pair of one-dimensional marginal distributions P_d and Q_d:

$$w_d := \mathrm{dist}(P_d, Q_d) \quad (d = 1, \ldots, D). \tag{4.2}$$

In practice, one needs to estimate these weights using samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \overset{i.i.d.}{\sim} P$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \overset{i.i.d.}{\sim} Q$, where $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,D})^\top \in \mathbb{R}^D$ and $\boldsymbol{y}_i = (y_{i,1}, \ldots, y_{i,D})^\top \in \mathbb{R}^D$ for $i = 1, \ldots, n$. For each $d = 1, \ldots, D$, let $\hat{P}_d := \frac{1}{n}\sum_{i=1}^n \delta_{x_{d,i}}$ be the empirical distribution of $x_{1,d}, \ldots, x_{n,d} \in \mathbb{R}$, and $\hat{Q}_d := \frac{1}{n}\sum_{i=1}^n \delta_{y_{i,d}}$ be the empirical distribution of $y_{1,d}, \ldots, y_{n,d} \in \mathbb{R}$, where δ_z for $z \in \mathbb{R}$ denotes the Dirac distribution at z. The $\hat{P}_d$ and $\hat{Q}_d$ are respectively consistent approximations of P_d and

Q_d. Thus, one can estimate the weights Eq. (4.2) by using $\hat{P}_d$ and $\hat{Q}_d$ as

$$\hat{w}_d := \mathrm{dist}(\hat{P}_d, \hat{Q}_d) \quad (d = 1, \ldots, D).$$

We use the following distance metrics in our experiments.

1. **Wasserstein distance.** We compute the Wasserstein-1 distance between $\hat{P}_d$ and $\hat{Q}_d$, which is identical to the L_1 distance between the cumulative distribution functions $\hat{F}_d$ of $\hat{P}_d$ and $\hat{G}_d$ of $\hat{Q}_d$:

   $$\hat{w}_d := W_1(\hat{P}_d, \hat{Q}_d) = \int_{\mathbb{R}} \left| \hat{F}_d(t) - \hat{G}_d(t) \right| dt \quad (d = 1, \ldots, D).$$

   We compute it using the implementation of `SciPy` library (Virtanen et al., 2020). We use the histogram-based thresholding algorithm in Section 3.3.1 to the weights $\hat{w}_1, \ldots, \hat{w}_D$ to select variables $\hat{S} \subset \{1, \ldots, D\}$.

2. **Maximum Mean Discrepancy.**

   We consider the approach of Lim et al. (2020). For each $d = 1, \ldots, D$, it computes the MMD between $\hat{P}_d$ and $\hat{Q}_d$,

   $$\hat{w}_d := \mathrm{MMD}_{k_1}(\hat{P}_d, \hat{Q}_d),$$

   where the kernel k_1 on $\mathbb{R}$ is the Gaussian kernel whose bandwidth is determined by the median heuristic in Section E or the IMQ (inverse multi-quadratic) kernel.[3] From $\hat{w}_1, \ldots, \hat{w}_D$, it then selects variables $\hat{S} \subset \{1, \ldots, D\}$ based on post-selection inference. This approach requires the number of candidate variables to be selected as a hyperparameter. To avoid confusion with the MMD-based approach in Chapter 3, we call this method `Mskernel`, following the naming of the authors' code, which we use in our experiments.[4]

4.4 Assessment

We investigate how the proposed framework in Algorithm 3 works based on synthetically generated data, for which we know ground-truth variables and time points where changes occur.

4.4.1 Data Setting

We set $D = 5$ and $T = 1,000$. We consider the following two different settings for data generation.

[3] More precisely, Lim et al. (2020) use the linear-time MMD estimator of Gretton et al. (2012a, Section 6).
[4] https://github.com/jenninglim/multiscale-features

Setting 1. We generate two time-series data $X := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{D \times T}$ and $Y := (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \in \mathbb{R}^{D \times T}$, where $\boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,D})^\top \in \mathbb{R}^D$ and $\boldsymbol{y}_t = (y_{t,1}, \ldots, y_{t,D})^\top \in \mathbb{R}^D$ for $t = 1, \ldots, T$, as

$$\begin{aligned} x_{t,d} &= t/T + \epsilon_{t,d} \text{ for } d = 1, \ldots, D \text{ and } t = 1, \ldots, T, \\ y_{t,d} &= \begin{cases} 0.25 + \epsilon'_{t,d} & \text{for } d = 4 \text{ and } t = 251, \ldots, 500, \\ t/T + \epsilon'_{t,d} & \text{otherwise,} \end{cases} \end{aligned} \tag{4.3}$$

where $\epsilon_{t,d} \overset{i.i.d.}{\sim} N(0, \sigma^2)$ and $\epsilon'_{t,d} \overset{i.i.d.}{\sim} N(0, \sigma^2)$ are independence zero-mean Gaussian noises with variance $\sigma^2 = 0.01$. We constructed X and Y so that they differ only in the 4-th variable, from time index $t = 250$ to 500. This information is not known to each method, and the aim is to identify them only from X and Y. Figure 4.3 shows one realisation of X and Y.

Setting 2. We generate $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{D \times T}$ and $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \in \mathbb{R}^{D \times T}$ as

$$\begin{aligned} x_{t,d} &= t/T + \epsilon_{t,d} \text{ for } d = 1, \ldots, D \text{ and } t = 1, \ldots, T, \\ y_{t,d} &= \begin{cases} 0.5 - (t - 250)/T + \epsilon'_{t,d} & \text{for } d = 4 \text{ and } t = 251, \ldots, 500, \\ t/T + \epsilon'_{t,d} & \text{otherwise,} \end{cases} \end{aligned} \tag{4.4}$$

where $\epsilon_{t,d} \overset{i.i.d.}{\sim} N(0, \sigma^2)$ and $\epsilon'_{t,d} \overset{i.i.d.}{\sim} N(0, \sigma^2)$ with $\sigma^2 = 0.01$. See Figure 4.4 for illustration.

Two time-series X and Y differ in the variable $d = 4$ on the period $t = 251, \ldots, 500$, i.e., $x_{251,4}, \ldots, x_{500,4}$ and $y_{251,4}, \ldots, y_{500,4}$, which are the same as the previous setting in Eq. (4.3). The difference from the previous setting is that we generate $y_{251,4}, \ldots, y_{500,4}$ so that their *marginal distribution* becomes the same as $x_{251,4}, \ldots, x_{500,4}$. Therefore, if one of subintervals $[t_{b-1} + 1, t_b]$ contains the period $[251, 500]$ entirely, then the proposed framework would fail to detect the change between $x_{251,4}, \ldots, x_{500,4}$ and $y_{251,4}, \ldots, y_{500,4}$.

Time-splitting Points. We use equally-spaced time-split points $0 < t_1 < \cdots < t_{B-1} < t_B = T$, with two options for the number B of the subintervals: $B = 10$ and $B = 2$.

Train-Test Ratio and Test Statistic. In Algorithm 3, we set $\rho_{\text{train}} = 0.8$, and use the Sliced Wasserstein distance (Bonneel et al., 2014) as a test statistic in the permutation test statistics for each subinterval.

Evaluation Metrics. To evaluate each variable selection method applied to each subinterval, we compute Precision $= |\hat{S} \cap S| / |\hat{S}|$ and Recall $= |\hat{S} \cap S| / |S|$, where $S \subset \{1, \ldots, D\}$ is the set of ground-truth variables and $\hat{S} \subset \{1, \ldots, D\}$ is the set obtained by the variable selection method.

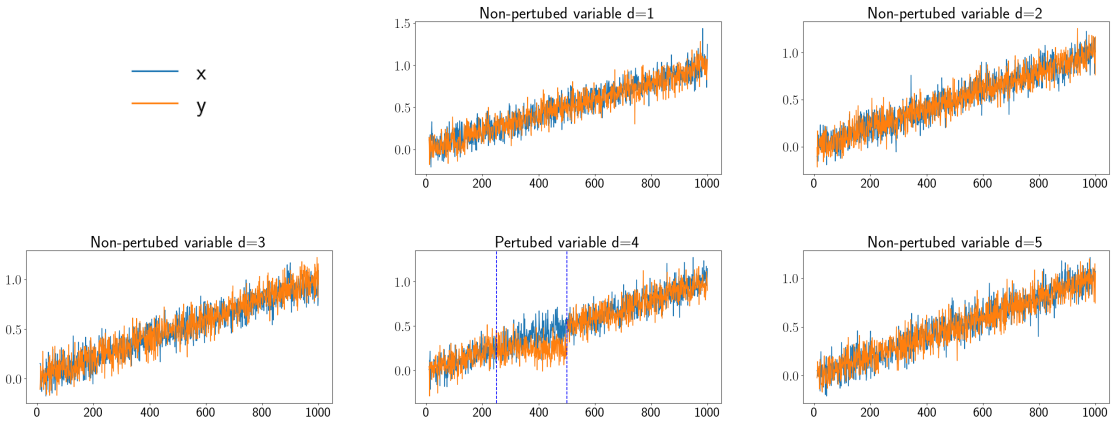


Figure 4.3: Illustration of two time-series data $X \in \mathbb{R}^{D \times T}$ and $Y \in \mathbb{R}^{D \times T}$ in Eq. (4.3). Each subfigure shows the trajectories of X and Y in each variable $d = 1, \ldots, D$, i.e., $x_{1,d}, \ldots, x_{T,d}$ and $y_{1,d}, \ldots, y_{T,d}$. The variable $d = 4$ from $t = 251$ to $t = 500$ is where X and Y differ.

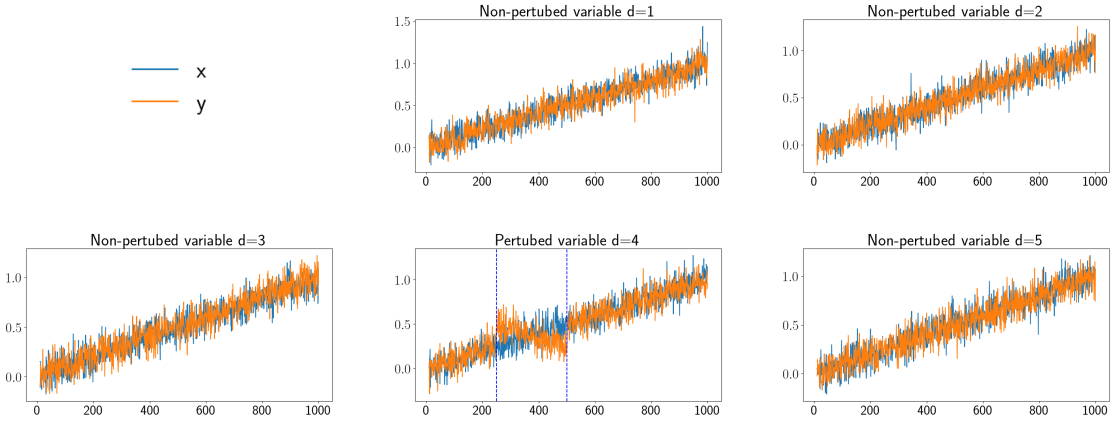


Figure 4.4: Illustration of two time-series data $X \in \mathbb{R}^{D \times T}$ and $Y \in \mathbb{R}^{D \times T}$ in Eq. (4.4). Each subfigure shows the trajectories of X and Y in each variable $d = 1, \ldots, D$, i.e., $x_{1,d}, \ldots, x_{T,d}$ and $y_{1,d}, \ldots, y_{T,d}$. The variable $d = 4$ from $t = 251$ to $t = 500$ is where X and Y differ.

4.4.2 Algorithm Configurations

Variable Selection Methods. For variable selection in Algorithm 3 (Line 5), we consider the following six methods. Three methods are based on MMD-based variable selection described in Section 3.4: (1) `MMD-Selection`, (2) `MMD-CV-AGG`, and (3) `MMD-Vanilla`, which optimises Eq. (3.2) without regularisation (the same method as in (Sutherland et al., 2017)). The other three are those based on marginal distribution comparisons: (4) `Wasserstein`, which computes the Wasserstein-distances of marginal distributions, (5) `Mskernel IMQ`, which computes the MMDs of marginal distributions using the IMQ kernel, and (6) `Mskernel Gaussian`, which uses the Gaussian kernel. As the `Mskernel` methods require the number of variables to be selected as an input, we give the true number of ground-truth variables (which is 1, as defined above).

Configurations for MMD based Variable Selection Methods The MMD optimisation problem is implemented by the Adam optimiser of *Pytorch* (Paszke et al., 2019). The Adam optimiser starts the optimisation with the learning rate of 0.01. A learning rate scheduler[5] monitors the objective value and decreases the learning rate by a factor of 0.5 until it reaches to 0.001. This learning rate scheduler waits for 10 epochs and updates the learning rate if the objective value does not dramatically change otherwise no updates. Two early stopping criteria are set to halt the optimisation, otherwise, the optimisation continues until 9,999 epochs. Please refer to Appendix G.1 for more details of these early stopping criteria.

The `MMD selection` and `MMD CV-AGG` are methods that automatically seek the regularisation parameter λ. `MMD selection` employs an automatic λ search described in Appendix F. An automated method in Appendix D.2 is for selecting the Λ input parameter of `MMD CV-AGG`. `MMD CV-AGG` is with the 5 cross-validations by the 6 candidates of λ.

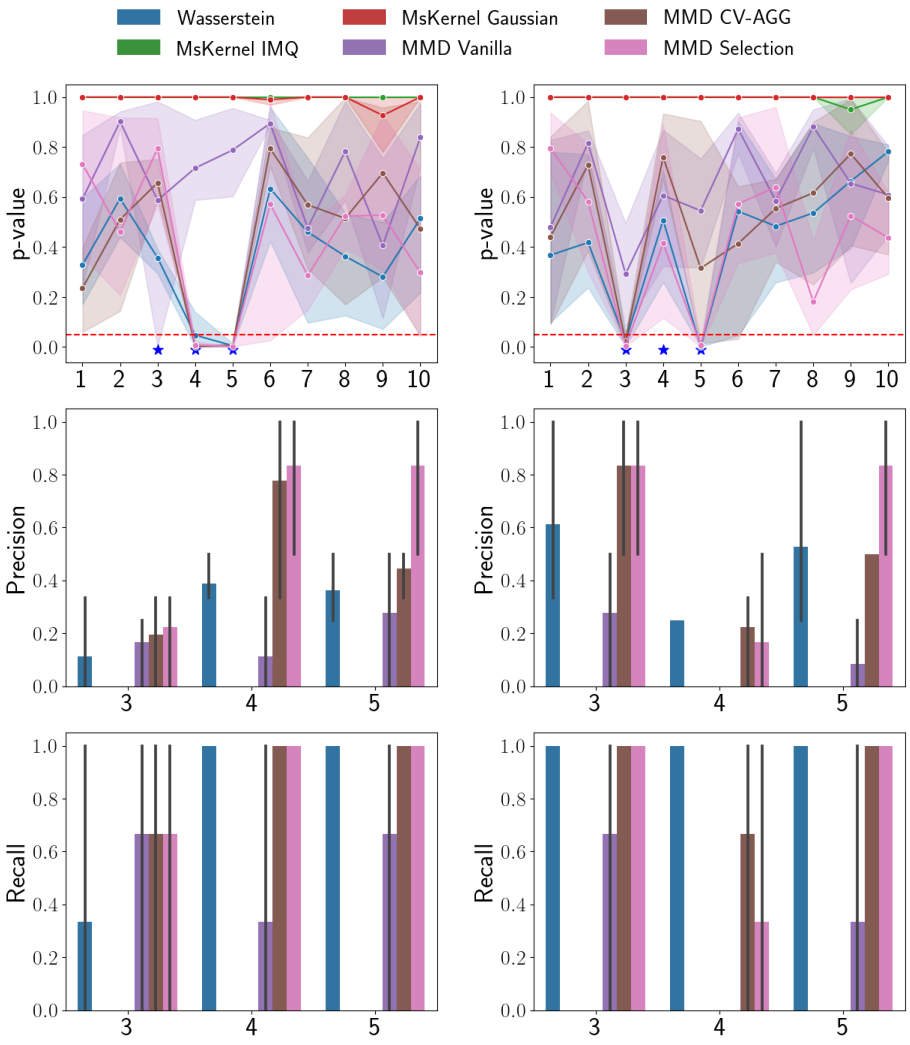
4.4.3 Results

For ease of comparison, we summarise the results for $B = 10$ in Figure 4.5 and $B = 2$ in Figure 4.6. Each of Figures 4.5 and 4.6 shows the means and standard deviations of p-values, precision, and recall over three independent experiments for each Settings 1 and 2. Precision and recall scores are shown only for subintervals overlapping the period $[251, 500]$ where changes in X and Y occur (as the recall is not well-defined for the other subintervals where there exists no ground-truth variables S). `MMD-Vanilla`, `MsKernel-Gaussian` and `MsKernel-IMQ`, failed to identity the differing variable $d = 4$ on the 3rd, 4th and 5th intervals in most settings, leading to zero precision and zero recall. We thus focus on discussing the other three methods.

Let us first study the case $B = 10$ in Figure 4.5, where each subinterval consists of 100 points. The 3rd ($[201, 300]$), 4th ($[301, 400]$) and 5th ($[401, 500]$) intervals overlap the period $[251, 500]$ where X and Y differ. We can make the following observations.

- In Setting 1, `MMD-CV-AGG`, `MMD-Selection` and `Wasserstein` gave low p-values on intervals 4 and 5, suggesting that these methods selected the correct variable ($d = 4$). (Recall that all methods use the same test statistic for the permutation test, so they only differ in the select variables.) Indeed, the recall is 1 for these methods on intervals 4 and 5: the correct variable is included in their selected variables. On the other

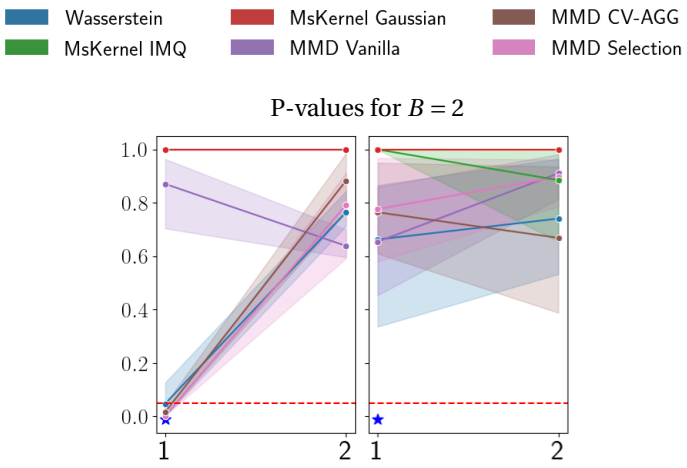
[5] The learning scheduler is defined as `ReduceLROnPlateau` in Pytorch. The documentation is at https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html



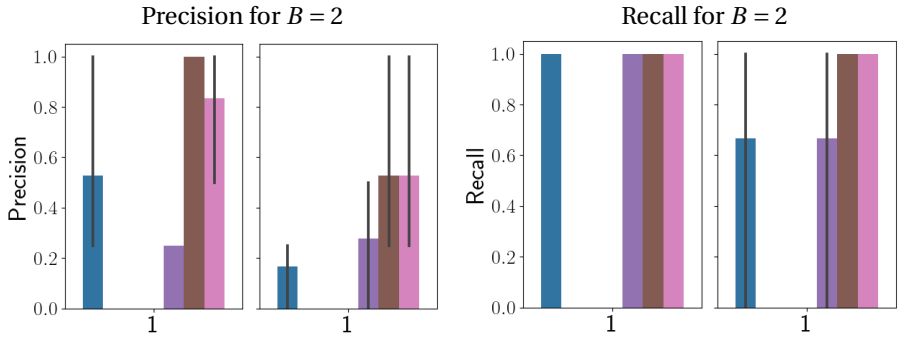
Setting 1 with $B = 10$. Setting 2 with $B = 10$.

Figure 4.5: Results of the experiments in Section 4.4 with $B = 10$ in Setting 1 (left column) and Setting 2 (right column). **Top**: The p-values over the $B = 10$ subintervals obtained by each method, where each line and the shaded area represent the means and standard deviations computed over three independent realisations of X and Y. The red dotted horizontal line indicates the value 0.05. The three blue stars indicate the three subintervals, 3, 4 and 5, that intersect with the changing period [251,500]. **Middle**: The precision scores for each method, where the bars and error bars represent the means and standard deviations over three independent realisations of X and Y. The numbers on the horizontal axis (3, 4 and 5) indicate the subintervals that intersect the changing period [251,500]. **Bottom**: The corresponding recall sores. Note that the precision and recall for `MsKernel-Gaussian`, `MsKernel-IMQ` and `MMD-Vanilla` were zero, so they are not shown.

hand, the precision is about 0.8 for `MMD-CV-AGG` and `MMD-Selection` and about 0.4 for `Wasserstein`, suggesting that `Wasserstein` selected more redundant variables.



Left: Setting 1. Right: Setting 2.



Left: Setting 1. Right: Setting 2.

Left: Setting 1. Right: Setting 2.

Figure 4.6: Results of the experiments in Section 4.4 with $B = 2$. In each figure, the left and right subfigures are the results for Settings 1 and 2, respectively. The top figure shows the p-values, the bottom left the precision scores, and the bottom right the recall scores. For details, see the caption of Figure 4.5.

- On interval 3 in Setting 1, the p-values are not small for these methods. The precision and recall are also lower than intervals 4 and 5. This is because variable selection is more difficult for interval 3. Indeed, the intersection of interval 3 and the differing period $[251,500]$ is $[251,300]$, which contains only 50 points and half of those for intervals 4 and 5. Moreover, as can be seen from Figure 4.3 ($d = 4$), the difference between X and Y appears only slightly on interval 3.

- In Setting 2, the p-values are small on intervals on 3 and 5 for `MMD-CV-AGG`, `MMD-Select` and `Wasserstein`. As the recall is 1 and the precision is from 0.5 to 0.8 for these methods, they were able to select the correct variable $d = 4$ on these intervals. As can be seen in Figure 4.4 ($d = 4$), the difference between X and Y is clear on these intervals, so correct variable selection was also possible for interval 3.

- On the other hand, interval 4 is more difficult for Setting 2, as the difference between X and Y is more subtle, thus producing large p-values. (The sample size of 100 may not be sufficient to identify the difference).
- Producing large p-values outside intervals 3, 4 and 5 is correct, as there is no difference between the generating processes of X and Y.

These observations suggest that Algorithm 3, using `MMD-CV-AGG`, `MMD-Select` or `Wasserstein`, can perform correct variable selection (at least in the considered settings) on the subintervals intersecting with the changing period and thus can detect such a period, if the differences between X and Y are clear enough for given sample sizes.

Now let us see the case $B = 2$ in Figure 4.6, where the entire interval is divided into two subintervals $[1,500]$ and $[501,1000]$, each consisting of 500 points. The changing period $[251,500]$ is contained in the first interval $[1,500]$. The following can be observed:

- In Setting 1, the p-values are small for `MMD-CV-AGG`, `MMD-Selection` and `Wasserstein` on interval 1, where the recall is 1 and the precision is higher than 0.5 for these methods. Since interval 1 contains the changing period $[251,500]$, this suggests that these methods were able to select the correct variable $d = 4$.
- In Setting 2, the p-values for interval 1 are large for all the methods, suggesting the failure of variable selection. Indeed, the precision of each method is lower than Setting 1. As can be seen from Figure 4.4 ($d = 4$), if we ignore the time order of points on interval 1 (which is essentially done by Algorithm 3), the marginal distributions for $d = 4$ are (nearly) identical for X and Y by construction. Therefore, one cannot detect the difference between X and Y on interval 1 by just looking at the marginal distributions for $d = 4$, so variable selection is harder than Setting 1. (Compare this situation with the case of $B = 10$, where variable selection was possible even in Setting 2.)
- However, variable $d = 4$ is correlated with other variables, and this correlation structure differs for X and Y. Therefore, it would still be possible to select variable $d = 4$ by identifying the change in the correlation structure. Indeed, the precision of `MMD-CV-AGG` and `MMD-Selection` is about 0.5, so they could identify the change for $d = 4$ to some extent. On the other hand, the precision of `Wasserstein` is about 0.2, which is the level of precision achievable by selecting all the five variables. This is unsurprising as the algorithm of `Wasserstein`, as designed here, does not consider the correlation structure among variables.

We can summarise the above observations as follows: If we make the number of subintervals B too small, each subinterval becomes longer, making it harder to detect changing variables.

Therefore, it is important to make B not too small so that each subinterval is short enough to detect the changes.

4.5 Demonstration: Surrogate Model Validation for Particle-based Fluid Simulation

We demonstrate how our approach can be used to validate a Deep Neural Network (DNN) model that emulates a particle-based fluid simulator. Motivated by the need for accelerating computationally expensive fluid simulations, there has been a recent surge of interest in developing DNN models that learn to approximate fluid simulations (e.g., Ummenhofer et al., 2020; Sanchez-Gonzalez et al., 2020; Prantl et al., 2022). However, model validation of a DNN emulator against the original simulator remains challenging, as both produce high-dimensional spatio-temporal outputs. Here, we demonstrate that our approach may be used to produce interpretable features that could be used by the human modeller for validation purposes.

4.5.1 Setup

Simulator. The ground-truth simulator is `Splish-Splash` (Bender et al., n.a), an open-source particle-based fluid simulator. Specifically, we consider the `WaterRamp` scenario from Sanchez-Gonzalez et al. (2020), which simulates water particles in a box environment, as illustrated in Figure 4.7.

DNN model. As the DNN model for learning the simulator, we use the `Deep Momentum-Conserving Fluids (DMCF)` model of Prantl et al. (2022), using the authors' code[6], trained with the default configuration on the dataset provided by the authors.[7]

Data Representation. Each of the simulator and the DNN model produces an output consisting of an array in $\mathbb{R}^{P\times T\times C}$, where $P = 1,444$ is the number of particles[8], $T = 600$ is the number of time steps, and $C = 2$ is the number of coordinates for each particle's position (i.e., two dimensions). We convert this array to a matrix in $\mathbb{R}^{D\times T}$, by dividing the two-dimensional coordinate space into $D = 256 = 16\times 16$ grids of equal-size squares. Thus, for a given output $X = (x_{t,d}) \in \mathbb{R}^{D\times T}$, each $x_{t,d}$ represents the number of particles on the d-th grid at the t-th time step, where $d = 1,\ldots,D$ and $t = 1,\ldots,T$. Intuitively, each grid represents a "sensor" that

[6] https://github.com/tum-pbs/DMCF/blob/main/models/cconv.py

[7] https://github.com/tum-pbs/DMCF/blob/96eb7fcdd5f5e3bdda5d02a7f97dfff86a036cfd/download_waterramps.sh

[8] The number of particles varies depending on the simulation random seed, which we set to 0.

counts the number of particles that pass the grid.

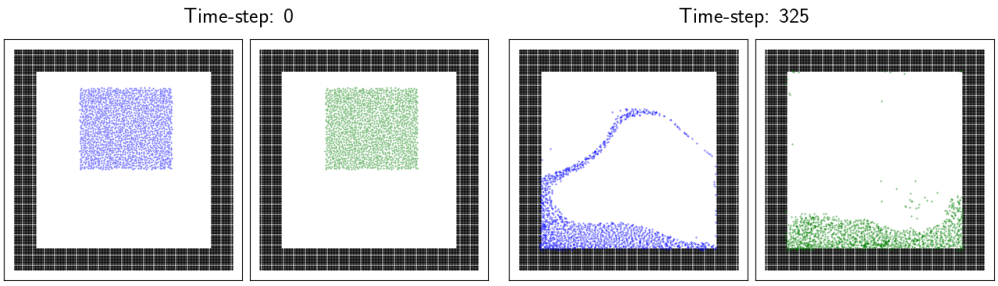


Figure 4.7: Snapshots of water particle distributions of the `WaterRamp` scenario in Section 4.5.1, at time $t = 0$ (left two figs) and time $t = 325$ (right two figs). The blue particles are from `Splish-Splash`, the ground-truth simulator, while the green ones are from the learned `DMCF` model.

Setting of Algorithm 3. We split the $T = 600$ time steps into $B = 6$ subintervals, each consisting of 100 steps: $t_1 = 100, t_2 = 200, \ldots, t_6 = 600$. For variable selection in Algorithm 3, we consider three methods: `Wassertstein`, `MMD-Selection`, and `MMD-CV-AGG`, as they performed better for the experiments in Section 4.4. We use the same configurations as Section 3.5.1 for these methods, with one modification. For the MMD-based methods, we use the dimension-wise *mean* (instead of median) heuristic (Section E) to set the length scales $\gamma_1, \ldots, \gamma_D$ of Eq. (2.3), as this leads to more stable results when the data is sparse, i.e., having many zeros, like the current setting.

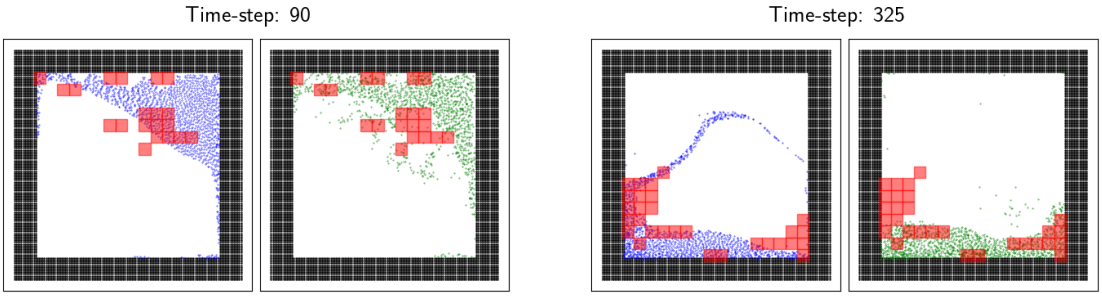
4.5.2 Results

Figure 4.8 shows selected results of variable selection; other results are available in Appendix K. More informative GIF animations illustrating selected variables are available on the authors' website[9]. Figure 4.9 shows p-values obtained with each variable selection method on each subinterval. We can make the following observations:

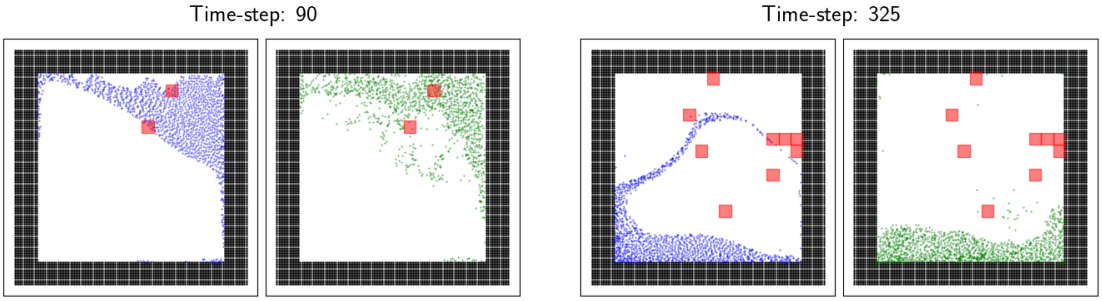
- Generally, selected variables (= red-highlighted grids) can indicate the discrepancies between the ground-truth simulator and the DNN model. For example, let us look at the grids selected by `MMD-CV-AGG` on the 4th interval $[301, 400]$ (Figure 4.8f). Many grids are selected along the blue particles' wave-like curve formed in the middle of the box (the ground-truth). However, such a wave-like curve does not exist for the green particles from the DNN model. Thus these selected grids capture successfully this discrepancy between the simulator and the DNN model.

- There are differences between the variables selected by the three methods. For example,

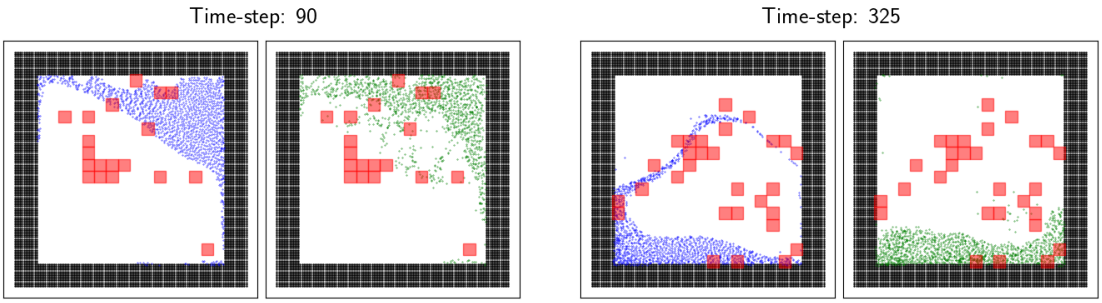
[9] https://kensuke-mitsuzawa.github.io/#/publications/mmd-paper-demo-particle-simulation



(a) `Wasserstein` (1st subinterval $[1,100]$) (b) `Wasserstein` (4th subinterval $[301,400]$)



(c) `MMD-Selection` (1st subinterval $[1,100]$) (d) `MMD-Selection` (4th subinterval $[301,400]$)



(e) `MMD-CV-AGG` (1st subinterval $[1,100]$) (f) `MMD-CV-AGG` (4th subinterval $[301,400]$)

Figure 4.8: Selected results from the experiments in Section 4.5.2. The top row shows the selected variables using Algorithm 3 with `Wasserstein`, the middle row with `MMD-Selection`, and the bottom row with `MMD-CV-AGG`. In each row, the left two figures show the selected variables for the 1st subinterval $[1,100]$, and the right two figures for the 4th subinterval $[301,400]$. The red grids represent the selected variables, the blue particles are those from the ground-truth simulator and the green ones are from the DNN model (at time $t = 90$ for the left and $t = 325$ for the right). Other results are available in Appendix K.

`Wasserstein` and `MMD-Selection` only partially capture the above discrepancy regarding the wave-like curve. On the other hand, `MMD-Selection` selects one grid along the ceiling of the box, which captures another discrepancy between the simulator and the DNN model: While there is no particle from the simulator, there are a few particles from the DNN model sticking on the ceiling, which are not physically plausible.

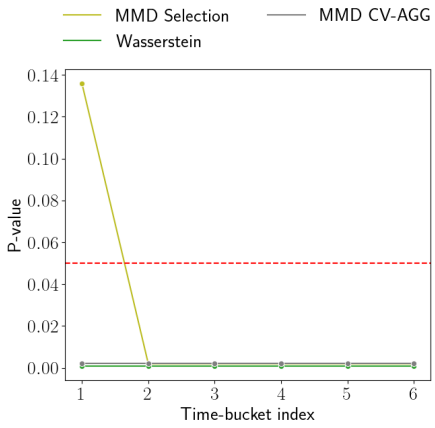


Figure 4.9: P-values obtained with Algorithm 3 on each subinterval with the `MMD-Selection`, `MMD-CV-AGG` and `Wasserstein` for the experiments in Section 4.5. "Time-bucket index" in the horizontal axis indicates the six subintervals. The red dashed line indicates the value 0.05.

- Let us look at Figure 4.9 on p-values. On the second to sixth subintervals, the p-values are all less than 0.05 for each method, indicating that the simulator and the DNN model are significantly different on these subintervals. On the first subinterval $[1, 100]$, `MMD-CV-AGG` and `Wasserstein` lead to p-values less than 0.05, while `MMD-Selection` yield a p-value higher than 0.05. The latter would be because `MMD-Selection` selects only two variables (the two red grids in Figure 4.8), which may not be enough for capturing the discrepancies between the two datasets on the 1st interval. (`MMD-Selection` tends to select few variables than `MMD-CV-AGG` in general.)
- On the other hand, the fact that `MMD-Selection` gives a larger p-value on the first subinterval suggests that the discrepancies between the simulator and the DNN model are less significant than the other subintervals. This is indeed the case, as the initial states are the same for the simulator and the DNN model.

From these observations, when using Algorithm 3 in practice, we recommend the user to try different variable selection methods (if possible) and compare the results. In this way, a more thorough analysis becomes possible.

4.6 Demonstration: Comparison of Microscopic Traffic Simulation Models

The model comparison analysis is an exploratory analysis for discovering and understanding discrepancies between models. This analysis becomes time-consuming work as a model has more variables. We apply the time slicing variable selection for this comparison to reduce the analysis workload.

The authors' website[10] provides further media content about this demonstration: animations of simulations and variable selection results.

4.6.1 Setup

We use SUMO (Lopez et al., 2018), a popular microscopic traffic simulator. We consider the simulation scenario named MoST (**Mo**naco **S**umo **T**raffic) developed by Codeca and Härri (2018). It models a realistic, large-scale traffic scenario inside the Principality of Monaco and the surrounding area on the French Riviera, as described in Figure 4.10. The scenario simulates various modes of transportation, such as passenger cars, public buses, trains, commercial vehicles, and pedestrians. Here, we focus on the simulation from 4 a.m. until 2 p.m., the duration being 6 hours (= 36,000 seconds).[11]

Simulation Scenarios. We consider two versions of the MoST scenario, one original and the other a modified version. We define the modified version[12] by blocking three roads, A8 (highway), D2564 and D51, in the original MoST scenario, at the locations marked "X" in red in Figure 4.10. This road blocking affects the vehicles that originally planned to use the A8 highway (where the top "X" blocks) to change their routes. One group of vehicles travel on the roads next to the A8 highway, as indicated by the purple arrows in Figure 4.10. As light-blue arrows indicate, another group changes their route to the south and passes inside Monaco, making the traffic there heavier than in the original scenario. These two groups of vehicles meet again at a junction[13] near the bottom "X" and travel back to the A8 highway, as indicated by orange arrows.[14]

Data Representation. We simulate both scenarios using the same random seed.[15] Let $D = 4,404$ be the number of road segments[16] in the simulation area, and $T = 3,600$ be the number of time steps from 4 a.m. to 2 p.m. (= 36,000 seconds) where each time step is 10 seconds.[17] For the original scenario, let $x_{d,t}$ be the number of vehicles on the road segment $d = 1,\dots,D$ during the time step $t = 1,\dots,T$, and define the data matrix as $X = (x_{t,d}) \in \mathbb{R}^{D\times T}$. For the modified scenario, we construct $Y = (y_{t,d}) \in \mathbb{R}^{D\times T}$ similarly.

[10] https://kensuke-mitsuzawa.github.io/#/publications/mmd-paper-demo-sumo-most

[11] In the SUMO implementation of the MoST scenario, this duration is from the 14,400th step to the 50,400th step, where one step corresponds to one second in the real world.

[12] This modified scenario is available on https://github.com/Kensuke-Mitsuzawa/sumo-sim-monaco-scenario

[13] The geographical coordinate of this junction is (43.7522398630394, 7.425510223260723).

[14] The vehicles in the opposite direction (towards the west) also change their routes similarly.

[15] We set the random seed value to 42.

[16] While "edge" is the proper SUMO terminology, we call it "road segment" for ease of understanding.

[17] Because each step in the SUMO simulator corresponds to one second, this means that we aggregate each 10 SUMO steps to form one time step in our data representation.

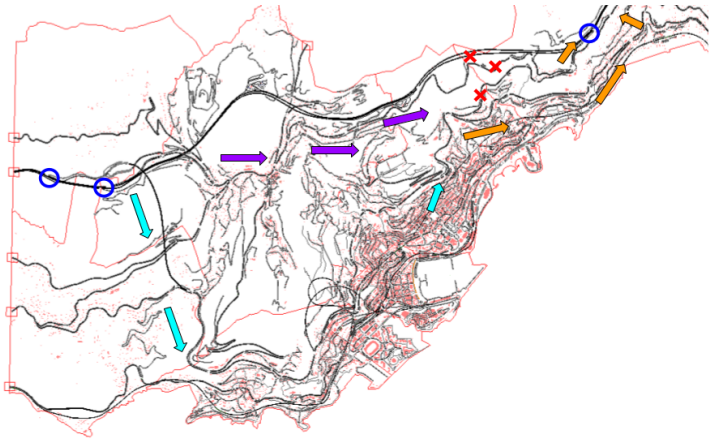


Figure 4.10: The simulated area of the MoST scenario inside and around Monaco in Section 4.6. The three red “X" marks indicate the blocked locations in the A8 highway (top), D2564 (middle) and D51 (bottom), in the modified scenario. The blue circles indicate ramps to the A8 highway. The purple and light-value arrows indicate the rerouting paths of vehicles that originally planned to travel via the A8 highway.

Variable Selection Methods. For variable selection in Algorithm 3, we try Wasserstein, MMD-Selection and MMD-CV-AGG, with the same configurations as Section 4.5.2. We set the time-splitting points as $t_1 = 500$, $t_2 = 1,000$, ... $t_7 = 3,500$ and $t_8 = 3,600$, resulting in $B = 8$ subintervals.

4.6.2 Results

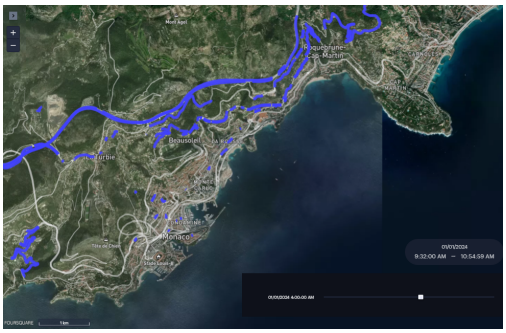
Selected Road Segments. To save space, we only describe road segments selected by MMD-CV-AGG, which performed well in the previous experiments, on interval 1 (4:00 am - 5:23 am), interval 3 (6:46 am - 8:09 am) and interval 5 (9:32 am - 10:55 am) in Figure 4.11. The purpose is to understand how the discrepancies between the two scenarios evolve over time. We can make the following observations:

- **Interval 1 (4:00 am - 5:23 am).** The A8 highway, indicated by the thick curve north of Monaco, is mainly selected. (Other road segments are also selected, but they are scattered.) The vehicles that travel on the A8 highway in the original scenario cannot pass there due to the road blocking in the modified scenario, and this discrepancy appears to be captured by the selected road segments.
- **Interval 3 (6:46 am - 8:09 am).** In addition to the A8 highway, many other road segments are selected. Particularly, the road running to the south direction from the west side of A8, the roads around the east side of A8, and the roads along the A8, are selected.

They correspond to the roads indicated by the light blue, orange and purple arrows in Figure 4.10. Moreover, more roads are selected in Monaco city than in interval 1. We can interpret this as follows: The traffic on these roads increased because the vehicles that could not pass `A8` changed their routes to bypass `A8` in the modified scenario.

(a) Selected road segments on interval 1 (4:00 am - 5:23 am).

(b) Selected road segments on interval 3 (6:46 am - 8:09 am.



(c) Selected road segments on interval 5 (9:32 am - 10:55 am).

Figure 4.11: Road segments selected by `MMD-CV-AGG` on the 1st (a), 3rd (b) and 5th (c) subintervals in the experiments of Section 4.6.

`MMD-CV-AGG` successfully selects roads where discrepancies are observed by rerouting vehicle groups as we illustrate in Figure 4.10. Among these buckets b_1, b_3, b_5, the highway *A8* is constantly selected. Plots at b_3, b_5 depict roads of two rerouting groups (purple and orange arrows in Figure 4.10). Roads by the rerouting group of light-blue arrow are observed at b_3.

Figure 4.12 represent time-series X, Y of the traffic count metrics at two representative roads. From the left plot, we affirm a high discrepancy between X, Y starting at the first bucket (4 a.m. until 5:23 a.m.), and this discrepancy continues until the seventh bucket (11 a.m. until 12:23). The right plot conveys a high discrepancy between X, Y mainly starting at the second bucket; the difference in a bucket of starting this discrepancy is due to travelling time. The right plot is at a road that is located at the west part of the scenario, and the left plot is at a

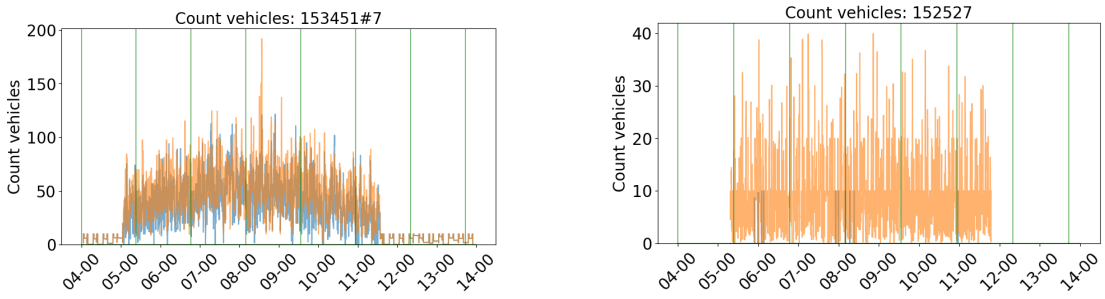


Figure 4.12: Time-series $x_{1,d},\ldots,x_{T,D}$ from the original scenario (blue) and $y_{1,d},\ldots,y_{T,D}$ from the modified scenario (orange) for two specific road segments d. The left figure is where the road segment d is at the centre of commune *La Turbie* (road id `153451#7`; near the left most purple arrow in Figure 4.10). The right figure is where d is at an entrance ramp to the `A8` highway and is located at the north part of commune *Dondéa* (road id `152527`; near the blue circle in the right side of Figure 4.10). These are road segments selected by all the methods.

road that is located at the east part of the scenario.

Permutation Tests. Figure 4.13 (left) shows p-values on each subinterval obtained with the three variable selection methods. From interval 2 to interval 5, all the methods resulted in low p-values, suggesting that the original and modified scenarios differ significantly on these subintervals. On the rest of the intervals, `Wasserstein` yielded high p-values, the reason of which may be that `Wasserstein` tends to select more redundant variables (thus leading to low precision), as the previous experiments suggest. On interval 7, `MMD-CV-AGG` selected no variable, and it returned p-value 1.0 (by definition of the algorithm; see Section G.2).

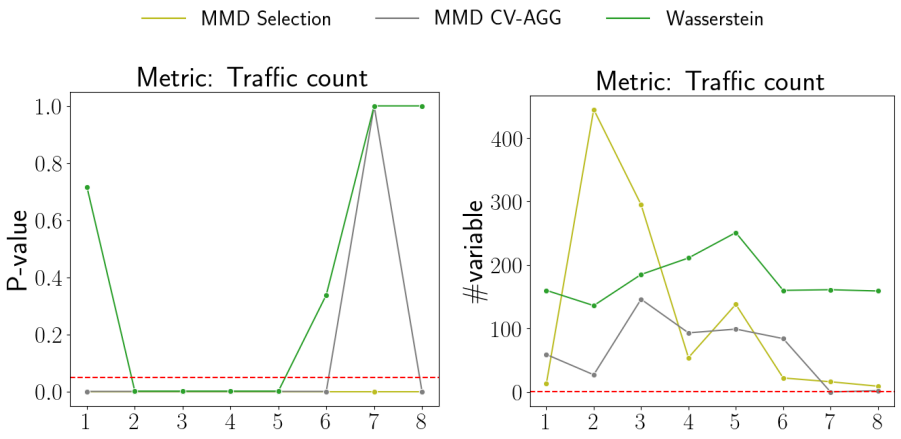


Figure 4.13: P-values (left) and the number of selected variables (right) of the three variable selection methods of the traffic count. The red dashed line represents $p = 0.05$ in the p-value plot (left) and zero variables in the variable count plot (right). More descriptions are in Section 4.6.2.

Chapter 5

Human-in-Loop Model Calibration with Variable Selection and MMD

In this chapter, we introduce an application of MMD as a distance metric for parameter calibration. The research content in this chapter mainly focuses on traffic simulation and its parameter calibration, which was requested by our industrial partner. First, we introduce related works in Section 5.1. The proposed parameter estimation (*Kernel ABC* with MMD) is in Section 5.2.3, and necessary notations are found in Section 5.2.1. Since the proposal realises the qualification of uncertainty, we elaborate on it in Section 5.2.2. A preliminary assessment of the parameter estimation task is in Section 5.3.

The research motivation is variable selection during model calibration. Model calibration is a procedure for estimating model parameters especially when the model is a black-box model (e.g. stochastic simulation model). In model calibration frameworks, we define a distance function between a model outcome and the counterpart (the real-world data or the other model's outcome), and we infer model parameters that minimise the distance.

An issue in model calibration is when the model outcome and the counterpart are high-dimensional data. Since a distance value is merely a summary statistic, it does not guarantee the validity of the calibrated parameter with the minimum distance in practice. Therefore, as we described in Section 1.4.2, model validation is necessary before deploying a model to social applications.

Although model validation confirms the validness of a model, ideally model calibration procedure is also able to provide humans with evidence, factors, or reasons for model calibration procedures. An example case of such model calibration is when computation costs (e.g. computational power or time) are expensive to execute simulations (Chen et al., 2017; Behrens

and Dias, 2015; Bharti et al., 2023; Zych et al., 2022; Najdek et al., 2021; Vijitpornkul and Marurngsith, 2015). **Human-in-loop model validation** is a practical solution to the calibration with expensive computation costs: the manual decision of halting or continuing calibration in every iteration. To realise the human-in-loop model calibration, humans are required to observe model outcomes and their counterparts and make decisions. However, this comparison analysis work is hard and time-consuming when the pair is high-dimensional data.

A solution is **a variable selection approach during model calibration**. At each iteration of model calibration, humans review the selected variables and decide if they continue model calibration or not. Selected variables and qualified uncertainty would be sufficient information for humans.

In this chapter, we introduce a preliminary work of integrating the proposed MMD-based variable selection into the model calibration algorithm. The proposed model calibration algorithm is *Kernel ABC* estimation employing an MMD estimator as a distance function (See Eq. (5.5)). The MMD estimator is optimised and selects variables during model calibration (See Chapter 3). Compared with other approaches (See the next section), the proposed calibration algorithm is superior in these two points; 1) humans can understand how model calibration goes, which is necessary information for human-in-loop calibration, 2) humans can review quantified uncertainty that is intuitive information for manual decision-making (detailed in Section 5.2.2).

5.1 Related Works regarding Parameter Calibration of Traffic Simulators

The parameters of a microscopic traffic simulator include those for vehicles (driving behaviours), the traffic control system, and the road network structure. These parameters affect the simulators' output, which includes all the vehicles' movements, in a complex way. Thus it is difficult to describe the functional relationship between the parameters and simulation output in the form of algebraic or differential equations. Consequently, previous works have handled a microscopic traffic simulator as a black-box function, and parameter calibration as black-box optimisation.

Simultaneous Perturbation Stochastic Approximation (SPSA) and *Genetic Algorithm* (GA), which are classic methods for black-box optimisation, have been popular choices in previous works on traffic simulator calibration. Indeed, Sha et al. (2020) listed 18 papers on traffic simulator calibration, of which 8 papers use SPSA and 2 papers use GA. However, both methods have some drawbacks in terms of accuracy and computational costs. SPSA is essentially a gradient descent algorithm with automatic differentiation. As such, SPSA approximates the gradient at each iteration numerically by evaluating the black-box function (i.e., by running

the simulator); therefore SPSA is computationally expensive. Moreover, since it is essentially a gradient descent algorithm, the optimal parameters found by SPSA are not guaranteed to be a global optimum (Islam et al., 2020). A drawback of GA is that it has its own hyperparameters that need to be determined appropriately. According to Islam et al. (2020), GA may produce "unpredictable results" because of its architecture, and the selection of appropriate hyperparameters that produce stable results may require several try-and-errors.

Approximate Bayesian Computation (ABC) is a generic approach to computing the *posterior distribution* of unknown parameters of a simulation model (Rubin, 1984; Sisson et al., 2018); see Section 5.2.3 for a formal definition of ABC. *Rejection ABC* is the simple and well-known version of ABC; it constructs the posterior distribution by accumulating candidate parameters that a distance of simulation output and the counterpart is less than the given threshold value. The posterior distribution computed by ABC is a conditional probability distribution of parameters given observed data. Therefore, the posterior distribution provides not only estimates of the parameters but also their *uncertainties*. This capability of quantifying uncertainties for model parameters is a major advantage of ABC (or Bayesian inference in general) over the optimisation-based approaches described above, since the latter can only produce point estimates and not their uncertainties. One can use the posterior distribution obtained with ABC for uncertainty quantification of simulation results as well. This can be done by first sampling several model parameters from the posterior distribution, and then running the simulator for each sampled parameter.

ABC has been widely used in many scientific disciplines relying on simulation modelling (Sisson et al., 2018). For example, in computational biology, ABC has been widely used for inferring the parameters of a simulation model for population genetics, where the parameters include those for population sizes, human gene evolution, rates of recombination and gene conversion, the strength of positive selection, and so on Csilléry et al. (2010).

In particular, we use *Kernel ABC* (Nakagome et al., 2013; Fukumizu et al., 2013), which is an extension of the common Rejection ABC algorithm. Kernel ABC is based on the Reproducing *Kernel Hilbert Space (RKHS)* methodology (Schölkopf and Smola, 2002), and can use more efficient simulations than Rejection ABC. One drawback of Rejection ABC is the need for specifying an appropriate threshold to determine which simulated data to accept; the specification of this threshold is not easy in practice, and it affects the computed posterior distribution significantly. On the other hand, Kernel ABC does not require specifying such a threshold. We propose a new approach to making Kernel ABC "parameter-free", which makes Kernel ABC much easier to use than Rejection ABC. This is one of our technical contributions to the literature. See Section 5.2.3 for details.

ABC Algorithms and Distance Metrics In general, any ABC-type algorithm requires an appropriate distance metric between observed and simulated datasets. Most works on ABC first summarise each of the observed and simulated datasets using summary statistics (such as the mean and variance), and then compute the Euclidean distance between the computed summary statistics. For example, the original paper of Kernel ABC (Nakagome et al., 2013) use the Euclidean distance between datasets. However, in general, it is difficult to manually define appropriate summary statistics. If the summary statistics do not capture relevant characteristics of datasets for inferring the model parameters, the resulting posterior distribution can become unreliable. Indeed, this problem occurs with traffic simulations. As Bhattacharyya et al. (2020) point out, many existing works on traffic simulator calibration use the mean of a dataset as a summary statistic, but the use of the mean does not capture the variations in the distributions of observed and simulated data, while such variations may be relevant for inferring the model parameters.

To overcome the difficulty of using summary statistics, we use MMD (Gretton et al., 2012a); See Section 2.2. MMD provides a distance metric between datasets without losing any information in the distributions of the datasets. Park et al. (2016) first propose to use MMD in ABC and show its advantage over ABC algorithms using summary statistics. Note that MMD itself contains hyperparameters, which have a strong influence on the effectiveness of MMD. Park et al. (2016) use the *median heuristic* (Garreau et al., 2018) to determine the hyperparameters; See Section E. Instead of using the median heuristic, we obtain a kernel function of MMD of which ARD parameters are optimised by the regularised optimisation problem in Section 3.4, and variables are already selected. We demonstrate that MMD with optimised ARD parameters improves Kernel ABC significantly compared with MMD solely with the median heuristic.

5.2 Simulator Calibration with Uncertainty Quantification

This section describes our approach to calibrating the parameters in a traffic simulation model and quantifying the uncertainties in the parameters and the resulting simulation outputs. These two tasks, calibration and uncertainty quantification, are vital for reliably using the simulator in traffic engineering applications. We introduce mathematical formulations for the calibration problem in Section 5.2.1, and uncertainty quantification of simulation outputs in Section 5.2.2, where we use the Bayesian approach in our formulations. We describe how to solve these problems by Kernel Approximate Bayesian Computation (Kernel ABC) in Section 5.2.3.

5.2.1 Basic Framework of the Bayesian Approach

To describe the Bayesian framework, we mathematically formulate our problem.

Model parameters. Let Θ be a set of *parameter vectors* of the simulation model. For instance, if there are $q \in \mathbb{N}$ scalar parameters in the simulator, we may define $\Theta := \mathbb{R}^q$, and each parameter vector $\theta = (\theta_1, \ldots, \theta_q)^\top \in \Theta$ consists of q scalar constants. Each parameter vector $\theta = (\theta_1, \ldots, \theta_q)^\top \in \Theta$ specifies all the configuration of the traffic simulator. Some of the parameters may specify the configuration of the road network, and others may specify the configuration of the driver model. For example, θ_1 may specify the width of a certain road segment, θ_2 the acceleration of the driver model, etc. One should include all uncertain aspects of the simulator configuration as parameters.

Simulator. We can interpret the traffic simulator as a black box that takes a parameter vector $\theta \in \Theta$ as input and produces data Y as an output. This Y takes a different form depending on how we define it. For instance, Y may take the form of a matrix of size $D \times T$, where D is the number of detectors and T is the total number of time points: $Y := (y_{d,t}) \in \mathbb{R}^{D \times T}$. In this case, each element $y_{d,t}$ may represent a certain statistic (e.g. traffic flow) at detector $d = 1, \ldots, D$ and time point $t = 1, \ldots, T$. If we are interested in multiple statistics, we may define the output Y as $Y = (Y_1, \ldots, Y_M)$, where M is the number of statistics and each $Y_m = (y^m_{d,t}) \in \mathbb{R}^{D \times T}$ is a $D \times T$ matrix such that each element $y^m_{d,t}$ represents the value of the m-th statistic (say speed) at detector $d = 1, \ldots, D$ and time point $t = 1, \ldots, T$. From now on, without loss of generality, we define Y as a $D \times T$ matrix, $Y \in \mathbb{R}^{D \times T}$, but an extension to the case of multiple statistics is straightforward.

The traffic simulator is stochastic in the sense that, even if one uses precisely the same parameters, the simulator produces different outputs for different runs if the random seeds are different. Therefore, the simulator as a black box can be mathematically defined as a *conditional probability distribution* $P(Y|\theta)$ of output data Y given a parameter vector $\theta \in \Theta$. To capture the characteristics of the simulator for a given parameter θ, one should run the simulator multiple times to obtain a *set* of output data:

$$S_\theta := \{Y_1, \ldots, Y_L\}, \quad Y_1, \ldots, Y_L \overset{i.i.d.}{\sim} P(Y|\theta),$$

where L is the number of simulation runs (e.g., $L = 20$) and $Y_\ell \in \mathbb{R}^{D \times T}$ for $\ell = 1, \ldots, L$. Here, $Y_1, \ldots, Y_L \overset{i.i.d.}{\sim} P(Y|\theta)$ denotes that $Y_1, \ldots, Y_L$ are generated from the simulator in an independently and identically distributed (i.i.d.) manner.

Real Data. We assume that we are given observed data from the real traffic system of interest. We assume that this is also given as a set of data:

$$S_{\text{real}} := \{Y_{\text{real},1}, \ldots, Y_{\text{real},L^*}\}, \tag{5.1}$$

where each $Y_{\text{real},\ell}$ should have the same form as the ones from the simulator. For example, if $Y \in \mathbb{R}^{D\times T}$, then we should have $Y_{\text{real},\ell} \in \mathbb{R}^{D\times T}$. Each $Y_{\text{real},\ell}$ may represent traffic flows recorded at D detectors and T time points on day $\ell = 1,\ldots,L^*$. The set S_{real} should consist of data sets $Y_{\text{real},1},\ldots,Y_{\text{real},L^*}$ that share the same common characteristics (e.g., the same day of a week; the same weather condition, etc.).

Prior distribution. We usually have specific prior knowledge or belief about plausible values of the parameters θ of the traffic simulator, such as the possible range of each parameter. In the Bayesian approach, such expert knowledge is expressed as a *prior probability distribution* $\pi(\theta)$ on the parameter space Θ, i.e. calibrated parameters in different conditions. This prior distribution may express our initial uncertainties about the simulation model parameters.

Posterior distribution. The focus of the Bayesian approach is to compute the *posterior distribution* on the parameter space Θ given the observed data S_{real} from the real traffic system:

$$P_\pi(\theta|S_{\text{real}}) = \frac{P(S_{\text{real}}|\theta)\pi(\theta)}{Z(S_{\text{real}})}, \tag{5.2}$$

where $P(S_{\text{real}}|\theta)$ is the conditional probability of observing S_{real} given that the simulation model with parameters θ is true, and $Z(S_{\text{real}}) := \int P(S_{\text{real}}|\theta)\pi(\theta)\,d\theta$ is the normalisation constant. The right-hand side is *Bayes' theorem*, and the left-hand side is the posterior distribution, which is the conditional distribution of the parameters θ given the real data S_{real}. The posterior distribution $P_\pi(\theta|S_{\text{real}})$ can be interpreted as possibilities over the parameters after observing the real data S_{data} updated from the initial possibilities $\pi(\theta)$.

5.2.2 Uncertainty Quantification with the Posterior Distribution

We explain here how the posterior distribution in Eq. (5.2) can be used for uncertainty quantification of simulations. To this end, suppose that the posterior $P_\pi(\theta|S_{\text{real}})$ (5.2) has been obtained.

To illustrate the idea, we generalise the notation for the simulator using the conditional probability distribution as

$$P(Y|\theta,a), \quad \theta\in\Theta,\ a\in\mathscr{A},$$

where $\mathscr{A}$ is a set of possible *actions* or *interventions* to the traffic system, such as different signal schedules. For the calibration task (i.e., the posterior distribution computation discussed later), we assume that the intervention $a_0 \in \mathscr{A}$ used for obtaining real data is known. Therefore, the previous notation for the simulator is recovered as $P(Y|\theta) := P(Y|\theta,a_0)$.

Suppose that we are interested in the effects of a certain intervention $a^* \in$ to the traffic system (e.g., a candidate signal schedule). Denote by $K_1, \ldots, K_\nu \in \mathbb{R}$ be *key performance indicators (KPIs)*[1] that one is interested in, where $\nu \in \mathbb{N}$. Each of these KPIs can be seen as a *function* of traffic data Y, and thus can be written as $K_\ell := f_\ell(Y)$ for an appropriate function f_ℓ for $\ell = 1, \ldots, \nu$. We are interested in quantifying uncertainties of these KPIs when using the intervention a^*, by taking into account also uncertainties on the simulator parameters θ.

What we need first is to derive the probability distribution of Y under the intervention a^*, which is given by

$$P(Y|a^*) := \int P(Y|\theta, a^*) P_\pi(\theta|S_{\text{real}}) d\theta. \tag{5.3}$$

Since the KPIs $K_1, \ldots, K_\nu$ are the functions of the data Y, their probability distributions are given by the probability distribution of Y (the so-called *push-forward measures*). The probability distribution in Eq. (5.3) is called the *Bayesian predictive distribution*.

In practice, the integral in Eq. (5.3) cannot be computed, because it involves the simulator $P(Y|\theta, a^*)$. Therefore in practice, the integral is approximated by a Monte Carlo average. More precisely, we first perform *sampling* from the posterior distribution $P_\pi(\theta|S_{\text{real}})$ to generate a set of parameter vectors:

$$\theta^{(1)}, \ldots, \theta^{(N)} \overset{i.i.d.}{\sim} P_\pi(\theta|S_{\text{real}}),$$

where $N \in \mathbb{N}$ is a user-specified number. Then, for each $i = 1, \ldots, N$, we run the simulator using the parameter vector $\theta^{(i)}$ and obtain the corresponding output data $Y^{(i)}$:

$$Y^{(i)} \sim P(Y|\theta^{(i)}, a^*), \quad i = 1, \ldots, N.$$

The predictive distribution in Eq. (5.3) is then approximated as

$$\hat{P}(Y|a^*) := \frac{1}{N} \sum_{i=1}^{N} \delta(Y - Y^{(i)}),$$

where $\delta(Y - Y^{(i)})$ is the Dirac delta distribution at $Y^{(i)}$ (i.e., the point mass at $Y^{(i)}$). Accordingly, the probability distributions of the KPIs can be approximately computed as

$$\hat{P}(K_\ell \mid a^*) := \frac{1}{N} \sum_{i=1}^{N} \delta(K_\ell - f_\ell(Y^{(i)})), \quad \ell = 1, \ldots, \nu,$$

where, as defined earlier, f_ℓ is a function that computes KPI K_ℓ from data Y. These probability distributions of KPIs are our uncertainty estimates for the KPIs. One can use them for one's own purpose.

[1] We dare to use the term "KPI" to clearly distinguish it from other types of metrics.

For example, suppose that we are interested in the mean and the variance of the KPI K_ℓ. These can be computed as

$$\hat{\mu}_\ell := \frac{1}{N}\sum_{i=1}^{N} f_\ell(Y^{(i)}), \quad \hat{\sigma}^2_\ell := \frac{1}{N-1}\sum_{i=1}^{N}(\hat{\mu}_\ell - f_\ell(Y^{(i)}))^2.$$

If the variance $\hat{\sigma}^2_\ell$ is extensive, our uncertainty about the KPI K_ℓ is significant, and vice versa. One can also use the values $f_\ell(Y^{(1)}),\ldots,f_\ell(Y^{(N)})$ to construct a histogram, which provides the landscape of the distribution $\hat{P}(K_\ell|a^*)$.

5.2.3 Approximate Bayesian Computation (ABC)

A challenge of the Bayesian approach is the computation of the posterior distribution Eq. (5.2) itself. In particular, in our setting, the conditional probability distribution $P(Y|\theta)$ cannot be written as a mathematical function since it is implicitly defined through the simulator. We can only access the conditional distribution $P(Y|\theta)$ via simulations (sampling), i.e., given some $\theta' \in \Theta$, we run the simulator and obtain the corresponding output $Y' \sim P(Y|\theta')$. Thus, we need to compute the posterior distribution by making use of this sampling procedure.

Approximate Bayesian Computation (ABC) is a Bayesian method for computing the posterior distribution by sampling from the conditional distribution $P(Y|\theta)$ (Rubin, 1984; Sisson et al., 2018). The main idea is to run the simulator multiple times for different parameter vectors and compare the obtained simulation outputs with the real observed dataset. Here, the comparison between simulated and real data is done by using an appropriately defined *distance metric* between two datasets. We propose to use MMD (Gretton et al., 2012a) explained in Section 2.2.

Rejection ABC

We first describe the simplest ABC approach named *Rejection ABC*.

We first generate parameters vectors $\theta^{(1)},\ldots,\theta^{(N)} \in \Theta$ from the prior distribution $\pi(\theta)$ on the parameter space Θ:

$$\theta^{(1)},\ldots,\theta^{(N)} \overset{i.i.d.}{\sim} \pi(\theta),$$

where $N \in \mathbb{N}$ is a user-specified number.

For each $i = 1,\ldots,N$, we run the simulator using the parameter vector $\theta^{(i)}$ multiple times and obtain a set of data:

$$S_{\theta^{(i)}} := \{Y_1^{(i)},\ldots,Y_L^{(i)}\}, \quad Y_1^{(i)},\ldots,Y_L^{(i)} \overset{i.i.d.}{\sim} P(Y|\theta^{(i)}).$$

Thus, we have obtained N simulated datasets $S_{\theta^1}, \ldots, S_{\theta^N}$. Now, we compare these datasets to the real data S_{real} in Eq. (5.1). To this end, as mentioned above, we use MMD.

Let $\varepsilon > 0$ be a user-specified small constant. We compute MMD between S_{real} and each of the simulated datasets $S_{\theta^{(1)}}, \ldots, S_{\theta^{(N)}}$, and only keep a subset of the simulated datasets whose MMD values are less than ε. More precisely, let $J_\varepsilon \subset \{1, \ldots, N\}$ be a set of indices such that

$$\begin{cases} i \in J_\varepsilon & \text{if} \quad \widehat{\text{MMD}}_U(S_{\text{real}}, S_{\theta^{(i)}}) < \varepsilon \\ i \notin J_\varepsilon & \text{if} \quad \widehat{\text{MMD}}_U(S_{\text{real}}, S_{\theta^{(i)}}) \geq \varepsilon, \end{cases}$$

where $\widehat{\text{MMD}}_U(S_{\text{real}}, S_{\theta^{(i)}})$ is the MMD estimate Eq. (2.2) computed for the two datasets S_{real} and $S_{\theta^{(i)}}$. Intuitively, the indices i in J_ε are such that the corresponding simulated dataset $S_{\theta^{(i)}}$ are "similar" to the real dataset S_{real}, as measured by MMD. Then, the posterior distribution $P_\pi(\theta|y^*)$ in Eq. (5.2) is approximated by the empirical distribution:

$$P_\pi(\theta|S_{\text{real}}) \approx \hat{P}_\pi(\theta|S_{\text{real}}) := \frac{1}{|J_\varepsilon|} \sum_{i \in J_\varepsilon} \delta(\theta - \theta^{(i)}), \tag{5.4}$$

where $|J_\varepsilon|$ is the cardinality of J_ε and $\delta(\theta - \theta^{(i)})$ denotes the Dirac delta distribution (or "point mass") at $\theta^{(i)} \in \Theta$. This entire procedure is called Rejection ABC.

One can perform uncertainty quantification for simulations in the way we explained in the previous section, by regarding the set of parameter vectors $\{\theta^{(i)} \mid i \in J_\varepsilon\}$ as generated from the posterior distribution $P_\pi(\theta|S_{\text{real}})$.

Rejection ABC is simple and illustrative for describing the idea of ABC in general. However, Rejection ABC is not efficient in the sense that a large number N of simulations may be needed for the approximate posterior $\hat{P}_\pi(\theta|S_{\text{real}})$ to be "close" to the true posterior $P_\pi(\theta|S_{\text{real}})$. Intuitively, this is because we need to take the threshold ε to small enough to make the approximate posterior accurate; but if ε is too small the probability of generating simulated data $S_{\theta^{(i)}}$ such that $\text{MMD}(S_{\text{real}}, S_{\theta^{(i)}}) < \varepsilon$ becomes small; thus the resulting number $|J_\varepsilon|$ of "accepted" parameter vectors becomes small, and the resulting posterior approximation Eq. (5.4) becomes inaccurate.

We thus propose to use a more sophisticated and efficient ABC technique, which is *Kernel ABC* explained in the next subsection.

Kernel ABC

Kernel ABC is a more efficient version of ABC based on *kernel methods* in machine learning (Nakagome et al., 2013; Kisamori et al., 2020). A theoretical backbone of the approach is the framework called *kernel mean embeddings* (Smola et al., 2007; Muandet et al., 2017), which

we do not explain here.

The procedure of Kernel ABC is the same as Rejection ABC until the generation of the simulated datasets $S_{\theta^{(1)}}, \ldots, S_{\theta^{(N)}}$. The difference is in the construction of the posterior approximation. To describe this, define a *second-level kernel* function $k_{\mathscr{S}}$ on datasets as

$$k_{\mathscr{S}}(S, S') := \exp\left(-\frac{\widehat{\mathrm{MMD}}_U^2(S, S')}{\gamma^2}\right), \tag{5.5}$$

where $\widehat{\mathrm{MMD}}_U(S, S')$ is the MMD estimate Eq. (2.2) between two datasets S and S', and $\gamma^2 > 0$ is a bandwidth parameter. We suppose that an appropriate MMD estimator $\widehat{\mathrm{MMD}}_U^2(S, S')$ is ready by methods in Section 3.4, and variables are already selected. These selected variables provide an interaction with humans for the Human-in-loop calibration.

We describe a mechanism for this kernel function. Intuitively, this kernel function measures the *similarity* between the two datasets S and S', as $\widehat{\mathrm{MMD}}_U(S, S')$ is a distance metric between S and S'. Namely, if $\mathrm{MMD}(S, S')$ is large, $k_{\mathscr{S}}(S, S')$ becomes small and approaches 0, and if $\widehat{\mathrm{MMD}}_U(S, S')$ is small $k_{\mathscr{S}}(S, S')$ becomes large and approaches 1.

The bandwidth parameter γ^2 is a hyperparameter of Kernel ABC. For instance, we can set γ^2 by the median heuristic (Garreau et al., 2018), i.e., as the median of the $N \times (N-1)$ pairwise MMD values $(\widehat{\mathrm{MMD}}_U^2(S_{\theta^{(i)}}, S_{\theta^{(j)}}))_{i \neq j}$ of simulated datasets $S_{\theta^{(1)}}, \ldots, S_{\theta^{(N)}}$: $\gamma^2 := \mathrm{median}((\widehat{\mathrm{MMD}}_U^2(S_{\theta^{(i)}}, S_{\theta^{(j)}}))_{i \neq j})$.

Now, using the above kernel function, we compare the real observed dataset S_{real} and each of the simulated datasets $S_{\theta^{(1)}}, \ldots, S_{\theta^{(N)}}$. This is done by first computing a *similarity vector* $\mathbf{k}(S_{\mathrm{real}}) \in \mathbb{R}^N$ defined by

$$\mathbf{k}(S_{\mathrm{real}}) := (k_{\mathscr{S}}(S_{\theta^{(1)}}, S_{\mathrm{real}}), \ldots, k_{\mathscr{S}}(S_{\theta^{(N)}}, S_{\mathrm{real}}))^\top \in \mathbb{R}^N. \tag{5.6}$$

That is, the vector $\mathbf{k}(S_{\mathrm{real}})$ quantifies the similarities of the simulated datasets $S_{\theta^{(1)}}, \ldots, S_{\theta^{(N)}}$ to the real observed data S_{real}. Then, we transform this similarity vector to a *weight vector* $\mathbf{w} \in \mathbb{R}^N$ for the associated parameter vectors $\theta^{(1)}, \ldots, \theta^{(N)}$ as

$$\mathbf{w} := (w_1, \ldots, w_N)^\top := (G + N\lambda I)^{-1}\mathbf{k}(S_{\mathrm{real}}) \in \mathbb{R}^N, \tag{5.7}$$

where $G \in \mathbb{R}^{N \times N}$ is the *kernel matrix* defined by $G_{ij} = k_{\mathscr{S}}(S_{\theta^{(i)}}, S_{\theta^{(j)}})$, $\lambda > 0$ is a regularisation constant, and $I \in \mathbb{R}^{N \times N}$ is the identity matrix (i.e., $I_{ij} = 1$ if $i = j$ and $I_{ij} = 0$ otherwise). Intuitively, each weight w_i represents the "importance" of the corresponding parameter vector $\theta^{(i)}$ in approximating the posterior distribution $P_\pi(\theta | S_{\mathrm{real}})$.

Finally, an approximation to the posterior distribution $P_\pi(\theta|S_{\text{real}})$ is given by

$$P_\pi(\theta|S_{\text{real}}) \approx \hat{P}_\pi(\theta|S_{\text{real}}) := \sum_{i=1}^{N} w_i \delta(\theta - \theta^{(i)}), \tag{5.8}$$

where $\delta(\theta - \theta^{(i)})$ denotes the Dirac delta distribution at $\theta^{(i)} \in \Theta$ and $w_1, \ldots, w_N$ are the weights defined above.

One can generate new parameter vectors $\tilde{\theta}^{(1)}, \ldots, \tilde{\theta}^{(N)} \in \Theta$ from the approximate posterior in Eq. (5.8) by using a sampling method called *kernel herding* (Chen et al., 2010); for the concrete algorithm of kernel herding, see e.g. Kisamori et al. (2020, Section 3.3) and Muandet et al. (2021, Algorithm 1).

Uncertainty quantification for simulations can then be carried out using the approach described in Section 5.2.2.

5.3 Empirical Assessment

This section presents numerical illustrations of the proposed approaches described so far. The parameter calibration assessment is for estimating road status parameters of a road network. We admit that the road network, normally in the context of traffic engineering, is considered as the given condition (or the given system), and road status is not for the target of calibration. The assessment setting is requested by our research collaborators who are interested in the calibration of the road status parameters, therefore, the assessment motivation is from the industrial motivation.

5.3.1 Simulation Settings

We define a grid-like road network consisting of 8 intersections described in Figure 5.2, which we call *Grid network*.[2] We consider a scenario where there are 16 origin-destination (OD) groups of vehicles, in which 15 groups travel from one corner of the network to another corner in the diagonal direction, and one group travels from the left-upper corner to the right-upper corner.

Each detector (Induction Loops Detector) records the *flow* – the number of vehicles that pass in front of the detector during one time step, which is approximately 1 second in the simulation world. We used the flow metric since it well represents differences in traffic. Also, it is a commonly used metric in calibrations of traffic simulations[3]. Our scenario consists of 3,600 time steps (= 1 hour in the simulation world). We aggregate the flow metrics of

[2] We generated the network using a SUMO script: https://sumo.dlr.de/docs/netgenerate.html

[3] According to Sha et al. (2020), eleven researchers used "flow" in an objective function of the calibration.

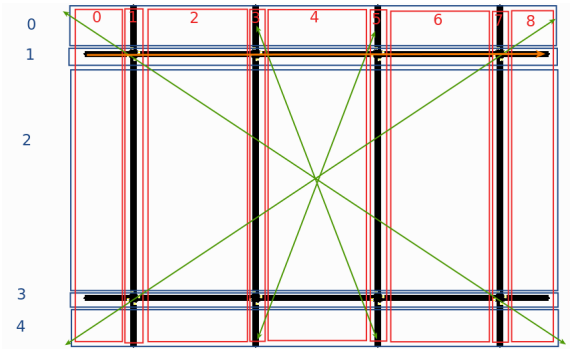


Figure 5.1: Illustration of the grid network scenario. Arrows indicate possible original-destination (OD) pairs. There are 16 distinct OD groups of vehicles. Most groups travel in diagonal directions as indicated by green arrows. One group (orange arrow) travels from the left-upper corner to the right-upper corner.

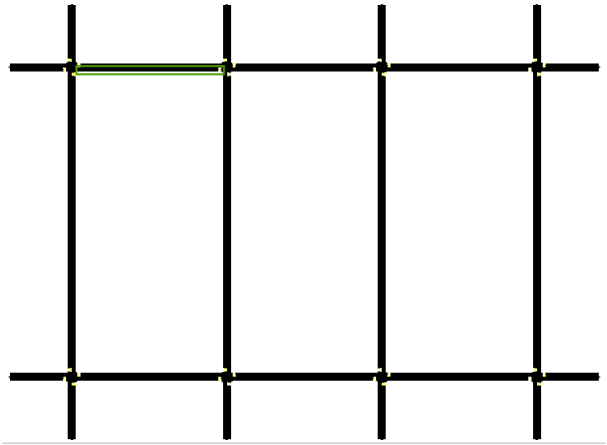


Figure 5.2: The grid network. The green box highlights the edge where the two binary parameters θ_1, θ_2 specify the conditions of two lanes. Yellow dots at intersections indicate detectors.

consecutive 300-time steps (= 5 minutes) as their average, resulting in $T = 12$ time intervals and the corresponding 12 values of average flows. There are $D = 64$ detectors in the network.

We generate a random seed of the simulator as $R + \epsilon$, where R is a discrete random variable uniformly sampled from $\{0, 1, \ldots, 9, 10\}$ and ϵ is a Gaussian random variable with mean 1 and variance 5.

5.3.2 Experiment settings

Binary parameters $\theta_1 \in \{0, 1\}$ and $\theta_2 \in \{0, 1\}$ control the conditions of two lanes in one network edge shown in Figure 11. As described in Figure 12, θ_1 parametrises the state of the upper lane in the edge, and θ_2 is the state of the lower lane. Specifically, $\theta_1 = 0$ indicates that the upper lane is *closed*; $\theta_1 = 1$ is that the upper lane is *open*. Similarly, $\theta_2 = 0$ indicates that the lower

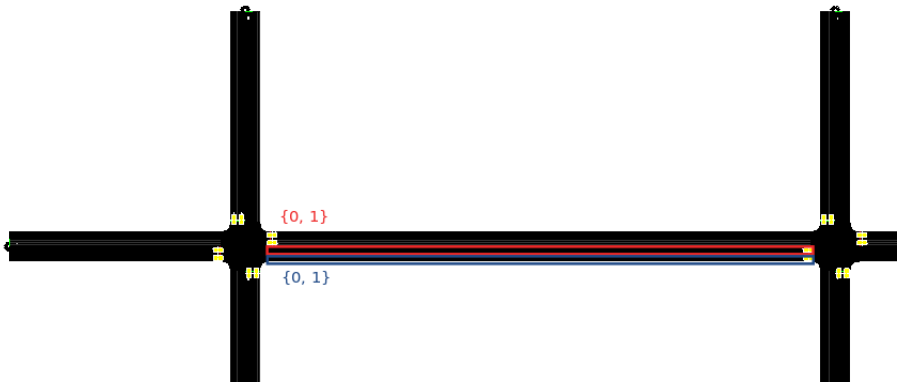


Figure 5.3: The parameterised edge in the grid network of Figure 5.2. The two parameters $\theta_1 \in \{0,1\}$ and $\theta_2 \in \{0,1\}$ specify the states of the upper lane (red) and the lower lane (blue), respectively. If $\theta_1 = 0$, the upper lane is closed; if $\theta_1 = 1$ it is open. If $\theta_2 = 0$, the lower lane is closed; if $\theta_2 = 1$, it is open.

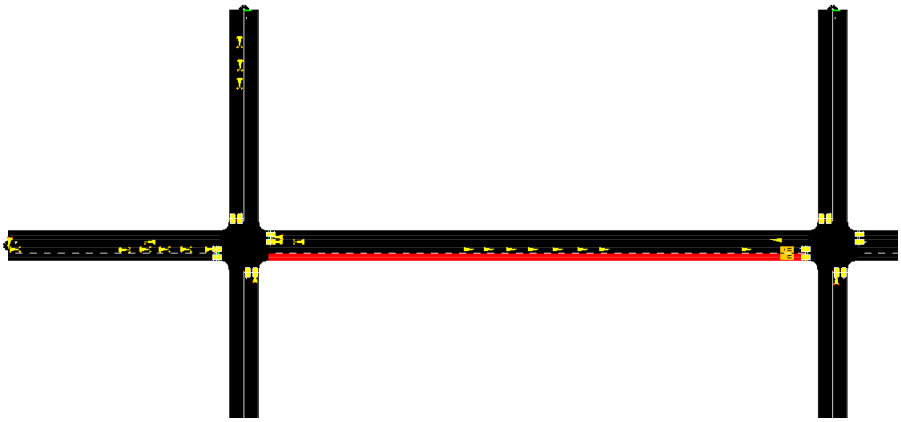


Figure 5.4: The network condition when the network parameter is $(1,0)^\top$. The red highlight represents the lane is closed.

lane is *closed*; $\theta_2 = 1$ is that the lower lane is *open*. We treat θ_1 and θ_2 as unknown parameters to be calibrated from observed data. Since the pair (θ_1, θ_2) can take possible 4 states,

$$(\theta_1, \theta_2) \in \Theta := \{(0,0), (0,1), (1,0), (1,1)\},$$

the calibration problem is to infer one of these four states from observed data.

Let $P(\mathbf{Y}|\theta)$ denote the conditional probability distribution of flow matrix $\mathbf{Y} \in \mathbb{R}^{D \times T}$ conditional on the parameter vector $\theta := (\theta_1, \theta_2) \in \Theta$. The traffic simulator determines this conditional distribution. We generate a "real" observed dataset S_{real} from the simulator using ground-truth parameters $\boldsymbol{\theta}_{\text{real}} := (\theta_{\text{real},1}, \theta_{\text{real},2}) \in \Theta$:

$$S_{\text{real}} = \{\mathbf{Y}_{\text{real}1}, \ldots, \mathbf{Y}_{\text{real}L^*}\} \overset{i.i.d.}{\sim} P(\mathbf{Y} \mid \boldsymbol{\theta}_{\text{real}}),$$

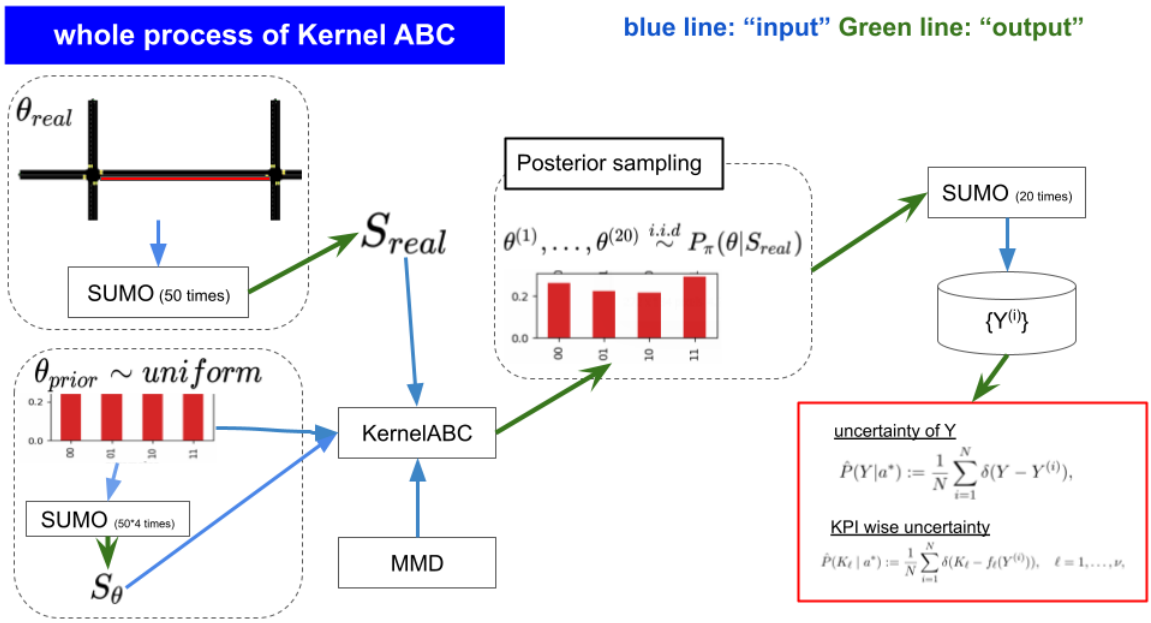


Figure 5.5: Estimation procedure with Kernel ABC.

where we set $L^* = 50$. We treat $\boldsymbol{\theta}_{\text{real}}$ as unknown. The calibration problem is to estimate θ_{real} from the observed dataset S_{real} and the simulator $P(\mathbf{Y}|\theta)$.

Figure 5.5 describes the estimation procedure using Kernel ABC. We define a prior distribution $\pi(\boldsymbol{\theta})$ on the parameter space $\Theta = \{(0,0),(0,1),(1,0),(1,1)\}$ as the uniform distribution. In practice, we implement this as preparing 20 samples for each of 4 possible states $(0,0),(0,1),(1,0),(1,1)$; thus we have in total 80 parameter samples, which we denote by $\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{80}$. Using each parameter sample $\boldsymbol{\theta}_i$ $(i = 1,\ldots,80)$, we run 50 simulations using random seeds, obtaining a simulated dataset

$$S_{\boldsymbol{\theta}_i} = \{\mathbf{Y}_{\theta_i,1},\ldots,\mathbf{Y}_{\theta_i,50}\} \overset{i.i.d.}{\sim} P(\mathbf{Y}|\theta_i).$$

We have 80 such simulated data $S_{\boldsymbol{\theta}_1},\ldots,S_{\boldsymbol{\theta}_{80}}$.

MMD Configuration

For optimising the ARD weights in MMD, we define a new objective function as the sum of 80 objective functions, each of which is given by Eq. (3.2), computed for S_{real} and S_{θ_i} where $i = 1,\ldots,80$. We use the L_2 regularisation with $\lambda_2 = 0.1$ (and thus $\lambda_1 = 0$) [4]. We set the length

[4]We did not use the automatic selection of the regularisation constant λ in this experiment, since we have conducted this assessment before we developed the method for automatic selection of λ. The choice of $\lambda_2 = 0.1$ is based on our manual analysis in which we observe optimised ARD parameters.

scales ($\gamma_{d,t}$) using the dimension-wise median heuristic described in Section E. We use the Adam optimiser with a learning rate of 0.001 and run 7,000 epochs, with all the ARD weights initialised as 1.0.

Kernel ABC Configuration

We determine the regularisation constant λ of Kernel ABC Eq. (5.7) as follows. First, the weights $\boldsymbol{w} = (w_1, \ldots, w_N)^\top$ in Eq. (5.7) given by Kernel ABC may be interpreted as representing posterior probabilities (here $N = 80$ in the current setting). Thus, good weights should be such that 1) their sum is close to 1 and 2) each weight is non-negative. Therefore, we define the following objective function for choosing λ:

$$c(\lambda) := \left| \sum_{i=1}^{N} w_i - 1 \right| + \sum_{i=1}^{N} |w_i| \tag{5.9}$$

where the first term quantifies a deviation of the sum of the weights from 1 and the second term a deviation from non-negativity[5] Note that the weights $\boldsymbol{w} = (w_1, \ldots, w_N)^\top$ depend on λ; see Eq. (5.7). We propose to choose λ that minimises the above criterion. Specifically, we compute the weights for each of $\lambda \in \{10^{-3}, 10^{-3+0.25}, \ldots, 10^{-0.25} + 10^{0}\}$, and choose λ that minimises the above criterion.

We choose the length scale γ in the second-level kernel in Eq. (5.5) using the median heuristic, as described in Section 5.2.3.

Uncertainty Quantification for Simulation Outputs

After obtaining a posterior distribution of the unknown parameters, we sample 50 times from this posterior distribution and run the simulator using each of these 50 parameter vectors.

For simplicity, we consider uncertainty quantification of one specific KPI: the *travel time*, defined as the median value of all vehicles' travel times.[6] This KPI can capture the effects of the lane blockage. However, note that it is also possible to perform uncertainty quantification for any other KPIs.

Baselines

For comparison, we consider the following four baseline methods.

1. Rejection ABC using summary statistics. This approach uses the Rejection ABC algo-

[5] Another possibility is $c(\lambda) := \left|\sum_{i=1}^{N} w_i - 1\right| + \sum_{i=1}^{N} \max(0, -w_i)$.

[6] We compute this using the *duration* XML attribute in *tripinfo_output.xml*

rithm in Section 5.2.3. We summarise each dataset (S_{real} or $S_{\theta_1}, \ldots, S_{\theta_N}$) by its mean. We use the Euclidean distance between the means as a distance metric.

2. Rejection ABC using MMD with optimised ARD weights. As a distance metric between datasets, this approach uses MMD with optimised ARD weights.

3. Kernel ABC using MMD with the dimension-wise median heuristic. In this baseline, we set all the ARD weights to 1.0, i.e., we do not optimise the ARD weights.

4. Kernel ABC using MMD with the standard median heuristic. All the dimensions have the same bandwidth parameter, which we set using the median heuristic. All the ARD weights are set to 1.0.

As explained in Section 5.2.3, Rejection ABC requires the specification of a threshold ε for deciding the acceptance of sampled parameters. Specifically, we select the threshold ε that matches a specific percentile of the distances between the observed dataset and simulated datasets (where the distance is either that between summary statistics or the MMD). The percentiles we use are 5%, 10%, 20%, 40% and 80%.

For the two baselines using Kernel ABC, we select the bandwidth γ of the second-level kernel and the regularisation constant λ in the same way as the proposed approach.

Evaluation Metrics for Estimated Parameters

To evaluate the estimated parameters, we define an error metric defined as $1 - p(\theta_{\text{real}})$, where $p(\theta)$ is the posterior probability of the parameter θ obtained with that method. Note that this "error" does not necessarily capture the quality of uncertainty quantification, but we use it here for ease of comparison.

5.3.3 Assessment Result: Estimated Parameters

We performed the experiment for each of the four cases of the true unknown parameter $\boldsymbol{\theta}_{\text{real}} \in \{(0,0),(0,1),(1,0),(1,1)\}$. Table 5.1 summarises the defined error metric for the estimated parameter, and Figure 5.6 shows the posterior distributions. Rejection ABC with optimised MMD performs the best in terms of the error metric, however, this method is **not implementable in practice** since it chooses the threshold based on the knowledge of the ground-truth parameter. Among the three approaches using Kernel ABC, the proposed approach with optimised MMD performs the best for the three cases $\boldsymbol{\theta}_{\text{real}} = (0,1),(1,0),(1,1)$. For the case $\boldsymbol{\theta}_{\text{real}} = (0,0)$, the lowest error is achieved by Kernel ABC with non-optimised MMD using the dimensional-wise median heuristic. The proposed approach with optimised MMD under-performs for estimating the posterior probability of the parameter $(0,0)$ since we might

Table 5.1: Estimated posterior probabilities of the four parameters obtained with Kernel ABC and the baselines. The columns from $(0,0)$ to $(1,1)$ represents the posterior probabilities of the four parameters. The column *error* represents the defined error metric for the estimated parameter. The symbol * for Rejection ABC-based approaches indicates using the knowledge of the ground-truth $\boldsymbol{\theta}_{\text{real}}$ to select the threshold, which are **not implementable in practice**. We mark error values with bold font where the error is the smallest among the methods except for not implementable methods.

method	$\boldsymbol{\theta}_{\text{real}}$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	error
Kernel ABC with opt. MMD (proposed)	$(0,0)$	0.439	0.182	0.202	0.176	0.56
Kernel ABC with opt. MMD (proposed)	$(0,1)$	0.0	0.97	0.01	0.18	**0.029**
Kernel ABC with opt. MMD (proposed)	$(1,0)$	0.001	0	0.987	0.011	**0.012**
Kernel ABC with opt. MMD (proposed)	$(1,1)$	0.004	0.031	0	0.963	**0.036**
Kernel ABC with non-opt. MMD (dim. med)	$(0,0)$	0.584	0.135	0.14	0.139	**0.415**
Kernel ABC with non-opt. MMD (dim. med)	$(0,1)$	0	0.93	0	0.0692	0.069
Kernel ABC with non-opt. MMD (dim. med)	$(1,0)$	0.015	0.015	0.941	0.0283	0.0586
Kernel ABC with non-opt. MMD (dim. med)	$(1,1)$	0	0.108	0.057	0.833	0.166
Kernel ABC with non-opt. MMD (single med)	$(0,0)$	0.25	0.25	0.248	0.25	0.749
Kernel ABC with non-opt. MMD (single med)	$(0,1)$	0.25	0.249	0.25	0.249	0.775
Kernel ABC with non-opt. MMD (single med)	$(1,0)$	0.234	0.243	0.283	0.238	0.716
Kernel ABC with non-opt. MMD (single med)	$(1,1)$	0.0.248	0.25	0.252	0.249	0.75
*Rejection ABC with summary statistics	$(0,0)$	1.000	0.000	0.000	0.000	0.000
*Rejection ABC with summary statistics	$(0,1)$	0.063	0.313	0.313	0.313	0.688
*Rejection ABC with summary statistics	$(1,0)$	0.500	0.031	0.438	0.031	0.563
*Rejection ABC with summary statistics	$(1,1)$	0.125	0.438	0.031	0.406	0.594
*Rejection ABC with optimised MMD	$(0,0)$	1.000	0.000	0.000	0.000	0.000
*Rejection ABC with optimised MMD	$(0,1)$	0.000	1.000	0.000	0.000	0.000
*Rejection ABC with optimised MMD	$(1,0)$	0.000	0.000	1.000	0.000	0.000
*Rejection ABC with optimised MMD	$(1,1)$	0.000	0.000	0.000	1.000	0.000

select improper regularisation parameter of the ARD weights optimisation. We manually tuned the regularisation parameter by checking visualisation of ARD weights[7]. Therefore, the proposed approach with optimised MMD can be further improved by using the automatic selection of the regularisation parameter.

Regularisation Parameter Selection of Kernel ABC Algorithm To compute the the posterior probability with Kernel ABC, we need to select the regularisation parameter λ in Eq. (5.7). As we show in Eq. (5.9), we propose an automatic selection of λ by minimising the criterion.

Table 5.4 shows the values of the objective function (5.9) for different values of λ, computed for each case of the ground-truth parameters. According to the criterion, we can see that λ should not be too small or large. By observing the estimated posterior probabilities in Figure 5.6, the

[7] At the time of this assessment, we have not developed the automatic regularisation parameter selection method described in Chapter 3.

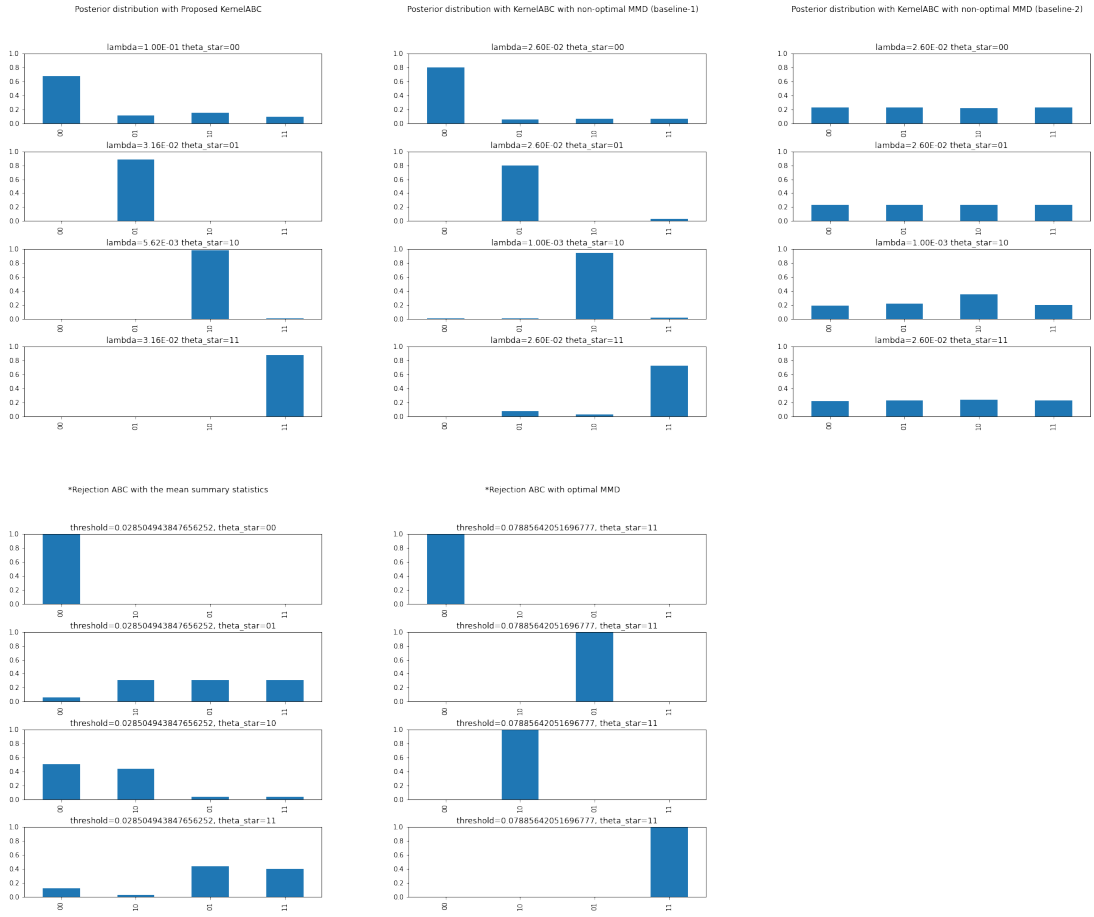


Figure 5.6: Calculated posterior probabilities of the four possible network parameters, for each of the four cases of the ground-truth parameter. The left-upper figure shows the results for Kernel ABC with optimised MMD (proposed), the centre-upper figure Kernel ABC with non-optimised MMD and the dimension-wise median heuristic, the right-upper figure Kernel ABC with non-optimised MMD and the single-bandwidth median heuristic, the left bottom figure Rejection ABC the mean summary statistic, and the centre-bottom figure Rejection ABC with optimised MMD. In each figure, the horizontal axis indicates the four possible parameters $\boldsymbol{\theta}$ and the vertical axis indicates posterior probabilities.

estimated posterior probabilities look reasonable. Thus, the automatic λ selection sounds work well, and therefore, Kernel ABC works well for estimating the unknown ground-truth parameters in this setting.

Posterior Estimation Quality by Kernel Length Scale Selection of MMD Estimator Figure 5.6 shows the estimated posterior distributions. The center-upper ("baseline-1") and right-upper ("baseline-2") plots represent the posterior distributions obtained from the two

baseline methods using Kernel ABC with non-optimised MMD. The differences in these two lie in the specification of the length scale (bandwidth) parameter of a Gaussian kernel. The center-upper ("baseline-1") is the result where the length scale is calculated by the dimension-wise median heuristic, and the right-upper ("baseline-2") is the result where the length scale is calculated by the single median heuristic. The plots of the two baselines represent that the specification of the length scale strongly influences the posterior distribution of the kernel ABC. In particular, the estimation quality is not excellent with the MMD using the single median heuristic, suggesting that the single median heuristic fails to capture key characteristics in the datasets. On the other hand, the Kernel ABC estimation with the dimension-wise median heuristic (center-upper) is similar with the estimated posterior distribution by the proposed approach (left-upper). However, the baseline-1 (center-upper) allows for certain probabilities of $\theta = (0,1)$ and $\theta = (1,0)$ when the real value of θ_{real} is $(1,1)$, while the proposed approach successfully suppresses these unpreferred probabilities for $\theta = (0,1)$ and $\theta = (1,0)$. These observations suggest that the MMD using the dimension-wise median heuristic can produce a reasonably good posterior distribution even without ARD-weights optimisation. However, MMD with ARD-weights optimisation can further improve the estimation accuracy.

Parameter Estimation Result by Rejection ABC (Baselines) We represent the difficulty of choosing the threshold in Rejection ABC.

Figure 5.7 shows a histogram of the distances between observed dataset S_{real} and simulated datasets $S_{\theta_1}, \ldots, S_{\theta_N}$, where the distance metric is either given by the summary statistic (left) or by the MMD (right). Each of these baseline methods sets a threshold for such distances to select parameters from $\theta_1, \ldots, \theta_N$ to accept. Table 5.2 summarises parameter estimation results depending on different thresholds (determined by percentile values). The results suggest the performance of Rejection ABC is sensitive to the threshold used. Based on Table 5.2, we selected the "best" threshold that achieves the highest F1 score for each setting, and show the histogram of the resulting accepted parameters in Figure 5.6 (bottom). Note that, in practice, one **cannot** choose the threshold in this way, because one does not know the ground-truth parameter and thus cannot calculate the F1 score. In each setting, the posterior distribution obtained with the optimised MMD concentrates on the ground-truth parameter, which suggests the effectiveness of the MMD as a distance metric. In contrast, the mean summary statistic does not provide sharp posterior distributions, suggesting that the mean statistic misses key characteristics in the datasets needed for parameter calibration.

5.3.4 Assessment Result: Uncertainty Quantification

Figure 5.8 shows the uncertainty distributions of the travel time KPI, computed from all the vehicles (left) and only from the AD group (right). The travel group "AD" is expected to be

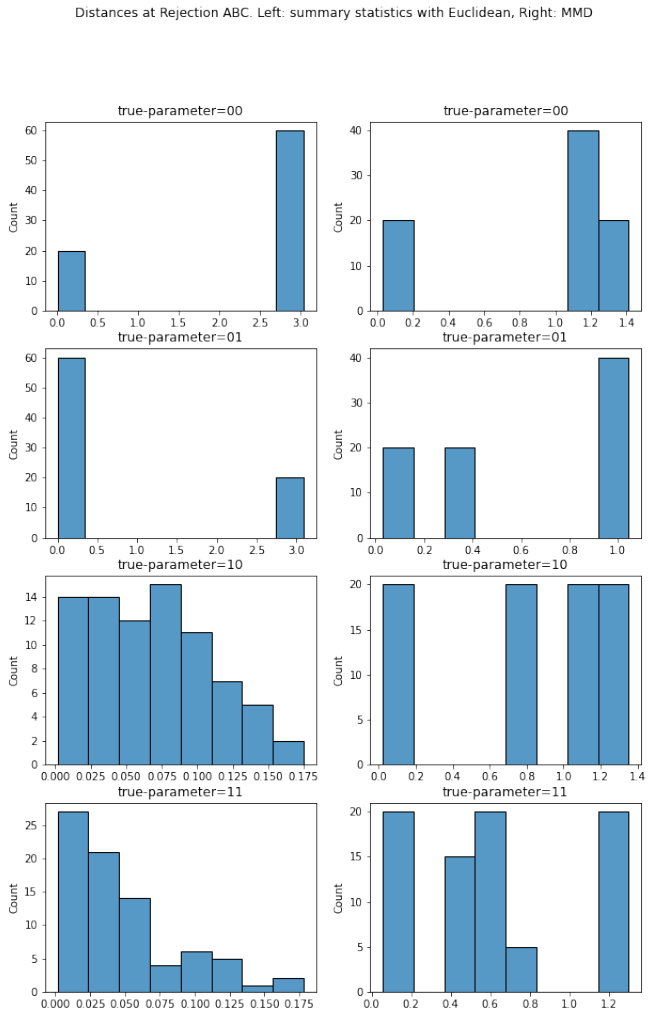


Figure 5.7: Distributions of the distance between 80 S_θ samples and S_{real} for Rejection ABC. The left side shows the distance of the summary statistics. The right side shows the distance of MMD.

influenced by the lane blockage in case of $\theta = (0,0)$, as the travel route goes through the edge where the two lanes may be closed. Our quantified uncertainties about travel time confirm that lane blockage events lead to longer travel times. The quantified travel time KPI do not significantly change in case of $\theta = (0,1),(1,0),(1,1)$. That may be because the vehicle driver's configuration of the traffic simulator ("SUMO"). The vehicle drivers attempt to drive through the original route if the one of lanes are available (either $\theta = (0,1)$ or $\theta = (1,0)$), and the travel time is not significantly affected by the lane blockage. Still, the travel time at $\theta = (0,1)$ and $\theta = (1,0)$ become with wider variances than the case of $\theta = (1,1)$. That would suggest traffic

Table 5.2: Performance of Rejection ABC per threshold.

true_parameter	metric	percentile	threshold	precision	recall	f
(0, 0)	summary-stats	5	0.020	1.000	0.2	0.333
(0, 0)	summary-stats	10	0.029	1.000	0.4	0.571
(0, 0)	summary-stats	20	0.057	1.000	0.8	**0.889**
(0, 0)	summary-stats	40	2.953	0.625	1	0.769
(0, 0)	summary-stats	80	2.994	0.313	1	0.476
(0, 1)	summary-stats	5	0.003	0.500	0.1	0.167
(0, 1)	summary-stats	10	0.009	0.375	0.15	0.214
(0, 1)	summary-stats	20	0.021	0.313	0.25	0.278
(0, 1)	summary-stats	40	0.033	0.333	0.55	**0.415**
(0, 1)	summary-stats	80	3.006	0.313	1	0.476
(1, 0)	summary-stats	5	0.009	0.500	0.1	0.167
(1, 0)	summary-stats	10	0.014	0.750	0.3	0.429
(1, 0)	summary-stats	20	0.024	0.563	0.45	0.500
(1, 0)	summary-stats	40	0.048	0.438	0.7	**0.538**
(1, 0)	summary-stats	80	0.107	0.313	1	0.476
(1, 1)	summary-stats	5	0.004	0.500	0.1	0.167
(1, 1)	summary-stats	10	0.008	0.375	0.15	0.214
(1, 1)	summary-stats	20	0.012	0.375	0.3	0.333
(1, 1)	summary-stats	40	0.029	0.406	0.65	**0.500**
(1, 1)	summary-stats	80	0.080	0.313	1	0.476
(0, 0)	MMD	5	0.037	1.000	0.2	0.333
(0, 0)	MMD	10	0.039	1.000	0.4	0.571
(0, 0)	MMD	20	0.045	1.000	0.8	**0.889**
(0, 0)	MMD	40	1.112	0.625	1	0.769
(0, 0)	MMD	80	1.348	0.313	1	0.476
(0, 1)	MMD	5	0.033	1.000	0.2	0.333
(0, 1)	MMD	10	0.036	1.000	0.4	0.571
(0, 1)	MMD	20	0.046	1.000	0.8	**0.889**
(0, 1)	MMD	40	0.348	0.625	1	0.769
(0, 1)	MMD	80	1.022	0.313	1	0.476
(1, 0)	MMD	5	0.026	1.000	0.2	0.333
(1, 0)	MMD	10	0.029	1.000	0.4	0.571
(1, 0)	MMD	20	0.035	1.000	0.8	**0.889**
(1, 0)	MMD	40	0.801	0.625	1	0.769
(1, 0)	MMD	80	1.299	0.313	1	0.476
(1, 1)	MMD	5	0.063	1.000	0.2	0.333
(1, 1)	MMD	10	0.066	1.000	0.4	0.571
(1, 1)	MMD	20	0.079	1.000	0.8	**0.889**
(1, 1)	MMD	40	0.509	0.625	1	0.769
(1, 1)	MMD	80	1.265	0.313	1	0.476

congestion occurs by the lane blockage, and the travel time becomes longer.

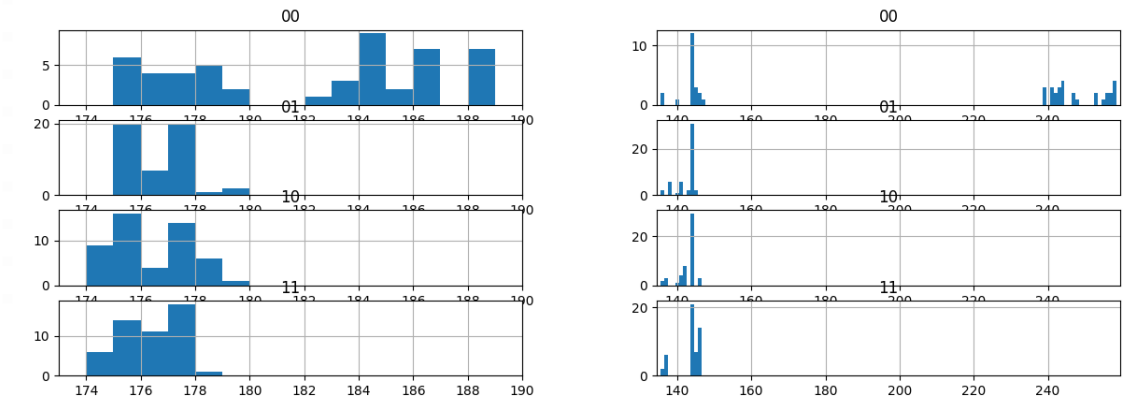


Figure 5.8: Empirical posterior predictive distributions of the travel time KPI computed from all the vehicles (left) and only from the AD group (right), for each of the four cases of the unknown ground-truth parameters. The horizontal axis indicates travel time (in seconds), and the vertical axis shows frequencies (histogram) in the 50 simulations from the posterior distribution.

Table 5.3: Means and variances of the uncertainty distributions for the travel time KPI.

group	$\boldsymbol{\theta}_{\text{real}}$	mean	variance
all	(0,0)	181.960	4.797
all	(0,1)	176.190	1.115
all	(1,0)	175.960	1.388
all	(1,1)	175.970	1.042
AD	(0,0)	203.98	52.355
AD	(0,1)	142.41	2.639
AD	(1,0)	142.67	2.529
AD	(1,1)	143.41	3.200

Table 5.4: Values of the criterion (5.9) for different regularisation constants λ, for each of the four cases of the ground-truth parameters $\boldsymbol{\theta}_{\text{real}}$.

$\lambda \setminus \boldsymbol{\theta}_{\text{real}}$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
1.78E-03	0.970	0.763	0.065	0.842
3.16E-03	0.834	0.563	0.041	0.651
5.62E-03	0.738	0.437	**0.024**	0.549
1.00E-02	0.651	0.367	0.046	0.459
1.78E-02	0.571	0.307	0.087	0.355
3.16E-02	0.472	**0.238**	0.150	**0.226**
5.62E-02	0.316	0.393	0.252	0.401
1.00E-01	**0.087**	0.559	0.402	0.551
1.78E-01	0.219	0.775	0.604	0.724
3.16E-01	0.582	1.032	0.855	0.947
5.62E-01	0.955	1.284	1.129	1.189
1.00E+00	1.286	1.502	1.387	1.426

Chapter 6

Conclusion and Future Work

This thesis addresses a ubiquitous problem in comparing a pair of high-dimensional data and offers two-sample testing variable selection as a solution for humans.

In Chapter 3, we focus on the ARD weights $a \in \mathbb{R}^D$ of the ARD kernel (2.3)

$$k(x, y) = \exp\left(-\frac{1}{D}\sum_{d=1}^{D}\frac{a_d^2(x_d - y_d)^2}{\gamma_d^2}\right),$$

and selects variables using the optimised ARD weights in the MMD estimator optimisation problem (Sutherland et al., 2017). To correctly capture the shape of differences between P, Q, we add a regularisation term to this optimisation problem Eq. (3.2). Since the regularised optimisation problem demands a parameter λ, we introduce automatic data-driven methods of choosing an appropriate λ (Section 3.4.1) and aggregating several optimised ARD weights (Section 3.4.2).

In Chapter 4, we address the practical demands of comparing a pair of high-dimensional time-series data (spatial-temporal data). The proposed framework divides a set of time steps into several buckets and executes the variable selection in each bucket. Since the framework is generic, one can employ any variable selection method, however, we empirically show that the MMD-optimisation-based variable selection methods are effective.

Chapter 5 focused on parameter calibration problems, especially when a model execution takes an expensive computational cost, and proposed the Human-In-Loop parameter calibration. The MMD-based variable selection gives humans information for judging if continuing or stopping the calibration iteration. By integrating the MMD estimator as a distance metric of KernelABC Eq. (5.5), parameter estimation is possible.

The proposed methods in this thesis can be applied to many practical problems and will pro-

vide humans with helpful insights. Yet, there is still space for improvement and we summarise some remaining challenges in the following.

Faster MMD Estimator Optimisation One potential future direction is to make the proposed methods scalable to large datasets. Since evaluating the objective function Eq. (3.2) requires quadratic complexity to the sample size, it can be slow when the sample size is large. A straightforward extension is to adapt the linear MMD estimator (Gretton et al., 2012b); We have empirically observed that the linear MMD estimator leads to poorer variable selection (See Appendix I.2). Yet, there are already many strategies for realising faster computation of MMD estimates, such as B-Tests (Zaremba et al., 2013), Fourier-feature type approximations (Jitkrittum et al., 2016). Some of them would be well-applicable to our variable selection approach.

Relationships between Variables (Interdependency) Our proposed method successfully picks a set of representative variables between two probability distributions. Still, in practical analysis work, there are demands for variable relationships; the question is "which variables are strongly dependent on the representative variables? How are their strength?". An example of such analysis demand is comparing image data as in Section 3.5.4. Although this example selects a set of pixels (variables) representing differences between cat and dog faces, these pixels are insufficient to confirm. Ideally, we can represent the relationships of the selected pixels and other surrounding pixels. It is possible to obtain this information in a post-processing approach (such as independently computing correlations to pixel-to-pixel); however, it is better to represent such variable-dependent information in an optimisation problem.

Asymptotic Analysis of Variable Selection by Regularised Optimisation Another key direction is developing a statistical theory to understand better the proposed methods, such as the consistency of the proposed methods for estimating the ground-truth variables S in Definition 1. To this end, one needs to analyse the asymptotic behaviour of the solution of the regularised objective Eq. (3.2), but this may be challenging because of the nonconcavity of the ratio objective Eq. (2.4) concerning the ARD weights.

Tracking Model Parameter from Selected Variables A part of model validation is between a surrogate model and the target system to be approximated (See an example in Section 4.5). When a surrogate model outcome does not reach the acceptable range of accuracy, a model validator naturally wonders what causes the error. Possibly, an error lies in the training step of a surrogate model, in the case where the surrogate model is tractable. In this case, ideally, variable selection can tell not only selected variables in the surrogate model outcomes

but also the relationships between selected variables and trainable model parameters. This relationship would help the model validator understand where the outcome errors originate from. Such research direction is possible when we set an MMD estimator as the objective function of the surrogate model and optimise the MMD estimator (ARD weights) and the surrogate model parameter together. Indeed, MMD is often used as an objective function in many machine-learning tasks (Sutherland et al., 2017).

Social Implementation for Generative Model Validation Recently[1], especially by emerging practical generative models, models are progressively more intimate to society than before. *Dall-E* (Ramesh et al., 2021) and *Stable Diffusion Model* (Rombach et al., 2022) are public available tools on the internet with easy-to-use WEB interfaces and let people recognise its practical application. Social attention is more important than ever to models, and model validation is crucially important for models to be trusted by society. For sure, our variable selection is an effective approach for validating these models; however, it is challenging to confirm these models for all use cases of end model users in society. Since models are widely accessible to society, model use cases would be more diverse such that the model developers are not able to cover all application cases. In a perfect world, all model users are conscious of the importance of model validation and conduct it for their applications; however, in practice, it is impossible. First of all, in most end-user-level application cases of generative models, there are no datasets to be compared. A possible workaround may be referring to model validation results that someone else conducted already and judging if further model validation is necessary or not. If there were such automated checking systems or software, model validation at the end-user level would ensure model outcome quality at a certain level. This future work is out of the variable selection framework; however, we believe that necessary for the social implementation of generative models in a desirable direction.

—·—

The research of variable selection in comparing datasets is not paid attention compared to other machine learning fields. We believe that this thesis lays the foundations for promoting the variable selection study.

[1] It would be controversial to say the exact year when models are more intimate to society. In this thesis work, we regard that 2021 and 2022, that *Stable Diffusion Model* and *Dall-E* are publicly available on the internet, are the years where models started to be intimate to society.

Chapter 7

Appendix

A History of Two Sample Testing

This section intends to provide a historical overview of two sample testing (TST). For the efforts of scientific researches toward TST with high dimensional data, readers are encouraged to refer to Section 2.1.

TST is a common problem in the modern statistics where one compares two sets of data points sampled from two probability distributions. Under TST, one sets the null hypothesis H_0 and the alternative hypothesis H_1. When two probability distributions are different, TST rejects the null hypothesis H_0. Before exploring the historical works of TST, we first would like to introduce early days of the modern statistics.

A.1 Historical Roots of Statistics

The statistics referred to data collection and description methods for a state in the long history of human beings, as the etymology of the term *statistics* represents that "It is derived from stato (state), and a statista is a man who deals with affairs of the state" (Hald, 1990b). The term *statistics* has gradually altered its meaning and definition in the context of modern science and started to seek hypotheses such that the definition represents "statistics is a mathematical and conceptual discipline that focuses on the relation between data and hypotheses." (Romeijn, 2022). Modern statistics came with the rapid growth of sciences in the early 20th century. Modern science started as scientists started to seek "how things work" instead of "absolute truth" (Pagels, 2009) and exploit statistics to answer this question.

A.2 Emergence of Modern Statistical Methods

Modern statistical tools have appeared in days of the early 20th century with the trend of the modern science. *Francis Galton* is said to be one of pioneers in the early age by connecting the statistics with the biology and the psychology, and introducing the concept of the correlation coefficient or regression (Fancher, 1997). *Karl Pearson*'s works (Barnard, 1992) are regarded as one of the breakthrough works in this early days in establishing the field of mathematical statistics, developing the chi-squared test (Kotz and Johnson, 1992). In a few years later in 1908, another breakthrough work *Student's t-distribution* and *Student's t-test* were by *William Gosset* (Student, 1992)[1]. Although Pearson's and Gosset's works are the fundamental works of the modern statistics, *Ronald Fisher* is often said to be the "founder of the modern statistics" (Rao, 1992). His work (Fisher, 1992) introduced not only F-distribution but also an essential concept of hypothesis testing: the null hypothesis, alternative hypothesis. In 1933, *Egon Pearson* and *Jerzy Neyman* introduced and added essential concepts for the hypothesis testing: *Type-I* and *Type-II* errors (Neyman and Pearson, 1992). These two errors has provided a clear framework for evaluating and comparing the performance of different hypothesis tests to statisticians.

In the development of modern statistics, TST has been regarded as an extension case or a subdomain of *hypothesis testing*. The primary distinction is found in the comparison's counterpart. Hypothesis testing involves comparing a given sample to a hypothesised distribution and determining how well the sample matches the distribution. In contrast, TST focuses on two sets of samples to determine if they likely originate from the same population.

A.3 Classic Nonparametric Two Sample Testing

Nonparametric statistics is the statistical approaches without knowing the underlying probability distribution, and the nonparametric TST is one part of it. The term "nonparametric" test became commonly in 1940s (Dodge, 2008), however, there was an attempt of the nonparametric testing even before.

It is said to be *John Arbuthnot* is the first person who used not only nonparametric hypothesis testing, but also hypothesis testing in the history of statistics, even though he did not use any terms of the modern statistics vocabularies such as p-value (Hald, 1990a). In 1710, Arbuthnot calculated a probability under the assumption of "the probability of male and female birth were equal" based on the birth records in London during 82 years. The calculated probability is $\frac{1}{2^{282}}$, and he commented that this probability as "Art, Not Chance, that governs" that we interpret rejecting the null hypothesis in the modern statistics. He thought this extreme

[1] Gosset used the penname *Student* since *Guinness*, for which he had worked, disallowded scientific publications of employees (enc, 2008a). At Guinness, Gosset has purshed to improve planting conditions of barley by the statistics. Since barley harvesting is once per year, the number of observations is limited. His Student-t test has been motivated by the practical problem.

low probability is a proof that human's sex ratio is the event given by God, since his central motivation has lied in proving the existing of God.

Despite Arbuthnot's work, the nonparametric testing domain has had less developments until 20th century (enc, 2008b). The well-known nonparametric testing work came in 1933 by *Andrey Kolmogorov*. He constructed the goodness-of-fit test between the Empirical Distribution Function (EDF) made with the collected samples and a true probability distribution. After that, *Vladimir Smirnov* extended Kolmogorov's work to TST in 1939. TST method by Kolmogorov and Smirnov is known as the Kolmogorov-Smirnov test (KS test) (Kolmogorov, 1992).

A notable work in the nonparametric TST is the *Mann-Whitney U test* by *Henry Mann* and *Donald Whitney* (Mann and Whitney, 1947). The Mann-Whitney U test measures the difference of the two sample distributions by the rank of the samples. This approach using the ranking is called *Ranking Based Two Sample Testing* nowadays.

Compared with the parametric testing, nonparametric testing has a strong advantage that it does not require the underlying assumptions about the probability distribution. On the other hand, nonparametric testing tends to mark more Type-II errors than the parametric testing when the data meet the parametric assumptions (Dodge, 2008). Therefore, nonparametric testing has required more samples for realising the compatible performance with the parametric testing.

B Model Validation and Two Sample Testing

While this thesis define model validation as "a procedure to confirm that model outcomes fall within the acceptable range of accuracy expected by the applicationt't', the term "model validation" may contain a wide range of procedures and approaches and refer to them. This section first breaks down the term "model validation" and clarify the connection between the model validation and two sample testing. Most of descriptions are from the work in Sargent (2010) that focuses on the development of a simulation model.

Before delving into the model validation, we want to clarify the term *model validator* and the stakeholders of the model development process. Sargent's work defines a term *team* consisting of two parts: *a model development team* and *model sponsors/model users*. The development team is responsible for the whole procedure of the developments, starting from designing and implementing the model, including verification and validations. On the other hand, the model sponsors/users are responsible for checking the model behaviours and making the decision about the verifications and validations. The model validation is done when the development team and model users make agreement about the model's behaviours. In this thesis work, we suppose that *model validator* can be the model development team or model sponsors/users.

B.1 Model Validation Procedures

A model is an approximation of the real world system. A model development procedure consists of four steps in the order: model verification, model validation, model credibility, and model accreditation.

Among these four steps, the model credibility and model accreditation, final two steps involve model users' confidence about the model and political intention by a government or society. The model credibility is "developing in (potential) users the confidence they require in order to use a model", and the model accreditation is "official certification that a model, simulation, or federation of models and simulations and its associated data are acceptable for use for a specific application", which is defined by the U.S.A Department of Defense. These two steps are rather subjective decisions by society or organisations, and are not the main focus of this thesis.

The model verification is a procedure to confirm that the model is correctly implemented and behaves as expected. Logical testing, code review, and unit testing are typical procedures for the model verification. In the model validation, the model validator confirms models' outcomes lie within the acceptable and expected range of accuracy. We note that the model verification and model validation are not always in the perfect waterfall order. The model validator may discover errors concerning the model verification (e.g. a logic error) in the model validation. In such cases, the model developer is required to execute the model verification and fix errors. This thesis mainly targets the model validation.

B.2 Model Validation Procedure

Sargent's work categorises the model validation into two procedures: *conceptual model validation* and *operational validity*. The conceptual model validation involves validation (and partial verification) of the model's underlying assumptions and logic. On the other hand, the operational validity intends to confirm that the model's outcomes lie within the acceptable range of accuracy.

Table 1 describes the classification of operational validity, based on whether the real system is observable and whether the validation approach is subjective or objective. The observable real system is one in which we have access to a counterpart model (or reference model) that allows us to obtain its outcomes on demands. In contrast, the non-observable system is one where we lack access to a counterpart model and have limited available data. An example of this would be developing a model that simulates natural events. Our model validation proposal in this thesis offers a solution for objective approaches in both observable and non-observable systems across the four categories in Table 1.

Table 1: Classification of approaches for the operation validity from (Sargent, 2010).

	Observable System	**Non-observable System**
Subjective Approach	Comparing a developed model with a counterpart using graphical displays.	Exploring a developed model behaviour.
Objective Approach	Comparing a developed model with a counterpart using statistical tests.	Comparing a developed model with given samples using statistical tests.

Parameter Calibration A model typically has a set of parameters that significantly influence its behaviour. Even when the models underlying logic is correct, improper parameters can lead to outcomes that do not align with those produced by a counterpart model or real-world data. Thus, parameter calibration is a vital step in validating the model. Normally, the parameter calibration is done during the operational validity. Then, the model validator checks the compatibility of the model outcomes and the counterpart. Parameter calibration has a long history as a field of research, which we discuss further in Chapter 5.

Model Validation Methods As we show in Table 1, the comparison is a fundamental procedure for the objective operational model validity. Sargent's work introduces three approaches for the operational model validity: *graphical comparison of data, confidence interval*, and *two sample testing*, and it recommends using the confidence interval or hypothesis tests to make the decision more objective.

The confidence interval is an approach that compares summary statistics (e.g. the mean or variance value) of the model's outcomes with the counterpart. In this sense, we could regard the confidence interval approach as a part of the hypothesis testing approach.

The model validation proposed in this thesis is categorised into two sample testing. As Sargent explains, two sample tests offer an objective approach to model validation, thus preferred approach. However, challenges emerge when both the developed model and its counterpart involve high-dimensional data. As Chapter 1 describes, this thesis presents interpretable model validation by demonstrating the supporting information provided by two sample tests.

C Preliminary Theoretical Analysis

We first present a theoretical analysis of MMD defined with a generalised version of the ARD kernel (2.3), defined below. For each variable $d = 1, \ldots, D$, let $\phi_d : \mathbb{R} \mapsto \mathbb{R}$ be a one-dimensional positive definite function. Then, for ARD weights $a_1, \ldots, a_d \geq 0$, we define a generalized ARD

kernel as

$$k(x,x') := \prod_{d=1}^{D} \phi_d\left(\sqrt{a_d^2(x_d - x'_d)^2/D}\right) \tag{1}$$

for $x = (x_1,\ldots,x_D)^\top \in \mathbb{R}^D$ and $x' = (x'_1,\ldots,x'_D)^\top \in \mathbb{R}^D$. For example, if $\phi_d(r) := \exp(-r^2/\gamma_d^2)$ for $r \geq 0$, the Gaussian ARD kernel in (2.3) is recovered, as we have $\prod_{d=1}^D \phi_d\left(\sqrt{a_d^2(x_d - x'_d)^2/D}\right) = \prod_{d=1}^D \exp\left(-\frac{1}{D}\frac{a_d^2(x_d-y_d)^2}{\gamma_d^2}\right) = \exp\left(-\frac{1}{D}\sum_{d=1}^D \frac{a_d^2(x_d-y_d)^2}{\gamma_d^2}\right)$. Likewise, if $\phi_d(r) := \exp(-|r|/\gamma_d)$, then a Laplace-type ARD kernel is obtained.

For $S \subset \{1,\ldots,D\}$, we define $k_S : \mathbb{R}^{|S|} \times \mathbb{R}^{|S|} \mapsto \mathbb{R}$ as the restriction of the ARD kernel (1) on S:

$$k_S(x_S, x'_S) := \prod_{d\in S} \phi_d\left(\sqrt{a_d^2(x_d - x'_d)^2/D}\right), \quad x_S, x'_S \in \mathbb{R}^{|S|}.$$

For $U = \{1,\ldots,D\}\backslash S$, $k_U : \mathbb{R}^{|U|} \times \mathbb{R}^{|U|} \mapsto \mathbb{R}$ is defined similarly. Then the kernel (1) can be written as the product $k(x,x') = k_S(x_S, x'_S)k_U(x_U, x'_U)$ for $x = (x_S, x_U) \in \mathbb{R}^D$ and $x' = (x'_S, x'_U) \in \mathbb{R}^D$.

Proposition 2 below shows an expression of MMD when $S \subset \{1,\ldots,D\}$ is the subset in Definition 1.

Proposition 2. *Let $S \subset \{1,\ldots,D\}$ be the subset in Definition 1, and let $U := \{1,\ldots,D\}\backslash S$. Then, for MMD* (2.1) *between P and Q with the generalized ARD kernel k in* (1)*, we have*

$$\mathrm{MMD}_k^2(P,Q) = \mathbb{E}[k_U(X_U, X'_U)]\mathrm{MMD}_{k_S}^2(P_S, Q_S). \tag{2}$$

where $X_U, X'_U \overset{i.i.d.}{\sim} P_U$. Moreover, we have

$$\max_{a\in\mathbb{R}^D} \mathrm{MMD}_k^2(P,Q) = \left(\prod_{d\in U} \phi_d(0)\right) \max_{a_S\in\mathbb{R}^{|S|}} \mathrm{MMD}_{k_S}^2(P,Q), \tag{3}$$

Proof. Appendix C.2. □

To understand Proposition 2, consider an example where $D = 5$, $S = \{1,3\}$ and $U = \{2,4,5\}$. Then Eq. (2) shows that the MMD between the 5-dimensional distributions P and Q can be written as the MMD between the 2-dimensional marginals $P_{\{1,3\}}$ and $Q_{\{1,3\}}$ multiplied by the constant $\mathbb{E}[k_{\{2,4,5\}}(X_{\{2,4,5\}}, X'_{\{2,4,5\}})]$. Consequently, as shown in Eq. (3), the maximum of $\mathrm{MMD}_k^2(P,Q)$ over the 5 ARD weights a_1, a_2, a_3, a_4, a_5 is equal to the product of the maximum of $\mathrm{MMD}_{k_{\{1,3\}}}^2(P_{\{1,3\}}, Q_{\{1,3\}})$ over the 2 ARD weights a_1, a_3 and the constant $\prod_{d\in\{2,4,5\}} \phi_d(0)$, which is obtained by setting the 3 ARD weights a_2, a_4, a_5 to be 0.

The above example suggests that, by maximizing the MMD over the ARD weights, it may be possible to identify the true variables S and the redundant ones U. However, this requires

that the marginal distribution $P_U = Q_U$ of the redundant variables U be non-singular, as Proposition 3 below suggests.

Proposition 3. *Consider the same setting as Proposition 2. Suppose that each $\phi_d : \mathbb{R} \mapsto \mathbb{R}$ in* (1) *satisfies $\phi(0) > \phi(r)$ for all $r > 0$. Moreover, let*

$$a^* \in \arg\max_{a \in \mathbb{R}^D} \mathrm{MMD}_k^2(P, Q)$$

be a maximiser of MMD between P and Q (which may not be unique). Let $V \subset U$ be such that for each $d \in V$, the variance of the marginal distribution P_d $(= Q_d)$ is not zero. Then, $a_V^* \in \mathbb{R}^{|V|}$ is the zero vector: $a_V^* = \mathbf{0}_{|V|}$.

Proof. See Appendix C.2. □

For example, consider the above example where $D = 5$, $S = \{1,3\}$ and $U = \{2,4,5\}$. Let $a^* = (a_1^*, a_2^*, a_3^*, a_4^*, a_5^*)$ be a maximiser of the MMD. If the variances of $P_2 = Q_2$ and $P_5 = Q_5$ are positive, and that of $P_4 = Q_4$ is zero, then $V = \{2,5\}$. In this case, Proposition 3 suggests that $a_2^* = 0$ and $a_5^* = 0$. However, the optimised weight a_4^* for the 4th variable may not be zero, as its marginal distribution $P_4 = Q_4$ has zero variance: this is the case where $P_4 = Q_4$ are a Dirac distribution δ_ξ for some fixed $\xi \in \mathbb{R}$, i.e., $X_4 \sim P_4 = Q_4$ satisfies $X_4 = \xi$ almost surely. The following example shows that, in such a case, the optimised ARD weight associated with a redundant variable $d \in U \setminus V$ can be non-zero.

Example 1. *Suppose that the d-th variable X_d of $X \sim P$ and the d-th variable Y_d of $Y \sim Q$ always take the same fixed value $\xi \in \mathbb{R}$. Then we have $d \in U \setminus V$, as $P_d = Q_d = \delta_\xi$ have zero variance. Let X_d' and Y_d' be i.i.d. copies of X_d and Y_d, respectively. For any ARD weight $a_d \in \mathbb{R}$, we then have*

$$\begin{aligned}&\phi_d(\sqrt{a_d^2(X_d - Y_d)^2}) = \phi_d(\sqrt{a_d^2(X_d - X_d')^2})\\ &= \phi_d(\sqrt{a_d^2(Y_d - Y_d')^2}) = \phi_d(\sqrt{a_d^2(\xi_d - \xi_d)^2}) = \phi_d(0).\end{aligned}$$

Therefore, the ARD weight a_d does not influence the value of the d-th kernel ϕ_d and thus that of the resulting kernel (1)*. Consequently,* any *value of the ARD weight a_d attains the maximum of the MMD in* (3)*. The same issue occurs with the objective function in* (2.4)*, since it depends on the ARD weights only through the kernel* (1)*.*

Example 1 suggests that one may fail to identify redundant variables if one optimises the ARD weights based only on the MMD or the objective function (2.4), as both depend on the datasets only through the kernel. In our example where $D = 5$, $S = \{1,3\}$, $U = \{2,4,5\}$ and $V = \{2,5\}$, if the 4th variable always takes the same value, say $\xi = 1.74$, for P and Q, this variable

is clearly redundant. However, the optimised ARD weight a_4^* can be any value to attain the maximum of MMD or the objective function (2.4). This issue provides one motivation for using regularisation, as explained next.

C.1 Proof of Proposition 1

Proof. We first prove that the uniqueness of S satisfying the conditions in Definition 1. To this end, suppose that there exist two subsets $S_1 \subset [D]$ and $S_2 \subset [D]$ satisfying the conditions in Definition 1. Then we shall prove that $S_0 = S_1 \cup S_2$ satisfies the conditions in Definition 1, which implies $S_1 = S_2$ (otherwise $S_1 \subsetneq S_0$ and $S_2 \subsetneq S_0$, which contradicts Definition 1).

To prove that S_0 satisfies Definition 1, suppose that S_0 does not satisfy it, i.e., there exists a non-empty subset $U_0 \subset S_0$ such that $P_{S_0} = P_{U_0} \otimes P_{S_0 \setminus U_0}$, $Q_{S_0} = Q_{U_0} \otimes Q_{S_0 \setminus U_0}$ and $P_{U_0} = Q_{U_0}$. Let $U_1 := U_0 \cap S_1$ and $U_2 := U_0 \cap S_2$. We either have $U_1 \neq \phi$ or $U_2 \neq \phi$, because

$$\begin{aligned}\phi &\neq U_0 \cap S_0 = U_0 \cap (S_1 \cup S_2) \\ &= (U_0 \cap S_1) \cup (U_0 \cap S_2) = U_1 \cup U_2.\end{aligned}$$

Without loss of generality, suppose $U_1 \neq \phi$. For $A \subset [D]$, let $A^c := [D] \setminus A$. Note that

$$\begin{aligned}&U_0 \setminus (U_0 \cap (S_0 \setminus S_1)) = U_0 \cap (U_0 \cap (S_0 \setminus S_1))^c \\ &= U_0 \cap (U_0^c \cup (S_0 \setminus S_1)^c) = U_0 \cap (S_0 \setminus S_1)^c \\ &= U_0 \cap (S_0 \cap S_1^c)^c = U_0 \cap (S_0^c \cup S_1) = U_0 \cap S_1 = U_1 \neq \phi.\end{aligned}$$

Similarly, we have

$$\begin{aligned}&(S_0 \setminus U_0) \setminus [(S_0 \setminus U_0) \cap (S_0 \setminus S_1)] \\ &= (S_0 \setminus U_0) \cap [(S_0 \setminus U_0) \cap (S_0 \setminus S_1)]^c \\ &= (S_0 \setminus U_0) \cap [(S_0 \setminus U_0)^c \cup (S_0 \setminus S_1)^c] = (S_0 \setminus U_0) \cap (S_0 \setminus S_1)^c \\ &= (S_0 \cap U_0^c) \cap (S_0 \cap S_1^c)^c = (S_0 \cap U_0^c) \cap (S_0^c \cup S_1) \\ &= U_0^c \cap [(S_0 \cap S_0^c) \cup (S_0 \cap S_1)] \\ &= U_0^c \cap S_1 = S_1 \setminus U_0 = S_1 \setminus U_1.\end{aligned}$$

Therefore,

$$
\begin{aligned}
P_{S_1}(x_{S_1}) &= \int P_{S_0}(x) dx_{S_0 \setminus S_1} \\
&= \int P_{U_0}(x_{U_0}) dx_{U_0 \cap (S_0 \setminus S_1)} \\
&\times \int P_{S_0 \setminus U_0}(x_{S_0 \setminus U_0}) dx_{(S_0 \setminus U_0) \cap (S_0 \setminus S_1)} \\
&= P_{U_1}(x_{U_1}) P_{S_1 \setminus U_1}(x_{S_1 \setminus U_1}),
\end{aligned}
$$

i.e., $P_{S_1} = P_{U_1} \otimes P_{S_1 \setminus U_1}$. Similarly, we have $Q_{S_1} = Q_{U_1} \otimes Q_{S_1 \setminus U_1}$. Since $P_{U_0} = Q_{U_0}$, $P_{U_1} = \int P_{U_0}(x_{U_0}) dx_{U_0 \cap (S_0 \setminus S_1)}$ and $Q_{U_1} = \int Q_{U_0}(x_{U_0}) dx_{U_0 \cap (S_0 \setminus S_1)}$, we have $P_{U_1} = Q_{U_1}$. This contradicts the assumption that S_1 satisfies the conditions in Definition 1. Therefore S_0 satisfies Definition 1.

We now prove the second assertion. Let $U \subset [D]$ be the largest subset such that $P = P_U \otimes P_{U^c}$, $Q = Q_U \otimes Q_{U^c}$ and $P_U = Q_U$, where $U^c := [D] \setminus U$. We show U^c satisfies conditions 1) and 2) in Definition 1.

To show condition 1), suppose there is $U' \subset U^c$ such that $P_{U^c} = P_{U'} \otimes P_{U^c \setminus U'}$, $Q_S = Q_{U'} \otimes Q_{U^c \setminus U'}$ and $P_{U'} = Q_{U'}$. Then $P = P_U \otimes P_{U'} \otimes P_{U^c \setminus U'} = P_{U \cup U'} \otimes P_{(U \cup U')^c}$, $Q = Q_U \otimes Q_{U'} \otimes Q_{U^c \setminus U'} = Q_{U \cup U'} \otimes Q_{(U \cup U')^c}$, and $P_{U \cup U'} = Q_{U \cup U'}$. However, U is the largest among such subsets, so we must have $U' = \phi$; thus condition 1) is satisfied.

To show condition 2), suppose there exists $S' \subset [D]$ such that $U^c \subset S'$ and condition 1) is satisfied. Define $U' := S' \setminus U^c = S' \cap U$. Because $U' \subset U$, $P = P_U \otimes P_{U^c}$, $Q = Q_U \otimes Q_{U^c}$ and $P_U = Q_U$, we have $P_{S'} = P_{U'} \otimes P_{U^c}$, $Q_{S'} = Q_{U'} \otimes Q_{U^c}$ and $P_{U'} = Q_{U'}$. Because S' satisfies condition 1), we must have $U' = \phi$. Therefore $U^c = S'$. This proves that $S = U^c$ satisfies condition 2).

□

C.2 Proof of Proposition 2

Proof. Let $X, X' \sim P$ be independent vectors from P, and write $X = (X_S, X_U)$ and $X' = (X'_S, X'_U)$. Note that, X_S and X_U are independent, and so are X'_S and X'_U, because $P = P_S \otimes P_U$ by assumption. Likewise, let $Y, Y' \sim Q$ be independent vectors from Q, and write $Y = (Y_S, Y_U)$ and $Y' = (Y'_S, Y'_U)$. Then X_S and X_U are independent, and so are X'_S and X'_U, because $Q = Q_S \otimes Q_U$ by assumption. As $P_U = Q_U$, we have X_U, X'_U, Y_U and Y'_U are i.i.d., which implies that

$\mathbb{E}[k_U(X_U, X'_U)] = \mathbb{E}[k_U(Y_U, Y'_U)] = \mathbb{E}[k_U(X_U, Y_U)]$. Using these properties, we have

$$
\begin{aligned}
&\mathrm{MMD}_k^2(P,Q) \\
&= \mathbb{E}[k(X,X')] + \mathbb{E}[k(Y,Y')] - 2\mathbb{E}[k(X,Y)] \\
&= \mathbb{E}[k_S(X_S,X'_S)k_U(X_U,X'_U)] + \mathbb{E}[k_S(Y_S,Y'_S)k_U(Y_U,Y'_U)] \\
&\quad - 2\mathbb{E}[k_S(X_S,Y_S)k_U(X_U,Y_U)] \\
&= \mathbb{E}[k_S(X_S,X'_S)]\mathbb{E}[k_U(X_U,X'_U)] \\
&\quad + \mathbb{E}[k_S(Y_S,Y'_S)]\mathbb{E}[[k_U(Y_U,Y'_U)] \\
&\quad - 2\mathbb{E}[k_S(X_S,Y_S)]\mathbb{E}[k_U(X_U,Y_U)] \\
&= \mathbb{E}[k_U(X_U,X'_U)]\left(\mathbb{E}[k_S(X_S,X'_S)] + \mathbb{E}[k_S(Y_S,Y'_S)]\right. \\
&\qquad \left. -2\mathbb{E}[k_S(X_S,Y_S)]\right) \\
&= \mathbb{E}[k_U(X_U,X'_U)]\mathrm{MMD}_{k_S}^2(P_S,Q_S),
\end{aligned}
$$

which proves the first assertion. For the second assertion, we have

$$
\begin{aligned}
&\max_{a_U \in \mathbb{R}^{|U|}} \mathbb{E}[k_U(X_U,X'_U)] \\
&= \max_{a_U \in \mathbb{R}^{|U|}} \mathbb{E}\left[\prod_{d\in U} \phi_d\left(\sqrt{a_d^2(X_d - X'_d)^2/D}\right)\right] \\
&\leq \prod_{d\in U} \phi_d(0),
\end{aligned}
$$

where the last inequality follows from each ϕ_d being a monotonically decreasing function. Therefore

$$
\begin{aligned}
&\max_{a\in\mathbb{R}^D} \mathrm{MMD}_k^2(P,Q) \\
&= \max_{a_U \in \mathbb{R}^{|U|}} \mathbb{E}[k_U(X_U,X'_U)] \max_{a_S\in\mathbb{R}^{|S|}} \mathrm{MMD}_{k_S}^2(P,Q) \\
&\leq \prod_{d\in U} \phi_d(0) \max_{a_S\in\mathbb{R}^{|S|}} \mathrm{MMD}_{k_S}^2(P,Q),
\end{aligned}
$$

which proves the assertion. □

Proof of Proposition 3

Proof. Let

$$
a^* := (a_S^*, a_T^*) \in \arg\max_{a\in\mathbb{R}^D} \mathrm{MMD}_k^2(P,Q). \tag{4}
$$

By Proposition 2, we have

$$a_S^* \in \arg\max_{a_S \in \mathbb{R}^{|S|}} \mathrm{MMD}^2_{k_S}(P_S, Q_S),$$
$$a_T^* \in \arg\max_{a_T \in \mathbb{R}^{|T|}} \mathbb{E}[k_U(X_U, X'_U)].$$

For a random variable $Z \in \mathbb{R}$, denote by $\mathrm{Var}[Z]$ the variance of Z. Then, by assumption, we have $\mathrm{Var}[X_i] = \mathrm{Var}[X'_i] > 0$ for $i \in V \subset T$. This implies that

$$\mathbb{E}[(X_i - X'_i)^2] = 2\mathrm{Var}[X_i] > 0,$$

where X'_i is an independent copy of X_i. Therefore, there exists a constant $\varepsilon_i > 0$ such that

$$\Pr((X_i - X'_i)^2 > \varepsilon_i^2) > 0. \tag{5}$$

We will show that $a_i^* = 0$ for $i \in V \subset T$ with proof by contradiction. To this end, suppose $a_i^* \neq 0$. Then,

$$\begin{aligned}
&\max_{a_T \in \mathbb{R}^{|T|}} \mathbb{E}[k_U(X_U, X'_U)] \\
&= \mathbb{E}\left[\prod_{d \in U} \phi_d\left(\sqrt{(a_d^*)^2 (X_d - X'_d)^2 / D}\right)\right] \\
&= \Pr((X_i - X'_i)^2 > \varepsilon_i^2) \\
&\quad \times \mathbb{E}\left[\prod_{d \in U} \phi_d\left(\sqrt{(a_d^*)^2 (X_d - X'_d)^2 / D}\right) \middle| (X_i - X'_i)^2 > \varepsilon\right] \\
&+ \Pr((X_i - X'_i)^2 \leq \varepsilon_i^2) \\
&\quad \times \mathbb{E}\left[\prod_{d \in U} \phi_d\left(\sqrt{(a_d^*)^2 (X_d - X'_d)^2 / D}\right) \middle| (X_i - X'_i)^2 \leq \varepsilon_i^2\right] \\
&\overset{(A)}{\leq} \Pr((X_i - X'_i)^2 > \varepsilon_i^2) \phi_i\left(a_i^* \varepsilon_i / \sqrt{D}\right) \prod_{d \in U \setminus \{i\}} \phi_d(0) \\
&\quad + \Pr((X_i - X'_i)^2 \leq \varepsilon_i^2) \prod_{d \in U} \phi_d(0) \\
&\overset{(B)}{<} \prod_{d \in U} \phi_d(0) = \max_{a \in \mathbb{R}^{|U|}} \mathbb{E}[k_U(X_U, X'_U)]
\end{aligned}$$

where (A) follows from ϕ_i being a monotonically decreasing function (as it is a positive definite function) and $a_i^* > 0$, and (B) from (5) and $\phi_{k_T}\left(a_i^* \varepsilon / \sqrt{D}\right) < \phi_i(0)$. This contradicts that $a_1^*, \ldots, a_D^*$ are a maximiser in (4), implying that the assumption $a_i^* \neq 0$ is false. Therefore $a_i^* = 0$ for $i \in V \subset T$.

□

D Parameter Range Selection for Regularisation Parameter

D.1 Heuristic Regularisation Parameter Range Selection

Algorithm 4 Generating Candidate regularisation Parameters

Input: λ_{lower}: the lower bound (default value: 0.01). N_λ: the number of candidate parameters (default value: 6). $(\mathbf{X}, \mathbf{Y})$: a pair of data. N_{history}: the number required for the stopping condition (default value: 3).

Output: Λ: a set of candidate regularisation parameters.

1: $\lambda = \lambda_{\text{lower}}$
2: $S_{\text{history}} \leftarrow []$ ▷ Initialize an empty list
3: **While** StopCriteria$(\hat{S}_\lambda, S_{\text{history}}, N_{\text{history}})$ == False; **do**
4: Obtain $a_\lambda \in \mathbb{R}^D$ by solving (3.2) using λ and $(\mathbf{X}, \mathbf{Y})$.
5: Obtain $\hat{S}_\lambda$ using a_λ as in Section 3.3.1.
6: AppendToList$(S_{\text{history}}, \hat{S}_\lambda)$
7: Update(λ).
8: **End While**
9: $\lambda_{\text{upper}} = \lambda$.
10: $\text{step}_\lambda = (\lambda_{\text{upper}} - \lambda_{\text{lower}})/(N_\lambda - 1)$.
11: $\Lambda = \{\lambda_{\text{lower}}, \lambda_{\text{lower}} + \text{step}_\lambda, \lambda_{\text{lower}} + 2\text{step}_\lambda \ldots,$
$\lambda_{\text{lower}} + (N_\lambda - 1)\text{step}_\lambda, \lambda_{\text{upper}}\}$.

1: Function StopCriteria$(\hat{S}_\lambda, S_{\text{history}}, N_{\text{history}})\{$
2: **if** $|\hat{S}_\lambda| == 1$ **then**
3: Return **True**
4: **else if** $S_{\text{history}}[-1] ==, \ldots, == S_{\text{history}}[-N_{\text{history}}]$ **then**
5: Return **True**
6: **else**
7: Return **False**
8: **end if**
9: }

1: Function Update$(\lambda_{\text{previous}})$ {
2: **if** $0 < \lambda_{\text{previous}} < 1.0$ **then**
3: Return $2\lambda_{\text{previous}}$
4: **else**
5: Return $\lambda_{\text{previous}} + 0.5$
6: **end if**
7: }

Algorithm Table 4 describes the procedure of the `heuristic-lambda-range`. In the algorithm, S_{history} is initialised as an empty list. The while-iteration from lines 3 to 7 continues as

long as the function StopCriteria returns False. StopCriteria checks two conditions: 1) if the number of selected variables $\hat{S}_\lambda$ is one, i.e., $|\hat{S}_\lambda| = 1$; and 2) if the selected variables do no change for the previous N_{history} iterations. In the latter (Line 4 of StopCriteria), $S_{\text{history}}[-1]$ refers to the set of variables in the last iteration, and $S_{\text{history}}[-N_{\text{history}}]$ is that in the N_{history}-th previous iteration.

In the main algorithm, line 4 optimises the ARD weights a_λ by numerically solving Eq. (3.2) using the current value of the regularisation parameter λ and data $(\mathbf{X}, \mathbf{Y})$. Based on these ARD weights, line 5 selects variables $\hat{S}_\lambda$ as explained in Section 3.3.1. In line 6, the function AppendToList appends $\hat{S}_\lambda$ to the list S_{history}, which is used in StopCriteria. In line 7, the function Update increases the regularisation parameter λ for the next iteration. If $0 < \lambda < 1$, the function Update increases λ by multiplying 2; if $1 < \lambda$, it adds 0.5 to λ. We have designed this procedure to speed up the search, as we observed that the upper bound is often above 1 in our preliminary analysis.

Once the upper bound λ_{upper} is determined by StopCriteria, the algorithm outputs a set of N_λ candidate parameters in Lines 10 and 11.

Drawbacks Two drawbacks in the `heuristic-lambda-range` are as follows: 1. a search value step for λ is heuristically defined (such as $2\lambda_{\text{previous}}$, $\lambda_{\text{previous}}+0.5$ in `Function Update`).; 2. λ_{lower} is fixed as a hyperparameter value. These drawbacks motivate us to propose the next method, MMD Model Selection Optuna.

The first drawback causes a failure that `heuristic-lambda-range` may jump over the optimal λ_{upper}. The search step of $2\lambda_{\text{previous}}$ is too rough, although the optimal λ_{upper}, empirically often, exists at a the range of > 1.0. We exemplify the failure case due to this rough search step; under more noisy datasets settings, we experience that too large λ lets select too much $\hat{S}_\lambda$; For example, let $D = 10$, $|\hat{S}_{\lambda=0.5}| = 5$ and $|\hat{S}_{\lambda=1.0}| = 10$. In this case, the optimal λ_{upper} would exist in the range of $[0.5, 1.0)$; however, due to $2\lambda_{\text{previous}}$, the `heuristic-lambda-range` jumps over the optimal λ.

The second drawback is that the lower bound λ_{lower} is fixed as a hyperparameter value. We expect that λ_{lower} lets us obtain the maximum number of selected variables, i.e. a λ where $|\hat{S}_\lambda|$ becomes the maximum. The λ_{lower} given as a hyperparameter is evidently not the optimal λ_{lower}.

D.2 Data-Driven Parameter Range Selection

As we mention earlier in Algorithm 4, this algorithm has two drawbacks. To overcome these drawbacks, we propose the MMD CV Aggregation Optuna algorithm using *Optuna* instead of

the heuristic approach.

We solve this trade-off by applying *Optuna* hyperparameter tuner (Akiba et al., 2019). Optuna is an open-source hyperparameter optimisation framework designed to automate the search for optimal hyperparameters using techniques like Bayesian optimisation and efficient sampling methods. We set the number of selected variables $|\hat{S}|$ as an objective function of the Optuna tuner. The Optuna-based-algorithm seeks λ_{lower} with maximising $|\hat{S}|$ and λ_{upper} with minimising $|\hat{S}|$. The Optuna-based-algorithm conducts the search for λ_{lower} where the default parameter search space is $[10^{-6}, 0.01]$ and the search for λ_{upper} where the default parameter search space is $[0.01, 2.0]$. When $\lambda_{\text{lower}}, \lambda_{\text{upper}}$ are found, then we set the Λ by $\Lambda = \{\lambda_{\text{lower}}, \lambda_{\text{lower}} + \text{step}_\lambda, \lambda_{\text{lower}} + 2\text{step}_\lambda \ldots, \lambda_{\text{lower}} + (N_\lambda - 1), \text{step}_\lambda, \lambda_{\text{upper}}\}$, where $\text{step}_\lambda = (\lambda_{\text{upper}} - \lambda_{\text{lower}})/(N_\lambda - 1)$ and N_λ is a hyperparameter (a fixed value by default is $N_\lambda = 10$).

E Kernel Length Scale Selection

Variable-wise Median Heuristic

We describe the variable-wise median heuristic, a method for selecting the length scales $\gamma_1, \ldots, \gamma_D$ in the ARD kernel (2.3). This method extends the standard median heuristic used in the kernel literature (e.g., Garreau et al., 2018).

For each $d = 1, \ldots, D$, we set γ_d as the median of pairwise distances of the d-th variable values in the datasets. To describe the heuristic more precisely, let $\mathbf{X} = \{X_1, \ldots, X_n\} \subset \mathbb{R}^D$ and $\mathbf{Y} = \{Y_1, \ldots, Y_m\} \subset \mathbb{R}^D$. For each $i = 1, \ldots, n$, write $X_i = (x_1^i, \ldots, x_D^i)^\top \in \mathbb{R}^D$, i.e., $x_d^i \in \mathbb{R}$ is the d-th entry of the vector X_i. Similarly, for $j = 1, \ldots, m$, let $Y_j = (y_1^j, \ldots, y_D^j)^\top \in \mathbb{R}^D$ with y_d^j denoting the d-th entry of Y_j. Then, for each $d = 1, \ldots, D$, define $L_d \subset \mathbb{R}$ as the set of the d-th variable values from $\mathbf{X}$ and $\mathbf{Y}$, i.e.,

$$L_d := \{x_d^i \mid i = 1, \ldots, n\} \cup \{y_d^j \mid j = 1, \ldots, m\},$$

We then set γ_d as the median of pairwise distances between elements in L_d:

$$\gamma_d^2 := \text{median}\left\{(z - z')^2 \mid z, z' \in L_d, \quad z \neq z'\right\}. \tag{6}$$

By setting the length scale γ_d in this way, the hope is that the scaled differences

$$\{(x_d^i - y_d^j)^2/\gamma_d^2 \mid i = 1, \ldots, n,\ j = 1, \ldots, m\} \tag{7}$$

would have approximately a similar level of variability across different dimensions $d = 1, \ldots, D$.

Minimum Value Replacement If the majority of the elements in L_d take the same value (e.g., $x_d^i = y_d^j = 0$ for many of $i = 1,\ldots,n$ and $j = 1,\ldots,m$), many of the pairwise differences in (6) become zero, so their median γ_d may be zero. This causes a problem because the length scale should be positive by definition. We propose the following procedure to address this issue. First, compute the D length scales $\gamma_1,\ldots,\gamma_D$ using the variable-wise median heuristic, and let $\gamma_{\min} > 0$ be the minimum among the *positive* computed length scales. Then, if the median (6) becomes zero for the d-th variable, we set $\gamma_d = \gamma_{\min}$.

Variable-wise Mean Heuristic In some the applications such as traffic simulation data in Section 3.5.3, we found that the variable-wise median heuristic makes many length scales $\gamma_1,\ldots,\gamma_D$ zero. This is caused by the data vectors $\mathbf{X} = \{X_1,\ldots,X_n\}$ and $\mathbf{Y} = \{Y_1,\ldots,Y_m\}$ being sparse in this application, i.e., each data vector has many zero entries. Accordingly, as explained above, the zero length scales are replaced by the minimum of the positive length scales. However, we found that optimisation of the ARD weights using the resulting length scales becomes unstable. The reason is that the replaced length scale, say γ_d, may be much smaller than some of the pairwise distances $(x_d^i - y_d^j)$ in the data.

For example, suppose for simplicity that $n = m = 3$, $x_d^1 = x_d^2 = 0$, $x_d^3 = 2.0$, and $y_d^1 = y_d^2 = y_d^3 = 0$. Then the set of pairwise squared distances in (6) is $\{0,0,0,0,0,0,0,0,0,4,4,4,4,4\}$. Then the median (6) is zero, and thus γ_d is set as the minimum $\gamma_{\min}$ of the positive length scales. If this minimum is, for example, $\gamma_{\min} = 0.1$, then the ratio (7) for $i = 3$ and $j = 1$ becomes $(x_d^3 - y_d^1)^2/\gamma_{\min}^2 = 4/0.1^2 = 400$. Since this value appears in the exponent of the ARD kernel (2.3), the kernel value becomes close to 0 for the initial ARD values $a_1 = \cdots = a_D = 1$, and thus the optimisation of the ARD weighs becomes unstable.

Therefore, for such sparse data, we suggest using the variable-wise *mean* heuristic, i.e., setting γ_d by replacing the median (6) by the mean of the pairwise distances. In the above example, by taking the mean of $\{0,0,0,0,0,0,0,0,0,0,4,4,4,4,4\}$, we have $\gamma_d^2 = 1.33$, so $(x_d^3 - y_d^1)^2/\gamma_d^2 = 4/1.33 = 3.00$; hence the resulting ARD kernel would not collapse to zero and the optimisation of the ARD weights becomes stable. Therefore, we use this variable-wise mean heuristic in the traffic simulation experiment in Section 3.5.3.

F MMD Model Selection Optuna: A Variant of MMD Model Selection with Optuna

MMD Model Selection seeks the best λ among the given Λ. However, there is no guarantee that the suitable λ is in the given Λ. Instead of relying on the given Λ, we let Optuna seek the best λ, which is a hyperparameter tuner (Akiba et al., 2019). Optuna is an open-source hyperparameter optimisation framework designed to automate the search for optimal hyperparameters

using techniques like Bayesian optimisation and efficient sampling methods.

Algorithm 5 describes the automatic regularisation parameter selection for `MMD Selection Optuna`. In this algorithm, Optuna seeks the best λ by maximising both of a test power on the validation dataset and a p-value obtained with the selected variables; the test power is an approximated value that the optimised MMD estimator can correctly distinguish two probability distributions, and a higher value is better; the p-value, $0 \leq p \leq 1.0$, is a metric for qualifying the selected variables, and a lower value is better.

We briefly describe the algorithm procedure. The algorithm requires splitting the given datasets into the training and validation datasets. The iteration number of parameter searching by Optuna N_{search} is necessary, which we set $N_{\text{search}} = 20$ by default. The output of this algorithm is a pair of the selected variables $\hat{S}_{\lambda^*}$ and the best regularisation parameter λ^*. The algorithm first initialises values for λ^* and the associated Optuna objective value ℓ^*_{Optuna} at line 1. Line 4 of this algorithm computes the ARD weights a_λ by solving the MMD optimisation problem with the training dataset and the given λ. Line 5 computes the test power using the optimised ARD weights on the given validation dataset. Line 7 computes the p-value by performing a permutation two-sample test on the validation dataset using the selected variables $\hat{S}_\lambda$ that the procedure of line 5 obtains. Line 8 defines the Optuna objective function as $\ell_{\text{val}}(a_\lambda)(1-p_\lambda)$, and, at line 9, Optuna updates the parameter search space. At lines 10-12, the best objective value and λ are updated when the computed objective value is larger than the current best value.

Algorithm 5 Automatic Data-driven Regularisation Parameter Selection
▷ **Input:** N_{search}: the iteration number of parameter searching by Optuna. $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$: training data. $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$: validation data.
▷ **Output:** $\hat{S}_{\lambda^*} \subset \{1, \ldots, D\}$: selected variables with the best regularisation parameter $\lambda^* \in \Lambda$.

1: Initialise λ^* and the associated Optuna objective function $\ell^*_{\text{Optuna}} = 0.0$.
2: **while** N_{search} times **do**
3: Optuna suggests λ.
4: Obtain ARD weights $a_\lambda \in \mathbb{R}^D$ by numerically solving (3.2) using $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ and λ.
5: Compute $\ell_{\text{val}}(a_\lambda) > 0$ by evaluating (2.4) on $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$.
6: Select variables $\hat{S}_\lambda \subset \{1, \ldots, D\}$ using a_λ as in Section 3.3.1.
7: Compute a p-value $0 \leq p_\lambda \leq 1$ by performing a permutation two-sample test on $(\mathbf{X}_{\text{val}}[:, \hat{S}_\lambda], \mathbf{Y}_{\text{val}}[:, \hat{S}_\lambda])$.
8: Compute the Optuna objective function $\ell_{\text{Optuna}} = \ell_{\text{val}}(a_\lambda)(1-p_\lambda)$.
9: Optuna reports the objective function value to the Optuna tuner.
10: **if** $\ell_{\text{Optuna}} > \ell^*_{\text{Optuna}}$ **then**
11: Update $\ell^*_{\text{Optuna}} = \ell_{\text{Optuna}}$ and $\lambda^* = \lambda$.
12: **end if**
13: **end while**

G Component of Variable Selection Algorithms

G.1 Early Stopping of MMD optimisation

MMD-based variable selection methods are based on the optimisation of the MMD estimator. The optimal epoch size of the optimisation is not known in advance, therefore, we set an abundant number of epochs and apply early stopping when a defined criterion activates. There are two early stoppers in our implementation: `convergence based stopper` and `variable selection based stopper`. The optimisation stops when either of the criteria activates.

The `convergence based stopper` monitors the loss value of Eq. 3.2 and halts the optimisation when the loss value keeps converging during a certain epoch, by default 100 epochs. The `variable selection based stopper` monitors the selected variables $\hat{S}$ and halts the optimisation when the selected variables are the same during a certain epoch, by default 100 epochs. The latter early stopper is necessary, from our experience, since the loss value often keeps decreasing, while some ARD weights clearly indicate high weight values and others keep almost zero. In this case, certain ARD weights keep increasing and therefore the loss values does not converge. To prevent this unnecessary optimisation, we apply the `variable selection based stopper`.

The configurations of these early stoppers are as follows: The `convergence based stopper` starts monitoring the loss value after 200 epochs. The stopping criteria activates $min(v)/max(v) < 0.001$ among 100 epochs window-span, where v the loss value, Eq. 3.2. The `variable selection based` stopper starts monitoring the selected variables after 400 epochs. After the 400 epochs, the stopper regularly executes the variable selection of $\hat{S}$ per 10 epochs.

G.2 MMD Optimisation when the Null Hypothesis H_0 is True

The truth of the two-sample testing problem is unknown when we apply the MMD-based variable selection methods. When the null hypothesis H_0 is true, there are two cases in the MMD optimisation; 1. the optimisation converges to a certain solution and we obtain $\hat{S}$ from the optimised ARD weights, 2. the objective value is impossible to compute because $\ell(a_1, ..., a_D)$ becomes a minus value.

The second case occurs because an MMD estimate value becomes a minus value. In this case, the optimisation is impossible to advance. Nevertheless, there may be a possibility that the truth is a rejection of H_0 even though given two samples are quite similar. The MMD estimate becomes a minus value because the initial ARD weights may be inappropriate. Therefore, we continue the optimisation problem without log operation in Eq. 3.2 when $\ell(a_1, ..., a_D) < 0.0$.

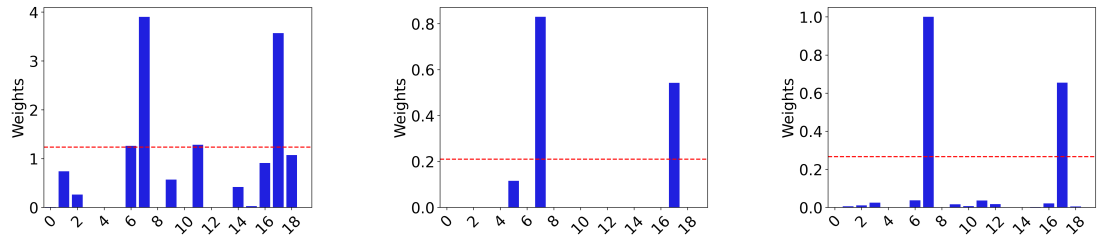
Once $\ell(a_1, ..., a_D) > 0.0$, we apply the log operation to the loss value. We let the optimisation continue even if $\ell(a_1, ..., a_D) < 0.0$ during 3000 epochs. If $\ell(a_1, ..., a_D) < 0.0$ always during 3000 epochs, we stop the optimisation and accept H_0. Since the optimisation is not done, there is no selected variables $\hat{S}$ in this case. We set the p-value as 1.0 in this case.

H Variable Selection with a Given Hyperparameter Threshold

In the context of variable selection, it is possible to introduce a hyperparameter τ for the optimised weights $a \in \mathbb{R}^D$, that controls the number of selected variables according to specific objectives,

$$\hat{S} := \{d \in \{1, \ldots, D\} \mid a_d^* > \tau\}.$$

This hyperparameter functions as a threshold for the weights, derived from either the optimized Automatic Relevance Determination (ARD) weights or their aggregated weights. By applying this threshold to the computed weights, one can effectively select variables for which the ARD weights exceed the predetermined threshold value.



(a) (Sutherland et al., 2017).

(b) MMD Model Selection (Algorithm 1).

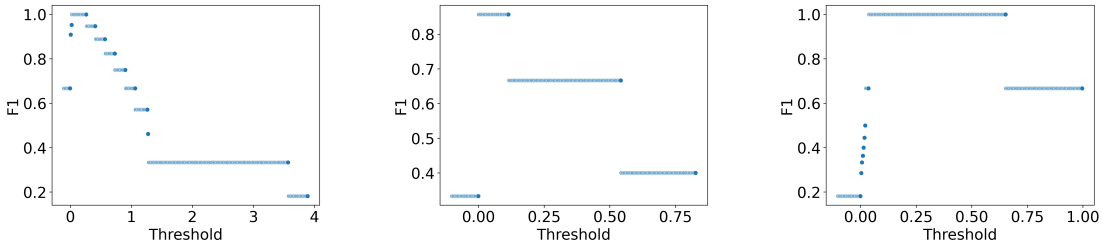
(c) MMD CV Aggregation (Algorithm 2).

Figure 1: The optimised ARD weights (Sutherland et al., 2017) (Left), optimised ARD weights by MMD Model Selection (Algorithm 1) (Middle), and aggregated weights by MMD CV Aggregation (Algorithm 2) (Right). The dataset is "Narrower Variance" described in Section 3.5.2. The red dashed lines are the given threshold τ.

Figure 1 illustrates examples of variable selection based on a thresholding approach for the dataset generated with the configuration of "Narrower Variance" in Section 3.5.2. The red line signifies a predetermined threshold. The three plots present, from the left, the ARD weights optimised by the baseline established in the work of (Sutherland et al., 2017), the ARD weights associated with the selected λ by the MMD Model Selection method (Algorithm 1), and the aggregated variable weights by MMD CV Aggregation (Algorithm 2). It is noteworthy that the ground-truth variables are located at the 7th and 17th indices.

H.1 Assessments of Variable Selection with a Given Hyperparameter Threshold

We conduct empirical and qualitative evaluations for variable selection utilising a threshold approach. For the weight distributions illustrated in Figure 1, we establish 100 threshold values and compute the corresponding F1 scores described in Section 3.5.2. Figure 2 presents the relationships between the threshold values and F1 scores for three distinct weight distributions.



(a) (Sutherland et al., 2017)

(b) MMD Model Selection (Algorithm 1).

(c) MMD CV Aggregation (Algorithm 2).

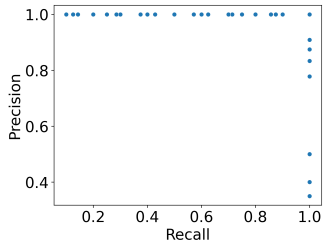
Figure 2: The relationship between the given threshold τ and F1 score. We conduct assessments with 100 of τ parameters for the optimised and aggregated weights in Figure 1.

The variable selection process exhibits stability with respect to the hyperparameter, as indicated by the alignment of points along a specific F1 value on the vertical axis. The scatter plot generated by the MMD CV Aggregation method (Right) reveals two distinct lines at F1 = 1.0 and F1 = 0.667. In contrast, the baseline plot (Left) demonstrates a more unstable relationship between the threshold and F1 scores when compared to the MMD CV Aggregation plot.

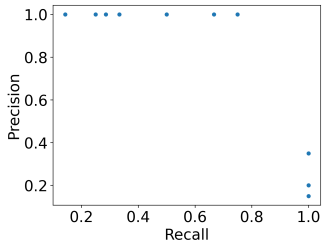
The assessment examples provided above illustrate the advantages of the MMD CV Aggregation approach when employing user-defined threshold selections.

H.2 Assessments of Variable Selection by Precision-Recall Curve

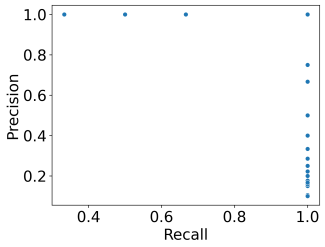
The Precision-Recall curve is commonly utilized for the assessment of machine learning models that incorporate a hyperparameter. However, it may not be entirely appropriate for our variable selection scenarios. Figure 3 illustrates the Precision-Recall curves corresponding to three distinct variable selection methodologies, as depicted in Figure 1. Despite evaluations conducted across 100 threshold values, the scatter plots exhibit a significant degree of duplication among the data points, resulting in a sparse representation.



(a) Baseline without regularisation (Sutherland et al., 2017).



(b) MMD Model Selection (Algorithm 1).



(c) MMD CV Aggregation (Algorithm 2).

Figure 3: Figure 1. The relationship between the precision and recall scores. We conduct assessments with 100 of τ parameters for the optimised and aggregated weights in Figure 1.

H.3 Discussion: Interpretable Information by Optimised or Aggregated Weights

While our research primarily focuses on variable selection, it may also be possible to derive weight distributions and facilitate visual interpretation. This allows for the analysis of the magnitudes and degrees of differences between two probability distributions.

In the current thesis, we have not assessed the quality of the visual representation of the extracted weights, as establishing the relationships between weight values and the magnitudes of contribution for differences between two probability distributions remains challenging. For instance, in Figure 1, we observe that two variables exhibit high weight values, with the seventh variable surpassing the weight of the seventeenth variable. However, at this juncture, we have not sufficiently elucidated the relationships between the weight values of these two variables. Although we designated two ground-truth variables as equivalent, it is important to note that these variables may possess differing contribution magnitudes within the specified datasets.

I Appendix: Synthetic Data Assessment (Appendix to Section 5.3)

I.1 Experiments with with Higher Noise Ratio $\rho = 0.8$

We perform the same experiments as in Section 3.5.2, with the rate ρ of ground-truth variables S being modified to 0.8, so that $|S| = 20 \times 0.8 = 16$. Figure 4 shows the results. Compared with the results for $\rho = 0.1$ in Section 3.5.2, all the approaches yield reasonably high scores across different settings of distributions P and Q. The relative easiness of the setting $\rho = 0.8$ can be attributed to the high number of ground-truth variables 16. For example, if one selects all the variables, i.e., $\hat{S} = \{1, \ldots, D\}$, then the Recall is 1 and the Precision is 0.8, so the F score is 0.89. However, in practice, one cannot take such a strategy because the value of ρ is typically

unknown. Given that we do not use the information of true ρ for variable selection, the obtained scores can be regarded as reasonably high. Among the approaches considered, the proposed methods, `model-selection` and `CV-aggregation`, yield stably high scores across the different distribution settings.

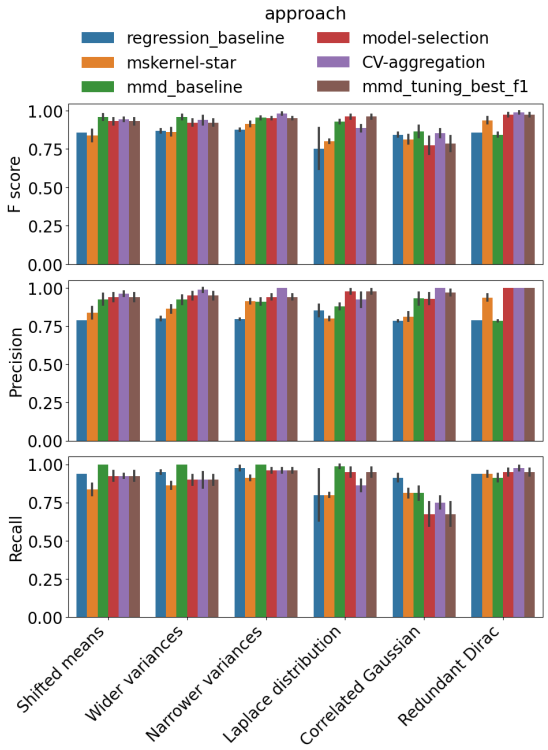


Figure 4: The results of the same experiments as Section 3.5.2 with $\rho = 0.8$, discussed in Section I.1.

I.2 Variable Detection with Linear MMD Estimator

A linear unbiased MMD estimator, proposed by Gretton et al. (Gretton et al., 2012a), is a computationally efficient alternative to the quadratic unbiased MMD estimator Eq. (2.2). The quadratic estimator's computational complexity is $O(n^2)$ for sample size $n = m$, which can be costly for large n. The linear MMD estimator's complexity is $O(n)$, thus offering a reduced computational cost, while its variance is larger than the quadratic estimator. Here, we compare the linear and quadratic estimators when used in the proposed methods for variable selection.

Given $\mathbf{X} = \{X_1, \ldots, X_n\} \subset \mathbb{R}^D$ and $\mathbf{Y} = \{Y_1, \ldots, Y_n\} \subset \mathbb{R}^D$ with sample size n being odd, the linear

MMD estimator is defined as

$$\mathrm{MMD}_l^2(\mathbf{X},\mathbf{Y}) := \frac{2}{n}\sum_{i=1}^{n/2} h((X_{2i-1},Y_{2i-1}),(X_{2i},Y_{2i}))$$

where $h((X,Y),(X',Y')) := k(X,X')+k(Y,Y')-k(X,Y')-k(X',Y)$. Assuming that n is divisible by 4, an unbiased estimator of its variance, which corresponds to Eq. (2.5) for the quadratic estimator, is given by (Gretton et al., 2012b, Eq. (7)):

$$\breve{\sigma}^2 := \frac{4}{n}\sum_{\ell=1}^{n/4} [h((X_{4\ell-3},Y_{4\ell-2}),(X_{4\ell-3},Y_{4\ell-2})) \\ -h((X_{4\ell-1},Y_{4\ell}),(X_{4\ell-1},Y_{4\ell}))]^2$$

Then, the ratio objective corresponding to Eq. (2.4) is defined as

$$\frac{\mathrm{MMD}_l^2(\mathbf{X},\mathbf{Y})}{\sqrt{\breve{\sigma}^2+C}}$$

Using this ratio objective in the regularised objective in Eq. (3.2), one can use the linear estimator in the proposed methods.

We perform the same experiments as Section 3.5.2 and make a comparison with the proposed methods using the linear estimator. Figure 5 shows the results of the F score, Precision and Recall. For all the settings of distributions P and Q, the evaluation scores for the linear estimator are significantly lower than the quadratic estimator. Specifically, in the "Wider variances" setting, the scores for the linear estimator are consistently low, approaching 0.0. For `mmd-tuning-best-f1`, the linear estimator consistently performs worse than the quadratic estimator. Since `mmd-tuning-best-f1` provides the highest possible F score for `model-selection`, the results indicate that the linear estimator is not suitable for accurate variable selection, when the sample size n is not too large so that the quadratic estimator is applicable.

Variable Detection with Linear MMD Estimator This section describes the results of the experiments with the linear MMD estimator. The corresponding results of $\rho = 0.1$ is found in Section Section I.2.

Figure 6 plots the comparisons of the linear MMD estimator with the quadratic MMD estimator for the variable detection task. Similarly as we describe in Section I.2, the significant drop in the evaluation scores occurs. Considering that the setting of $\rho = 0.8$ is easier problem than the one of $\rho = 0.1$, we could conclude that the quadratic MMD estimator is more suitable for the variable detection task.

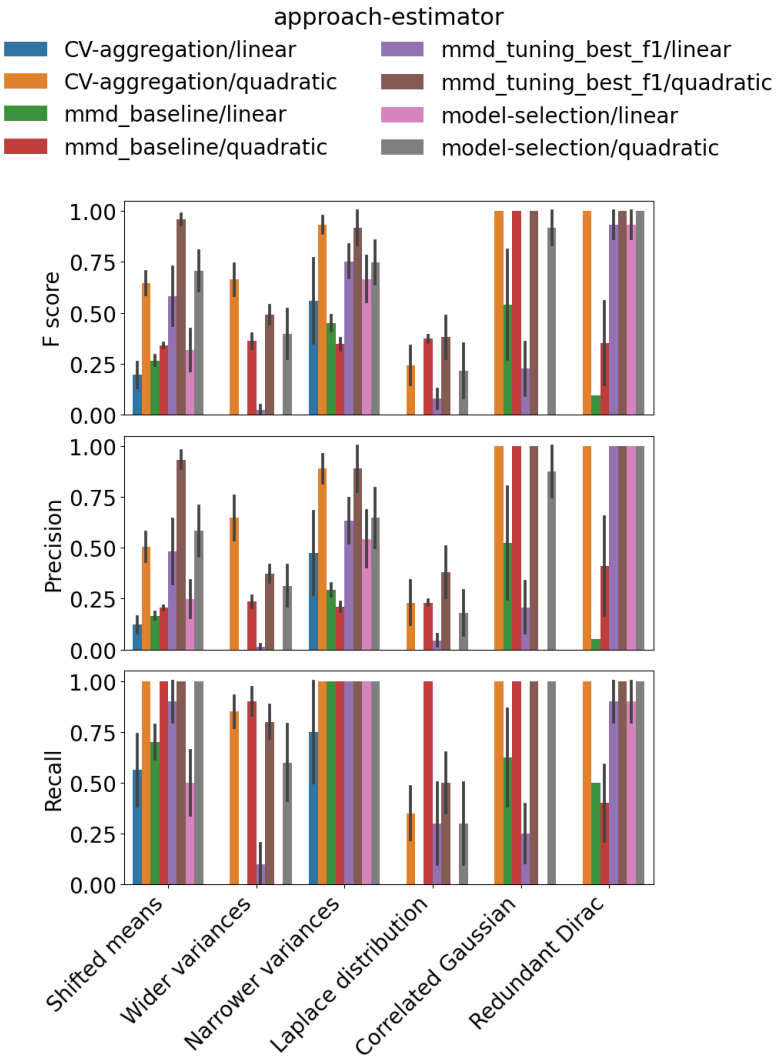


Figure 5: Comparison of the linear and quadratic MMD estimators when used in the proposed approaches (`CV-aggregation` and `model-selection`) and other baselines (`mmd-baseline` and `mmd-tuning-best-f1`). The left and right figures are the results for $\rho = 0.1$. The top, middle and bottom rows show the F score, Precision and Recall for each method and each setting of distributions P and Q, with the means and standard deviations obtained from 10 independently repeated experiments. Note that the linear estimator often leads to the complete failure of variable detection, in which case no bar is shown.

I.3 Assessing the Effects of Dimensionality

We assess the effects of the dimensionality D, i.e., the total number of variables, on the variable selection performance. We consider the same setting as Section 3.5.2, with P and Q being D-dimensional distributions in the following way.

Let $P = \mathcal{N}(\mathbf{0}_D, \Sigma)$, where the covariance matrix $\Sigma \in \mathbb{R}^{D \times D}$ is diagonal with D diagonal elements

being

$$(\underbrace{1,1,\ldots,1}_{10},\underbrace{2,2,\ldots,2}_{10},\ldots,\underbrace{D/10,D/10,\ldots,D/10}_{10})$$

We define the other distribution Q by defining how its random vector $Y \sim Q$ is generated. Let $S \subset \{1,2,\ldots,D\}$ be ground-truth variables with $\rho = 0.1$, so that $|S| = D \times 0.1$. Let $\tilde{X}_S \sim \mathcal{N}(\mathbf{0}_{|S|}, \Sigma_S)$, where $\Sigma_S \in \mathbb{R}^{|S|\times|S|}$ is the submatrix of Σ restricted to the variables S. Let $Z_S \sim \mathcal{N}(\mathbf{0}_{|S|}, I_{|S|})$. Then we define $Y_S = \tilde{X}_S \odot Z_S$, where $\odot$ denotes the element-wise product. Let $Y_{\{1,\ldots,D\}\setminus S} \sim \mathcal{N}(0_{D-|S|}, \Sigma_{\{1,\ldots,D\}\setminus S})$. Finally, define $Y = (Y_S, Y_{\{1,\ldots,D\}\setminus S}) \in \mathbb{R}^D$, and Q is defined as the distribution of Y.

We consider various values for the dimensionality: $D = 20, 50, 100, 200, 300, 400, 500$. We fix the sample size $N = 200$. For each value of D and each method, we experiment as in Section 3.5.2 and repeat it 10 times to derive evaluation scores' means and standard deviations.

Figure 7 shows the results. As a general trend, all the approaches, except the regression baseline, experience a drop in their F scores as the dimensionality D increases. This drop can be attributed to the variable selection task becoming harder for higher D. The decrease of the F score appears to be milder for `CV-aggregation` than the other methods, and the `CV-aggregation` attains similar F scores for the largest D as `mmd-tuning-best-f1`, which is not implementable in practice and provides the best possible performance of `model-selection`.

When comparing `mmd-baseline` with the proposed methods (`CV-aggregation` and `model-selection`, we observe that `mmd-baseline` experiences a drastic drop in the F score as the dimensionality D increases, eventually resulting in the zero F score for $D = 500$. In contrast, the proposed methods yield reasonable F scores even for $D = 500$. This difference can be attributed to the regularisation approach used in the proposed methods, which proves to be more effective for higher dimensionality D.

I.4 Alternative MMD-based Objective

We compare the proposed objective function Eq. (3.2) for optimising the ARD weights with an alternative objective function where the ratio objective Eq. (2.4) is replaced by an unbiased MMD estimate $\widehat{\mathrm{MMD}}^2_U(\mathbf{X},\mathbf{Y})$ in Eq. (2.2), i.e.,

$$\min_{a\in\mathbb{R}^D} -\log \widehat{\mathrm{MMD}}^2_U(\mathbf{X},\mathbf{Y}) + \lambda \sum_{d=1}^{D} |a_d| \tag{8}$$

Historically, before Gretton et al. (Gretton et al., 2012b) proposed the ratio objective as an objective function for hyper-parameter tuning in the MMD, Sriperumbudur et al. (2009) discussed that one could use the MMD itself as an objective function for hyper-parameter tuning of MMD. The latter approach optimises kernel parameters to maximise the MMD. It

corresponds to removing the denominator in the ratio objective Eq. (2.4), thus resulting in Eq. (8). Gretton et al. (Gretton et al., 2012b) showed that maximising the ratio objective yields higher test power than the MMD maximisation. We will investigate whether this property translates to the variable selection task, i.e., whether the ratio-based objective Eq. (3.2) leads to better variable selection performance than the MMD-based objective Eq. (8).

We perform the same experiments as in Section 3.5.2 for both $\rho = 0.1$ and $\rho = 0.8$ (recall this constant ρ represents the ratio of the number of ground-truth variables over the total number of variables). Here, we compare the performance of Algorithm 1 (`model-selection`), `mmd-baseline` and `mmd-tuning-best-f1`, which use the ratio-based objective Eq. (3.2), and their MMD-based objective versions using Eq. (8).

Figure 8 describes the Precision, Recall and F scores for each setting and each method, obtained from 10 independently repeated experiments. The MMD-objective versions tend to yield lower values of the evaluation criteria than the ratio-objective versions. In particular, the former yields significantly lower Recall than the latter for $\rho = 0.8$. While the MMD-based objective yields better Precision for some settings, this improvement in Precision is enabled by sacrificing the Recall. Figure 10 shows the number of selected variables for each method and each setting. It is evident that the MMD objective results in much fewer variables than the ground-truth variables for $\rho = 0.8$. Based on these observations, we conclude that the ratio objective is more appropriate than the MMD objective for variable selection in two-sample testing.

I.5 Comparing Different CV Aggregation Strategies

In the process of developing Algorithm 2 (`CV-aggregation`), there were several candidate ways of computing the aggregation score vector $\hat{\Pi}_\lambda = (\hat{\Pi}_{\lambda,1}, \ldots, \hat{\Pi}_{\lambda,D})^\top \in \mathbb{R}^D$ for each candidate regularisation parameter λ. Here, we compare these different choices and how we have arrived at our choice for Algorithm 2. We use the notation for Algorithm 2 in Section 3.4.2 in the following.

First, there are the following five ways of defining $\hat{\Pi}_\lambda$ based on the selected variables $S^i_\lambda \subset \{1, \ldots, D\}$, where $i = 1, \ldots, K$:

1. "`plane-variable`": $\hat{\Pi}_{\lambda,d} = \frac{1}{K} \sum_{i=1}^K I\{d \in \hat{S}^i_\lambda\}$ for $d = 1, \ldots, D$.
2. "`p-value-variable`": $\hat{\Pi}_{\lambda,d} = \frac{1}{K} \sum_{i=1}^K (1 - p^i_\lambda)\ I\{d \in \hat{S}^i_\lambda\}$ for $d = 1, \ldots, D$.
3. "`test-power-variable`": $\hat{\Pi}_{\lambda,d} = \frac{1}{K} \sum_{i=1}^K \ell_{\text{val}}(a^i_\lambda)\ I\{d \in \hat{S}^i_\lambda\}$ for $d = 1, \ldots, D$.
4. "`p-value-filter-variable`": $\hat{\Pi}_{\lambda,d} = \frac{1}{K} \sum_{i=1}^K I(p^i_\lambda < 0.05) I\{d \in \hat{S}^\lambda_i\}$ for $d = 1, \ldots, D$.

5. "`p-value-filter-test-power-variable`": $\hat{\Pi}_{\lambda,d} = \frac{1}{K}\sum_{i=1}^{K} I(p_\lambda^i < 0.05)\ell_{\text{val}}(a_\lambda^i) I\{d \in \hat{S}_i^\lambda\}$ for $d = 1, \ldots, D$.

We also have the following five ways based on the normalised ARD weights $\tilde{a}_\lambda^i$, where $i = 1, \ldots, K$:

1. "`plane-ard`": $\hat{\Pi}_\lambda = \frac{1}{K}\sum_{i=1}^{K} \tilde{a}_\lambda^i$.
2. "`p-value-ard`": $\hat{\Pi}_\lambda = \frac{1}{K}\sum_{i=1}^{K} (1 - p_\lambda^i)\tilde{a}_\lambda^i$.
3. "`test-power-ard`": $\hat{\Pi}_\lambda = \frac{1}{K}\sum_{i=1}^{K} \ell_{\text{val}}(a_\lambda^i)\tilde{a}_\lambda^i$.
4. "`p-value-filter-ard`": $\hat{\Pi}_\lambda := \frac{1}{K}\sum_{i=1}^{K} I(p_\lambda^i < 0.05)\tilde{a}_\lambda^i$.
5. "`p-value-filter-test-power-ard`": $\hat{\Pi}_\lambda = \frac{1}{K}\sum_{i=1}^{K} I(p_\lambda^i < 0.05)\ell_{\text{val}}(a_\lambda^i)\tilde{a}_\lambda^i$.

Note that the last approach is the one used in Algorithm 2.

Figure 11 describes the results of these candidate ways for computing the score vector $\hat{\Pi}_\lambda$ in Algorithm 2 and the other baseline methods. It shows F scores obtained in the same way as in Section 3.5.2, for two settings of the rate of the ground-truth variables S, $\rho = 0.1$ and $\rho = 0.8$.

One can observe that "`p-value-filter-test-power-ard`" exhibits the highest stability among the ten candidate methods for Algorithm 2. Consequently, we have decided to adopt this way for Algorithm 2.

J Data Generation Process for Section 3.5.3

We define a grid-like road network consisting of eight intersections, as described in Figure 12; we call it *grid network*. We define two scenarios for traffic simulation on the grid network: scenario P and scenario Q, the latter being a perturbed version of the former. In scenario P, there are two groups of vehicles, one moving from the top-left corner to the bottom-right corner, and the other moving in the opposite direction. Scenario Q is defined via a perturbation to scenario P by blocking two specific lanes visualised in Figure 12.

Each intersection in the grid network has 8 sensors, as shown in Figure 12c, resulting in 64 sensors in total in the network. Each sensor emulates an induction loops detector and measures the flow at the sensor, i.e., the number of vehicles passing in front of the sensor during one time step, representing one second in the simulation world. Each scenario consists of 3,600 time steps, corresponding to one hour, during which 200 vehicles are simulated.

We implement the grid network and the two scenarios using *SUMO*, a widely used traffic simulator.[2] For each simulation, SUMO generates an XML file containing the sensors' measurements, from which we construct a $D_{\text{sensor}} \times D_{\text{time}}$ matrix. Here, $D_{\text{sensor}} = 64$ is the number of sensors, and $D_{\text{time}} = 12$ is the number of time periods, each being 5 minutes. Specifically, for each $s = 1, \ldots, D_{\text{sensor}}$ and $t = 1, \ldots, D_{\text{time}}$, the (s, t)-th element of the matrix is defined as the average flow at the s-th sensor during the t-th period, calculated by taking the average of the flows of the 300 time steps (= 5 minutes) of the t-th period.

Since the majority of the parameters of the SUMO are controlled by the value of a random seed, an excessive seed value could produce an abnormal simulation result. To avoid such an abnormal simulation, we control the random seed by generating it as $R + \varepsilon$, where R is uniformly sampled from $\{0, 1, \ldots, 9, 10\}$ and ε is a Gaussian random variable with mean 1 and variance 5.

K Particle Simulation

K.1 Setups

We trained the DMCF model (Prantl et al., 2022) having convolutions layers[3]. The DNN model's training configuration is from the author on Github repository[4]. The training data is from a ready-to-use dataset of the Waterramp scenario (Sanchez-Gonzalez et al., 2020) that a particle simulator *SPlish Splash* generates. The DNN model recursively predicts the next time step's particle blobs, i.e. the DNN model's input is all particles geo-coordinate at $t-1$, and the prediction output is the particles geo-coordinate at t. Because of this recursive prediction, prediction errors are accumulated over time and particle geo coordinate would become more and more dissimilar to the Splish-Splash simulation if the DNN model's prediction is not accurate. Figure 2 represents an example of the particle blobs from the Splish-Splash simulation and the DNN model at different time steps. The Splish-Splash output and the DNN model prediction apparently look dissimilar at the time step of $t = 285$.

K.2 Naive Approach: a naive L1 distance v.s. variable selection methods

An L1 distance (Absolute difference) is a naive but intuitive method for observing discrepancies. We compute the average L1 distance per observation sensor $d \in \{1, ..., D\}$ at each time bucket $b \in \{1, ...B\}$,

[2] We use version 1.4.0 of SUMO. https://www.eclipse.org/sumo/

[3] The codebase for the training the DNN model is at https://github.com/tum-pbs/DMCF/blob/main/models/cconv.py

[4] https://github.com/tum-pbs/DMCF/blob/96eb7fcdd5f5e3bdda5d02a7f97dfff86a036cfd/configs/WaterRamps.yml

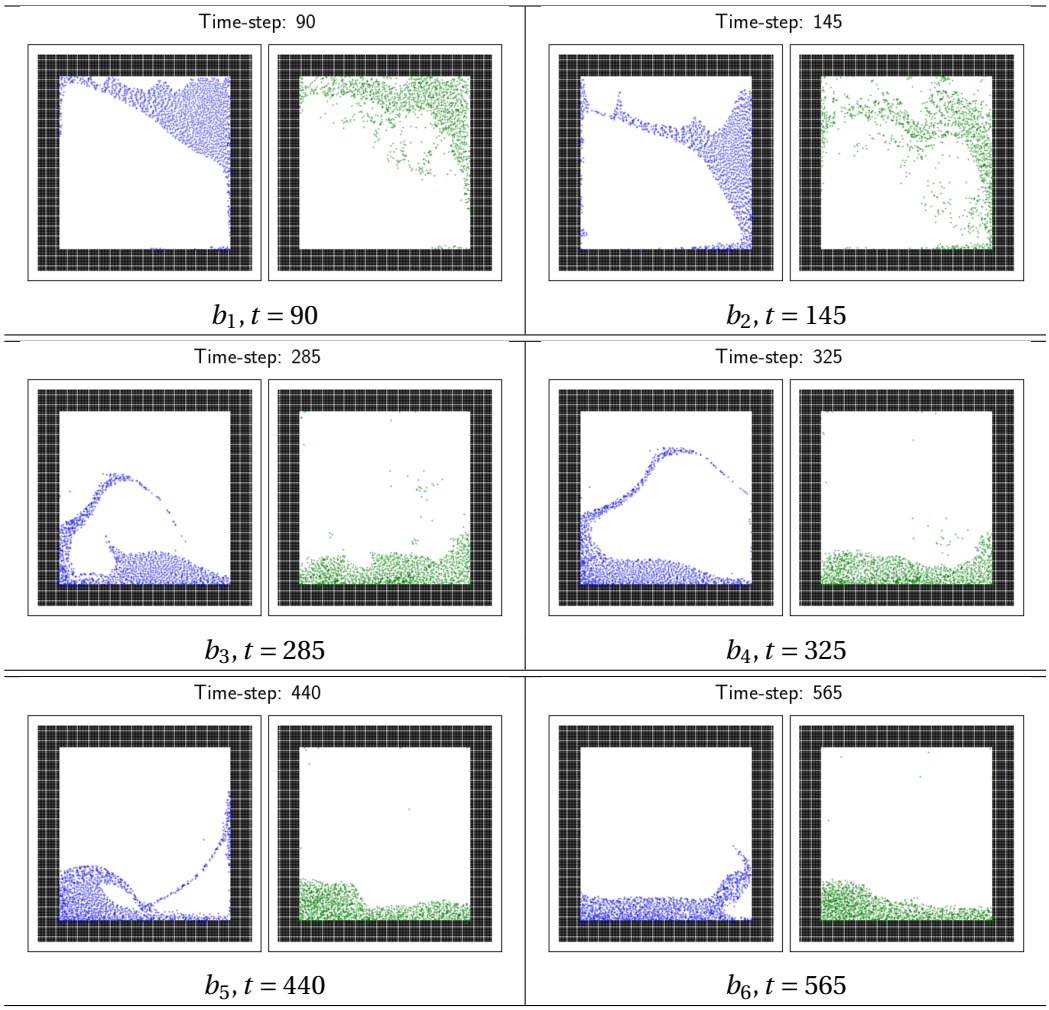


Table 2: Particle blobs from a Splish-Splash simulation and the DNN model. blue particle blobs are from the Splish-Splash simulation, and green particle blobs are from the DNN model.

$$Avg\ L1_b^d = \frac{1}{|I_b|} \sum_{i=t_{b-1}+1}^{t_b} |x_i^d - y_i^d|, \tag{9}$$

where x_i^d and y_i^d are the counts of particles at a sensor d from the Splish-Splash simulation and the DNN model, respectively, and $|I_b|$ is the number of observations at the time bucket b. Figure 14 depicts comparisons of heatmaps by the average L1 distance and the variable selection methods. The heatmap by the averaged L1 distance, Figure 14a, shows that there are significant differences at buckets b_2 and b_3, which matches with our observation with the snapshots in Figure 2. However, a drawback of the averaged L1 distance is that there are too many sensors representing discrepancies, and it would be time-consuming analysis work to confirm such sensors. Another drawback would be that the averaged L1 distance

naturally focuses on sensors showing huge discrepancies; thus there is a possibility of missing discrepancies occurring by outliers which will be crucial information for the analysis of the DNN model's output.

Figures (14b - 14d) are heatmaps by the variable selection algorithms. Compared with the heatmap by the averaged L1 distance, the variable selection methods are able to dramatically reduce the number of variables (sensors). Therefore, analysts can focus on the selected sensors and investigate the discrepancies. Regarding discovering discrepancies that occurred by outliers, the MMD-based variable selection method successfully discovers such sensors, as we discuss Section 4.5.2. Table 3 is a comparison of the selected variables in b_2, b_4. As we point out in Section 4.5.2, the DNN model prediction has unnatural water particles that seem to stick the ceil of the box at buckets b_2 until b_6. `MMD Selection` and `MMD CV` successfully discover such sensors, and the `Wasserstein-based` method does not.

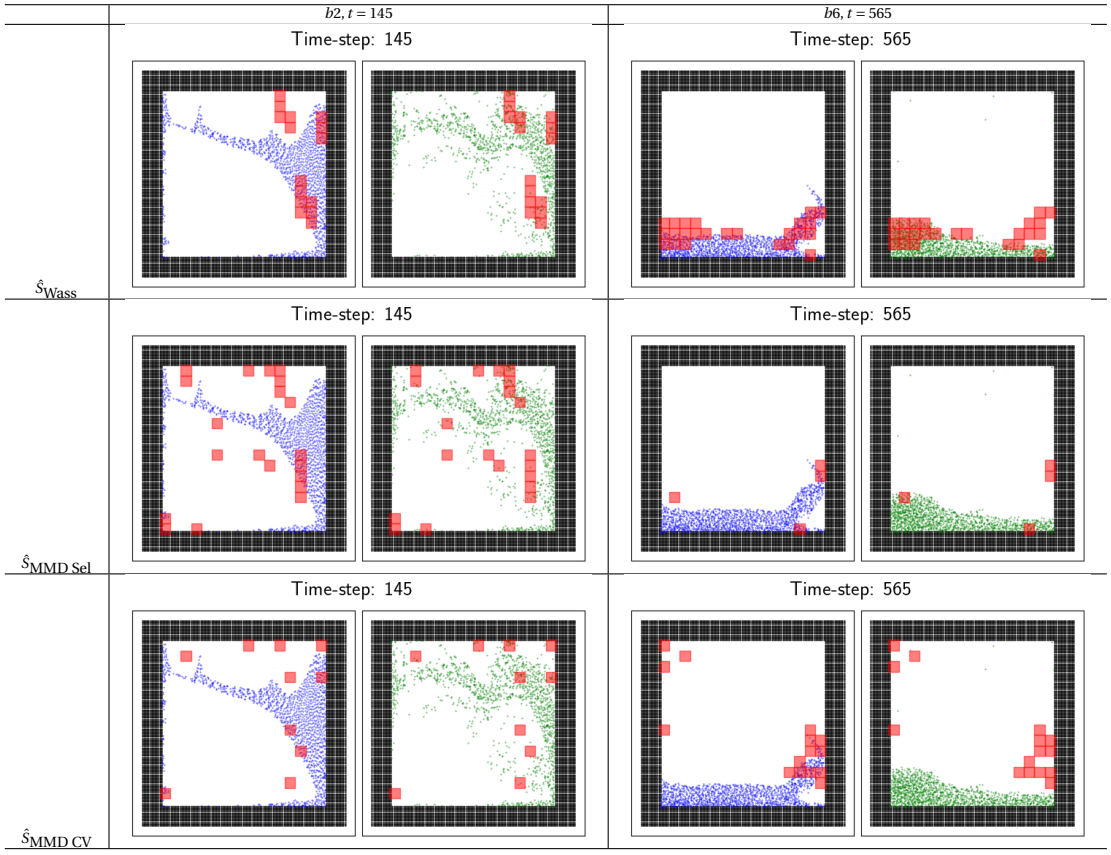
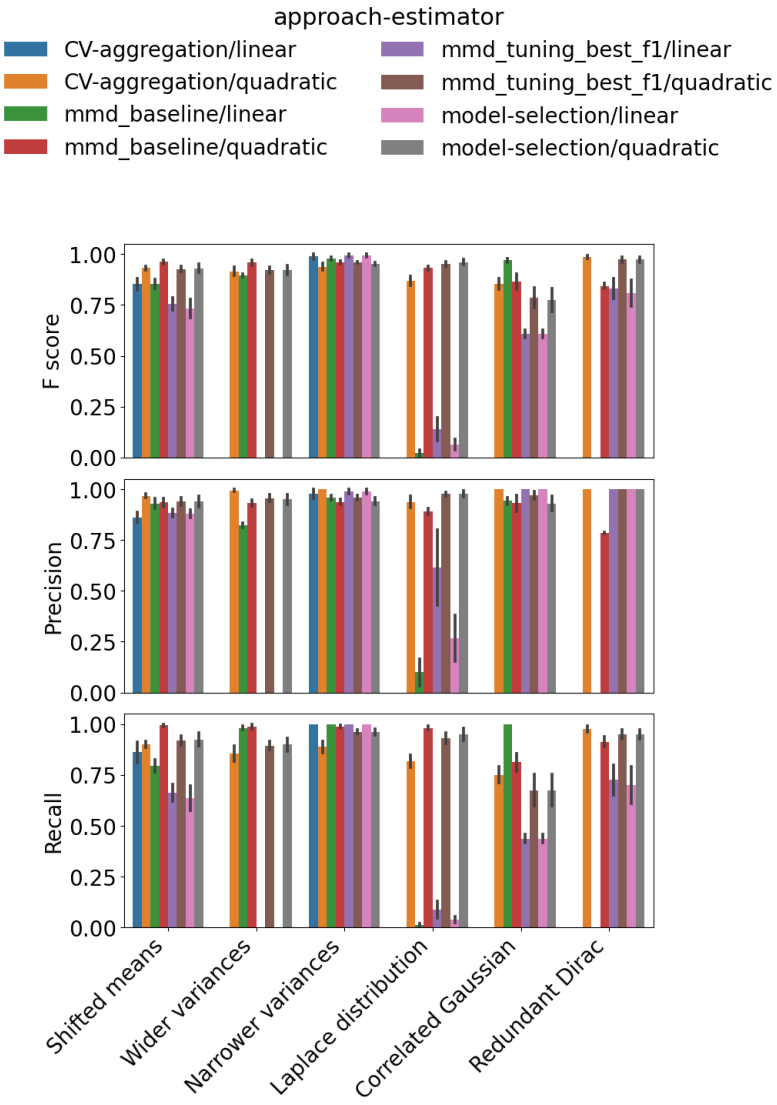


Table 3: Comparison of three variable selection methods. In each table element, the blue particles (left) are from the Splish-Splash simulator, and the green particles (right) are from the DMCF model. The red highlighting boxes are sensors selected using the variable selection method. We select and represent representative time steps at each bucket b. We do not show the red highlighting box when p-value > 0.05. More descriptions are in Section 4.5.2.



(a) $\rho = 0.8$

Figure 6: The comparison of the linear and quadratic MMD estimators for the variable detection task with $\rho = 0.8$. The corresponding results of $\rho = 0.1$ is found in Section I.2 and Figure 5.

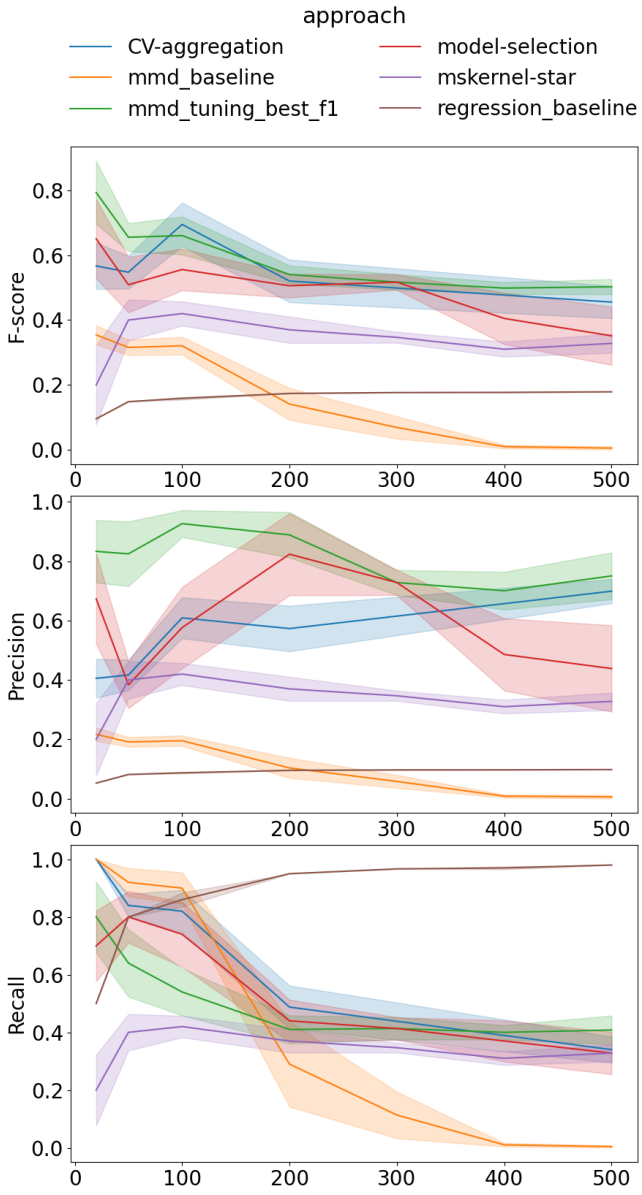


Figure 7: The results of the experiments on assessing the effects of the dimensionality D in Appendix I.3. The top, middle and bottom figures show the F score, Precision and Recall for each method and each D, with the mean and standard deviation computed from 10 independently repeated experiments. The horizontal axis indicates the dimensionality D.

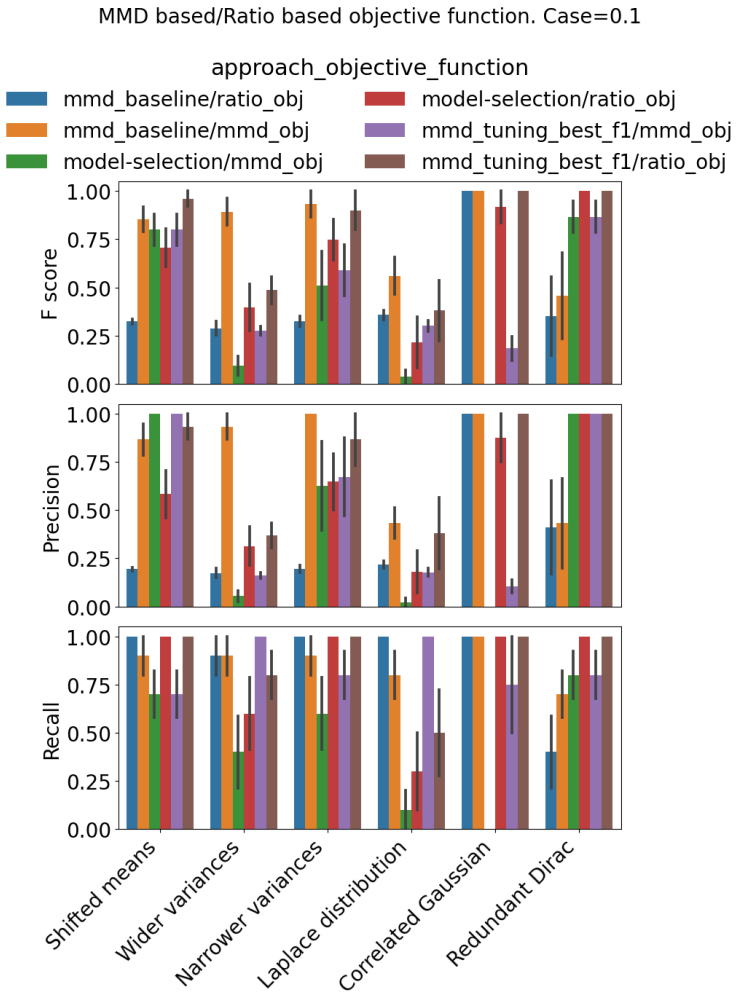


Figure 8: Results of comparative experiments between the ratio-based objective (`ratio_obj`) and the MMD objective-based objective (`mmd_obj`) in Appendix I.4. Note that the evaluation scores of `model_selection/mmd_obj` for "Correlated Gaussian" with $\rho = 0.1$ are all zero, hence no bars. The corresponding results of $\rho = 0.8$ is found in Figure 9.

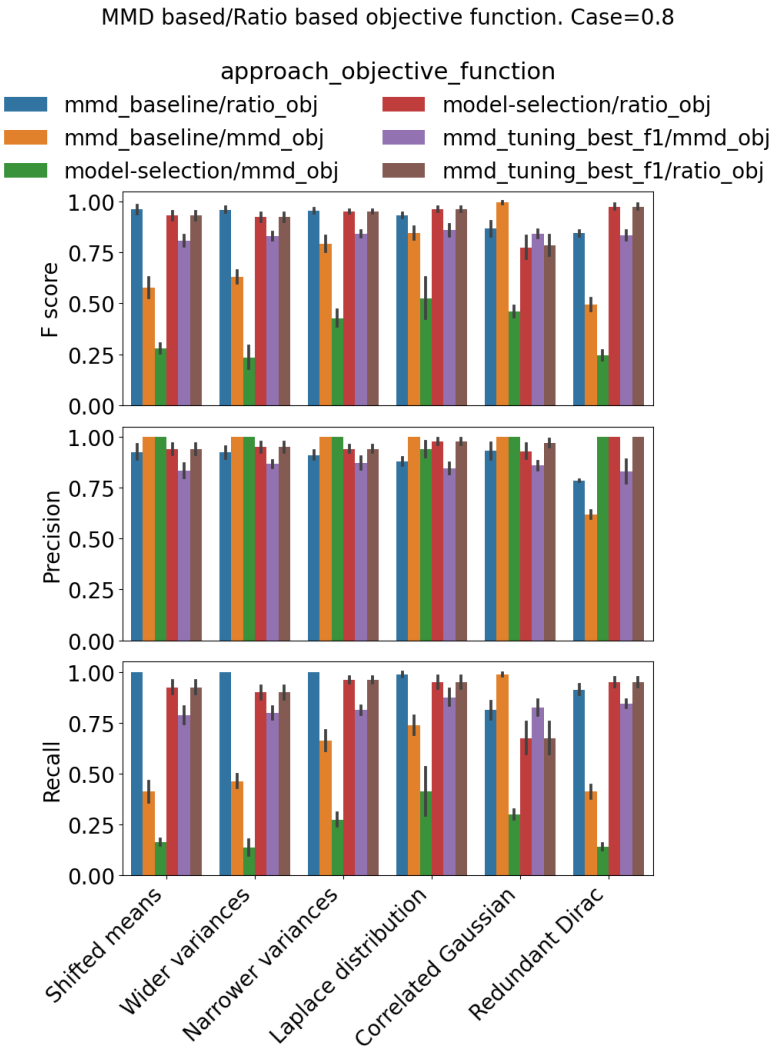


Figure 9: The description is found in Section 8.

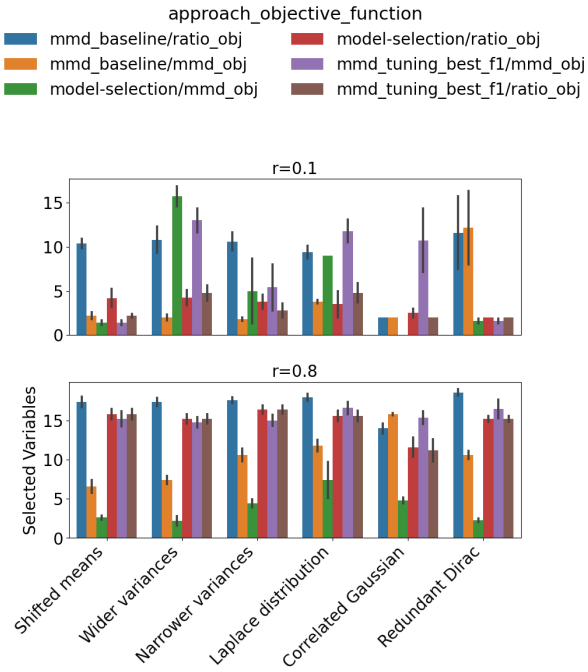


Figure 10: The number of selected variables in the comparative experiments between the ratio-based and MMD-based objective functions in Appendix I.4. The top figure shows the results with $\rho = 0.1$, where the number of ground-truth variables is $20 \times 0.1 = 2$. The bottom figure is for $\rho = 0.8$, where the true number is $20 \times 0.8 = 16$.

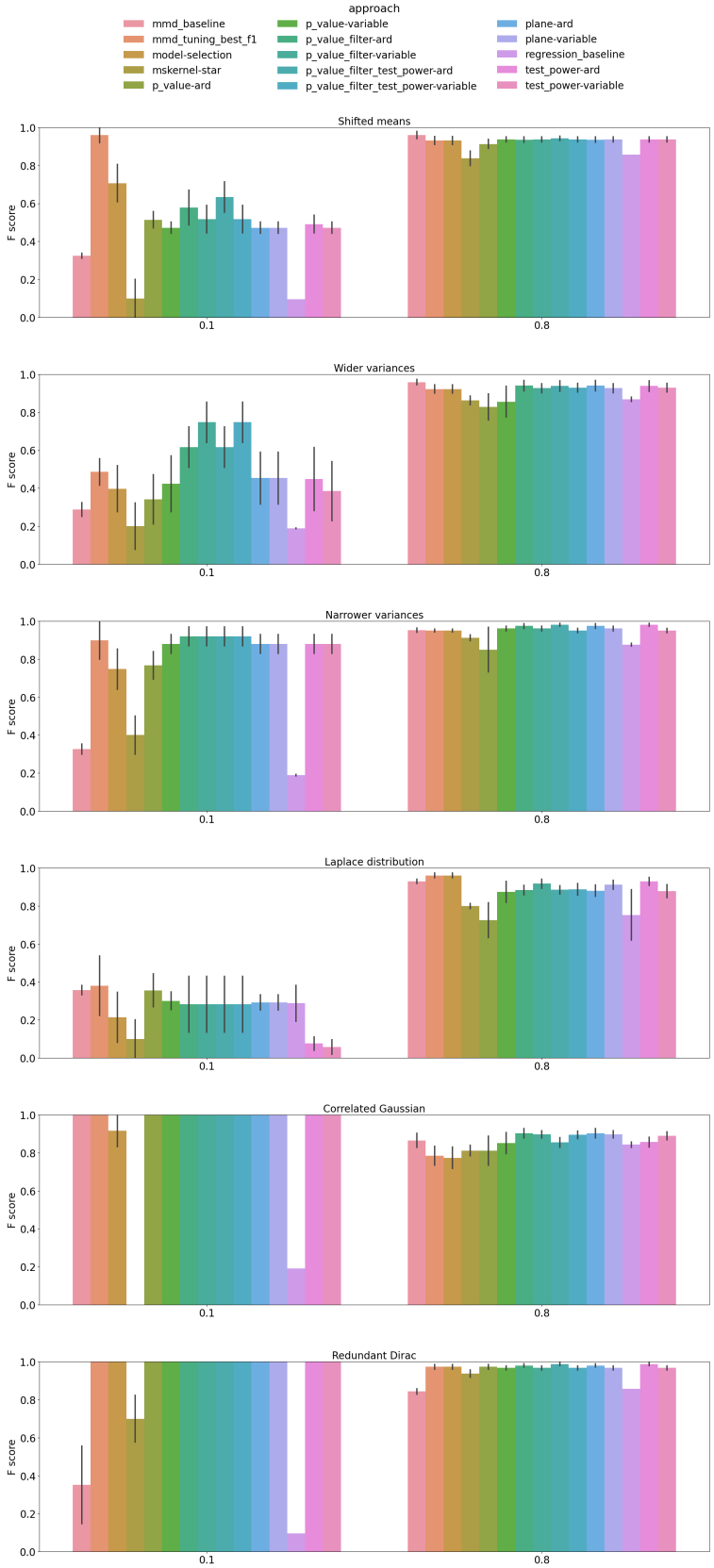
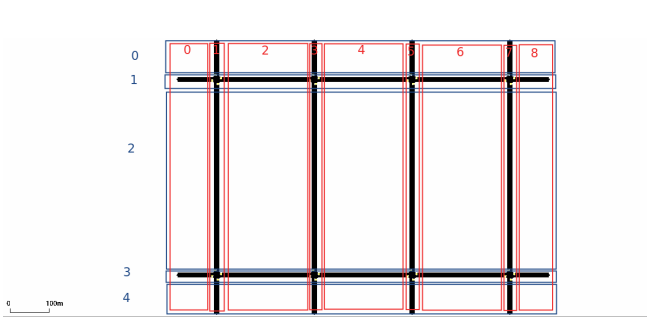
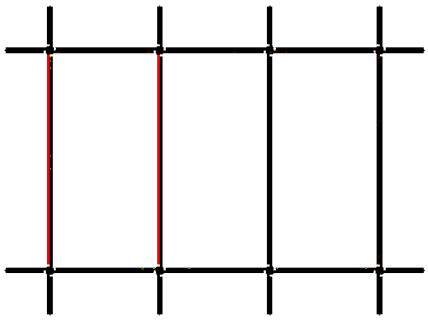


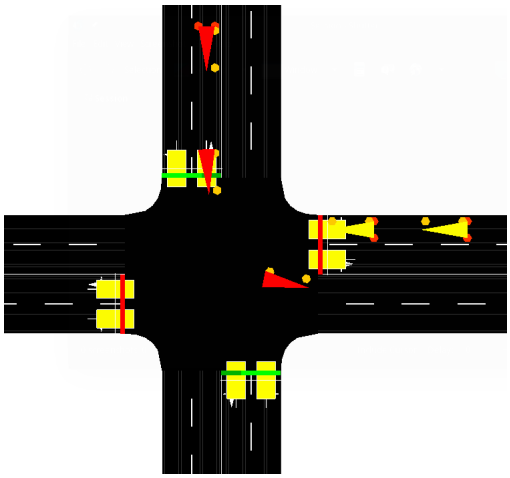
Figure 11: Comparison of the ten candidate aggregation methods for Algorithm 2 and the other baseline approaches. Note that "`p-value-filter-test-power-ard`" is the one adopted in Algorithm 2 in the main body of the paper.



(a) The grid road network.



(b) The two closed lanes in the grid network, highlighted in red.



(c) An intersection in the gird network, with 8 sensors (induction loop detectors) indicated by rectangles and vehicles by triangles.

Figure 12: Illustrations of the traffic road network used in the experiments of Section 3.5.3. Figure 12a describes the grid network. Each edge (= road) and each intersection are named according to the row index (blue) and the column index (red). In each scenario, there are two groups of vehicles, one travelling from the left-upper edge (row 1, column 0) to the right-bottom edge (row 3, column 8) and the other travelling in the opposite direction (from the right-bottom to the left-upper). Figure 12b highlights the two lanes blocked in scenario Q in red. Figure 12c describes an intersection, where two sensors are located at the end of two lanes, resulting in eight sensors for each intersection.

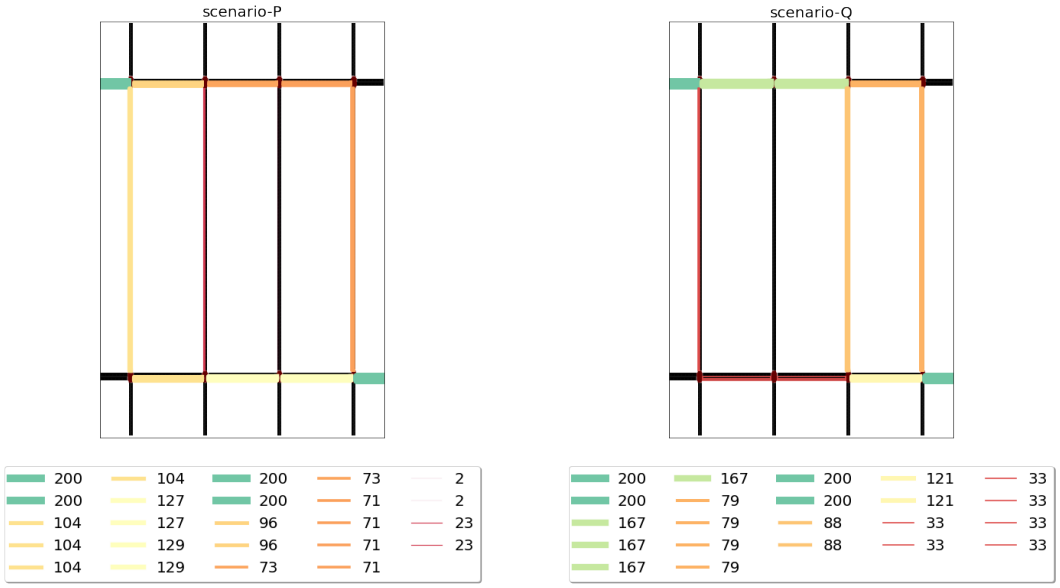
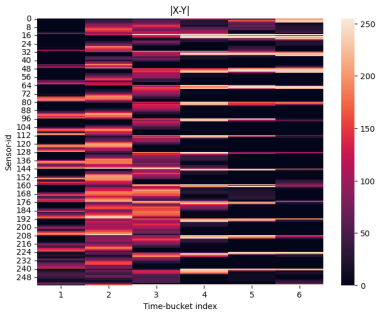
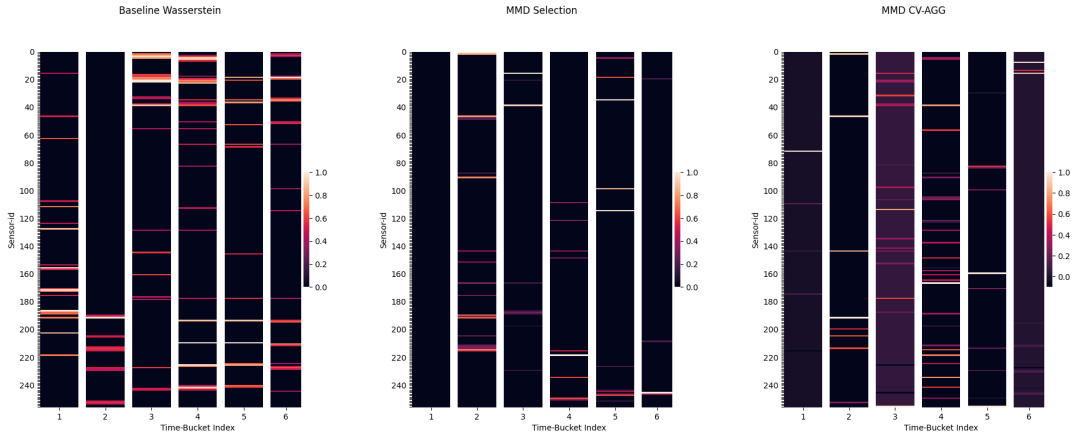


Figure 13: The number of vehicles passing each road in one simulation of scenario P (left) and scenario Q (right). Each edge's line width and colour indicate the corresponding number of vehicles shown in the legend box.



(a) Heatmaps by the average L1 distance



(b) `Wasserstein-based` (c) `MMD selection` (d) `MMD CV`

Figure 14: Comparison of heatmaps. The horizontal and vertical axes are the indices of time buckets b and dimension (sensor-id). 14a: A heatmap of the average L1 distance per sensor at each time bucket. (14b - 14d) Heatmaps by the variable selection methods. The heatmap is based on weights $w \in \mathbb{R}^D$ per dimension at each time bucket. The weight is normalised in $[0.0, 1.0]$ per bucket b for ease of comparison. The heatmap shows only selected variables $\hat{S}$, and the rest of the values are 0.0. When a p-value by the permutation test $p_{\text{test}_b} > 0.05$, then all values at b are 0.0. More descriptions are in Section K.2.

Bibliography

[1] Gosset, william sealy. In *The Concise Encyclopedia of Statistics*, pages 234–235. Springer New York, New York, NY, 2008a. ISBN 978-0-387-32833-1. doi: 10.1007/978-0-387-32833-1_171. URL https://doi.org/10.1007/978-0-387-32833-1_171.

[2] Nonparametric test. In *The Concise Encyclopedia of Statistics*, pages 376–377. Springer New York, New York, NY, 2008b. ISBN 978-0-387-32833-1. doi: 10.1007/978-0-387-32833-1_283. URL https://doi.org/10.1007/978-0-387-32833-1_283.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019. doi: 10.1145/3292500.3330701.

[4] G. Aneiros, S. Novo, and P. Vieu. Variable selection in functional regression models: A review. *Journal of Multivariate Analysis*, 188:104871, 2022. ISSN 0047-259X. doi: 10.1016/j.jmva.2021.104871.

[5] M. Aoshima and K. Yata. Two-sample tests for high-dimension, strongly spiked eigenvalue models. *Statistica Sinica*, 28(1):43–62, 2018. ISSN 10170405, 19968507.

[6] Z. Bai and H. Saranadasa. EFFECT OF HIGH DIMENSION: BY AN EXAMPLE OF A TWO SAMPLE PROBLEM. *Statistica Sinica*, 6:311–329, 1996.

[7] O. Balci. Validation, verification, and testing techniques throughout the life cycle of a simulation study. In *Proceedings of the 26th Conference on Winter Simulation*, WSC '94, pages 215–220. Society for Computer Simulation International, 1994. ISBN 078032109X.

[8] G. A. Barnard. Introduction to pearson (1900) on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 1–10. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_1. URL https://doi.org/10.1007/978-1-4612-4380-9_1.

[9] A. C. Bathke, S. W. Harrar, and L. V. Madden. How to compare small multivariate samples using nonparametric tests. *Computational Statistics & Data Analysis*, 52(11):4951–4965, 2008. ISSN 0167-9473. doi: 10.1016/j.csda.2008.04.006.

[10] J. Behrens and F. Dias. New computational methods in tsunami science. *Philosophical Rransactions. Series A, Mathematical, Physical, and Engineering Sciences*, 373, 10 2015. doi: 10.1098/rsta.2014.0382.

[11] J. Bender et al. SPlisHSPlasH Library, n.a. URL https://github.com/InteractiveComputerGraphics/SPlisHSPlasH.

[12] A. Bharti, M. Naslidnyk, O. Key, S. Kaski, and F.-X. Briol. Optimally-weighted estimators of the maximum mean discrepancy for likelihood-free inference. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2289–2312. PMLR, 2023.

[13] K. Bhattacharyya, B. Maitra, and M. Boltze. Calibration of Micro-Simulation Model Parameters for Heterogeneous Traffic using Mode-Specific Performance Measure. *Transportation Research Record: Journal of the Transportation Research Board*, 2674(1):135–147, jan 2020. ISSN 0361-1981. doi: 10.1177/0361198119900130.

[14] M. Biswas and A. K. Ghosh. A nonparametric two-sample test applicable to high dimensional data. *Journal of Multivariate Analysis*, 123:160–171, 2014. ISSN 0047-259X. doi: 10.1016/j.jmva.2013.09.004.

[15] N. Bonneel, J. Rabin, G. Peyre, and H. Pfister. Sliced and Radon Wasserstein Barycenters of Measures. *Journal of Mathematical Imaging and Vision*, 2014.

[16] R. Bono, M. J. Blanca, J. Arnau, and J. Gómez-Benito. Non-normal distributions commonly used in health, education, and social sciences: A systematic review. *Frontiers in Psychology*, 8, 2017. ISSN 1664-1078. doi: 10.3389/fpsyg.2017.01602.

[17] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics*, 22(14):e49–e57, 07 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl242.

[18] G. E. Box. Robustness in the strategy of scientific model building. *Robustness in Statistics*, pages 201–236, 1979.

[19] F. Briol, A. Barp, A. B. Duncan, and M. A. Girolami. Statistical inference for generative models with maximum mean discrepancy. *arXiv e-print 1906.05944*, 2019. doi: arXiv:1906.05944v1. URL https://arxiv.org/abs/1906.05944v1.

[20] F. Cazals and A. Lhéritier. Beyond Two-sample-tests: Localizing Data Discrepancies in High-dimensional Spaces. In *IEEE/ACM International Conference on Data Science and Advanced Analytics*, IEEE/ACM International Conference on Data Science and Advanced Analytics, page 29, Paris, France, Oct. 2015. sample-selection.

[21] S. X. Chen and Y.-L. Qin. A two-sample test for high-dimensional data with applications to gene-set testing. *The Annals of Statistics*, 38(2):808–835, 2010. doi: 10.1214/09-AOS716.

[22] X. Chen, X. Huang, C. Jiao, M. G. Flanner, T. Raeker, and B. Palen. Running climate model on a commercial cloud computing environment: A case study using community earth system model (cesm) on amazon aws. *Computers & Geosciences*, 98:21–25, 2017. ISSN 0098-3004. doi: 10.1016/j.cageo.2016.09.014.

[23] Y. Chen, M. Welling, and A. Smola. Supersamples from kernel-herding. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010)*, pages 109–116, 2010.

[24] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[25] L. Codeca and J. Härri. Monaco SUMO traffic (MoST) scenario: A 3D mobility scenario for cooperative ITS. *EPiC Series in Engineering*, 2:43–55, 2018.

[26] J. Cortés, M. Mahecha, M. Reichstein, and A. Brenning. Accounting for multiple testing in the analysis of spatio-temporal environmental data. *Environmental and Ecological Statistics*, 27 (2):293–318, 2020.

[27] K. Csilléry, M. G. B. Blum, O. E. Gaggiotti, and O. François. Approximate Bayesian Computation (ABC) in practice. *Trends in Ecology & Evolution*, 25(7):410–418, 2010. ISSN 0169-5347. doi: https://doi.org/10.1016/j.tree.2010.04.001.

[28] R. Cutura, M. Aupetit, J.-D. Fekete, and M. Sedlmair. Comparing and Exploring High-Dimensional Data with Dimensionality Reduction Algorithms and Matrix Visualizations. In *AVI' 20 - International Conference on Advanced Visual Interfaces*, Sept. 2020. doi: 10.1145/3399715.3399875.

[29] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6): 1293–1305, 2019. doi: 10.1109/JAS.2019.1911747.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[31] Y. Dodge. Nonparametric statistics. In *The Concise Encyclopedia of Statistics*, pages 375–376. Springer New York, New York, NY, 2008. ISBN 978-0-387-32833-1. doi: 10.1007/978-0-387-32833-1_282.

[32] T. Duong. Local significant differences from nonparametric two-sample tests. *Journal of Nonparametric Statistics*, 25(3):635–645, 2013.

[33] T. Duong and I. Koch. Highest density difference region estimation with application to flow cytometric data. *Biometrical Journal*, 51(3):504–521, 2009.

[34] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994. ISBN 0412042312.

[35] J. Fan and R. Li. Statistical challenges with high dimensionality: Feature selection in knowledge discovery. In *Proceedings of the International Congress of Mathematicians*, pages 595 – 622, 2006.

[36] R. E. Fancher. *Introduction to Galton (1889) Co-Relations and Their Measurement, Chiefly from Anthropometric Data*, pages 1–16. Springer New York, New York, NY, 1997. ISBN 978-1-4612-0667-5. doi: 10.1007/978-1-4612-0667-5_1. URL https://doi.org/10.1007/978-1-4612-0667-5_1.

[37] R. A. Fisher. Statistical methods for research workers. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 66–70. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_6. URL https://doi.org/10.1007/978-1-4612-4380-9_6.

[38] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[39] J. H. Friedman. On multivariate goodness–of–fit and two–sample testing. *Statistical Problems in Particle Physics, Astrophysics, and Cosmology*, 1:311, 2003.

[40] J. H. Friedman and L. C. Rafsky. Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests. *The Annals of Statistics*, 7(4):697–717, 1979. ISSN 00905364.

[41] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel measures of conditional dependence. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[42] K. Fukumizu, L. Song, and A. Gretton. Kernel Bayes' rule: Bayesian inference with positive definite kernels. *Journal of Machine Learning Research*, 14:3753–3783, 2013.

[43] R. Gao, F. Liu, J. Zhang, B. Han, T. Liu, G. Niu, and M. Sugiyama. Maximum mean discrepancy test is aware of adversarial attacks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3564–3575. PMLR, 18–24 Jul 2021.

[44] D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic. *arXiv e-prints 1707.07269v3*, 2018. doi: arXiv:1707.07269v3.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, Oct. 2020. ISSN 0001-0782. doi: 10.1145/3422622.

[46] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012a.

[47] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. *The Twenty-Sixth Annual Conference on Neural Information Processing Systems (NIPS)*, 25, 2012b.

[48] R.-j. Gu and W.-b. Xu. An improved manifold learning algorithm for data visualization. In *2006 International Conference on Machine Learning and Cybernetics*, pages 1170–1173, 2006. doi: 10.1109/ICMLC.2006.258599.

[49] A. Hald. *A History of Probability and Statistics and Their Applications before 1750*. Wiley, New York, 1990a. ISBN 0-471-47129-1.

[50] A. Hald. *A History of Probability and Statistics and Their Applications before 1750*. Wiley, New York, 1990b. ISBN 0-471-47129-1.

[51] S. Hara, T. Katsuki, H. Yanagisawa, T. Ono, R. Okamoto, and S. Takeuchi. Consistent and Efficient Nonparametric Different-Feature Selection. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 130–138. PMLR, 20–22 Apr 2017.

[52] N. Henze. A Multivariate Two-Sample Test Based on the Number of Nearest Neighbor Type Coincidences. *The Annals of Statistics*, 16(2):772 – 783, 1988. doi: 10.1214/aos/1176350835.

[53] S. Hido, T. Idé, H. Kashima, H. Kubo, and H. Matsuzawa. Unsupervised change analysis using supervised learning. In *Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data mining*, pages 148–159, 2008.

[54] H. Hotelling. Relations between two sets of variates. In N. L. Kotz, Samueland Johnson, editor, *Breakthroughs in Statistics: Methodology and Distribution*, pages 162–190. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_14.

[55] J. Islam, P. M. Vasant, B. M. Negash, M. B. Laruccia, M. Myint, and J. Watada. A holistic review on artificial intelligence techniques for well placement optimization problem. *Advances in Engineering Software*, 141:102767, 2020. ISSN 0965-9978. doi: 10.1016/j.advengsoft.2019.102767.

[56] W. Jitkrittum, Z. Szabó, K. Chwialkowski, and A. Gretton. Interpretable distribution features with maximum testing power. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 181–189, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[57] B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[58] I. Kim, A. B. Lee, and J. Lei. Global and local two-sample tests via regression. *Electronic Journal of Statistics*, 13(2):5253 – 5305, 2019. doi: 10.1214/19-EJS1648. related-work, sample-selection.

[59] K. Kisamori, M. Kanagawa, and K. Yamazaki. Simulator calibration under covariate shift with kernels. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1244–1253, Online, 26–28 Aug 2020. PMLR.

[60] A. Kolmogorov. On the empirical determination of a distribution function. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 106–113. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_10. URL https://doi.org/10.1007/978-1-4612-4380-9_10.

[61] M. R. Kosorok and S. Ma. Marginal asymptotics for the large p, small n paradigm: With applications to microarray data. *The Annals of Statistics*, 35(4):1456 – 1486, 2007. doi: 10.1214/009053606000001433.

[62] S. Kotz and N. L. Johnson. *Breakthroughs in Statistics*. Springer New York, NY, 1992. ISBN 978-1-4612-4380-9.

[63] J. M. Kübler, W. Jitkrittum, B. Schölkopf, and K. Muandet. A witness two-sample test. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 1403–1419. PMLR, 28–30 Mar 2022a.

[64] J. M. Kübler, V. Stimper, S. Buchholz, K. Muandet, and B. Schölkopf. AutoML two-sample test. In *Advances in Neural Information Processing Systems*, volume 35, pages 15929–15941. Curran Associates, Inc., 2022b.

[65] J. N. Lim, M. Yamada, W. Jitkrittum, Y. Terada, S. Matsui, and H. Shimodaira. More powerful selective kernel tests for feature selection. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 820–830. PMLR, 26–28 Aug 2020.

[66] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[67] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland. Learning deep kernels for non-parametric two-sample tests. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[68] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, 2017. doi: 10.1109/TVCG.2016.2640960.

[69] J. R. Lloyd and Z. Ghahramani. Statistical model criticism using kernel two sample tests. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[70] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[71] D. Lopez-Paz and M. Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations (ICLR)*, 2017.

[72] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, 1947. doi: 10.1214/aoms/1177730491.

[73] J. Marinković. Multivariate statisticsmultivariate statistics. In W. Kirch, editor, *Encyclopedia of Public Health*, pages 973–976. Springer Netherlands, Dordrecht, 2008. ISBN 978-1-4020-5614-7. doi: 10.1007/978-1-4020-5614-7_2264.

[74] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society (Series B)*, 72:417–473, 2010. doi: 10.1111/j.1467-9868.2010.00740.x.

[75] K. Mitsuzawa, M. Kanagawa, S. Bortoli, M. Grossi, and P. Papotti. Variable selection in maximum mean discrepancy for interpretable distribution comparison. *arXiv e-print 2311.01537*, 2023. doi: arXiv:2311.01537. URL https://arxiv.org/abs/2311.01537.

[76] K. Mitsuzawa, M. Grossi, S. Bortoli, and M. Kanagawa. Variable selection for comparing high-dimensional time-series data. *arXiv e-print 2412.06870*, 2024. doi: arXiv:2412.06870. URL https://arxiv.org/abs/2412.06870.

[77] K. Muandet, K. Fukumizu, B. K. Sriperumbudur, and B. Schölkopf. Kernel mean embedding of distributions : A review and beyond. *Foundations and Trends in Machine Learning*, 10(1–2): 1–141, 2017.

[78] K. Muandet, M. Kanagawa, S. Saengkyongam, and S. Marukatat. Counterfactual mean embeddings. *Journal of Machine Learning Research*, 22(162):1–71, 2021.

[79] J. W. Mueller and T. Jaakkola. Principal differences analysis: Interpretable characterization of differences between distributions. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[80] M. Najdek, H. Xie, and W. Turek. Scaling simulation of continuous urban traffic model for high performance computing system. In *Computational Science – ICCS 2021*, pages 256–263, Cham, 2021. Springer International Publishing. ISBN 978-3-030-77961-0.

[81] S. Nakagome, K. Fukumizu, and S. Mano. Kernel approximate Bayesian computation in population genetic inferences. *Statistical Applications in Genetics and Molecular Biology*, 12 (6):667–678, 2013.

[82] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. In *Breakthroughs in Statistics: Foundations and Basic Theory*, pages 73–108. Springer New York, New York, NY, 1992. ISBN 978-1-4612-0919-5. doi: 10.1007/978-1-4612-0919-5_6. URL https://doi.org/10.1007/978-1-4612-0919-5_6.

[83] H. R. Pagels. Perfect Symmetry The Search for the Beginning of Time. 2009.

[84] M. Park, W. Jitkrittum, and D. Sejdinovic. K2-abc: Approximate bayesian computation with kernel embeddings. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 398–407, Cadiz, Spain, 09–11 May 2016. PMLR.

[85] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

[86] L. Prantl, B. Ummenhofer, V. Koltun, and N. Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics. In *Conference on Neural Information Processing Systems*, 2022.

[87] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN 0262170051.

[88] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[89] C. R. Rao. R. A. Fisher: The Founder of Modern Statistics. *Statistical Science*, 7(1):34 – 48, 1992. doi: 10.1214/ss/1177011442.

[90] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[91] J.-W. Romeijn. Philosophy of Statistics. In *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022.

[92] P. R. Rosenbaum. An exact distribution-free test comparing two multivariate distributions based on adjacency. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(4):515–530, 2005. doi: 10.1111/j.1467-9868.2005.00513.x.

[93] D. B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, pages 1151–1172, 1984.

[94] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[95] R. G. Sargent. Verification and validation of simulation models. In *Proceedings of the 2010 Winter Simulation Conference*, pages 166–183, 2010. doi: 10.1109/WSC.2010.5679166.

[96] E. Scharwächter and E. Müller. Two-sample testing for event impacts in time series. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 10–18, 2020. doi: 10.1137/1.9781611976236.2.

[97] M. F. Schilling. Multivariate two-sample tests based on nearest neighbors. *Journal of the American Statistical Association*, 81(395):799–806, 1986. doi: 10.1080/01621459.1986.10478337.

[98] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002. ISBN 9780262256933. doi: 10.7551/mitpress/4175.001.0001.

[99] M. Sedghi, G. Atia, and M. Georgiopoulos. Non-linear manifold clustering based on conformity index. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pages 607–611, 2020. doi: 10.1109/IEEECONF51394.2020.9443279.

[100] D. Sha, K. Ozbay, and Y. Ding. Applying Bayesian Optimization for Calibration of Transportation Simulation Models. *Transportation Research Record: Journal of the Transportation Research Board*, 2674(10):215–228, oct 2020. ISSN 0361-1981. doi: 10.1177/0361198120936252.

[101] L. Shi, W. Tong, H. Fang, U. Scherf, J. Han, R. K. Puri, F. W. Frueh, F. M. Goodsaid, L. Guo, Z. Su, et al. Cross-platform comparability of microarray technology: intra-platform consistency and appropriate data analysis procedures are essential. *BMC bioinformatics*, 6:1–14, 2005.

[102] M. Sinn, A. Ghodsi, and K. Keller. Detecting change-points in time series by maximum mean discrepancy of ordinal pattern distributions. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pages 786–794. AUAI Press, 2012. ISBN 9780974903989.

[103] S. A. Sisson, Y. Fan, and M. Beaumont. *Handbook of Approximate Bayesian Computation*. Chapman and Hall/CRC, 2018. ISBN 9781315117195. doi: 10.1201/9781315117195.

[104] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert space embedding for distributions. In *Proceedings of the International Conference on Algorithmic Learning Theory*, volume 4754, pages 13–31. Springer, 2007.

[105] H. Song and H. Chen. Generalized kernel two-sample tests. *Biometrika*, 111(3):755–770, 11 2023. ISSN 1464-3510. doi: 10.1093/biomet/asad068.

[106] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, G. Lanckriet, and B. Schölkopf. Kernel Choice and Classifiability for RKHS Embeddings of Probability Distributions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 22, 2009.

[107] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:15171561, aug 2010. ISSN 1532-4435.

[108] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.

[109] Student. The probable error of a mean. In S. Kotz and N. L. Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, pages 33–57. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_4.

[110] D. J. Sutherland, H.-Y. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, and A. Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *International Conference on Learning Representations*, 2017.

[111] O. Thas. *Comparing Distributions*. Springer New York, NY, 1 edition, 2010. ISBN 978-0-387-92709-1. doi: 10.1007/978-0-387-92710-7.

[112] G. Thompson. Asymptotic distribution of rank statistics under dependencies with multivariate application. *Journal of Multivariate Analysis*, 33(2):183–211, 1990. ISSN 0047-259X. doi: 10.1016/0047-259X(90)90045-J.

[113] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[114] H. Tyralis and G. Papacharalampous. Variable selection in time series forecasting using random forests. *Algorithms*, 10(4), 2017. ISSN 1999-4893. doi: 10.3390/a10040114.

[115] B. Ummenhofer, L. Prantl, N. Thuerey, and V. Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020.

[116] S. Vijitpornkul and W. Marurngsith. Simulating crowd movement in agent-based model of large-scale flood. In *2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications*, pages 1–6, 2015. doi: 10.1109/ICAICTA.2015.7335368.

[117] C. Villani. *Optimal Transport: Old and New*. Springer, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9.

[118] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[119] J. Wang, S. S. Dey, and Y. Xie. Variable selection for kernel two-sample tests. *arXiv e-print 2302.07415v3*, 2023. URL https://arxiv.org/abs/2302.07415v3.

[120] R. L. Wasserstein and N. A. Lazar. The ASA statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016. doi: 10.1080/00031305.2016.1154108.

[121] M. West. Bayesian Factor Regression Models in the Large p, Small n Paradigm. *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, 07 2003. doi: 10.1093/oso/9780198526155.003.0053. seems the 1st paper of using "large p, small n".

[122] G. Wynne and A. B. Duncan. A kernel two-sample test for functional data. *Journal of Machine Learning Research*, 23(1), jan 2022. ISSN 1532-4435.

[123] M. Yamada, Y. Umezu, K. Fukumizu, and I. Takeuchi. Post selection inference with kernels. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and*

Statistics, volume 84 of *Proceedings of Machine Learning Research*, pages 152–160. PMLR, 09–11 Apr 2018.

[124] M. Yamada, D. Wu, Y.-H. H. Tsai, H. Ohta, R. Salakhutdinov, I. Takeuchi, and K. Fukumizu. Post selection inference with incomplete maximum mean discrepancy estimator. In *International Conference on Learning Representations*, 2019.

[125] W. Zaremba, A. Gretton, and M. Blaschko. B-test: A non-parametric, low variance kernel two-sample test. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[126] W. Zheng, F.-Y. Wang, and C. Gou. Nonparametric Different-Feature Selection Using Wasserstein Distance. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 982–988, 2020. doi: 10.1109/ICTAI50040.2020.00153.

[127] D. Zhou and H. Chen. A new ranking scheme for modern data and its application to two-sample hypothesis testing. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 3615–3668. PMLR, 2023. a conference.

[128] M. Zych, M. Najdek, M. Paciorek, and W. Turek. Distributed architecture for highly scalable urban traffic simulation. In *Computational Science – ICCS 2022*, pages 517–530, Cham, 2022. Springer International Publishing. ISBN 978-3-031-08760-8.