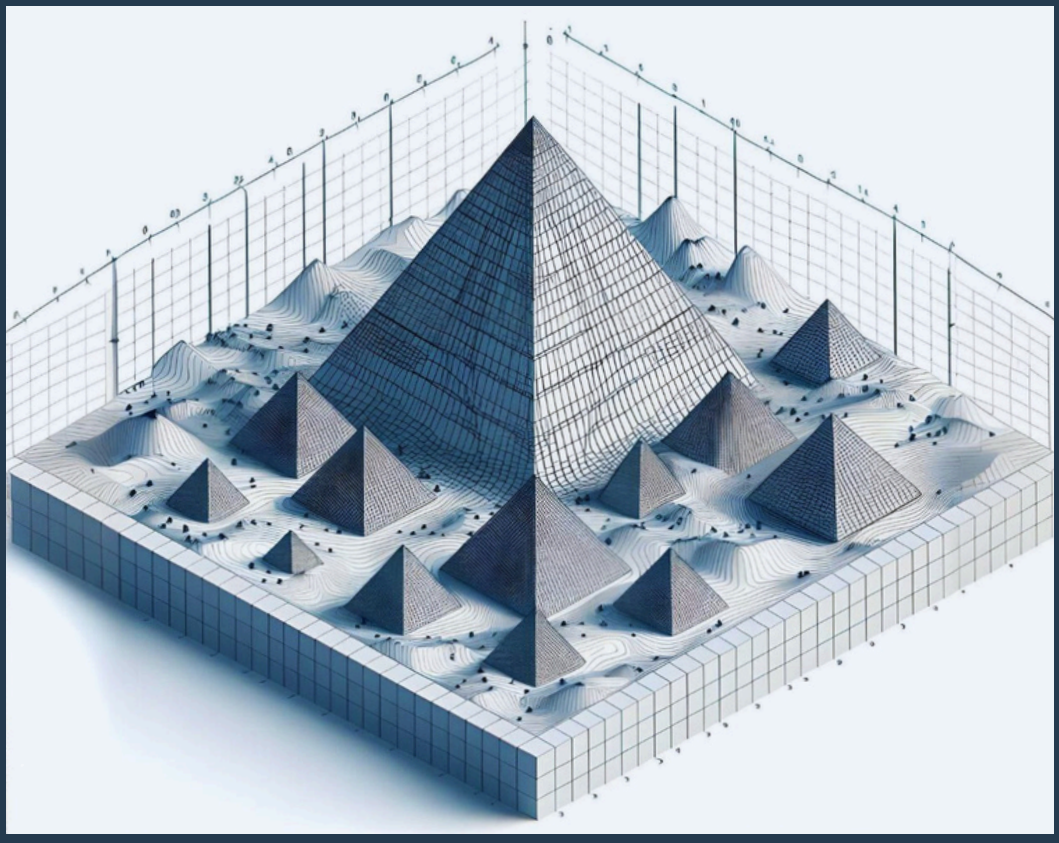
Machine Learning Beyond Point Predictions: Uncertainty Quantification



Rafael Izbicki

v0.1

Machine Learning

Beyond Point Predictions:

Uncertainty Quantification

v0.1 — Please do not distribute.

Last compiled on September 26, 2024

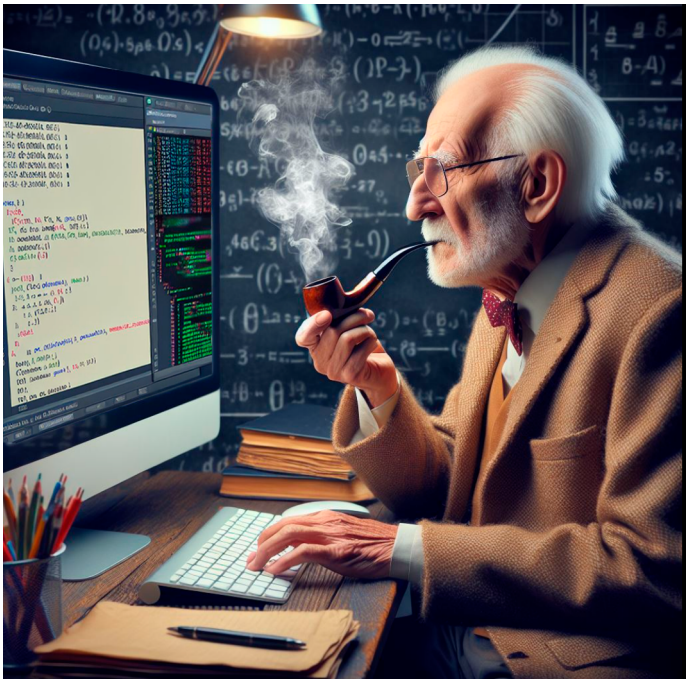
Title: Machine Learning learning beyond point predictions: uncertainty quantification. Version 0.1. Last compiled on September 26, 2024.

Author: Rafael Izbicki.

Cover: This cover was created through the assistance of AI. Special thanks to Lea Veras, Michel Helcias Montoril, and Tiago Mendonça for their valuable suggestions.

Rafael Izbicki

Machine Learning Beyond Point Predictions: Uncertainty Quantification



Solum certum nihil esse certi.

The only certain thing is that nothing is certain.

Pliny the Elder

To all of those who deeply shaped the way I think:

My teachers
My students
My family

Contents

.

.

.

Foreword

This book. Over the past few decades, Machine Learning has achieved remarkable success in generating accurate predictions across diverse domains. However, point predictions alone are often insufficient; understanding and effectively representing the uncertainties in these predictions is crucial for informed decision-making in real-world applications. As a result, there has been a surge in research focused on developing methods to quantify this uncertainty. This book aims to explore some of these techniques, with a particular emphasis on approaches that I find compelling and to which I have contributed through my research.

The two cultures. Breiman (2001a) distinguishes between two key cultures in statistical modeling: the *data modeling culture*, which dominates traditional statistics, and the *algorithmic modeling culture*, prevalent in machine learning. The data modeling culture emphasizes the interpretation of model parameters, operating under the assumption that the model accurately reflects the underlying data-generating mechanism. This leads to a focus on goodness-of-fit tests to ensure the model's validity. In contrast, the algorithmic modeling culture prioritizes prediction accuracy, where models are evaluated based on their predictive performance using loss functions, rather than an assumption of correctness.

Integrating these approaches is particularly advantageous for Uncertainty Quantification. While loss functions are effective for comparing models, they provide only a relative measure of performance and do not assess the model's overall adequacy. Consequently, they fall short of answering critical questions such as, "Should we continue searching for better estimates, or is our current model fit sufficient?" Goodness-of-fit techniques, however, can address this gap by evaluating model adequacy and suggesting potential improvements. Thus, combining these cultures

.

enables a more comprehensive model evaluation. This integrated approach is emphasized in Chapter 4, which focuses on using goodness-of-fit methods to assess the adequacy of density models, thereby enriching the model evaluation process with practical insights.

Theory. This book presents theoretical results about the techniques we explore. Good theory isn't just an abstract exercise; it provides a fresh perspective on problems, complementing what we learn from practical applications and simulations. While some sections delve into more advanced mathematics to deepen your understanding, you can still grasp the book's core concepts even if you choose to skip these parts, especially if you're focused on an introductory overview of Uncertainty Quantification.

Prerequisites. We assume that the reader is familiar with the foundations of probability and statistical inference. An excellent introduction to the topic at the desired level is provided by Schervish and DeGroot (2014). Readers will also benefit from having taken an introductory class on machine learning or regression analysis.

Research Papers. Parts of this book are heavily based on the following research papers:

- Izbicki, R., & Lee, A. B. (2017). Converting high-dimensional regression to high-dimensional conditional density estimation. *Electronic Journal of Statistics*, *11*(2), 2800–2831
- Izbicki, R., Lee, A. B., & Pospisil, T. (2019). ABC–CDE: Toward approximate bayesian computation with complex high-dimensional data and limited simulations. *Journal of Computational and Graphical Statistics*, *28*(3), 481–492
- Izbicki, R., Shimizu, G. T., & Stern, R. B. (2020). Distribution-free conditional predictive bands using density estimators. *Proceedings of Machine Learning Research (AISTATS Track)*
- Dalmasso, N., Pospisil, T., Lee, A. B., Izbicki, R., Freeman, P. E., & Malz, A. I. (2020b). Conditional density estimation tools in python and r with applications to photometric redshifts and likelihood-free cosmological inference. *Astronomy and Computing*, 100362

- Dalmasso, N., Izbicki, R., & Lee, A. (2020a). Confidence sets and hypothesis testing in a likelihood-free inference setting. *International Conference on Machine Learning*, 2323–2334

- Dalmasso, N., Masserano, L., Zhao, D., Izbicki, R., & Lee, A. B. (2021). Likelihood-free frequentist inference: Bridging classical statistics and machine learning in simulator-based inference. *arXiv preprint arXiv:2107.03920*

- Zhao, D., Dalmasso, N., Izbicki, R., & Lee, A. B. (2021). Diagnostics for conditional density models and bayesian inference algorithms. *Uncertainty in Artificial Intelligence*, 1830–1840

- Dey, B., Zhao, D., Newman, J. A., Andrews, B. H., Izbicki, R., & Lee, A. B. (2022). Calibrated predictive distributions via diagnostics for conditional coverage. *arXiv preprint arXiv:2205.14568*

- Izbicki, R., Shimizu, G., & Stern, R. B. (2022). CD-split and HPD-split: Efficient conformal regions in high dimensions. *Journal of Machine Learning Research*

- Masserano, L., Dorigo, T., Izbicki, R., Kuusela, M., & Lee, A. (2023). Simulator-based inference with waldo: Confidence regions by leveraging prediction algorithms and posterior estimators for inverse problems. *International Conference on Artificial Intelligence and Statistics*, 2960–2974

- Cabezas, L. M., Otto, M. P., Izbicki, R., & Stern, R. B. (2024). Regression trees for fast and adaptive prediction intervals. *Information Sciences*, 121369

- Nakazono, L., R Valença, R., Soares, G., Izbicki, R., Ivezić, Ž., R Lima, E., T Hirata, N., Sodré Jr, L., Overzier, R., Almeida-Fernandes, F., et al. (2024). The quasar catalogue for s-plus dr4 (qucats) and the estimation of photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, *531*(1), 327–339

- Xiao, Y., Soares, G., Bastos, L., Izbicki, R., & Moraga, P. (2024). Dengue nowcasting in brazil by combining official surveillance data and google trends information. *medRxiv*. https://doi.org/10.1101/2024.09.02.24312934

- Comito, M. B., Izbicki, R., do Canto Hubert Junior, P., & Moura, F. (2024). Improving decision-making in construction: Nonparametric modeling of weather-induced delays [in prep.]

.

Code and Data. Notebooks for reproducing some of the analyses presented in this book are available at https://github.com/rizbicki/UQ4ML. These notebooks are referenced throughout the book for further exploration.

Errors. This book certainly contains various errors. If you encounter any mistakes, whether grammatical or related to mathematics, or if you have additional suggestions, please email them to rafaelizbicki (at) gmail.com.

Acknowledgments. I am deeply grateful to my long-term collaborators, Ann B. Lee and Rafael B. Stern, whose significant contributions were crucial in developing much of the research that forms the foundation of this book. Their insights have been invaluable, and this work would not have been possible without their partnership.

I am also grateful to Brett Andrews, Luben Miguel Cruz Cabezas, Mateus Borges Comito, Victor Coscrato, Biprateep Dey, Nic Dalmasso, Tomasso Dorigo, Marco Henrique de Almeida Inacio, Paulo do Canto Hubert Junior, Mikael Kuusela, Alex Malz, Luca Masserano, Filipe Moura, Lilianne Nakazono, Jeff Newman, Mateus Piovezan Otto, Taylor Pospisil, Tiago Mendonça dos Santos, Gilson Shimizu, Gabriela Soares, Raquel Ruiz Valença, and David Zhao for their exceptional contributions as co-authors.

I gratefully acknowledge the São Paulo Research Foundation (FAPESP), the Brazilian National Council for Scientific and Technological Development (CNPq), the Coordination for the Improvement of Higher Education Personnel (CAPES), and the Federal University of São Carlos (UFSCar). Their support was essential in making this project possible.

Finally, I am deeply thankful to my family—Meyer, Deborah, Sarah, Lea, Bianca, and Lila for their constant support and understanding.

Chapter 1

Introduction

The Fortune Teller. Georges de La Tour, 1630s, The Metropolitan Museum of Art, New York.

> Maturity is the capacity to endure uncertainty.
>
> John Huston Finley

Machine learning (ML) has made remarkable strides in recent years, driven by larger datasets and more sophisticated algorithms. These advancements have enhanced the accuracy of predictions across a variety of fields, from healthcare diagnostics to market trend forecasting. However, despite these improvements, the relationship between the inputs to a machine learning model, represented by $\mathbf{x}$, and the target variable, y, often involves significant uncertainty. This uncertainty can stem not only from the limitations of the model but also from the inherent unpredictability of the phenomena being studied, given the available features $\mathbf{x}$.

The central goal of this book is to provide practical tools and theoretical insights for enhancing the reliability of machine learning predictions by incorporating uncertainty quantification. We focus on developing methods that go beyond simple point predictions to offer a more nuanced understanding of the uncertainty inherent in various prediction problems, enabling more informed decision-making across diverse applications.

1.1 Motivating Example: Life Expectancy and Per Capita GDP

To illustrate the challenges of uncertainty in prediction, consider the relationship between GDP per capita ($\mathbf{x}$) and life expectancy (y) in 211 countries in 2012, as shown in Figure 1.1. GDP per capita is often used as a proxy for a country's wealth or development, and it generally correlates with better health outcomes, including longer life expectancy. However, no machine learning model that outputs a single y for each $\mathbf{x}$ can perfectly predict life expectancy across all countries using GDP alone. This is because GDP per capita, while informative, is not the sole determinant of life expectancy. Other factors, such as healthcare quality, education, and lifestyle, contribute to the observed outcomes, leading to variability even among countries with similar GDPs.

Even with an infinite amount of data, perfect predictions remain impossible in

such cases. The reason is that even the most advanced machine learning algorithms cannot overcome the fact that $\mathbf{x}$ does not uniquely determine y. This inherent variability limits the precision of point predictions, often making them insufficient for informed decision-making. For example, countries with a GDP per capita of around \$1000 display much larger variability in life expectancy compared to those with a GDP per capita of around \$80000. As a result, point predictions for countries with $\mathbf{x} = \$1000$ are far less reliable than for countries with $\mathbf{x} = \$80000$, even though this disparity is not captured in the point predictions alone. This illustrates the need for models that go beyond predicting the most likely outcome—they must also communicate the uncertainty around their predictions.

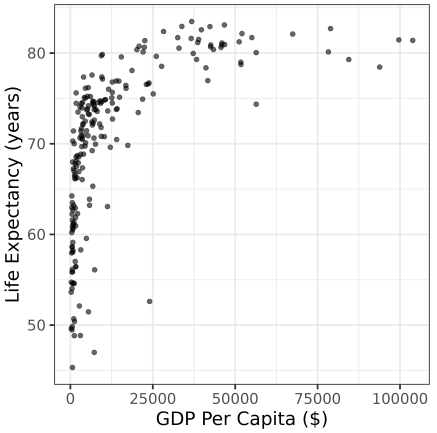


Figure 1.1: Life expectancy (y) versus GDP per Capita ($\mathbf{x}$) of 211 countries. It is impossible to give perfect predictions for y using $\mathbf{x}$ only. Moreover, countries with a GDP per capita of $\approx \$1000$ have much greater variability in their life expectancy than countries with a GDP per capita of $\approx \$80000$.

1.2 Understanding Aleatoric and Epistemic Uncertainty

The uncertainty associated with predicting a new label Y given features $\mathbf{x}$ can be divided into two main categories[1]:

- **Aleatoric Uncertainty:** This type of uncertainty arises when the same feature

[1]There are several nuances in this classification; see Hüllermeier and Waegeman (2021) and references therein for a thorough discussion.

vector $\mathbf{x}$ corresponds to different possible labels y. It captures the inherent variability in the distribution of Y conditional on $\mathbf{x}$, which we denote by $Y|\mathbf{x}$. For example, in the case of GDP and life expectancy, two countries with the same GDP per capita may have different life expectancies due to factors such as healthcare or education. Aleatoric uncertainty can be measured by the conditional distribution of Y given $\mathbf{x}$. Figure 1.2 illustrates this concept: the green and blue curves represent the conditional densities $f(y|\mathbf{x}_a)$ and $f(y|\mathbf{x}_b)$, showing that while the optimal point predictions for $\mathbf{x}_a$ and $\mathbf{x}_b$ (given by the true regression functions $\mathbb{E}[Y|\mathbf{x}_a]$ and $\mathbb{E}[Y|\mathbf{x}_b]$; see Chapter 2 for a mathematical explanation) are identical, the associated uncertainties differ. This reflects the inherent variability in the output for each input. In the context of classification (that is, a prediction problem where Y is a qualitative label), $f(y|\mathbf{x})$ is the probability of the label given the features, $\mathbb{P}(Y = y|\mathbf{x})$.

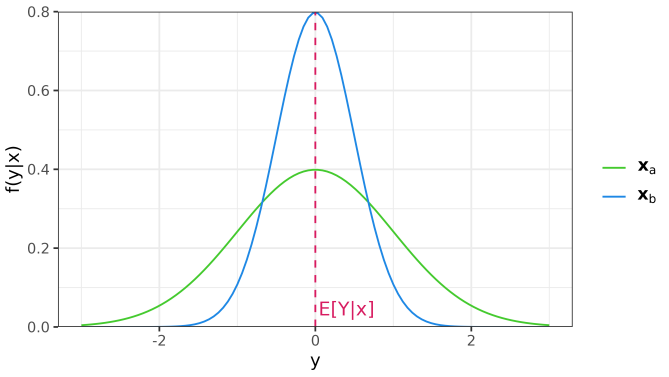


Figure 1.2: Conditional densities $f(y|\mathbf{x})$ for inputs $\mathbf{x} = \mathbf{x}_a$ (green) and $\mathbf{x} = \mathbf{x}_b$ (blue), showing identical optimal point predictions but different uncertainties. The dashed line marks the optimal point prediction $r(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}]$.

- **Epistemic Uncertainty:** This uncertainty is due to a lack of knowledge about the true distribution of $Y|\mathbf{x}$. Increasing the size of the dataset can help reduce epistemic uncertainty, but it does not affect aleatoric uncertainty. Epistemic uncertainty is particularly important in scenarios with limited data. Figure 1.3 illustrates epistemic uncertainty: the first plot shows three regression models that are compatible with the same dataset, each producing different predictions due to sparse data in certain regions, highlighting the uncertainty that

arises from the lack of data. The second plot shows the epistemic uncertainty associated to the regression function as measured by a Gaussian process fit to the same data (see Section 6.2 for Gaussian processes). These bands visually depict epistemic uncertainty, showing greater uncertainty in areas with fewer data points and more confidence in regions with denser data.

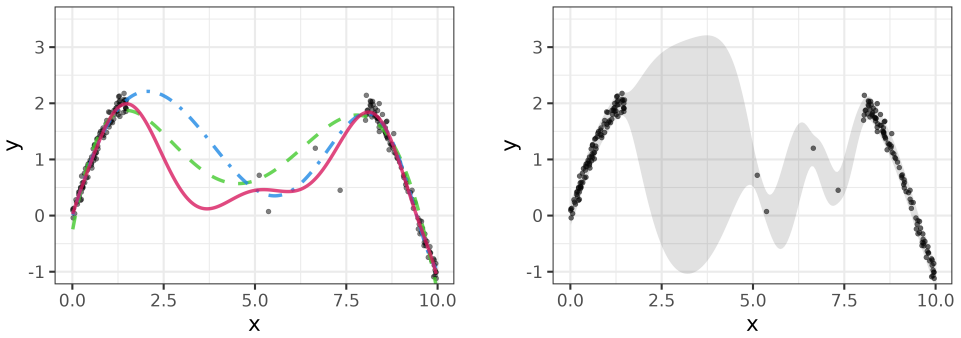


Figure 1.3: Illustration of epistemic uncertainty. Left: Comparison of three regression models, each producing different predictions due to sparse data in certain regions. Right: Gaussian process regression with uncertainty bands, where the shaded area represents epistemic uncertainty.

In many prediction problems, it is important to distinguish between these two types of uncertainty, as they require different treatment. While epistemic uncertainty can be mitigated by gathering larger datasets or improving the model, aleatoric uncertainty can only be reduced by measuring additional covariates. Ignoring this distinction can lead to overconfident predictions and flawed decision-making.

In the next section, we will briefly explore real-world machine learning problems where quantifying uncertainty is crucial. These examples illustrate the real-world impact of the methods we will cover.

1.3 Applications of Uncertainty Quantification in Machine Learning

Uncertainty Quantification (UQ) techniques are increasingly recognized as vital for enhancing the reliability and usefulness of predictive models across various domains. By providing insights into the limitations and confidence levels of model predictions, UQ supports better decision-making and more effective risk assessment. Below are some recent applications of UQ in machine learning:

- **Medical Image Segmentation:** UQ is essential for addressing inherent ambiguities in medical image segmentation, such as variations in anatomical definitions and partial volumes. Traditional deep learning models often rely on point estimates, which may fail to capture the full spectrum of plausible segmentations. UQ techniques enable the inference of distributions over possible segmentations, providing a more comprehensive quantification of uncertainty—crucial in clinical settings where inter-rater variability and consensus-based decisions are common (Selvan et al., 2020).

- **Fault Diagnosis:** UQ is crucial for fault diagnosis because it provides a measure of confidence in the diagnostic outcomes, which is essential for making informed decisions in critical systems. UQ allows engineers and system operators to understand not just what the diagnosis indicates, but also how reliable that diagnosis is. This is particularly important in scenarios where false positives or negatives can lead to significant consequences, such as unnecessary maintenance actions or undetected failures (Mian et al., 2024).

- **Residential Load Forecasting:** Accurate residential load forecasting is critical for efficient energy management, yet traditional probabilistic methods often struggle with the inherent uncertainty in household electricity consumption patterns. UQ techniques provide a more complete probabilistic representation of future consumption, which is essential for informed decision-making in modern power systems (Afrasiabi et al., 2020).

- **Astronomy:** UQ is indispensable in astronomy, particularly for estimating distances to galaxies (photo-z estimation; see Chapter 7 for details) and performing likelihood-free cosmological inference (see Chapter 9). While traditional methods typically offer single-point estimates, UQ methods deliver a more nuanced

picture, enabling more accurate inferences in scenarios where multiple redshifts or cosmological parameter values align with the observed data.

- **Validating Soil Property Predictions:** When ML models are used as a replacement or supplement to traditional soil laboratory analysis, rigorous uncertainty assessment becomes indispensable. Without a clear understanding of the model's confidence in its predictions, it is difficult to gauge their reliability for critical applications such as digital soil mapping or soil carbon accounting (Bejani and Ghatee, 2021; England and Viscarra Rossel, 2018; Padarian et al., 2022; Searle et al., 2021). UQ also facilitates the integration of predictions from various models or sensors, as understanding the uncertainty associated with each prediction is crucial for appropriately weighting different sources (Horta et al., 2015).

- **Land-Use Land-Cover (LULC) Classification:** UQ enhances the reliability of LULC maps by identifying areas with high classification uncertainty, which is critical for accurate environmental and ecological analysis. This ensures that users are aware of the confidence levels in different regions of the map, improving the robustness of downstream applications (Valle et al., 2023).

- **Cancer Risk Assessment:** In the context of cancer risk assessment, uncertainty quantification helps in providing more accurate and personalized predictions, reducing the number of unnecessary biopsies while maintaining high sensitivity for cancer detection. This enhances decision-making in clinical settings by offering clearer insights into the confidence level of each diagnosis (Fröhlich et al., 2024).

- **Robotics Motion Planning:** UQ can be used in robotics for ensuring safe and effective motion planning, particularly when robots operate in unfamiliar environments. For instance, it can be used to enable robots to create prediction regions in real-time, ensuring that their movements remain safe even under changing conditions (Marques and Berenson, 2024).

- **Autonomous Vehicles:** Uncertainty quantification plays a critical role in the safety and reliability of autonomous vehicles by providing confidence measures for the vehicle's perception and decision-making systems. It helps in identifying scenarios where the model's predictions are uncertain, allowing the system to

take precautionary actions, such as slowing down or handing control back to a human driver (McAllister et al., 2017).

- **Predicting Infectious Disease Waves:** UQ plays a crucial role in predicting waves of infectious diseases, such as dengue, COVID-19, and Influenza. By quantifying uncertainty in real-time models, UQ enhances the reliability of forecasts, enabling better preparation and response to potential outbreaks (Codeço et al., 2016; Ray and Reich, 2018; Wu et al., 2020); see also Chapter 8.

- **Assessing Model Confidence:** UQ is critical in scenarios involving limited or out-of-domain data, where it helps to establish the confidence level of a model's predictions. This is particularly important in applications where model predictions influence critical decisions, ensuring more trustworthy outcomes (Padarian et al., 2022).

Some of these examples will be explored in greater depth throughout the book.

1.4 Outline of the Book

In this book, we explore various methods for estimating uncertainties in prediction problems. The structure of the book is as follows:

The first part covers the statistical foundations of uncertainty quantification in machine learning. It starts with Chapter 2, which reviews supervised learning for classification and regression. Chapter 3 then covers methods to estimate conditional densities, modeling the aleatoric uncertainty of the target Y given the input $\mathbf{x}$. Chapter 4 introduces tools to evaluate these estimates. Chapter 5 explains how to compute prediction bands for Y from estimated densities and discusses alternative methods based on conformal inference. Some of these take into account both the aleatoric and the epistemic uncertainties. Chapter 6 then provides an overview of techniques that directly model both the aleatoric and the epistemic uncertainty on Y.

The second part of the book contains applications of the tools introduced in previous chapters. Chapter 7 applies some of the UQ techniques to the photo-z problem from cosmology, while Chapter 8 applies them to dengue epidemic nowcasting. Chapter 9 applies the techniques to likelihood-free inference problems, where the goal is to infer parameters θ of a statistical model in which a tractable likelihood

function is unavailable. Finally, Chapter 10 demonstrates how to predict the duration of construction projects by incorporating weather-related delays using historical meteorological data.

Chapter 11 concludes the book by summarizing the key topics explored and briefly discussing further challenges in UQ.

Part I

Foundations of Uncertainty Quantification in ML

Chapter 2

Review of Supervised Learning

A Philosopher Lecturing on the Orrery. Joseph Wright of Derby, 1766, Derby Museum and Art Gallery, Derby.

> Forecasting is the art of saying what will happen and then explaining why it didn't.
>
> Unknown Author

The main objective of a supervised model is predict the value a random variable $Y \in \mathcal{Y}$ using information from a vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d := \mathcal{X}$. When Y is a quantitative variable, we have a problem of *regression*. For situations where Y is qualitative, we have a problem of *classification*. In order to achieve this goal, we assume we have access to a *labeled dataset*

$$(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$$

where both $\mathbf{x}$'s and y's are known.

2.1 Notation and Assumptions

> Independence is happiness.
>
> Susan B. Anthony

The variable Y is often called the response variable, dependent variable, or label. On the other hand, the observations contained in $\mathbf{x} = (x_1, \ldots, x_d)$ are generally referred to as explanatory variables, independent variables, features, attributes, predictors, or covariates.

Throughout the book, unless stated otherwise, we assume that the labeled daset is independent and identically distributed (i.i.d.) to $(\mathbf{X}, Y)$. See the notation used in Table 2.1.

Table 2.1: Notation used in the book.

Response	Covariates			
Y_1	$X_{1,1}$	$\ldots$	$X_{1,d}$	$(= \mathbf{X}_1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
Y_n	$X_{n,1}$	$\ldots$	$X_{n,d}$	$(= \mathbf{X}_n)$

The goal of a supervised model is to create a function

$$g : \mathbb{R}^d \longrightarrow \mathcal{Y}$$

that has good predictive power. That is, g must be such that, given *new* i.i.d. observations $(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$, we have

$$g(\mathbf{x}_{n+1}) \approx y_{n+1}, \ldots, g(\mathbf{x}_{n+m}) \approx y_{n+m}.$$

In a regression problem, we assume that $\mathcal{Y} \subset \mathbb{R}$, while in a classification problem, we assume (without loss of generalization) that $\mathcal{Y} = \{0, 1, \ldots, K-1\}$ for some $K > 1$. We often denote the feature space by $\mathcal{X}$.

2.2 Loss Functions and Risk

In supervised learning, constructing a good prediction function $g : \mathbb{R}^d \longrightarrow \mathcal{Y}$ requires a criterion to measure its performance. This is typically done using a *loss function*, which quantifies the error in a prediction and provides a way to guide the learning process.

A loss function is defined as

$$\begin{aligned} L &: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, \\ &(g(\mathbf{x}), y) \mapsto L(g(\mathbf{x}), y), \end{aligned}$$

where $g(\mathbf{x})$ is the predicted value and y is the true label. The loss function evaluates how well the prediction $g(\mathbf{x})$ matches the true outcome y. In regression tasks, a commonly used loss function is the quadratic loss:

$$L(g(\mathbf{x}), y) = (y - g(\mathbf{x}))^2, \tag{2.1}$$

which penalizes larger deviations between the prediction and the actual outcome more heavily.

For classification, the loss function $L(g(\mathbf{x}), y)$ is often designed so that $L(k, k) = 0$ for every $k \in \mathcal{Y}$. Here, $L(k, j)$ represents the penalty for predicting class j when the true class is k. A common choice is the zero-one loss, $L(k, j) = \mathbb{I}(k \neq j)$, which assigns an equal penalty to all misclassifications, even though this assumption may not be

appropriate in many real-world applications (Assunção et al., 2023).

The performance of a prediction function g is then assessed through the *risk*, which is the expected loss over new, unseen data:

$$R(g) = \mathbb{E}\left[L(g(\mathbf{X}), Y)\right],$$

where $(\mathbf{X}, Y)$ represents a random variable drawn from the same distribution as the training data. The risk measures how well the function g performs on average, with lower risk indicating better performance.

In classification problems, the risk under the zero-one loss is defined as

$$R(g) = \mathbb{P}\left(Y \neq g(\mathbf{X})\right), \tag{2.2}$$

which represents the probability that the classifier misclassifies a new sample. A more intuitive and commonly used measure is the classifier's *accuracy*, given by $1 - R(g) = \mathbb{P}(Y = g(\mathbf{X}))$. Minimizing the risk is equivalent to maximizing the accuracy. However, it is important to note some limitations of this metric when the goal is to quantify uncertainties (see Section 3.1.3).

2.2.1 Optimal Solutions for Different Loss Functions

Different choices of loss functions lead to different optimal prediction functions. For instance:

Theorem 1 (Regression function)**.** *In a regression problem, the function g that minimizes the risk $R(g) = \mathbb{E}\left[(Y - g(\mathbf{X}))^2\right]$ is the* regression function*, which is the conditional expectation:*

$$r(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}].$$

This result indicates that, under quadratic loss, the best prediction function is the one that outputs the average value of Y given $\mathbf{x}$. While $r(\mathbf{x})$ is generally unknown in practical scenarios, this insight suggests that regression problems can be tackled by estimating the regression function $r(\mathbf{x})$.

Theorem 2 (Bayes classifier)**.** *In classification, the function g that minimizes the risk is*

called the Bayes classifier. *It is defined as:*

$$g_{Bayes}(\mathbf{x}) := \arg\min_{j \in \mathcal{Y}} \sum_{k \in \mathcal{Y}} L(k,j)\mathbb{P}(Y = k|\mathbf{x}). \tag{2.3}$$

In the binary classification case (that is, $K = 2$) with $L(k,k) = 0$ for all $k \in \mathcal{Y}$, this reduces to the decision rule:

$$g_{Bayes}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbb{P}(Y = 1|\mathbf{x}) \geq \frac{L(0,1)}{L(0,1)+L(1,0)}, \\ 0, & \text{otherwise}. \end{cases} \tag{2.4}$$

Under the zero-one loss, this further simplifies to:

$$g_{Bayes}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbb{P}(Y = 1|\mathbf{x}) \geq \frac{1}{2}, \\ 0, & \text{otherwise}. \end{cases}$$

In the general classification case with a zero-one loss, the Bayes classifier becomes:

$$g_{Bayes}(\mathbf{x}) = \arg\max_{j \in \mathcal{Y}} \mathbb{P}(Y = j|\mathbf{x}).$$

As in the regression case, $\mathbb{P}(Y = k|\mathbf{x})$ cannot be computed in real-world problems. However, this theorem suggests that classification problems can be solved by estimating such probabilities.

The objective of supervised learning is, therefore, to develop methods that estimate these functions effectively, ensuring low risk and therefore reliable predictions.

2.3 Model Selection: Overfitting and Underfitting

> The best preparation for tomorrow is doing your best today.
>
> H. Jackson Brown Jr

In practical problems, it is common to fit multiple models to $g(\mathbf{x})$ and search for the one with the best predictive power, i.e., the one with the lowest risk. This is exemplified below in a regression context.

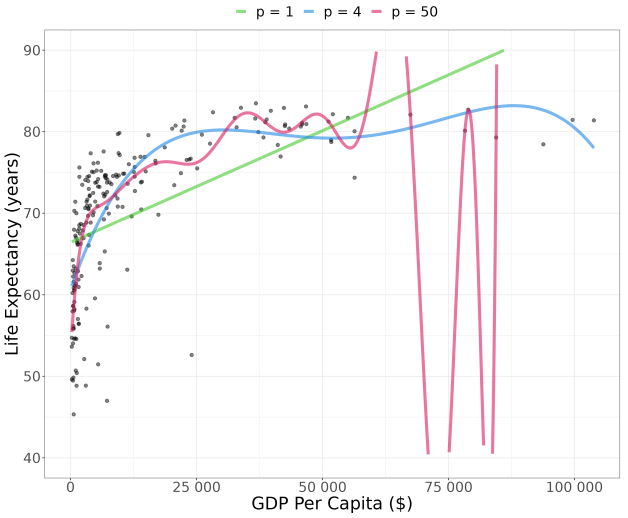


Figure 2.1: GPD per Capita and Life Expectancy in 211 countries and the fitted prediction models g. Each g is a polynomial with p degrees.

Exemple 2.1. [Life Expectancy and Per Capita GDP] Recall the example from Section 1.1. The challenge is to utilize this data to predict the life expectancy of countries where the GDP per Capita is known but life expectancy is not. To achieve this, one can estimate $\mathbb{E}[Y|\mathbf{x}]$, where Y is the life expectancy in a given country, and $\mathbf{x}$ is its GDP per Capita. The figure also shows the fitting of three different models for the prediction function:

$$g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i, \text{ for } p \in \{1, 4, 50\}.$$

In other words, we fit three polynomial regressions: one of degree 1, one of degree 4, and one of degree 50. The fits were performed (that is, β_i were estimated) using the least squares method (Section 2.6.1). While the degree-1 model is too simplistic for the data, the model with $p = 50$ (i.e., 50th-degree) is too complex and seems to provide a function g that will not make good predictions on new observations. The model with $p = 4$ seems to be the most reasonable in this case. We say that the model with $p = 1$ suffers from *underfitting*, while the model with $p = 50$ suffers from *overfitting* (it over-adapts to this specific sample but has low generalization power). The goal of this section is to describe methods for choosing the best predictive model

among those that were fitted, that it, the model with the highest predictive power. In this example, we want a method that automatically chooses $p = 4$.

□

The goal of a model selection method is to select a good function g. Therefore, we want to choose a function g from a class of candidates $\mathbb{G}$ that has good predictive power (low risk). Since $R(g)$ is unknown, it needs to be estimated to evaluate the function $g \in \mathbb{G}$. In the next section, we will present methods to estimate this risk.

2.3.1 Data Splitting and Cross-Validation

The *observed risk* (also known as the *training error*, or the *training risk*), defined by

$$\frac{1}{n}\sum_{i=1}^{n} L(g(\mathbf{X}_i), Y_i), \tag{2.5}$$

is a very optimistic estimator of the true risk. If used for model selection, it leads to overfitting, which is a perfect fit to the data. This happens because g was chosen to fit well $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$.

One way to solve this problem is to divide the dataset into two parts: *training* and *validation*:

$$\overbrace{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_s, Y_s)}^{\text{Training (e.g., 70\%)}}, \overbrace{(\mathbf{X}_{s+1}, Y_{s+1}), \ldots, (\mathbf{X}_n, Y_n)}^{\text{Validation (e.g., 30\%)}}.$$

We use the training set exclusively to estimate g (e.g., estimate the coefficients of the linear regression), and the validation set *only to estimate* $R(g)$ via

$$\hat{R}(g) = \frac{1}{n-s}\sum_{i=s+1}^{n} L(g(\mathbf{X}_i), Y_i) =: \widehat{R}(g), \tag{2.6}$$

i.e., we evaluate the error on the validation set.

A good practice for choosing which samples will be used for the training set and which will be used for the validation set is to do it **randomly**. Thus, a random number generator is used to choose which samples will be used for training and which will be used for validation. This procedure avoids problems when the dataset is previously

sorted according to some covariate (e.g., the person who collected the data may have ordered the observations according to some variable).

Since the validation set was not used to estimate the parameters of g, the estimator from Equation 2.6 is consistent by the law of large numbers.

For any loss function L, the risk can be approximated by

$$R(g) \approx \frac{1}{n-s} \sum_{i=s+1}^{n} L(g(\mathbf{X}_i); Y_i) := \widehat{R}(g).$$

The procedure of dividing the data into two parts and using one part to estimate the risk is called *data splitting*. A variation of this method is *cross-validation*, which uses the entire sample. For example, in *leave-one-out cross-validation* (LOOCV) (Stone, 1974), the estimator used is given by

$$\hat{R}(g) = \frac{1}{n} \sum_{i=1}^{n} L(g_{-i}(\mathbf{X}_i); Y_i),$$

where g_{-i} is fitted using all observations except the i-th one, i.e., using

$$(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{i-1}, Y_{i-1}), (\mathbf{X}_{i+1}, Y_{i+1}), \ldots, (\mathbf{X}_n, Y_n).$$

Alternatively, we can use *k-fold cross-validation*. In this approach, we randomly divide the data into k disjoint *folds* of approximately the same size. Let $L_1, \ldots, L_k \subset \{1, \ldots, n\}$ be the indices associated with each of the folds. The idea of *k-fold cross-validation* is to create k estimators of the regression function, $\hat{g}_{-1}, \ldots, \hat{g}_{-k}$, where $\hat{g}_{-j}$ is created using all observations in the dataset except those in fold L_j. The estimator of the risk is given by

$$\hat{R}(g) = \frac{1}{n} \sum_{j=1}^{k} \sum_{i \in L_j} L(g_{-j}(\mathbf{X}_i); Y_i).$$

Note that when $k = n$, we get LOOCV. For more details, see Wasserman (2006).

Training vs. Validation vs. Test. Just like the error on the training set is very optimistic because each $g \in \mathbb{G}$ is chosen to (approximately) minimize it, the error on the validation set *evaluated on the g with the lowest error on the validation set* is also optimistic, especially if many prediction methods are evaluated on the validation set.

This can lead to overfitting to the validation set, i.e., it may lead to choosing a function g that performs well on the validation set but not on new observations. One way to circumvent this problem is to split the original dataset into three parts: training, validation, and test. The training and validation sets are used as described above. The test set is used to estimate the error of the best prediction function determined using the validation set. This way, we can know if the risk of the best g found is indeed small.

Several variations of these ideas can be used. For example, suppose we want to compare a polynomial regression with a lasso. A reasonable procedure is: (i) choose the best polynomial degree and the lasso *tuning parameter* λ using *data splitting* within the training set (i.e., dividing the training set into two parts: training and validation), and (ii) compare the performance of the best polynomial regression with the best lasso (according to criterion (i)) on the test set. This way, the test will be used to compare only two models. Alternatively, we can replace (i) with cross-validation within the training set. In general, ideally, we use one of the splits (the validation) to choose the *tuning parameters* of each method, and the other (the test) to compare the best version of each method.

Confidence Intervals for Risk. Using the test set, we can also create a confidence interval for the risk. Let $(\tilde{\mathbf{X}}_1, \tilde{Y}_1), \ldots, (\tilde{\mathbf{X}}_m, \tilde{Y}_m)$ be the elements of the test set (i.e., they were not used for training or validation of the model). An unbiased estimator for the risk of g (which was estimated based on the training and validation sets) is given by

$$\widehat{R}(g) = \frac{1}{m} \sum_{k=1}^{m} \underbrace{L(g(\tilde{\mathbf{X}}_k); \tilde{Y}_k)}_{W_k}.$$

Since $\widehat{R}(g)$ is an average of i.i.d. random variables, the Central Limit Theorem implies that

$$\widehat{R}(g) \approx \text{Normal}\left(R(g), \frac{1}{m}\mathbb{V}[W_1]\right).$$

Since $W_1, \ldots, W_m$ are i.i.d., we can estimate $\mathbb{V}[W_1]$ with

$$\widehat{\sigma^2} = \frac{1}{m} \sum_{k=1}^{m} \left(W_k - \overline{W}\right)^2,$$

where $\overline{W} = \frac{1}{m}\sum_{k=1}^{m} W_k$. Thus, an approximate 95% confidence interval for $R(g)$ is

given by

$$\widehat{R}(g) \pm 2\sqrt{\frac{1}{m}\widehat{\sigma^2}}. \tag{2.7}$$

Equation 2.7 also provides insight into how to choose the size of the division between training and validation. We can choose the smallest m such that the size of the confidence interval for the risk is as small as we want. This idea is especially interesting when the sample size is large. In this case, the size of the validation set can be much smaller than the training set, since estimating the risk is a much easier task than creating a prediction function.

2.4 Bias and Variance Tradeoff

> Simplicity is an exact medium
> between too little and too much.
>
> Sir Joshua Reynolds

The quadratic loss function of Equation 2.1 is such that its risk can be decomposed as

$$R(g) = \int \mathbb{V}[Y|\mathbf{X}=\mathbf{x}]d\mathbb{P}(\mathbf{x}) + \int \left(r(\mathbf{x}) - \mathbb{E}[g(\mathbf{X})|\mathbf{X}=\mathbf{x}]\right)^2 d\mathbb{P}(\mathbf{x}) + \int \mathbb{V}[g(\mathbf{X})|\mathbf{X}=\mathbf{x}]d\mathbb{P}(\mathbf{x}).$$

This decomposition has the following three key components:

- $\mathbb{V}[Y|\mathbf{X}=\mathbf{x}]$, which is the *intrinsic variance* of the response variable, which does not depend on the chosen function g and therefore cannot be reduced;
- $\left(r(\mathbf{x}) - \mathbb{E}[g(\mathbf{X})|\mathbf{X}=\mathbf{x}]\right)^2$ is the square of the estimator $\widehat{g}$ bias, and
- $\mathbb{V}[g(\mathbf{X})|\mathbf{X}=\mathbf{x}]$ is the variance of the estimator $\widehat{g}$.

The values of the last two items can be reduced if we choose an appropriate g. Roughly speaking, complex models (e.g. models with many parameters) have relatively low bias but high variance since they are hard to estimate. In contrast, models

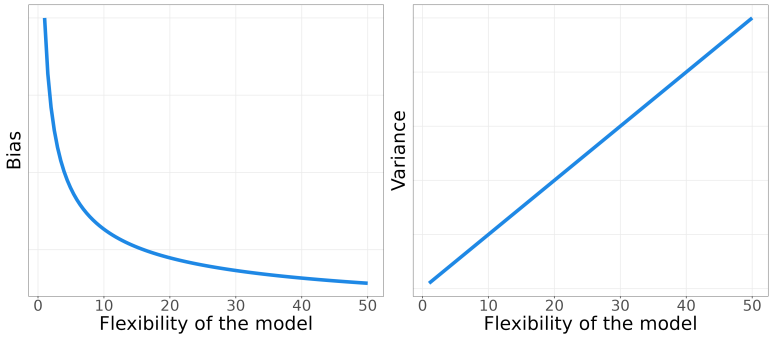


Figure 2.2: Typical behaviour of the bias and the variance of a model as a function of its complexity.

that are less flexible have low variance but very high bias as they are too simplistic to describe the data-generating model. Therefore, to achieve good predictive power, one should choose the model that has the right amount of complexity. Figure 2.2 qualitatively illustrates the trade-off between bias and variance.

Exemple 2.2. We revisit Example 2.1. Here our goal is to select the best estimator within the class

$$\mathbb{G} = \left\{ g(x) : g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i, \text{ for } p \in \{1, 2, \ldots, 50\} \right\}.$$

Figure 2.3 (left panel) displays the quadratic error, estimated via cross validation, for each $g \in \mathbb{G}$. As the risks for $p > 11$ are excessively high, we only show the behavior for $p \leq 11$. Figure 2.1 (right panel) demonstrates that, indeed, minimizing the error on the validation set leads to a good fit.

□

The bias-variance tradeoff also occurs for other loss functions. However, the details are somewhat different. Suppose $\mathbb{G}$ is a class of prediction functions (for example, all classifiers based on logistic regression),

$$R_{or} := \inf_{g \in \mathbb{G}} R(g)$$

is the best risk (that is, the oracle risk) that can be achieved using prediction functions

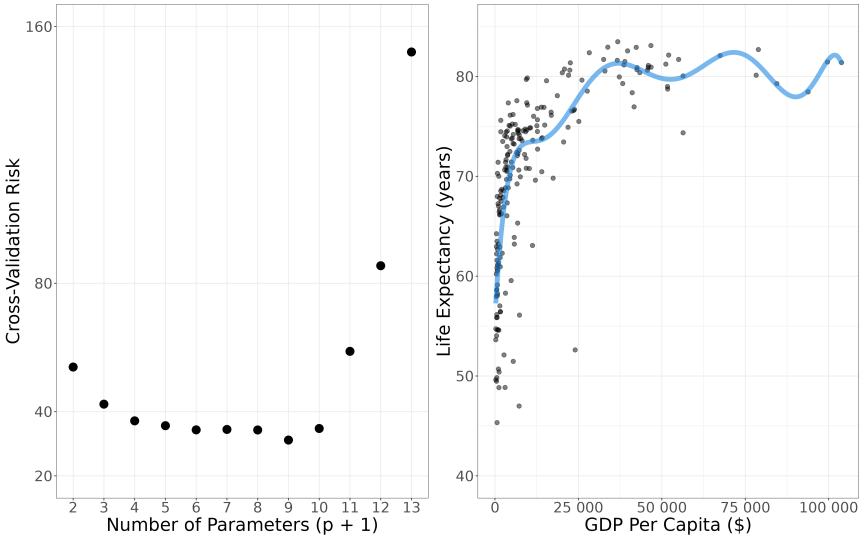


Figure 2.3: GPD per Capita and Life Expectancy in 211 countries and the fitted prediction models g. Each g is a polynomial with p degrees.

from $\mathbb{G}$, and

$$R_* := \inf_{g:\mathcal{X}\longrightarrow\mathcal{Y}} R(g)$$

is the risk of the best prediction function among all prediction functions. Then, for all $g \in \mathbb{G}$,

$$R(g) = R_* + T_1 + T_2,$$

where R_* is the risk of the best prediction function (intrinsic variance), $T_1 = R_{or} - R_*$ is the approximation error (a general version of the bias), and $T_2 = R(g) - R_{or}$ is the estimation error (a general version of the variance). Thus, $\mathbb{G}$ cannot be too large or too small.

2.5 Tuning Parameters

The role of the parameter p (degree of the polynomial) in Example 2.1 is to control the trade-off between bias and variance. The optimal value of p depends on n and $r(\mathbf{x})$. The parameter p is called a *tuning parameter*.

Several regression methods have one or more *tuning parameters*. In this book, we

will always choose them through cross-validation or *data splitting*, although these are not the only ways to do this (see, for example, Wasserman 2006).

2.6 Methods to create prediction functions

2.6.1 Parametric Methods

> Nature is pleased with simplicity.
> And nature is no dummy.
>
> Sir Isaac Newton

> The penalty may be removed, the crime is eternal.
>
> Ovid

One way to create a prediction function g is to (i) parameterize it (that is, assume it belongs to a class of functions that can be described by a finite number of parameters), (ii) find parameter values for the parameters that lead to a function that has good fit. In practice, step (ii) is done by designing and optimizing a loss function.

Some examples of methods that fall into this framework are:

- **[Linear regression via least squares]** Linear regression uses a linear form for the prediction function, meaning that the prediction function used can be written as

$$g(\mathbf{x}) = \boldsymbol{\beta}^\intercal \mathbf{x} = \beta_0 x_0 + \beta_1 x_1 + \ldots + \beta_d x_d, \tag{2.8}$$

  where we adopt the convention $x_0 \equiv 1$, and where $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_d)$. Note that x_i is not necessarily the i-th original variable; we can create new covariates that are functions of the original ones (e.g., $x_i^2, x_i x_j$, etc.; see Example 2.1). One way to estimate the coefficients $\boldsymbol{\beta}$ of linear regression is by using the method of least squares, which consists of finding the solution to

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_{i,1} - \ldots - \beta_d x_{i,d})^2, \tag{2.9}$$

which mimics the squared loss function. The prediction function is then given by $g(\mathbf{x}) = \widehat{\boldsymbol{\beta}}^{\intercal}\mathbf{x}$.

- **[Lasso]** The lasso (Tibshirani, 1996) also assumes a linear form for the regression function. However, it instead minimizes the function

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 x_{i,1} - \ldots - \beta_d x_{i,d})^2 + \lambda\sum_{j=1}^{d}|\beta_j|, \tag{2.10}$$

where λ is a tuning parameter. The L_1 penalty $\sum_{j=1}^{d}|\beta_j|$ decreases the variance of the least-squares solution, thus leading to a smaller risk if there is not much increase in the bias. This is the case when several covariates are not useful to predict Y and therefore have small coefficients β_j. The solution induced by Equation 2.10 has many zeros (that is, the vector $\widehat{\boldsymbol{\beta}}$ is sparse). Therefore, the resulting model is also easy to interpret.

- **[Logistic Regression]** In a binary classification problem (that is, $K = 2$), a logistic regression assumes that

$$\mathbb{P}(Y = 1|\mathbf{x}) = \frac{e^{\beta_0 + \sum_{i=1}^{d}\beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{d}\beta_i x_i}}.$$

The coefficients $\boldsymbol{\beta}$ are often estimated by maximizing the log-likelihood function

$$\sum_{k=1}^{n}\log\left[\left(\frac{e^{\beta_0 + \sum_{i=1}^{d}\beta_i x_{k,i}}}{1 + e^{\beta_0 + \sum_{i=1}^{d}\beta_i x_{k,i}}}\right)^{y_k}\left(\frac{1}{1 + e^{\beta_0 + \sum_{i=1}^{d}\beta_i x_{k,i}}}\right)^{1-y_k}\right].$$

This corresponds to minimizing the cross-entropy between the vector $(y_1, \ldots, y_n)$ and the estimated probabilities. Similarly to the lasso, a penalty that decreases the variance of this estimator may be introduced.

Once the probabilities are estimated, a classifier can be chosen via

$$g(\mathbf{x}) = \mathbb{I}\left(\widehat{\mathbb{P}}(Y = 1|\mathbf{x}) > K\right),$$

which mimics the Bayes classifier (Theorem 2) and K is given by the loss function. Alternatively, other metrics for classification can be used to choose K, such as the sensitivity and the specificity.

Although a parametric model may be incorrect in the sense that the best prediction function does not precisely align with the specified parametric shape, it can still yield good predictions. This is primarily because the parametric assumption substantially reduces the estimation error (as discussed in Section 2.4) of the model.

2.6.2 K-Nearest Neighbors

> We're neighbors and we're going to pull together.
>
> Rick Perry

The k-nearest neighbors (KNN) (Benedetti, 1977; Stone, 1977) is a nonparametric approach to create prediction functions. As the name implies, KNN constructs a prediction function, denoted as $g(\mathbf{x})$, by considering the labels of the closest neighbors to a given point $\mathbf{x}$ within the training dataset. Specifically, in the context of regression, the KNN estimate for the regression function is defined as follows:

$$g(\mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_{\mathbf{x}}} y_i, \tag{2.11}$$

where $\mathcal{N}_{\mathbf{x}}$ is the set of the k observations closest to $\mathbf{x}$, that is,

$$\mathcal{N}_{\mathbf{x}} = \left\{ i \in \{1, \ldots, n\} : d(\mathbf{x}_i, \mathbf{x}) \leq d_{\mathbf{x}}^k \right\},$$

and $d_{\mathbf{x}}^k$ is the distance from the k-th nearest neighbor of $\mathbf{x}$ to $\mathbf{x}$. In other words, the regression function evaluated at $\mathbf{x}$ is estimated using a local average of the responses of the k nearest neighbors to $\mathbf{x}$ in the covariate space.

The tuning parameter k can be chosen through cross-validation. A high value of k leads to a very simple model (approaching a constant as $k \longrightarrow \infty$), resulting in high bias but low variance. Conversely, a low value of k leads to an estimator with high variance but low bias.

In a classification context, Equation 2.11 can be replaced by a majority voting mechanism. Specifically, the predicted class label $\hat{y}(\mathbf{x})$ for a given point $\mathbf{x}$ is determined by the most frequent class among its k nearest neighbors. Formally, this can

be expressed as:

$$g(\mathbf{x}) = \arg\max_{c \in \mathcal{Y}} \sum_{i \in \mathcal{N}_{\mathbf{x}}} \mathbb{I}(y_i = c).$$

This approach essentially assigns the class that appears most frequently among the k nearest neighbors.

Alternatively, probabilities of each class can be estimated by calculating the proportion of neighbors that belong to each class. This probabilistic version of KNN assigns a probability to each class label based on the relative frequency of that label among the k nearest neighbors. Formally, the estimated probability that a point $\mathbf{x}$ belongs to class c is given by:

$$\hat{\mathbb{P}}(Y = c|\mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_{\mathbf{x}}} \mathbb{I}(y_i = c).$$

2.6.3 Trees

Trees consist of a non-parametric methodology that leads to highly interpretable results. A tree is constructed through recursive partitioning in the covariate space. Each partition is called a node, and each final result is called a leaf; see Figure 2.4 for an example.

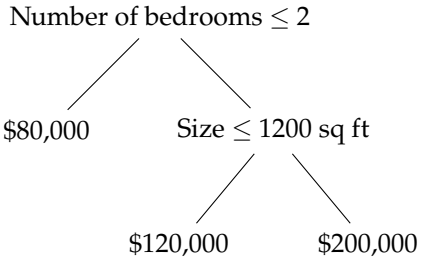


Figure 2.4: Regression tree that predicts the price of an apartment.

The use of the tree to predict a new observation is done as follows: starting from the top, we check if the condition described at the top (first node) is satisfied. If it is, we go left. Otherwise, we go right. This process continues until we reach a leaf.

Formally, a tree creates a partition of the covariate space into distinct and disjoint regions: $R_1, \ldots, R_j$. In a regression context, the prediction for the response Y of an

observation with covariates $\mathbf{x}$ that are in R_k is given by

$$g(\mathbf{x}) = \frac{1}{|\{i : \mathbf{x}_i \in R_k\}|} \sum_{i:\mathbf{x}_i \in R_k} y_i. \tag{2.12}$$

That is, to predict the value of the response for $\mathbf{x}$, we observe the region to which observation $\mathbf{x}$ belongs and then calculate the average of the response variable values for the training set samples belonging to that same region. In a classification context, this aggregation can be performed by using the mode of the Y's that fall in R_k for example.

The creation of the structure of a tree is done through two major steps: (i) creating a complete and complex tree and (ii) pruning that tree to avoid overfitting. In step (i), the goal is to create a tree that leads to "pure" partitions, meaning partitions in which the values of Y in the training set observations in each leaf are homogeneous. To achieve this, one first defines a measure of homogeneity of the leaves of a tree T. In regression, it is common to use the mean squared error,

$$\mathcal{P}(T) = \sum_{R} \sum_{i:\mathbf{x}_i \in R} \frac{(y_i - \widehat{y}_R)^2}{n},$$

where $\widehat{y}_R$ is the predicted value for the response of an observation belonging to region R. In classification, one often uses the Gini index:

$$\sum_{R} \sum_{c \in \mathcal{Y}} \widehat{p}_{R,c}(1 - \widehat{p}_{R,c}),$$

where $\widehat{p}_{R,c}$ represents the proportion of observations classified as belonging to category c among those that fall within region R.

Finding the optimal value of T to minimize $\mathcal{P}(T)$ is practically impossible due to its computational complexity. Therefore, we employ a heuristic approach to construct a tree with high homogeneity. This approach involves creating recursive binary splits, as illustrated in Figure 2.5. Initially, the algorithm divides the covariate space into two distinct regions, as depicted in Figure 2.5 (a). To determine the first partition (R_1, R_2), we systematically explore all possible combinations of covariates x_i and cut points t_1. We select the combination that results in the greatest homogeneity. We then define:

$$R_1 = \{\mathbf{x} : x_i < t_1\} \text{ and } R_2 = \{\mathbf{x} : x_i \geq t_1\}.$$

Once we establish these regions, the root node of the tree is determined. In the subsequent step, we aim to further partition either R_1 or R_2, as shown in Figure 2.5 (b). To decide on the new split, we employ the same strategy: we explore all combinations of covariates x_i and cut points t_2 to find the one that maximizes homogeneity. Importantly, we must now decide which region to split, either R_1 or R_2. Suppose we choose to partition R_1, using covariate x_j and cut point t_2. This partition results in $\{R_{1,1}, R_{1,2}\}$, as depicted in Figure 2.5 (b). Consequently, we define:

$$R_{1,1} = \{\mathbf{x} : x_i < t_1, x_j < t_2\},\ R_{1,2} = \{\mathbf{x} : x_i < t_1, x_j \geq t_2\}, \text{ and } R_2 = \{\mathbf{x} : x_i \geq t_1\}.$$

This recursive procedure continues, as shown in Figures 2.5 (c) and (d), until we obtain a tree with only a small predetermined number of observations in each leaf node. For example, we may halt the process when all leaf nodes contain fewer than five observations.

The tree created using this process produces good results for the training set, but overfitting is highly likely. This leads to poor predictive performance on new observations. Therefore, we proceed to step (ii), which is called pruning. The goal of this step is to make the regression tree smaller and less complex to reduce the variance of this estimator. In this step of the process, each node is removed one by one, and the prediction error on the validation set is observed. Based on this, decisions are made about which nodes will remain in the tree.

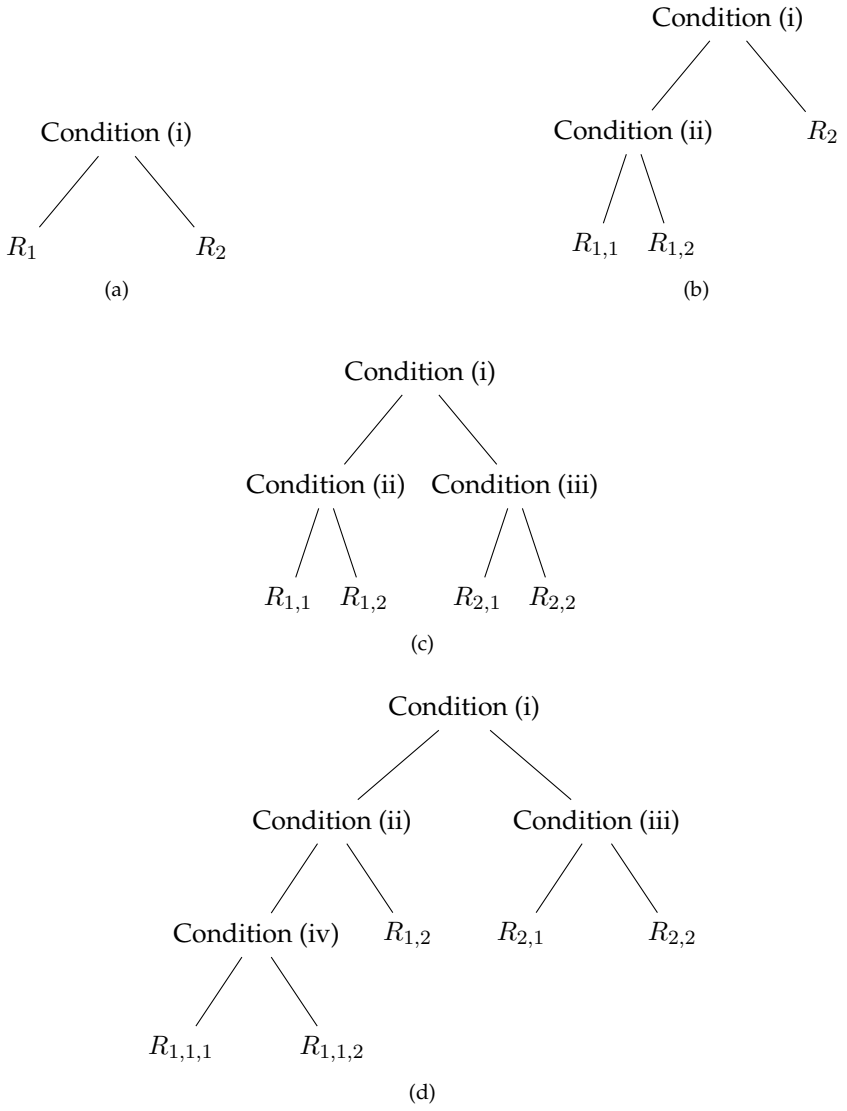


Figure 2.5: Recursively growing a tree.

2.6.4 Bagging and Random Forests

An interesting feature of regression trees is that they are highly interpretable. However, they tend to have low predictive power when compared to other estima-

tors. Bagging and random forests (Breiman, 2001b) are methods that overcome this limitation by combining multiple trees to make a prediction for the same problem.

To motivate this approach, imagine that in a regression context, we have two prediction functions for Y, $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$. The risks of these (conditional on $\mathbf{x}$) are given, respectively, by

$$\mathbb{E}\left[(Y - g_1(\mathbf{x}))^2|\mathbf{x}\right] \text{ and } \mathbb{E}\left[(Y - g_2(\mathbf{x}))^2|\mathbf{x}\right].$$

Now, consider the combined estimator $g(\mathbf{x}) = (g_1(\mathbf{x}) + g_2(\mathbf{x}))/2$. We have

$$\begin{aligned}
&\mathbb{E}\left[(Y - g(\mathbf{x}))^2|\mathbf{x}\right] = \\
&= \mathbb{V}[Y|\mathbf{x}] + \frac{1}{4}\left(\mathbb{V}[g_1(\mathbf{x}) + g_2(\mathbf{x})|\mathbf{x}]\right) + \left(\mathbb{E}[Y|\mathbf{x}] - \frac{\mathbb{E}[g_1(\mathbf{x})|\mathbf{x}] + \mathbb{E}[g_2(\mathbf{x})|\mathbf{x}]}{2}\right)^2.
\end{aligned}$$

Therefore, if $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ are uncorrelated ($\text{Cor}(g_1(\mathbf{x}), g_2(\mathbf{x})|\mathbf{x}) = 0$), unbiased ($\mathbb{E}[g_1(\mathbf{x})|\mathbf{x}] = \mathbb{E}[g_2(\mathbf{x})|\mathbf{x}] = r(\mathbf{x})$), and have the same variance ($\mathbb{V}[g_1(\mathbf{x})|\mathbf{x}] = \mathbb{V}[g_2(\mathbf{x})|\mathbf{x}]$), then

$$\mathbb{E}\left[(Y - g(\mathbf{x}))^2|\mathbf{x}\right] = \mathbb{V}[Y|\mathbf{x}] + \frac{1}{2}\mathbb{V}[g_i(\mathbf{x})|\mathbf{x}] \leq \mathbb{E}\left[(Y - g_i(\mathbf{x}))^2|\mathbf{x}\right] \tag{2.13}$$

for $i = 1, 2$. Therefore, it is better to use the combined estimator $g(\mathbf{x})$ than to use $g_1(\mathbf{x})$ or $g_2(\mathbf{x})$ separately. Although we consider only two estimators of the regression function in this example, the conclusion remains valid when combining any number B of estimators.

Building on this, bagging, which stands for "bootstrap aggregating," works by creating B distinct trees using B bootstrap samples from the original data[1]. Each tree is built using the steps from Section 2.6.3, but without pruning, which allows the trees to be grown fully. After constructing the trees, bagging combines their predictions to create a more stable estimator. In regression, the aggregated prediction function is given by:

$$g(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} g_b(\mathbf{x}),$$

where $g_b(\mathbf{x})$ is the prediction from the b-th tree. In classification, the final prediction can be obtained by taking the majority vote among the trees. B

[1]A bootstrap sample is a random sample with replacement of the same size as the original sample.

Random forests enhance the concept of bagging by introducing additional randomness into the tree-building process. Beyond using bootstrap samples, random forests randomly select a subset of features at each node split. This reduces the correlation between trees, leading to more accurate predictions compared to bagging alone. Specifically, rather than considering all d covariates at each node, a random subset of $m < d$ covariates is chosen at every split. These m covariates are randomly drawn from the original set, and a new subset is selected for each node as the tree grows. In classification tasks, it is common to select $\sqrt{p}$ features from a total of p features at each split, while in regression tasks, typically $p/3$ features are used.

2.6.5 Boosting

Just like random forests and bagging, boosting also involves the aggregation of different prediction functions. However, this combination is done differently. There are various variations and implementations of boosting. We will address one approach here, in the regression context.

In boosting, the estimator $g(\mathbf{x})$ is constructed incrementally. Initially, we assign the value of $g(\mathbf{x}) \equiv 0$. This estimator has high bias but low variance (specifically, zero). At each step, we update the value of g in order to decrease the bias and increase the variance of the new function. This is done by adding a function that predicts the residuals $r_i = Y_i - g(\mathbf{x}_i)$. One way to do this is by using a regression tree. It's important for this tree to have shallow depth to avoid overfitting. Additionally, instead of simply adding this function outright, we add it multiplied by λ (called the "learning rate"), a factor between 0 and 1 that is intended to prevent overfitting. Formally, this version of boosting follows the algorithm below:

1. We define $g(\mathbf{x}) \equiv 0$ and $r_i = y_i \; \forall i = 1, \ldots, n$.
2. For $b = 1, \ldots, B$:
   (a) We fit a tree with p leaves to $(\mathbf{x}_1, r_1), \ldots, (\mathbf{x}_n, r_n)$. Let $g^b(\mathbf{x})$ be its respective prediction function.
   (b) We update g and the residuals: $g(\mathbf{x}) \leftarrow g(\mathbf{x}) + \lambda g^b(\mathbf{x})$ and $r_i \leftarrow Y_i - g(\mathbf{x})$.
3. We return the final model $g(\mathbf{x})$.

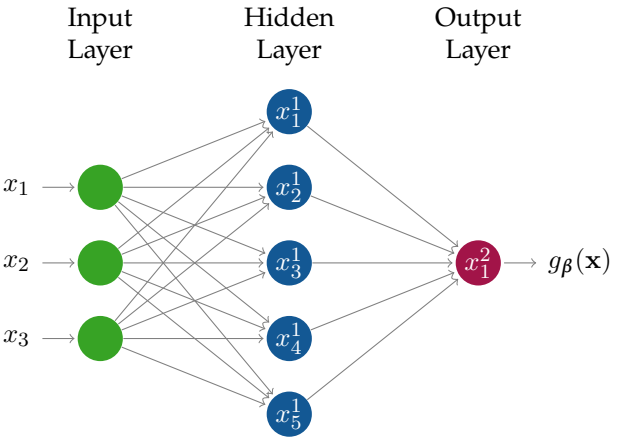


Figure 2.6: Example of a neural network with one hidden layer.

Thus, the tuning parameters for this version of boosting are B, p, and λ. Contrary to random forest, large values of B typically lead to overfitting. Thus, B is typically chosen by data splitting.

Modern implementations of boosting, such as XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018), have significantly advanced the technique by introducing optimizations like gradient-based learning, regularization, parallel processing, and efficient handling of categorical variables. These improvements make boosting more accurate and scalable.

2.6.6 Neural Networks

> With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.
>
> John von Neumann

Artificial neural networks are a quite old concept in artificial intelligence (McCulloch and Pitts, 1943; Rosenblatt, 1958). Mathematically, in the context of regression, they are a non-linear estimator of $r(\mathbf{x})$ that can be graphically represented by a structure like the one in Figure 2.6.

The nodes on the left side of the figure represent the inputs of the network, which

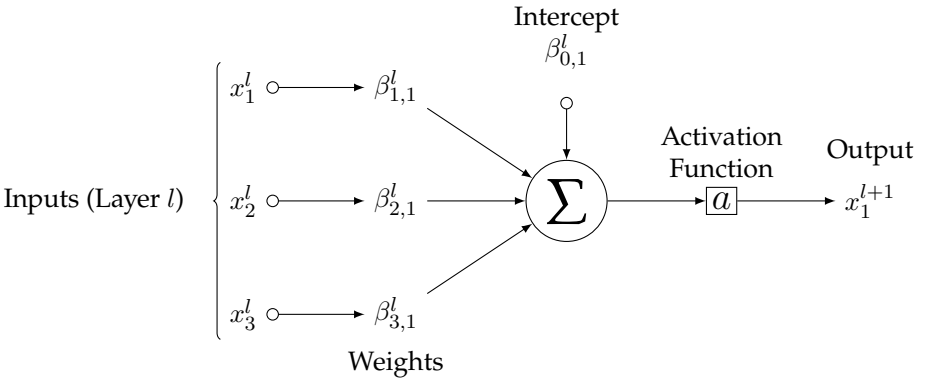


Figure 2.7: Example of the processing that occurs within a neuron located in layer $l+1$. The output x_1^{l+1} is given by $a\left(\beta^l_{0,j} + \sum_{i=1}^3 \beta^l_{i,1} x^l_i\right)$.

are the covariates of the dataset (e, there are three of them). The nodes in the second layer are called nodes in the hidden layer of the network. Each arrow represents a weight (parameter) β. Each node in this layer represents a transformation of the nodes from the previous layer.

For this network, if $\mathbf{x} = (x_1, x_2, x_3)$ is the input vector, then a given neuron j in the hidden layer is computed as

$$x^1_j := a\left(\beta^0_{0,j} + \sum_{i=1}^{3} \beta^0_{i,j} x^0_i\right) \text{ where } x^0_i = x_i \text{ for } i = 1,2,3,$$

where a is a user-defined function called the *activation function*. The superscript in this equation denotes the layer of the network. Once the values of x^1_j for all neurons j in the hidden layer are calculated, the output of the model can be computed. For this example, the output is given by

$$x^2_1 := a\left(\beta^1_{0,j} + \sum_{i=1}^{5} \beta^1_{i,1} x^1_i\right) =: g_{\boldsymbol{\beta}}(\mathbf{x}).$$

This procedure is represented in more detail in Figure 2.7.

Some common choices for the activation function are:

- **Identity:** $a(z) = z$

- **Logistic:** $a(z) = \frac{1}{1+\mathbf{e}^{-z}}$
- **Hyperbolic Tangent:** $a(z) = \frac{\mathbf{e}^{z}-\mathbf{e}^{-}z}{\mathbf{e}^{z}+\mathbf{e}^{-z}}$
- **ReLU (Rectified Linear Unit):** $a(z) = \max\{0, z\}$
- **Leaky ReLU:** $a(z) = \begin{cases} 0.01z & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$

In general, a neural network can have multiple hidden layers and a different number of neurons in each layer. These are choices made by the user. Figure 2.8 presents a schematic of a neural network with four hidden layers, each with six neurons.

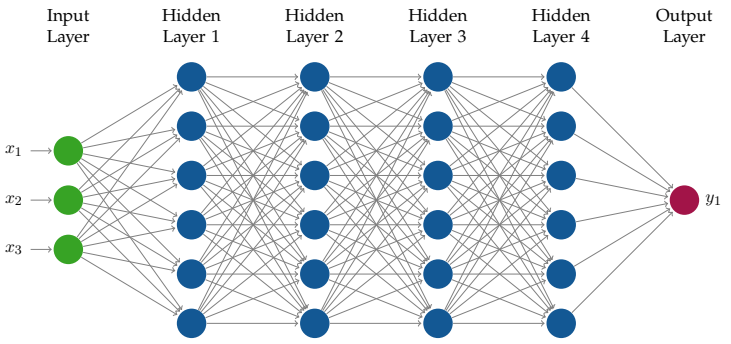


Figure 2.8: Example of a feed-forward neural network.

The propagation of input covariates in the neural network is done sequentially through these layers. Consider a network with H hidden layers, each with d_h neurons ($h = 1, \ldots, H$), and let $\beta_{i,j}^{l}$ be the weight assigned to the connection between input i in layer l and output j in layer $l+1$, $l = 0, \ldots, H$. Here, $h = 0$ denotes the input layer, and $H+1$ denotes the output layer. The estimator of an artificial neural network for the regression function takes the form

$$g(\mathbf{x}) = x_1^{H+1} = a\left(\beta_{0,1}^{H} + \sum_{i=1}^{d_H} \beta_{i,1}^{H} x_i^{H}\right) \tag{2.14}$$

where, for all $l = 1, \ldots, H$ and $j = 1, \ldots, d_{l+1}$,

$$x_j^{l+1} = a\left(\beta_{0,j}^l + \sum_{i=1}^{d_l} \beta_{i,j}^l x_i^l\right).$$

Notice that if $a(z) = z$ and there are no hidden layers, a neural network represents a regular linear regression. Indeed, as a linear regression, a neural network also parametrizes g. However, such parametrization is typically more complex, and the architecture of the network (and therefore its parametrization) is usually taylored for each dataset.

Another way to represent a neural network is through matrix notation. For $l = 0, \ldots, H$, let

$$\mathbb{H}_l = \begin{pmatrix} \beta_{0,1}^l & \beta_{1,1}^l & \cdots & \beta_{d_l,1}^l \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{0,d_{l+1}}^l & \beta_{1,d_{l+1}}^l & \cdots & \beta_{d_l,d_{l+1}}^l \end{pmatrix}.$$

In this case, $d_0 = d$ is the number of neurons in the input layer, and d_{H+1} is the number of neurons in the output layer. The estimator of Equation 2.14 can be written as the following composition of functions:

$$g(\mathbf{x}) = a \circ \mathbb{H}_H \ldots \tilde{a} \circ \mathbb{H}_1 \cdot \tilde{a} \circ \mathbb{H}_0 \cdot \tilde{\mathbf{x}},$$

where $\tilde{\mathbf{x}} = (1, \mathbf{x})$ and $\tilde{a}(\mathbf{y}) = (1, a(\mathbf{y}))$.

The structure of neural networks presented here can be highly generalized. For instance, each function a applied at each layer can be different. Moreover, the network's output doesn't need to be a single real number. Neural networks also allow for more complex structures like feedback (outputs from neurons that feed back into inputs of earlier layers) and others.

Indeed, the output of a classification network usually consists in k values,

$$g_0(\mathbf{x}), \ldots, g_{k-1}(\mathbf{x})$$

which represent estimates of the probabilities $\mathbb{P}(Y = i|\mathbf{x})$, $i = 0, \ldots, k-1$. Figure 2.9 illustrates a typical architecture.

In the context of classification, the common practice involves employing the *softmax* function as the activation function for the final layer. This function is mathemat-

ically represented as follows:

$$a_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{|\mathcal{Y}|} e^{z_j}},$$

which is essentially the logistic function. Here, $\mathbf{z}$ denotes the vector encompassing all input values within the last layer, and a_i denotes the activation function applied to the i-th neuron in the output layer. The primary purpose of this activation function is to ensure that the output values fall within the range of zero to one, while also guaranteeing that the entire vector sums up to one.

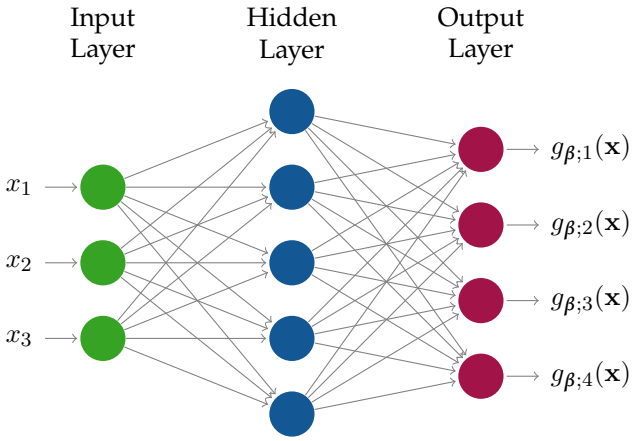


Figure 2.9: Example of a classification neural network with one hidden layer in the case where Y takes four possible values.

In order to estimate the parameters of a neural network, one first needs to define a differentiable loss function. In regression, one often uses the mean squared error

$$\text{MSE}(g_{\boldsymbol{\beta}}) = \frac{1}{n}\sum_{i=1}^{n}\left(g_{\boldsymbol{\beta}}(\mathbf{x}_i) - y_i\right)^2,$$

where $\boldsymbol{\beta}$ denotes all parameters from the network (the collection of all matrices $\mathbb{H}_0, \ldots, \mathbb{H}_H$). In classification, it is usual to use the cross-entropy

$$\text{CE}(g_{\boldsymbol{\beta},0}, \ldots, g_{\boldsymbol{\beta},k-1}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=0}^{k-i}\mathbb{I}(y_i = j)\log(g_{\boldsymbol{\beta};j}(\mathbf{x}_i)).$$

Once the loss function is defined, the next step is to minimize it with respect to the parameters β of the neural network. This is typically done using gradient-based optimization methods, such as gradient descent or its variants (e.g., stochastic gradient descent, Adam). The gradient of the loss function with respect to the parameters β is computed using backpropagation, which efficiently applies the chain rule to propagate the error backward through the network layers.

The optimization process iteratively updates the parameters β in the direction that decreases the loss function, aiming to reach a minimum. However, reaching the true global minimum of the loss function may not be desirable. For instance, consider the behavior of the 50-degree polynomial fit used to minimize the MSE in Example 2.1. Such a fit, while minimizing the training error, can lead to overfitting. Therefore, in practice regularization needs to be performed to ensure that the resulting estimated function is smooth and generalizes effectively to new, unseen data.

Early stopping is one such technique used to prevent overfitting. It halts the training process before the model fully converges to the minimum of the loss function, stopping when the validation loss stops improving, which signals the onset of overfitting. The best model parameters (those that resulted in the lowest validation loss) are then used as the final model.

In addition to early stopping, the choice of optimization method plays a crucial role in preventing poor generalization. One of the most commonly used methods is stochastic gradient descent (SGD). Unlike batch gradient descent, which computes gradients using the entire training set, SGD updates the model parameters based on gradients computed from a single training sample point or a small mini-batch. This introduces noise into the optimization process, which prevents the model from settling into sharp minima – often associated with overfitting – and encourages finding flatter minima that generalize better.

Another widely used technique to prevent overfitting in neural networks is dropout. Dropout is a regularization method where, during each training iteration, a random subset of neurons in the network is "dropped out," or temporarily removed from the network (that is, set to zero). This forces the network to learn more robust features that are not reliant on specific neurons, as different subsets of neurons are active at different times. At test time, dropout is turned off, and the full network is used, but with the weights scaled down to account for the dropped neurons during training. In Section 6.4 we discuss how dropout can also be used for UQ.

For a more in-depth exploration of this topic, we recommend referring to Good-

fellow et al. (2016) and Zhang et al. (2021).

2.7 Summary

In this chapter, we reviewed the fundamentals of supervised learning, where the goal is to predict a target variable (label) Y using a set of input variables (features) $\mathbf{x}$. We saw that this task can be categorized as regression when Y is quantitative and classification when Y is qualitative.

We explored the concepts of loss functions and risk, emphasizing that different loss functions lead to different optimal solutions, such as the regression function in regression problems and the Bayes classifier in classification problems. We discussed the importance of model selection and saw how challenges like overfitting and underfitting can impact model performance. Techniques like data splitting, cross-validation, and understanding the bias-variance tradeoff were introduced as essential tools for improving model selection and predictive accuracy.

The chapter also introduced various supervised learning methods. We saw examples of parametric methods, like linear regression, lasso, and logistic regression, which involve specific assumptions about the form of the prediction function. Nonparametric methods, such as k-nearest neighbors and decision trees, were discussed for their flexibility in modeling complex data. We explored ensemble methods like bagging, random forests, and boosting, which combine multiple models to enhance predictive performance. Lastly, we looked at neural networks, noting their power in capturing complex relationships through interconnected layers.

Chapter 3

Quantifying Aleatoric Uncertainty with Conditional Densities

Under the Wave off Kanagawa. Katsushika Hokusai, 1831, The Metropolitan Museum of Art, New York.

> Prediction is very difficult,
> especially if it is about the future.
>
> Niels Bohr

Instead of producing a single point prediction $g(\mathbf{x})$ for Y, it is often more informative to assess the uncertainty surrounding the new Y. This chapter focuses on estimating the conditional density $f(y|\mathbf{x})$, which reflects the aleatoric uncertainty in Y, where we use f to represent both discrete and continuous labels. In contrast, Chapters 5 and 6 explore epistemic uncertainty as well.

For example, in a classification problem, knowing the conditional density $f(y|\mathbf{x}) = \mathbb{P}(Y = y|\mathbf{x})$ helps us address questions such as "What is the probability that the predicted label for a new sample is correct?" This probability is given by $\mathbb{P}(Y = g(\mathbf{X})|\mathbf{x}) = f(g(\mathbf{x})|\mathbf{x})$.

We begin the chapter by presenting loss functions that can be used to assess conditional density estimators. Then, we introduce several estimators of $f(y|\mathbf{x})$ both for classification and regression problems. Finally, we discusse how to estimate specific quantiles of $f(y|\mathbf{x})$. Notice that both quantiles and the regression $\mathbb{E}[Y|\mathbf{x}] = \int y f(y|\mathbf{x}) dy$ are properties of y are a function of f.

3.1 Loss Functions

Similar to the standard supervised learning approach, when it comes to comparing and fine-tuning different Conditional Density Estimators (CDEs), it is essential to define a suitable loss function. In this chapter, we define the loss function as a mapping denoted by:

$$\begin{aligned} L\colon \mathcal{F} \times \mathcal{X} \times \mathcal{Y} &\to \mathbb{R} \\ (\widehat{f}, \mathbf{x}, y) &\mapsto L(\widehat{f}; \mathbf{x}, y), \end{aligned}$$

where $\widehat{f}$ is an estimate of f and $\mathcal{F}$ denotes the class of all conditional density estimates.

Typically, loss functions for conditional density estimation are required to be proper - their implies risk must be minimized by the true conditional density $f(y|\mathbf{x})$:

Definition 1 (Proper loss function). *A CDE loss function L is proper if, and only if,*

$$f = \arg\min_{\widehat{f} \in \mathcal{F}} \mathbb{E}\left[L(\widehat{f}; \mathbf{X}, Y)\right],$$

where f denotes the true conditional density of $Y|\mathbf{x}$.

For a loss function to be useful, the risk implied by it must be easy to estimate. Few loss functions have this property; in the sequence, we explore a few of them.

3.1.1 The L_2 loss and the Brier Score

The L_2-loss is given by

$$L_2(\widehat{f}; \mathbf{x}, y) = \int \left(\widehat{f}(y'|\mathbf{x}) - f(y'|\mathbf{x})\right)^2 dy',$$

where the integral is a summation for classification problems.

The L_2-risk is therefore given by

$$\begin{aligned}\mathbb{E}\left[L_2(\widehat{f}; \mathbf{X}, Y)\right] &= \iint \left(\widehat{f}(y|\mathbf{x}) - f(y|\mathbf{x})\right)^2 dP(\mathbf{x})dy \\ &= \iint \widehat{f}^2(y|\mathbf{x})dP(\mathbf{x})dy - 2\iint \widehat{f}(y|\mathbf{x})dP(\mathbf{x}, y) + C, \end{aligned} \tag{3.1}$$

where C is a constant that does not depend on the estimator and $P(\mathbf{x}, y)$ is the joint distribution of $(\mathbf{X}, Y)$. It is easy to see from Equation 3.1 that the L_2 loss is proper.

Given a validation sample $(\mathbf{X}'_1, Y'_1), \ldots, (\mathbf{X}'_m, Y'_m)$, the L_2-risk can be estimated (up to C, which is not required for model selection and parameter tuning) by

$$\frac{1}{m}\sum_{i=1}^{m} \int \widehat{f}^2(y|\mathbf{x}'_i)dy - 2\frac{1}{m}\sum_{i=1}^{m} \widehat{f}(y'_i|\mathbf{x}'_i).$$

Notice that, in the classification, this estimate can be rewritten as

$$\begin{aligned}
&\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\widehat{f}^2(j|\mathbf{x}_i') - 2\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\widehat{f}(j|\mathbf{x}_i')\mathbb{I}(y_i'=j)\\
&\quad=\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\left(\widehat{f}(j|\mathbf{x}_i') - \mathbb{I}(y_i'=j)\right)^2 - \frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\mathbb{I}(y_i'=j)\\
&\quad=\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\left(\widehat{f}(j|\mathbf{x}_i') - \mathbb{I}(y_i'=j)\right)^2 - 1.
\end{aligned}$$

It follows that minimizing the estimate of the L_2-risk is equivalent to minimizing

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\left(\widehat{f}(j|\mathbf{x}_i') - \mathbb{I}(y_i'=j)\right)^2, \tag{3.2}$$

which is known as the Brier score (Brier, 1950).

3.1.2 The cross-entropy loss (or the negative loglikelihood)

> Only entropy comes easy.
>
> Anton Chekhov

The cross-entropy loss is defined as

$$L_{\text{CE}}(\widehat{f};\mathbf{x},y) = -\log\left(\widehat{f}(y|\mathbf{x})\right), \tag{3.3}$$

and therefore the CE-risk is given by

$$\mathbb{E}\left[L_{\text{CE}}(\widehat{f};\mathbf{X},Y)\right] = -\mathbb{E}\left[\log\left(\widehat{f}(Y|\mathbf{X})\right)\right].$$

This risk can be estimated by the observed negative log-likelihood of a validation sample

$$-\frac{1}{m}\sum_{i=1}^{m}\log\left(\widehat{f}(y_i'|\mathbf{x}_i')\right),$$

which, in the case of classification, is often written as

$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\mathbb{I}(y_i'=j)\log\left(\widehat{f}(j|\mathbf{x}_i')\right).$$

The reason the cross-entropy loss has this name is that its associated risk is the expected cross-entropy between f and $\widehat{f}$:

$$\begin{aligned}&\mathbb{E}\left[L_{\text{CE}}(\widehat{f};\mathbf{X},Y)\right]=\mathbb{E}\left[\mathbb{E}\left[L_{\text{CE}}(\widehat{f};\mathbf{X},Y)|\mathbf{X}\right]\right]\\&=\mathbb{E}\left[H\left(f(\cdot|\mathbf{X}),\widehat{f}(\cdot|\mathbf{X})\right)\right],\end{aligned}$$

where $H(p,q)=-\int p(y)\log(q(y))dy$ denotes the cross-entropy between distributions p and q. Because $H(p,q)$ is minimized when $p=q$, it follows from this that the cross-entropy loss is also proper.

The cross-entropy loss can also be characterized in terms of the Kullback-Leibler divergence. Indeed,

$$H\left(f(\cdot|\mathbf{x}),\widehat{f}(\cdot|\mathbf{x})\right)=H\left(f(\cdot|\mathbf{x})\right)+D_{\text{KL}}\left(f(\cdot|\mathbf{x})\|\widehat{f}(\cdot|\mathbf{x})\right),$$

where $H\left(f(\cdot|\mathbf{x})\right)=-\int f(y|\mathbf{x})\log f(y|\mathbf{x})\,dy$ is the entropy of the true distribution f, and

$$D_{\text{KL}}\left(f(\cdot|\mathbf{x})\|\widehat{f}(\cdot|\mathbf{x})\right)=\int f(y|\mathbf{x})\log\frac{f(y|\mathbf{x})}{\widehat{f}(y|\mathbf{x})}\,dy,$$

is the Kullback-Leibler divergence between f and $\widehat{f}$. As $H\left(f(\cdot|\mathbf{X})\right)$ does not depend on $\widehat{f}$, this shows that

$$\arg\min_{\widehat{f}\in\mathcal{G}}\mathbb{E}\left[L_{\text{CE}}(\widehat{f};\mathbf{X},Y)\right]=\arg\min_{\widehat{f}\in\mathcal{G}}\mathbb{E}\left[D_{\text{KL}}\left(f(\cdot|\mathbf{X})\|\widehat{f}(\cdot|\mathbf{X})\right)\right],$$

where $\mathcal{G}$ is any set of estimated densities. That is, minimizing the expected cross-entropy loss corresponds to minimizing the expected Kullback-Leibler divergence between f and $\widehat{f}$.

The cross-entropy loss is sensitive to the tails of the predicted distribution because it penalizes small predicted probabilities heavily. When the true outcome y has a low predicted probability $\widehat{f}(y|\mathbf{x})$, the logarithm of this small value leads to a large loss. This means that even a few low-probability events can significantly increase

the overall loss, pushing the model to give more weight to rare or extreme outcomes. While this helps capture infrequent events, it also makes the model more sensitive to outliers and noise (Bowman, 1985; Hall, 1987).

3.1.3 Accuracy, F1-scores and related metrics: Improper Metrics for Probabilistic Classification

Accuracy, defined as $1 - \mathbb{P}(Y \neq g(\mathbf{X}))$ (Section 2.2), is widely used to tune hyperparameters in many machine learning frameworks and libraries. For example, in Python's `scikit-learn`, accuracy is often the default evaluation metric for probabilistic classification models, particularly in functions like `GridSearchCV` and `RandomizedSearchCV`. Until recently, this was also true for XGBoost's `eval_metric` parameter, which now defaults to logloss. However, the default scoring method for XGBoost remains accuracy, meaning that functions like `GridSearchCV` still rely on accuracy when tuning XGBoost models.

Although accuracy can yield reasonable results for classification tasks, using it to tune parameters can lead to suboptimal models when uncertainty quantification is important. This occurs because accuracy is an improper metric for probabilistic models.

To demonstrate this, consider two probabilistic models. The first, $\widehat{\mathbb{P}}(Y = y|\mathbf{x})$, perfectly estimates the class probabilities, meaning $\widehat{\mathbb{P}}(Y = y|\mathbf{x}) = \mathbb{P}(Y = y|\mathbf{x})$. The second model, $\widehat{\mathbb{P}}'(Y = y|\mathbf{x})$, distorts the true probabilities by pushing them closer to 0.5 when they are near 0 or 1, and moving them further from 0.5 when they are close to it. Specifically, the model is defined as:

$$\widehat{\mathbb{P}}'(Y = y|\mathbf{x}) = \begin{cases} \frac{3}{2} - \mathbb{P}(Y = y|\mathbf{x}), & \text{if } \mathbb{P}(Y = y|\mathbf{x}) > \frac{1}{2} \\ \frac{1}{2} - \mathbb{P}(Y = y|\mathbf{x}), & \text{otherwise} \end{cases}$$

This manipulated model, $\widehat{\mathbb{P}}'(Y = y|\mathbf{x})$, produces the same classification decisions as the perfectly calibrated model, $\widehat{\mathbb{P}}(Y = y|\mathbf{x})$. This occurs because

$$\widehat{\mathbb{P}}(Y = y|\mathbf{x}) > 1/2 \text{ if and only if } \widehat{\mathbb{P}}'(Y = y|\mathbf{x}) > 1/2.$$

As a result, both models achieve identical accuracy. However, $\widehat{\mathbb{P}}'(Y = y|\mathbf{x})$ offers distorted probability estimates that fail to reflect the true level of uncertainty in

the predictions. It moves probabilities closer to 0.5 when they should reflect high certainty (closer to 0 or 1), and moves them away from 0.5 when the actual uncertainty is greater. This distortion undermines the model's ability to quantify uncertainty.

In fact, any model that shifts probabilities above 0.5 to values even further above 0.5, and probabilities below 0.5 to values even lower, will maintain the same accuracy as $\widehat{\mathbb{P}}(Y = y|\mathbf{x})$. This limitation extends to other metrics that only rely on classification decisions based on a fixed thresholding of $\widehat{\mathbb{P}}(Y = y|\mathbf{x})$, such as the F1-score and balanced accuracy. Therefore, tuning models based on these metrics can result in poorly calibrated probability estimates, misrepresenting uncertainty. This becomes particularly problematic when probabilistic models are required to provide reliable rankings of the observations.

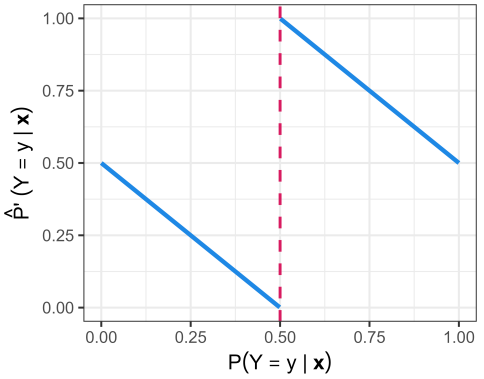


Figure 3.1: Comparison between the true class probabilities $\mathbb{P}(Y = y|\mathbf{x})$ and the distorted probabilities $\widehat{\mathbb{P}}'(Y = y|\mathbf{x})$, illustrating how accuracy can fail to penalize poor probability estimates.

3.2 Probabilistic Classifiers

There is a plethora of methods to estimate $f(y|\mathbf{x})$ in classification problems. Indeed, probabilistic classification's goal is to estimate $f(y|\mathbf{x})$. Many of the methods seen in Chapter 2 (such as the logistic regression) directly estimate such densities.

Regression can also be used to obtain probabilistic classifiers. Indeed, for a fixed i, consider the transformed variable $Z := \mathbb{I}(Y = i)$. Then, by construction, the

regression of Z on $\mathbf{x}$ is

$$\mathbb{E}[Z|\mathbf{x}] = \mathbb{P}(Y = i|\mathbf{x}) = f(i|\mathbf{x}).$$

Notice also that the mean squared error for estimating Z is the Brier Score for estimating $\mathbb{I}(Y = i)$ (recall Equation 3.2). This suggests that by using regression methods to $\mathbb{E}[Z|\mathbf{x}]$ one is effectively estimating $\mathbb{P}(Y = i|\mathbf{x})$ using the Brier Score to tune parameters and do model selection.

3.3 Parametric Approaches

Statistical models often imply a conditional density model. For instance, in a standard Gaussian linear regression model, the assumption is that

$$Y|\mathbf{x} \sim N\left(\boldsymbol{\beta}^\intercal \mathbf{x}; \sigma^2\right).$$

Generalized linear models (GLM; Nelder and Wedderburn 1972) extend this model by allowing the conditional density to be other members within the exponential family.

The parameters of such models can be estimated for instance via maximum likelihood estimation. Denoting by θ the parameters of the model, this corresponds to obtaining

$$\widehat{\theta} = \arg\max_\theta \mathcal{L}(\theta; \mathcal{D}) = \arg\max_\theta \prod_{i=1}^n f_\theta(y_i|\mathbf{x}_i) f(\mathbf{x}_i) = \arg\max_\theta \prod_{i=1}^n f_\theta(y_i|\mathbf{x}_i),$$

where $\mathcal{D}$ denotes the full dataset and we use the notation $f_\theta(y|\mathbf{x})$ to emphasize the dependency of the conditional model on θ. Once $\widehat{\theta}$ is obtained, an estimate of the conditional density, $\widehat{f}(y|\mathbf{x}) := f_{\widehat{\theta}}(y|\mathbf{x})$, is readily available. For instance, in the Gaussian linear model, the estimated conditional density is given by

$$\widehat{f}(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi\widehat{\sigma}^2}} e^{-\frac{(y - \widehat{\boldsymbol{\beta}}^\intercal \mathbf{x})^2}{2\widehat{\sigma}^2}},$$

where

$$\widehat{\boldsymbol{\beta}} = (\mathbb{X}^\intercal \mathbb{X})^{-1} \mathbb{X}^\intercal \mathbb{Y}, \tag{3.4}$$

is the least squares solution, with

$$\mathbb{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix},$$

$\mathbb{Y} = (y_1, \ldots, y_n)$, and

$$\widehat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \widehat{\boldsymbol{\beta}}^{\mathsf{T}} \mathbf{x}_i \right)^2. \tag{3.5}$$

Of course, other estimators (such as the lasso described in Chapter 2) can also be used.

3.4 FlexCode

> Be flexible, but stick to your principles.
>
> Eleanor Roosevelt

FlexCode (Izbicki and Lee, 2017) is an acronym for *Flex*ible nonparametric *co*nditional *d*ensity *e*stimation via regression. It introduces a fully nonparametric approach to conditional density estimation by reframing the problem as an orthogonal series, where regression is utilized to estimate expansion coefficients. This innovative strategy enables efficient estimation of conditional densities in high dimensions, leveraging the successes of high-dimensional regression. The versatility of *FlexCode* lies in its ability to adapt to different types of sparse structures within the data and handle various data scenarios.

For instance, in settings characterized by submanifold structures, *FlexCode* adjusts to the intrinsic dimensionality of the data through the use of a carefully chosen regression method. Options include nearest neighbors, local linear, tree-based, or spectral series regression (Bickel and Li, 2007; Kpotufe and Dasgupta, 2012; Kpotufe, 2011; Lee and Izbicki, 2016). Similarly, when the number of relevant covariates (i.e., those influencing the distribution of Y) is small, *FlexCode* constructs effective conditional density estimators using regression techniques such as lasso, SAM, Rodeo, or other

additive-based methods (Lafferty and Wasserman, 2008; Meier et al., 2009; Tibshirani, 1996; Yang and Tokdar, 2015). The adaptability of *FlexCode* extends to handling different types of covariates, including discrete data, mixed data types, functional data, circular data, and more. These diverse data types often require specialized techniques (Di Marzio et al., 2016). An R implementation of *FlexCode* can be found at https://github.com/rizbicki/FlexCoDE, while a Python implementation can be found at https://github.com/lee-group-cmu/FlexCode.

For simplicity, let's assume that Y lies in the interval $[0,1]$. In cases where this assumption does not hold, we can map Y to this specified interval. We start by specifying an orthonormal basis $(\phi_i)_{i\in\mathbb{N}}$ in $\mathbb{R}$. This basis will be used to model the density $f(y|\mathbf{x})$ *as a function of* y. As we shall see, each coefficient in the expansion can be directly estimated via a regression. Note that there is a wide range of (orthogonal) bases one can choose from to capture any challenging shape of the density function of interest (Mallat, 1999). For example, when dealing with reasonably smooth functions $f(z|\mathbf{x})$, the Fourier basis emerges as a natural choice:

$$\phi_1(y) = 1; \qquad \phi_{2i+1}(y) = \sqrt{2}\sin(2\pi i y),\ i \in \mathbb{N}; \qquad \phi_{2i}(y) = \sqrt{2}\cos(2\pi i y),\ i \in \mathbb{N}$$

Alternatively, one can opt for wavelets or similar bases to effectively capture inhomogeneities in the density. In instances where discrete responses need to be modeled, indicator functions serve as a suitable choice (Izbicki and Lee, 2016, Sec. 4.2).

For fixed $\mathbf{x} \in \mathbb{R}^d$ and $f(\cdot|\mathbf{x}) \in \mathcal{L}^2(\mathbb{R})$, we we express f as a series using basis functions $(\phi_i)i \in \mathbb{N}$ with coefficients $\beta_i(\mathbf{x})$:

$$f(y|\mathbf{x}) = \sum_{i\in\mathbb{N}} \beta_i(\mathbf{x})\phi_i(y). \tag{3.6}$$

Since the basis functions $(\phi_i)i \in \mathbb{N}$ are orthogonal, the expansion coefficients are determined by the inner product:

$$\beta_i(\mathbf{x}) = \langle f(.|\mathbf{x}), \phi_i \rangle = \int_{\mathbb{R}} \phi_i(y) f(y|\mathbf{x}) dy = \mathbb{E}\left[\phi_i(Y)|\mathbf{x}\right]. \tag{3.7}$$

In other words, each $\beta_i(\mathbf{x})$ in Equation (3.6) is a regression function or conditional expectation. This implies that, for a fixed i, we can estimate $\beta_i(\mathbf{x})$ by regressing $\phi_i(y)$ on $\mathbf{x}$ using the transformed sample $(\mathbf{X}_1, \phi_i(Y_1)), \ldots, (\mathbf{X}_n, \phi_i(Y_n))$.

The *FlexCode* estimator for $f(y|\mathbf{x})$ is defined as follows:

$$\widehat{f}(y|\mathbf{x}) := \sum_{i=1}^{I} \widehat{\beta}_i(\mathbf{x})\phi_i(y), \tag{3.8}$$

where the coefficients $\widehat{\beta}_i(\mathbf{x})$ are obtained through regression:

$$\widehat{\beta}_i(\mathbf{x}) = \hat{\mathbb{E}}\left[\phi_i(Y)|\mathbf{x}\right],$$

capturing how the density varies across the covariate space. The parameter I determines the series expansion cutoff and plays a crucial role in balancing the bias-variance tradeoff in the final density estimate. In general, a smaller value of I results in a smoother density. The optimal value for the tuning parameter I is determined through the validation set, utilizing the L_2 loss function for estimation.

Notice that each β_i is fitted independently and does not depend on I. Thus, the optimal value of I can be chosen in a fast manner; one does not need to refit β_i's for each I.

The estimated densities are not necessarily proper density functions. Thus, after the initial $\hat{f}$ is fitted, FlexCode applies the same techniques as in Izbicki and Lee (2016, Section 2.2) to remove potentially negative values and to make sure it integrates to one. Morever, it also remove spurious bumps that are artifacts of the Fourier series using the technique described in Izbicki and Lee (2016, Section 2.2) .

3.4.1 Variable Importance

If the regression method used to estimate β_i provides variable importance metrics, these metrics can be directly utilized by FlexCode. Specifically, let $u_{i,j}$ denote a measure of importance of the j-th feature in estimating regression β_i. For instance, for random forests, $u_{i,j}$ may represent the mean decrease in the Mean Squared Error (Breiman, 2001b); for sparse additive models , $u_{i,j}$ may be value of the indicator function for the j0th summary statistic when estimating $\beta_i(\mathbf{x})$ (Ravikumar et al., 2009). We define an *importance measure* for the j-th feature *in estimating* $f(y|\mathbf{x})$ according to

$$u_j := \frac{1}{I}\sum_{i=1}^{I} u_{i,j}. \tag{3.9}$$

This importance measures adds interpretability to FlexCode.

3.4.2 Theory

This section shows bounds and rates for *FlexCode* (Eq. 3.8). We use the notation $\widehat{f}_I(z|\mathbf{x})$ to indicate its dependence on the cutoff I. Proofs and further details can be found in Izbicki and Lee (2017).

For every $s > \frac{1}{2}$ and $0 < c < \infty$, let $W_\phi(s,c) = \{f = \sum_{i\geq 1} \theta_i \phi_i : \sum_{i\geq 1} a_i^2 \theta_i^2 \leq c^2\}$, where $a_i \sim (\pi i)^s$, denote the Sobolev space. We assume that f belongs to a set of functions which are not too "wiggly":

Assumption 1 (Smoothness in y direction). $\forall \mathbf{x} \in \mathcal{X}$, $f(y|\mathbf{x}) \in W_\phi(s_\mathbf{x}, c_\mathbf{x})$, *where* $f(y|\mathbf{x})$ *is viewed as a function of* y, *and* $s_\mathbf{x}$ *and* $c_\mathbf{x}$ *are such that* $\inf_\mathbf{x} s_\mathbf{x} \stackrel{def}{=} \beta > \frac{1}{2}$ *and* $\int_\mathcal{X} c_\mathbf{x}^2 d\mathbf{x} \stackrel{def}{=} C < \infty$.

We make the assumption that each function $\beta_i(\mathbf{x})$ is estimated using a regression method with a convergence rate of $O(n^{-\frac{2\alpha}{2\alpha+p}})$. Here, α typically represents a parameter associated with the smoothness of the $\beta_i(\mathbf{x})$ function, while p denotes either the number of relevant covariates or the intrinsic dimension of $\mathbf{x}$. To put it differently, we posit that each regression process is capable of adapting to the sparse structure inherent in the data. This formal assertion is explicitly presented as Assumption 2.

Assumption 2 (Regression convergence). *For every* $i \in \mathbb{N}$, *there exists some* $p \in \mathbb{N}$ *and* $\alpha > 0$ *such that*

$$\mathbb{E}\left[\int \left(\widehat{\beta}_i(\mathbf{x}) - \beta_i(\mathbf{x})\right)^2 d\mathbf{x}\right] = O(n^{-2\alpha/(2\alpha+p)})$$

Theorem 3. *Under Assumptions 1 and 2, an upper bound on the risk of the CDE from Equation 3.8 is*

$$\mathbb{E}\left[\iint \left(\widehat{f}_I(y|\mathbf{x}) - f(y|\mathbf{x})\right)^2 dy d\mathbf{x}\right] \leq IO\left(n^{-2\alpha/(2\alpha+p)}\right) + O(I^{-2\beta})$$

Corollary 1. *Under Assumptions 1 and 2, it is optimal to take*

$$I \asymp n^{\frac{2\alpha}{(2\alpha+p)(2\beta+1)}},$$

which yields the rate

$$O\left(n^{-\frac{2\beta}{2\beta+p\frac{2\beta+1}{2\alpha}+1}}\right)$$

for the FlexCode *estimator.*

In conclusion, the convergence rate of *FlexCode* is solely contingent on p, representing the "true" dimension of the problem. Notably, the rate approaches minimax efficiency concerning p. In the isotropic scenario, where $\mathbf{x}$ and y exhibit the same degree of smoothness ($\alpha = \beta$), the rate is given by

$$O\left(n^{-\frac{2\alpha}{2\alpha+p\frac{2\alpha+1}{2\alpha}+1}}\right),$$

which closely aligns with the minimax rate $O\left(n^{-\frac{2\alpha}{(2\alpha+1+p)}}\right)$ of a conditional density estimator with p covariates (Izbicki and Lee, 2016). The variance lies in the multiplicative factor $\frac{2\alpha+1}{2\alpha}$ which approaches 1 as the function f becomes smoother. While *FlexCode*'s rate is marginally slower than the optimal rate, the estimator demonstrates significantly greater speed compared to the standard minimax rate,

$$O\left(n^{-\frac{2\alpha}{(2\alpha+1+d)}}\right),$$

typical for nonparametric conditional density estimators in $\mathbb{R}^d$. In essence, despite the presence of d covariates, *FlexCode* successfully mitigates the curse of dimensionality, effectively behaving as if only $p \ll d$ covariates were at play.

For a version of FlexCode for (dependent) time series data, see Grivol et al. (2023).

3.5 Mixture Models and Networks

In the context of conditional density estimation, mixture models (Quandt, 1958; Tatiana et al., 2009) assume that the conditional density $f(y|\mathbf{x})$ can be written as a mixture of parametric densities,

$$f(y|\mathbf{x}) = \sum_{i=1}^{m} \alpha_i(\mathbf{x})\phi(y|\theta_i(\mathbf{x})), \tag{3.10}$$

where $\alpha_i(\mathbf{x})$'s are nonnegative coefficients such that $\sum_{i=1}^{m} \alpha_i(\mathbf{x}) = 1$ and ϕ is the density of a predefined parametric distribution over y with parameters $\theta_i(\mathbf{x})$. For

instance, $\phi(y|\theta_i(\mathbf{x}))$ can be the density of a gaussian distribution:

$$\phi(y|\theta_i(\mathbf{x})) = \frac{1}{\sqrt{2\pi\sigma_i^2(\mathbf{x})}} \exp\left\{-\frac{(y-\mu_i(\mathbf{x})^2)}{2\sigma_i^2(\mathbf{x})}\right\},$$

where $\theta_i(\mathbf{x}) = (\mu_i(\mathbf{x}), \sigma_i(\mathbf{x}))$. One then models each parameter by imposing a parametric relationship, such as

$$\mu_i(\mathbf{x}) = \left(\boldsymbol{\beta}^{(i)}\right)^t \mathbf{x} \text{ and } \sigma_i(\mathbf{x}) = \exp\left[\left(\boldsymbol{\gamma}^{(i)}\right)^t \mathbf{x}\right].$$

These parameters are typically fitted using the Expectation Maximization algorithm (EM; Dempster et al. 1977).

A mixture density network (Bishop, 1994) instead , employs a neural network to parameterize the functions $\alpha_i(\mathbf{x})$ and $\theta_i(\mathbf{x})$. Specifically, it designs a network that accepts $\mathbf{x}$ as its input and outputs a set of pairs $(\alpha_i(\mathbf{x}), \theta_i(\mathbf{x}))_{i=1}^m$. The user must decide the value of m, which defines the network's output structure. For the output layers corresponding to $\alpha_i(\mathbf{x})$, a softmax function is commonly applied to ensure these values lie on a simplex. Rather than using the EM to estimate the parameters of the network, this approach leverages standard neural network techniques. The cross-entropy loss function is frequently employed for this purpose. Figure 3.2 shows an example of one architecture that can be used to estimate a mixture of two gaussian distributions.

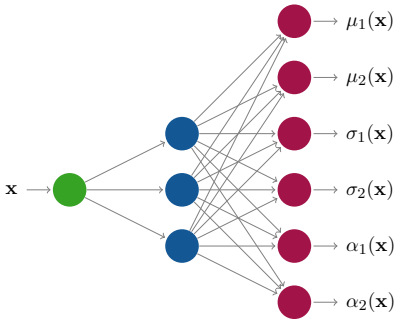


Figure 3.2: Example of a Gaussian mixture density network structure with input x, hidden layers, and outputs $\mu_1(x)$, $\mu_2(x)$, $\sigma_1(x)$, $\sigma_2(x)$, $\alpha_1(x)$, and $\alpha_2(x)$. The outputs are combined according to Equation 3.10 to give the estimated density.

3.6 Normalizing Flows

> I used to just live my life by going with the flow.
>
> Jin

In this section, we assume $\mathbf{y} \in \mathbb{R}^m$, that is, the response can be multivariate. Normalizing flows model $f(\mathbf{y}|\mathbf{x})$ by creating a *bijective* transformation of $\mathbf{y}$,

$$T^{\mathbf{x}} : \mathbb{R}^m \longrightarrow \mathbb{R}^m,$$

such that $T^{\mathbf{x}}(\mathbf{Y})|\mathbf{x}$ has density approximately $p_0(\cdot)$[1], where p_0 is chosen a priori. For instance, one may take p_0 to be a Normal$(\mathbf{0}, \mathbf{I}_m)$.

A key reason why this idea is useful is that $T^{\mathbf{x}}$ can be easily used to approximate $f(\mathbf{y}|\mathbf{x})$. Indeed, because $T^{\mathbf{x}}$ is bijective and $T^{\mathbf{x}}(\mathbf{Y})|\mathbf{x}$ has density approximately $p_0(\cdot)$, it follows that

$$f(\mathbf{y}|\mathbf{x}) \approx p_0\left(T^{\mathbf{x}}(\mathbf{y})\right) \left|\det J_{T^{\mathbf{x}}}(\mathbf{y})\right|, \tag{3.11}$$

where

$$J_{T^{\mathbf{x}}}(\mathbf{y}) = \begin{bmatrix} \frac{\partial T_1^{\mathbf{x}}}{\partial y_1} & \cdots & \frac{\partial T_1^{\mathbf{x}}}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial T_m^{\mathbf{x}}}{\partial y_1} & \cdots & \frac{\partial T_m^{\mathbf{x}}}{\partial y_m} \end{bmatrix}$$

is the Jacobian matrix with all $m \times m$ partial derivatives of $T^{\mathbf{x}}$ with respect to $\mathbf{y}$. Thus, if a good $T^{\mathbf{x}}$ was known, the right-hand side of Equation 3.11 could be used to approximate $f(\mathbf{y}|\mathbf{x})$ – all of its terms are known.

Another feature of normalizing flows is that it is easy to sample from the estimated conditional density. Indeed, if $\mathbf{Z} \sim p_0(\cdot)$, then by construction $(T^{\mathbf{x}})^{-1}(\mathbf{Z})$ has approximately the same distribution as $\mathbf{Y}|\mathbf{x}$. Thus, one can sample from the estimated conditional density by sampling from the base distribution p_0 and then applying the inverse of the transformation that was learned.

In practice, to learn $T^{\mathbf{x}}$, this function is parameterized using a neural network with weights ϕ. This network takes $\mathbf{x}$ and $\mathbf{y}$ as inputs, and it outputs the vector

[1] $p_0(\cdot)$ may actually depend on $\mathbf{x}$ so that, $p_0(\cdot|\mathbf{x})$

$T^{\phi,\mathbf{x}}(\mathbf{y})$. Given a training sample $(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)$, the weights ϕ are typically trained by minimizing the cross-entropy loss of the implied conditional density. That is, one searches for the weights ϕ that minimize

$$-\frac{1}{n}\sum_{i=1}^{n}\log\left[p_0\left(T^{\phi,\mathbf{x}_i}(\mathbf{y}_i)\right)\left|\det J_{T^{\phi,\mathbf{x}_i}}(\mathbf{y}_i)\right|\right]$$

There are several approaches to parameterize $T^{\mathbf{x}}$ using neural networks. Modern methods focus on creating transformations that are both expressive, increasing the likelihood of approximating well the true density, and computationally efficient, ensuring that the Jacobians in Equation 3.11 are calculated with minimal cost. Typically, $T^{\mathbf{x}}$ is build using a composition of small building blocks, that is, $T^{\mathbf{x}} = S_B \circ \ldots \circ S_2 \circ S_1$, where we omit $\mathbf{x}$ from S's to simplify the notation. Each S_b is a bijective transformation whose Jacobian is easy to compute. This approach is useful because

(i) $T^{\mathbf{x}}$ is by construction bijective and its inverse can be easily computed as $S_1^{-1} \circ S_2^{-1} \circ \ldots S_B^{-1}$, and

(ii) the determinant of its Jacobian can be computed using the chain rule:

$$\det J_{T^{\mathbf{x}}}(\mathbf{y}) = \det J_{S_B}(S_{B-1} \circ \ldots \circ S_2 \circ S_1(\mathbf{y})) \ldots \det J_{S_1}(\mathbf{y}).$$

Each S is parametrized using a neural network structure. In this way, the full network will consist of the nested layers $S_1 \to S_2 \to S_B$, as shown in Figure 3.3 for a setting with no features $\mathbf{x}$.

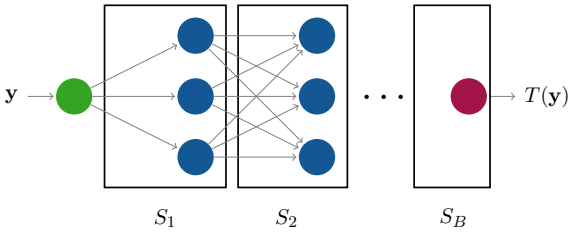


Figure 3.3: Pictorial representation of a simplified normalizing flow structure without features. The inputs are transformed through multiple stages of invertible transformations, S_1, S_2, …, S_B, leading to the output $T(\mathbf{y})$. Note that this illustration does not depict the exact transformations S.

An example of one of such building blocks S_b is the affine coupling layer (Dinh

et al., 2016), which splits its input $\mathbf{u}$ into two components, $\mathbf{u}_1$ and $\mathbf{u}_2$ and is given by

$$S(\mathbf{u}) = (\mathbf{u}_1, a(\mathbf{u}_1) \odot \mathbf{u}_2 + b(\mathbf{u}_1)).$$

Here, a and b are functions parametrized by the network: both take $\mathbf{u}_1$ as input and pass it through the layers of a neural network with weights that are estimated when training the normalizing flow. The symbol $\odot$ represents the Hadamard (element-wise) product. This transformation remains bijective even if a and b are not.

Several such building blocks have been designed, such as invertible 1×1 convolutions, split priors (Dinh et al., 2016), masked autoregressive flows (Papamakarios et al., 2017), and neural autoregressive flows (Huang et al., 2018). See Papamakarios et al. (2021) for a comprehensive introduction to such models.

3.7 The Ratio Trick

The ratio trick has appeared in very different fields and with different goals (e.g., Cheng and Chu 2004; Cranmer et al. 2020; Cranmer et al. 2015; Gutmann et al. 2018; Qin 1998; Sugiyama et al. 2010b). Essentially, it converts the goal of estimating a quantity (usually a ratio between two densities) into the problem of estimating a classifier.

Here, we explore a version of this trick based on Dalmasso et al. (2020a) and Okuno and Polo (2021). In this context, the ratio tricks converts the problem of estimating the conditional density of a continuous vector $\mathbf{y}$ given $\mathbf{x}$ into the problem of estimating a probabilistic classifier.

Specifically, we start by generating a labeled sample $\mathcal{T} = \{(\mathbf{X}'_i, \mathbf{Y}'_i, S_i)\}_{i=1}^m$. This dataset contains all original training samples $(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)$, and also additional artificial samples $(\tilde{\mathbf{X}}_1, \tilde{\mathbf{Y}}_1), \ldots, (\tilde{\mathbf{X}}_{\tilde{n}}, \tilde{\mathbf{Y}}_{\tilde{n}})$, where each pair $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ is drawn from $\tilde{\mathbf{X}} \sim U(\{\mathbf{X}_1, \ldots, \mathbf{X}_n\})$[2] and $\tilde{\mathbf{Y}}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}} \sim g(\mathbf{y}|\mathbf{X} = \tilde{\mathbf{x}})$. Here, the conditional distribution $g(\mathbf{y}|\mathbf{x})$ is chosen by the user.

Define the odds at $(\mathbf{y}; \mathbf{x})$ as

$$\mathbb{O}(\mathbf{y}; \mathbf{x}) := \frac{\mathbb{P}(S = 1|\mathbf{x}, \mathbf{y})}{\mathbb{P}(S = 0|\mathbf{x}, \mathbf{y})}. \tag{3.12}$$

[2]One can also sample from the marginal distribution of $\mathbf{x}$, $f(\mathbf{x})$ if this is an option. This is the case on the LFI problem studied in Chapter 9.

By construction, it holds that

$$\mathbb{O}(\mathbf{y};\mathbf{x}) = \frac{f(\mathbf{x},\mathbf{y}|S=1)}{f(\mathbf{x},\mathbf{y}|S=0)} = \frac{f(\mathbf{y}|\mathbf{x})}{g(\mathbf{y}|\mathbf{x})}. \tag{3.13}$$

Thus,

$$f(\mathbf{y}|\mathbf{x}) = g(\mathbf{y}|\mathbf{x})\mathbb{O}(\mathbf{y};\mathbf{x}).$$

This suggests we estimate f by using the plugin estimate

$$\widehat{f}(\mathbf{y}|\mathbf{x}) := g(\mathbf{y}|\mathbf{x})\frac{\widehat{\mathbb{P}}(S=1|\mathbf{x},\mathbf{y})}{\widehat{\mathbb{P}}(S=0|\mathbf{x},\mathbf{y})}, \tag{3.14}$$

where $\widehat{\mathbb{P}}(S=1|\mathbf{x},\mathbf{y})$ is obtained using any probabilistic classifier (such as those discussed in Chapter 2).

3.8 Cal-PIT

The Cal-PIT reshaping technique discussed in Section 4.2 can also be interpreted as a conditional density estimator.

3.9 Other Conditional Density Estimators

In the realm of low-dimensional feature spaces, many conditional density estimators rely on the initial estimation of $f(y,\mathbf{x})$ and $f(\mathbf{x})$. A common practice involves using kernel density estimators, as pioneered by Rosenblatt (1969). The subsequent combination of these estimates is expressed by $f(y|\mathbf{x}) = \frac{f(y,\mathbf{x})}{f(\mathbf{x})}$, often implemented through

$$\widehat{f}(y|\mathbf{x}) = \frac{\sum_{i=1}^{n} K_{h_x}(\|\mathbf{x}-\mathbf{X}_i\|)K_{h_y}(y-Y_i)}{\sum_{i=1}^{n} K_{h_x}(\|\mathbf{x}-\mathbf{X}_i\|)},$$

where $K_h(t) = h^{-d}K(t/h)$ denotes a kernel with bandwidth h in d dimensions.

Several advancements have been made to enhance this approach. Notable works, such as those by Hyndman et al. (1996) and Ichimura and Fukuda (2010), introduce different criteria and shortcuts for parameter tuning, along with efficient implementations of these methods.

Alternative techniques for low-dimensional settings include locally polynomial

regression (Fan et al., 1996), least squares approaches (Sugiyama et al., 2010a), and density estimation through quantile estimation (Takeuchi et al., 2009). Further exploration of these methods and others is available in the comprehensive review by Bertin et al. (2016).

Hall et al. (2004) presents a method for parameter tuning in kernel density estimators that automatically identifies relevant components of $\mathbf{x}$ for $f(y|\mathbf{x})$. Similarly, Shiga et al. (Shiga et al., 2015a) propose a conditional estimator that selects relevant components, assuming an additive structure in $f(y|\mathbf{x})$.

Efromovich (2010) introduces an orthogonal series estimator, offering automatic dimension reduction on $\mathbf{x}$ when certain components are conditionally independent of the response. For high-dimensional feature spaces, Izbicki and Lee (2016) propose a spectral basis-based orthogonal series approach, while Inácio and Izbicki (2018) and Dalmasso et al. (2020b) present a version of FlexCode where $\beta_i(\mathbf{x})$ is estimated using neural networks.

3.10 Quantile Regression

Quantile regression aims to estimate one property of the distribution of $Y|\mathbf{x}$: its *conditional quantiles*. Formally, the α-conditional quantile of Y at $\mathbf{x}$, $q_\alpha(\mathbf{x})$, is the function such that $\mathbb{P}(Y \leq q_\alpha(\mathbf{x})|\mathbf{X} = \mathbf{x}) = \alpha$ (we assume here that the distribution of $Y|\mathbf{x}$ is strictly continuous). In other words, $q_\alpha(\mathbf{x}) = F^{-1}(\alpha|\mathbf{x})$, where F is the cumulative distribution of Y conditional on $\mathbf{x}$. Note that $q_{1/2}(\mathbf{x})$ is the conditional median.

3.10.1 Pinball Loss

To assess the accuracy of an estimate of $q_\alpha(\mathbf{x})$, we will use the loss function called the *pinball loss*. Let $g : \mathcal{X} \longrightarrow \mathbb{R}$ be an estimate of $q_\alpha(\mathbf{x})$. The pinball loss of g evaluated at $(\mathbf{x}, y)$ is defined as

$$L_\alpha(g, \mathbf{x}, y) = (g(\mathbf{x}) - y)(\mathbb{I}(y \leq g(\mathbf{x})) - \alpha). \tag{3.15}$$

The risk derived from this loss function is minimized precisely by the solution $g(\mathbf{x}) = q_\alpha(\mathbf{x})$:

$$\arg\min_g \mathbb{E}\left[L_\alpha(g, \mathbf{X}, Y)\right] = q_\alpha(\mathbf{x}).$$

This indicates that it is suitable for evaluating estimates of $q_\alpha(\mathbf{x})$.

Figure 3.4 illustrates the behavior of this loss function as a function of $g(\mathbf{x})$ for $\alpha = 10\%$. According to it, underestimating y is less bad than overestimating this quantity.

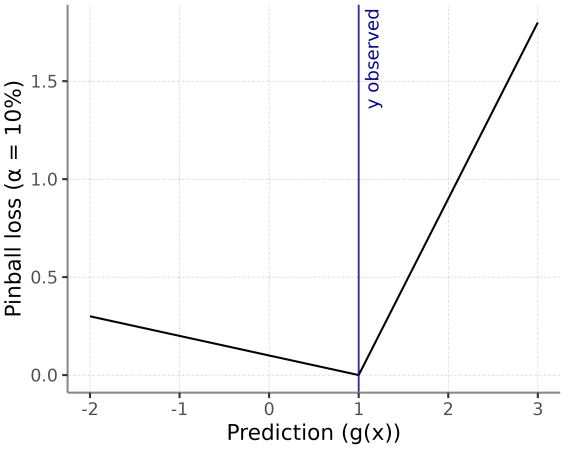


Figure 3.4: Pinball loss for $\alpha = 10\%$.

3.10.1.1 Estimation the quantile function

- **[Parametric Methods]** One way to estimate $q_\alpha(\mathbf{x})$ is to assume that it depends linearly on the covariates. In other words, we can assume that

$$q_\alpha(\mathbf{x}) = \beta^t \mathbf{x}.$$

  We can then search for

$$\arg\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} L_\alpha(g_\beta, \mathbf{x}_i, y_i),$$

  where $g_\beta(\mathbf{x}) := \beta^t \mathbf{x}$ (Koenker and Bassett Jr, 1978).

- **[KNN]** The idea of the k-nearest neighbors (KNN) method for estimating $q_\alpha(\mathbf{x})$ (Ma et al., 2016) is to use the responses Y of the k-nearest neighbors to $\mathbf{x}$.

Formally, we define

$$g(\mathbf{x}) = \widehat{q}_\alpha\left(\{y_i\}_{i\in\mathcal{N}_\mathbf{x}}\right), \tag{3.16}$$

where $\widehat{q}_\alpha(\mathcal{S})$ is the α-quantile of $\mathcal{S}$ and $\mathcal{N}_\mathbf{x}$ is the set of the k nearest observations to $\mathbf{x}$, i.e.,

$$\mathcal{N}_\mathbf{x} = \left\{ i \in \{1,\dots,n\} : d(\mathbf{x}_i, \mathbf{x}) \leq d^k_\mathbf{x} \right\}$$

and $d^k_\mathbf{x}$ is the distance from $\mathbf{x}$ to the k-th nearest neighbor of $\mathbf{x}$.

The tuning parameter k can be chosen through cross-validation using the pinball loss as a metric.

- **[Random Forests]** Meinshausen and Ridgeway (2006) proposes estimating quantiles from an estimate of the conditional cumulative distribution. This estimate is a local version of the estimate given by the empirical cumulative distribution function:

$$\widehat{F}(y|\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) \mathbb{I}(y_i \leq y),$$

where $w_i(\mathbf{x})$ is a measure of similarity between $\mathbf{x}$ and the i-th observation in the training set obtained through a random forest. Specifically, it is defined as follows: let $R^b_\mathbf{x}$ be the leaf where the observation $\mathbf{x}$ falls in the b-th tree, and define the weight (associated with that tree) as

$$w_i(\mathbf{x}; b) = \frac{\mathbb{I}(\mathbf{x}_i \in R^b_\mathbf{x})}{\sum_{j=1}^{n} \mathbb{I}(\mathbf{x}_j \in R^b_\mathbf{x})}.$$

In other words, $w_i(\mathbf{x}; b)$ is the proportion of observations in the training set that fall in the same leaf as $\mathbf{x}$. The combined weight is then given by the average of these weights:

$$w_i(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} w_i(\mathbf{x}; b).$$

This method has several advantages over KNN. For example, random forests automatically perform variable selection in their construction, so the quantile estimates inherit this property and are therefore more robust in scenarios with many irrelevant variables. Additionally, random forests automatically take into

account interactions and non-linearities of the covariates.

- **[Neural Networks]** A neural network can be trivially used to estimate quantile regression. To do this, simply use the pinball loss as the loss function.

Figure 3.5 displays the 10th and 90th quantiles estimated by boosting, linear, and random forest quantile regression models for predicting life expectancy based on GDP per capita using the data described in Section 1.1. The code to generate this figure is available at the Quantile Regression Notebook.

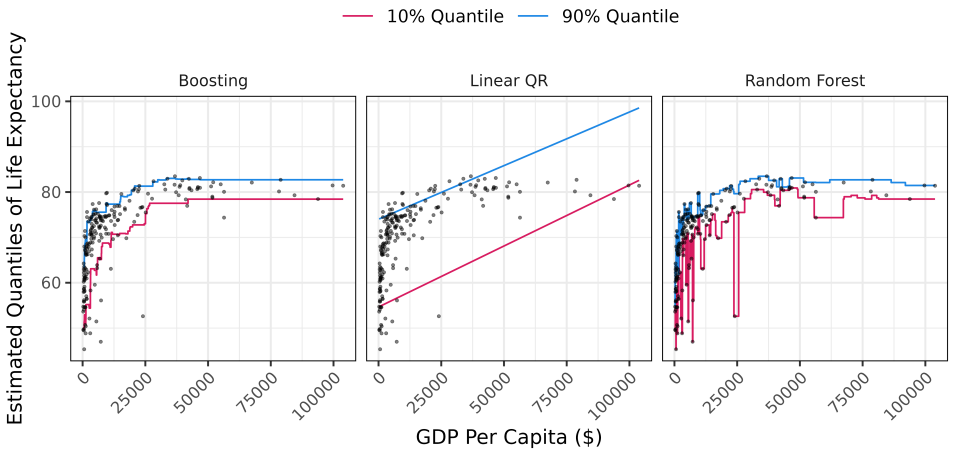


Figure 3.5: Comparison between boosting, linear, and random forest quantile regression models for Predicting Life Expectancy based on GDP per Capita, using 10th and 90th percentiles.

3.11 Simulated Example: Gaussian Distribution

Here, we compare various methods for CDE on a toy example which involves the creation of a multivariate dataset, where the conditional relationship between features and target variables is known. The data is generated according to:

$$\mathbf{X} \sim \mathcal{U}(0,1)^d,$$

with $d = 50$, and

$$Y|\mathbf{x} \sim N(x_1, 1).$$

We compare FlexCode, where the β coefficients are estimated using random forests (Section 3.4), against a Gaussian mixture density network (Section 3.5) and normalizing flows (Section 3.6). The detailed implementation and code are available in the CDE Notebook. Figure 3.6 illustrates the estimated conditional densities at six test points. FlexCode consistently demonstrates superior performance over the other methods, as further supported by the L_2 loss values shown in Table 3.1.

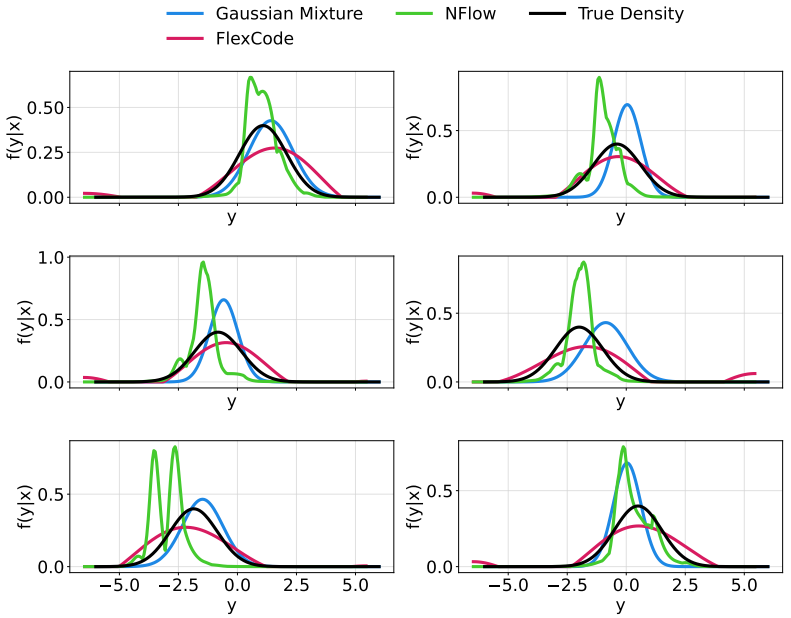


Figure 3.6: Comparison of estimated conditional density functions using FlexCode with random forest-based β estimates, Gaussian mixture density networks, and normalizing flows on a simulated Gaussian dataset. The true conditional density $Y|\mathbf{x} \sim N(x_1, 1)$ is shown for reference.

Table 3.1: Estimated losses (with standard errors) for different CDE methods on a simulated Gaussian dataset. FlexCode, with random forest-based β estimates, outperforms the Gaussian mixture density network and normalizing flows.

Method	L_2 Loss (Standard Error)
Gaussian Mixture	0.1154 (0.0121)
FlexCode	-0.2657 (0.0015)
NFlow	0.1430 (0.0083)

3.12 Example: Twitter Location Prediction

We leverage FlexCode to predict the geographical origin of tweets based on their content. By utilizing samples with known locations, we train our model to estimate the full conditional distribution of latitude and longitude ($f(\mathbf{y}|\mathbf{x})$), where $\mathbf{x}$ represents covariates extracted from tweets and $\mathbf{y} = (y_1, y_2)$ denotes the latitude/longitude pair.

Our dataset comprises approximately 8,000 tweets from the USA in July 2015 containing the keyword "beach". We extract 500 covariates using a bag-of-words method, focusing on the most frequent unigrams and bigrams (Manning, 2009). Given our anticipation that only a subset of the 500 covariates is pertinent to tweet location, we implement FlexCode through sparse additive models. As the response is bivariate, we employ a tensor product of Fourier basis functions.

In Figure 3.7, two instances of estimated densities are illustrated. FlexCode successfully determines tweet locations even in ambiguous scenarios, such as the presence of Long Beach in both California and Connecticut (as depicted in the bottom right plot of Figure 3.7).

Notably, FlexCode-SAM, based on sparse additive models, allows us to identify the most relevant covariates for location prediction. In Figure 3.7 (left), the terms "beachin", "boardwalk", and "daytona" are present in at least 33% of the estimated regression functions. Conversely, for the example on the right, relevant covariates include "long beach", "island", "long", and "haven".

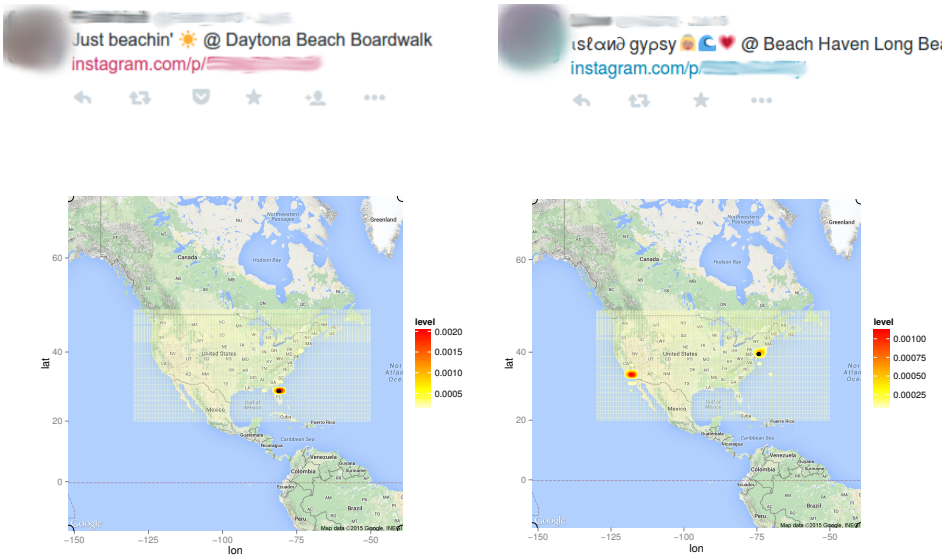


Figure 3.7: Top: Two tweets with the keyword "beach". Bottom: Level sets of the estimated probability density of the tweet locations given the content of the tweets. The black dots indicate their true locations. Image adapted from Izbicki and Lee (2017).

3.13 Summary

In this chapter, we introduced two metrics to compare the performance of estimators $\widehat{f}$ of the conditional density f: the L_2 loss and the cross-entropy loss. Both loss functions are proper and can be easily estimated if an i.i.d. test dataset is available. We then covered how, in classification, f is estimated using probabilistic classifiers tuned with an appropriate loss function. In particular, we saw that accuracy, F1-score and related metrics are not appropriate for tuning probabilistic classifiers. We also explored several approaches to estimate this conditional density in a regression context, ranging from methods that leverage existing regression methods and probabilistic classifiers (FlexCode and the Ratio trick) to methods that use modern neural network architectures (such as normalizing flows). Finally, we saw that specific quantiles of distribution $Y|\mathbf{x}$ maybe be directly estimated using quantile regression.

Chapter 4

Diagnostics and Recalibration

The Anatomy Lesson of Dr. Nicolaes Tulp. Rembrandt, 1632, Mauritshuis, The Hague.

> A correct diagnosis is
> three-fourths of the remedy.
>
> Mahatma Gandhi

In Chapter 3, we covered various conditional density estimators and discussed different loss functions to help choose the best fit for a given dataset. However, these loss functions have limitations—they don't tell us if the best estimate we found, say $\hat{f}$, is actually good enough. They also don't provide guidance on how to improve $\hat{f}$ or which specific data points ($\mathbf{x}'s$) need attention.

This chapter aims to address these practical questions that are crucial for trusting uncertainty quantification based on $\hat{f}$. We will focus on evaluating the overall quality of the estimate, figuring out how to enhance it, and identifying specific areas in the dataset that require improvement.

Secions 4.1 and 4.2 deal with regressio problems, while Section 4.3 discusses usual techniques for classification calibration.

4.1 PIT Values: Evaluating Calibration in Regression

The standard way to address the question of whether $\hat{f}$ is reasonable is to test the global consistency hypothesis:

Definition 2 (Global Consistency). *An estimate $\widehat{f}(y|\mathbf{x})$ is globally consistent with the density $f(y|\mathbf{x})$ if the following null hypothesis holds:*

$$H_0 : \widehat{f}(y|\mathbf{x}) = f(y|\mathbf{x}) \text{ for almost every } \mathbf{x} \in \mathcal{X} \text{ and } y \in \mathbb{R}. \tag{4.1}$$

This means that, under global consistency, the estimated conditional density $\widehat{f}(y|\mathbf{x})$ accurately represents the true underlying density $f(y|\mathbf{x})$ across the entire input space $\mathcal{X}$ and output space $\mathbb{R}$. Notice that, in this hypothesis, we consider $\widehat{f}$ is a particular fixed conditional density estimate, that is, there is no randomness associated to it.

In a regression setting, one way of testing H_0 is by computing PIT values:

Definition 3 (PIT). *Fix $\mathbf{x} \in \mathcal{X}$ and $y \in \mathbb{R}$. The probability integral transform of y at $\mathbf{x}$, as*

modeled by the conditional density estimate $\widehat{f}(y|\mathbf{x})$*, is*

$$PIT(y;\mathbf{x}) := \int_{-\infty}^{y} \widehat{f}(y'|\mathbf{x})dy' =: \widehat{F}(y|\mathbf{x}). \tag{4.2}$$

The PIT value therefore represents the cumulative probability, according to the estimated density $\widehat{f}$, of observing a value less than or equal to y given the input $\mathbf{x}$. See Figure 4.1 for an illustration.

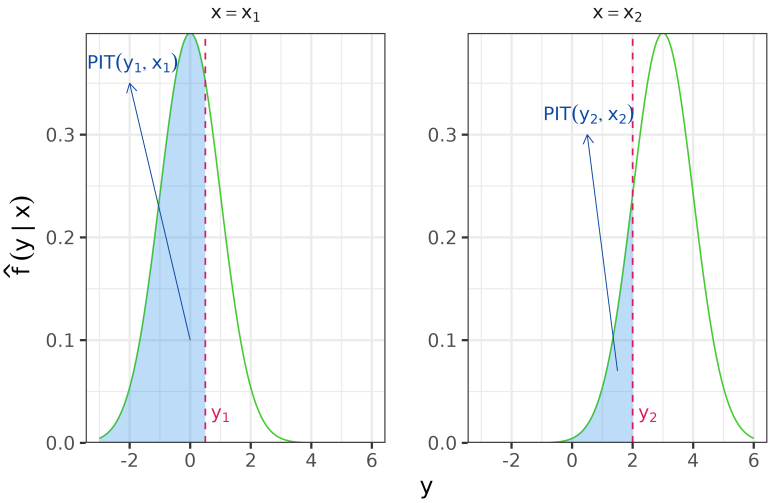


Figure 4.1: Illustration of the PIT values for two distributions, $\widehat{f}(y|\mathbf{x}_1)$ and $\widehat{f}(y|\mathbf{x}_2)$. The vertical dashed lines represent the true observed values $y = y_1$ and $y = y_2$, respectively, for each distribution. The highlighted areas indicate the regions of interest under the curves for the PIT values.

If the conditional density model $\widehat{f}(y|\mathbf{x})$ is globally consistent with the true density $f(y|\mathbf{x})$, then the PIT values should be uniformly distributed over the interval $[0,1]$. Specifically, if the null hypothesis H_0 (Equation 4.1) holds true, the random variables $\text{PIT}(Y_1;\mathbf{X}_1),\ldots,\text{PIT}(Y_n;\mathbf{X}_n)$ will be independent and identically distributed as $\text{Unif}(0,1)$. Here, $(\mathbf{X}_1,Y_1),\ldots,(\mathbf{X}_n,Y_n)$ represent a holdout set that was not used in the training of $\hat{f}$. This result follows from a fundamental property in probability theory: when you apply the cumulative distribution function of a random variable to the variable itself, the outcome is uniformly distributed on the interval $[0,1]$. Therefore, if $\widehat{f}(y|\mathbf{x})$ correctly models the true conditional density, the PIT values will reflect this

uniformity.

The uniformity of PIT values under H_0 is a widely used criterion for assessing the goodness-of-fit of conditional density models in practice (Bordoloi et al., 2010; Cook et al., 2006; Tanaka et al., 2018). For example, practitioners often plot a histogram of the PIT values to visually inspect deviations from uniformity. Additionally, formal hypothesis tests, such as the Kolmogorov–Smirnov test, can be employed to evaluate whether the PIT values follow a uniform distribution statistically.

While these methods provide valuable insights, it's important to note that an estimate $\hat{f}$ might pass the uniformity test even if it is not a good model. The following theorem illustrates this limitation:

Theorem 4 (Insensitivity to Covariate Transformations; Zhao et al. (2021)). *Suppose there exists a function $g : \mathcal{X} \longrightarrow \mathcal{Z}$, where $\mathcal{Z} \subseteq \mathbb{R}^k$ for some k, that satisfies*

$$\widehat{f}(y|\mathbf{x}) = f(y|g(\mathbf{x})). \tag{4.3}$$

Let $(\mathbf{X}, Y) \sim F_{\mathbf{X},Y}$. Then PIT$(Y;\mathbf{X}) \sim$ Unif$(0,1)$.

Many models naturally lead to estimates that could satisfy the condition in Equation 4.3, even without being globally consistent. In fact, clearly misspecified models $\widehat{f}$ can yield uniform PIT values and "pass" an associated goodness-of-fit test regardless of the sample size. To illustrate, consider a scenario where $\widehat{f}(y|\mathbf{x})$ is derived from a linear model. In this case, $\widehat{f}(y|\mathbf{x})$ will depend on $\mathbf{x} \in \mathbb{R}^d$ solely through $g(\mathbf{x}) := \beta^T\mathbf{x}$, where $\beta \in \mathbb{R}^d$. Consequently, it is possible to have $\widehat{f}(y|\mathbf{x}) = f(y|g(\mathbf{x}))$ even when $\widehat{f}(y|\mathbf{x})$ substantially deviates from the true distribution $f(y|\mathbf{x})$.

As another example, consider a conditional density estimator incorporating variable selection techniques (Dalmasso et al., 2020b; Izbicki and Lee, 2017; Shiga et al., 2015b). Such an estimator could satisfy $\widehat{f}(y|\mathbf{x}) = f(y|g(\mathbf{x}))$ with $g(\mathbf{x}) := (\mathbf{x})_S$, where $S \subset 1, \ldots, d$ denotes a subset of covariates. Thus, a test of the overall uniformity of PIT values is no guarantee that we are correctly modeling the relationship between y and the predictors $\mathbf{x}$.

A test that overcomes this issue is given by Zhao et al. (2021) and described in Section 4.2.

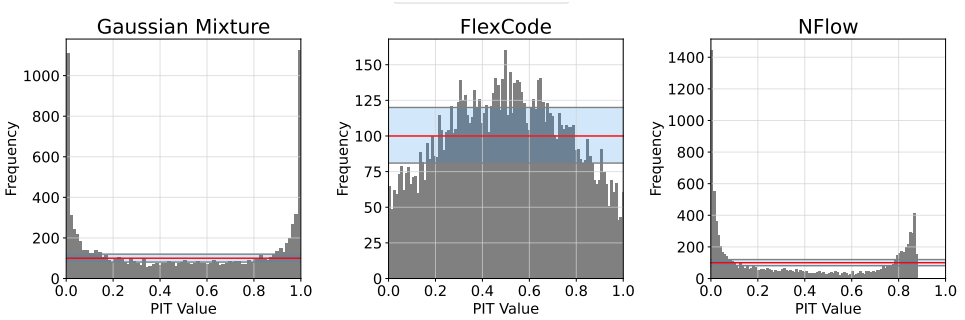


Figure 4.2: PIT histograms for the Gaussian Mixture, FlexCode, and NFlow models. Each panel shows the distribution of PIT values, with the red line representing the expected uniform distribution under proper calibration. The shaded region shows 95% confidence interval for the bar heights under the global consistency hypothesis.

4.1.1 Simulated Example: Gaussian Distribution Revisited

Figure 4.2 displays the histograms of the PIT for the three models discussed in Section 3.11. The code for this implementation is available in the CDE Notebook. Under proper model calibration, these histograms should follow a uniform distribution, represented by the red line in each panel. The shaded region indicates the 95% confidence interval for the bar heights, assuming a uniform distribution. The plot suggests that although none of the models are fully calibrated, FlexCode demonstrates a better alignment with calibration.

4.2 Conditional PIT values

The primary objective of conditional PIT values is to overcome the limitations highlighted in Theorem 4 (Zhao et al., 2021). This approach is based on the following key result:

Theorem 5 (Local Consistency and Pointwise Uniformity). *Global consistency is achieved if and only if the distribution of* $\text{PIT}(Y;\mathbf{x})$*, given* $\mathbf{x}$*, is uniform over* $(0,1)$.

Building on this result, Zhao et al. (2021) suggests testing H_0 by evaluating whether $\text{PIT}(Y;\mathbf{x})$, conditional on $\mathbf{x}$, follows a uniform distribution over $(0,1)$. This testing problem is then reformulated as follows. For each $\gamma \in (0,1)$, define the cumu-

lative distribution function of the PIT at $\mathbf{x}$ as

$$r^{\hat{f}}(\mathbf{x};\gamma) := \mathbb{P}\left(\text{PIT}(Y;\mathbf{x}) < \gamma | \mathbf{x}\right). \tag{4.4}$$

Since $r^{\hat{f}}(\mathbf{x};\gamma)$ represents the conditional cumulative distribution function of the PIT at $\mathbf{x}$, it follows that for each fixed $\mathbf{x}$,

$\widehat{f}(y|\mathbf{x}) = f(y|\mathbf{x})$ for almost every $y \in \mathcal{Y}$ if, and only if, $r^{\widehat{f}}(\mathbf{x};\gamma) = \gamma$ for every $\gamma \in (0,1)$.

In other words, global consistency holds if, and only if, $r^{\widehat{f}}(\mathbf{x};\gamma) = \gamma$ for every $\gamma \in (0,1)$. This insight suggests that we can test H_0 by estimating $r^{\widehat{f}}(\mathbf{x};\gamma)$ and assessing its deviation from γ.

To estimate $r^{\widehat{f}}$, note that this quantity can be interpreted as the regression of the indicator variable

$$W^{\gamma} := \mathbb{I}(\text{PIT}(Y;\mathbf{X}) < \gamma)$$

on $\mathbf{X}$. Therefore, $r^{\hat{f}}(\mathbf{x};\gamma)$ can be estimated by calculating $W_i^{\gamma} := \mathbb{I}(\text{PIT}(Y_i;\mathbf{X}_i) < \gamma)$ and performing a regression of W on $\mathbf{X}$ using the transformed dataset

$$\{(\mathbf{X}_1, W_1^{\gamma}), \ldots, (\mathbf{X}_n, W_n^{\gamma})\}.$$

Once $r^{\widehat{f}}$ is estimated (for simplicity, we denote its estimate by $\widehat{r}(\mathbf{x};\gamma)$), we can test H_0 using the test statistic

$$S := \frac{1}{n}\sum_{i=1}^{n} T(\mathbf{X}_i), \tag{4.5}$$

where

$$T(\mathbf{x}) := \frac{1}{|G|}\sum_{\alpha \in G}\left(\widehat{r}(\mathbf{x};\alpha) - \alpha\right)^2 \tag{4.6}$$

quantifies the deviation of $\widehat{f}(y|\mathbf{x})$ from $f(y|\mathbf{x})$ at each point $\mathbf{x}$.

The distribution of T under the null hypothesis can be approximated using Monte Carlo sampling, leveraging the fact that under H_0, $W \sim U(0,1)$. Each Monte Carlo sample consists of $(\mathbf{X}_1, W_1^b), \ldots, (\mathbf{X}_n, W_n^b)$, with $W_i^b \sim U(0,1)$. The resulting test is referred to as the Global Consistency Test (GCT).

$T(\mathbf{x})$ can also be used as a test statistic to test *local consistency* at $\mathbf{x}$, which is defined as follows:

Definition 4 (Local Consistency). *Fix $\mathbf{x} \in \mathcal{X}$. An estimate $\widehat{f}(y|\mathbf{x})$ is locally consistent*

with the density $f(y|\mathbf{x})$ at fixed $\mathbf{x}$ if the following null hypothesis holds:

$$H_0(\mathbf{x}) : \widehat{f}(y|\mathbf{x}) = f(y|\mathbf{x}) \text{ for every } y \in \mathbb{R}. \tag{4.7}$$

Local consistency therefore means that, at the specific point $\mathbf{x}$, the estimated conditional density $\widehat{f}(y|\mathbf{x})$ perfectly matches the true conditional density $f(y|\mathbf{x})$ for all possible values of y. In other words, the estimation is accurate at this particular location in the feature space.

The local null hypothesis $H_0(\mathbf{x})$ can be tested by using the test statistic $T(\mathbf{x})$ and Monte Carlo sampling to determine (an approximate) null distribution. We denote such test by Local Consistency Test (LCT).

Under certain conditions, LCT approximately controls the Type I error rate. Specifically, we require that the regression estimator used to construct $\widehat{r}$ is local in the following sense:

Assumption 3 (Local Regression Estimator). *There exists $\epsilon > 0$ such that the estimated regression at $\mathbf{x}$, denoted by $\widehat{r}(\mathbf{x}, \gamma)$, only utilizes the sample points $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ for which $\mathbf{X}_i$ lies within the ball $B(\mathbf{x}; \epsilon)$, a neighborhood around $\mathbf{x}$ with radius ϵ.*

This assumption ensures that the regression estimate $\widehat{r}(\mathbf{x}, \gamma)$ at a point $\mathbf{x}$ is influenced solely by the data points in the vicinity of $\mathbf{x}$, thus making the estimation highly localized.

Given this assumption, we can state the following theorem:

Theorem 6. *Fix $\epsilon \in \mathbb{R}$. Under the null hypothesis*

$$H_0^\epsilon(\mathbf{x}) : \widehat{f}(y|\mathbf{x}) = f(y|\mathbf{x}) \text{ for almost every } y \in \mathcal{Y} \text{ and for all } \mathbf{x}' \in B(\mathbf{x}; \epsilon),$$

and under Assumption 3, for any significance level $0 < \alpha < 1$,

$$\lim_{B \longrightarrow \infty} \mathbb{P}\left(p(\mathbf{x}) \leq \alpha\right) = \alpha,$$

where $p(\mathbf{x})$ is the Monte Carlo p-value computed using $T(\mathbf{x})$ as the test statistic.

This theorem indicates that when the regression estimator is local, the LCT will control the Type I error rate at the specified level α asymptotically. In other words, the probability of incorrectly rejecting the null hypothesis $H_0^\epsilon(\mathbf{x})$ (which asserts that the estimated conditional density $\widehat{f}(y|\mathbf{x})$ is consistent with the true density $f(y|\mathbf{x})$ within

the neighborhood $B(\mathbf{x};\epsilon)$) will approach α as the number of Monte Carlo samples B increases indefinitely.

4.2.1 Diagnostics

Conditional PIT (Probability Integral Transform) values offer valuable insights into the nature of deviations between the estimated conditional density $\hat{f}$ and the true conditional density f at any given location $\mathbf{x}$. One effective way to analyze these deviations is through the use of "amortized local P-P plots" (ALPs), which plot $\widehat{r}_\alpha(\mathbf{x})$ against α.

P-P plots (Probability-Probability plots) traditionally serve as a diagnostic tool to compare the empirical CDF of a sample with a theoretical CDF (Gibbons and Chakraborti, 2014), helping to assess how well the model aligns with the expected distribution. In the context of ALPs, these plots allow us to visualize the agreement between the estimated conditional CDF and the uniform distribution, which is expected under the true model. The alignment, or lack thereof, reveals where and how the model's performance may deviate from the true conditional density, providing detailed interpretive information about potential modes of deviation.

Figure 4.3 illustrates how these ALPs can highlight such deviations effectively. Importantly, these plots can be constructed without requiring knowledge of the true conditional density $f(y|\mathbf{x})$, making them a powerful tool for model diagnostics and interpretation.

4.2.2 Recalibration

If the estimate $\hat{f}$ is not consistent, conditional PIT values can be used to reshape it, aiming to enhance calibration. This approach aligns conceptually with the principles of smooth tests, where the rejection of the null hypothesis prompts the proposal of an alternative model that better fits the data (Algeri and Zhang, 2022; Ghosh and Bera, 2002; Neyman, 1937a; Reis and Izbicki, 2023).

This can be done by noticing that[1]

$$\text{PIT}(\cdot;\mathbf{x}) \leq \gamma \text{ if, and only if, } Y \leq \widehat{F}^{-1}(\gamma|\mathbf{x}).$$

[1]We assume that for every $\mathbf{x} \in \mathcal{X}$, both $F(y|\mathbf{x}) := \int_{-\infty}^{y} f(y'|\mathbf{x})dy'$ and $\widehat{F}(y|\mathbf{x}) := \int_{-\infty}^{y} \widehat{f}(y'|\mathbf{x})dy'$ are continuous in y, and $\widehat{F}$ dominates F; see Section 4.2.3 for details.

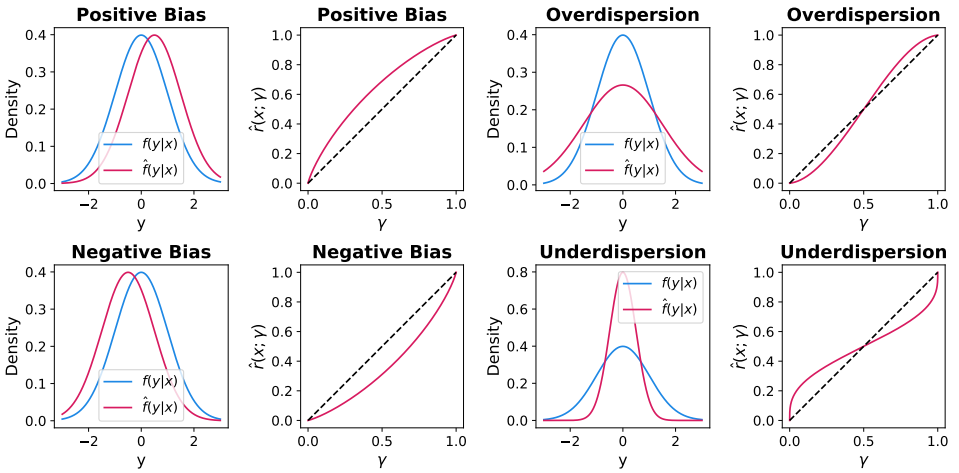


Figure 4.3: P-P plots assess the fit between a density model and actual data, revealing bias (*left panel*) and dispersion (*right panel*). "Amortized Local P-P Plots" (ALPs) compare conditional densities $\widehat{f}(y|\mathbf{x})$ at any location $\mathbf{x}$ in the feature space $\mathcal{X}$.

Thus, the conditional distribution of $\text{PIT}(Y;\mathbf{x})|\mathbf{x}$ completely determines the conditional distribution of $Y|\mathbf{x}$, which is exactly what $\hat{f}(y|\mathbf{x})$ attempts to model. Consequently, having an estimate of the distribution of $\text{PIT}(Y;\mathbf{x})|\mathbf{x}$ automatically provides an estimate of $f(y|\mathbf{x})$. The work by Dey et al. (2022) leverages this observation and incorporates the estimate of $r^{\hat{f}}(\cdot;\mathbf{x})$ to introduce a recalibration framework for $\widehat{f}(y|\mathbf{x})$ in situations where the null hypothesis H_0 is rejected.

Concretely, Dey et al. (2022) proposes that $F(y|\mathbf{x})$ is restimated through the P-P map

$$\widetilde{F}(y|\mathbf{x}) := \hat{r}^{\hat{f}}\left(\hat{F}(y|\mathbf{x});\mathbf{x}\right), \tag{4.8}$$

where $\hat{r}^{\hat{f}}$ is the regression estimator of the PIT-CDF (Equation 4.4).

By construction, if $\widehat{r} = r$ (that is, the regression r is well estimated), then

$$\widetilde{F}(y|\mathbf{x}) = F(y|\mathbf{x}),$$

that is, the true generating process will be recovered.

4.2.3 Theory

Next, we explore the relationship between the performance of the recalibrated estimator, denoted as $\widetilde{F}$, and the mean squared error of the regression function $\widehat{r}^{\widehat{f}}$. Additionally, we present convergence rates. Further details and proofs can be found in Dey et al. (2022).

We make the assumption that the true distribution of Y given $\mathbf{x}$, as well as its initial estimate, are continuous:

Assumption 4. *For every $\mathbf{x} \in \mathcal{X}$, $\widehat{F}(\cdot|\mathbf{x})$ and $F(\cdot|\mathbf{x})$ are continuous functions.*

Moreover, we assume that $\widehat{F}$ dominates F:

Assumption 5. *For every $\mathbf{x} \in \mathcal{X}$, $\widehat{F}(\cdot|\mathbf{x})$ dominates $F(\cdot|\mathbf{x})$.*

Additionally, we impose a constraint on the distribution of $F(\cdot|\mathbf{x})$, ensuring that it cannot allocate excessive mass in regions where the initial estimate $\widehat{F}(\cdot|\mathbf{x})$ places little mass:

Assumption 6. *There exists $K > 0$ such that, for every $\mathbf{x} \in \mathcal{X}$, the Radon-Nikodym derivative of $F(\cdot|\mathbf{x})$ with respect to $\widehat{F}(\cdot|\mathbf{x})$ is bounded above by K.*

Under these assumptions, the next theorem establishes a connection between the performance of the recalibrated PD and the regression function:

Theorem 7. *Under Assumptions 4, 5 and 6,*

$$\mathbb{E}\left[\int\int \left(\widetilde{F}(y|\mathbf{x}) - F(y|\mathbf{x})\right)^2 dP(y,\mathbf{x})\right] = K\mathbb{E}\left[\int\int \left(\widehat{r}^{\widehat{f}}(\gamma;\mathbf{x}) - r^{\widehat{f}}(\gamma;\mathbf{x})\right)^2 d\gamma dP(\mathbf{x})\right].$$

To derive convergence rates for the recalibrated PD, we assume a convergence rate of $O(n^{-\kappa})$ for the regression method:

Assumption 7. *The regression method used to estimate $r^{\widehat{f}}$ is such that its convergence rate is given by*

$$\mathbb{E}\left[\int\int \left(\widehat{r}^{\widehat{f}}(\gamma;\mathbf{x}) - r^{\widehat{f}}(\gamma;\mathbf{x})\right)^2 d\gamma dP(\mathbf{x})\right] = O\left(\frac{1}{n^{\kappa}}\right)$$

for some $\kappa > 0$.

Many methods satisfy Assumption 7 for some value κ, typically related to the dimension of $\mathcal{X}$ and the smoothness of the true regression r (see, for instance, Györfi et al. 2002).

Under these assumptions, we have the following the rate of convergence for $\widetilde{F}$:

Corollary 2. *Under Assumptions 4, 5, 6 and 7,*

$$\mathbb{E}\left[\int\int\left(\widetilde{F}(y|\mathbf{x})-F(y|\mathbf{x})\right)^2 dP(y,\mathbf{x})\right]=O\left(\frac{1}{n^{\kappa}}\right).$$

4.2.4 Monotonic Neural Networks to Estimate the Regression Function

Dey et al. (2022) estimate $r^{\hat{f}}(\gamma;\mathbf{x})$ by improving upon Zhao et al., 2021. The key idea is to first augment the calibration data $(\mathbf{X}_1,Y_1),\ldots,(\mathbf{X}_n,Y_n)$ by drawing multiple values $\gamma_{i,1},\ldots,\gamma_{i,K}\sim U(0,1)$ for each data point ($i=1,\ldots,n$), and then regress the random variable

$$W_{i,j}:=\mathbb{I}(\text{PIT}(Y_i;\mathbf{X}_i)\leq\gamma_{i,j})$$

on both $\mathbf{X}_i$ and $\gamma_{i,j}$ using the augmented calibration sample $\mathcal{D}'=\{(\mathbf{X}_i,\gamma_{i,j},W_{i,j})\}_{i,j}$, for $i=1,\ldots,n$ and $j=1,\ldots,K$. As $r^{\hat{f}}(\gamma;\mathbf{x})$ is a non-decreasing function of γ, Dey et al. (2022) use monotonic neural networks (Wehenkel and Louppe, 2019) for most applications with complex inputs, though any other suitable regression method may be used.

Python code to implement this full diagnostics and recalibration framework can be found at https://github.com/lee-group-cmu/Cal-PIT.

4.2.5 Example: Neural Density Inference for Galaxy Images

In this toy example, $\mathbf{x}\in\mathbb{R}^{400}$ represents an image of an elliptical galaxy generated by `GalSim`, an open-source toolkit for simulating realistic images of astronomical objects (Rowe et al., 2015). Our response variable is y, the galaxy's rotation angle with respect to the x-axis. For illustration, our data is simulated as a mixture of a larger population with $\lambda=0.7$ (spheroidal galaxies), and a smaller population with $\lambda=0.1$ (elongated galaxies). We then simulate a sample of images as follows: first, we draw λ and y from a prior distribution given by

$$\mathbb{P}(\lambda=0.7)=1-\mathbb{P}(\lambda=0.1)=0.9$$
$$Y\sim Unif(-\pi,\pi)$$

Then we sample 20×20 galaxy images $\mathbf{X}$ according to the data model $\mathbf{X}|\lambda, y \sim \texttt{GalSim}(a, \lambda)$, where

$$a|\lambda = 0.7 \sim N(y, 0.05)$$
$$a|\lambda = 0.1 \sim 0.5Laplace(y, 0.05) + 0.5Laplace(y, 0.0005).$$

We then fit a convolutional mixture density network (ConvMDN; see Section 3.5), which gives us an estimate of $f(y|\mathbf{x})$. We allow K, the number of mixture components, to vary. According to the KL divergence loss computed on a separate test sample with 1000 images, the best fit of $f(\theta|\mathbf{x})$ is achieved by a ConvMDN model with $K = 7$.

Here, the ConvMDN model with the smallest KL loss fails the global test that uses the statistic in Equation 4.5 ($p < 0.001$), so we turn to the diagnostics described in Section 4 to understand why. Figure 4.4 plots the test galaxy images along their first two principal components. The local tests of Equation 4.6 show that the ConvMDN model generally fits the density well for the main population of spheroidal galaxies ($\lambda = 0.7$), but fails to properly model the smaller population of elongated galaxies ($\lambda = 0.1$). P-P plots at selected test points indicate severe bias in the density estimates for the $\lambda = 0.1$ population. These plots suggest that an effective way of obtaining a better approximation of the density is by improving the fit for the $\lambda = 0.1$ population (by obtaining more data in that region of the feature space, using a different model class, etc). For instance, CDE models not based on mixtures could be more effective.

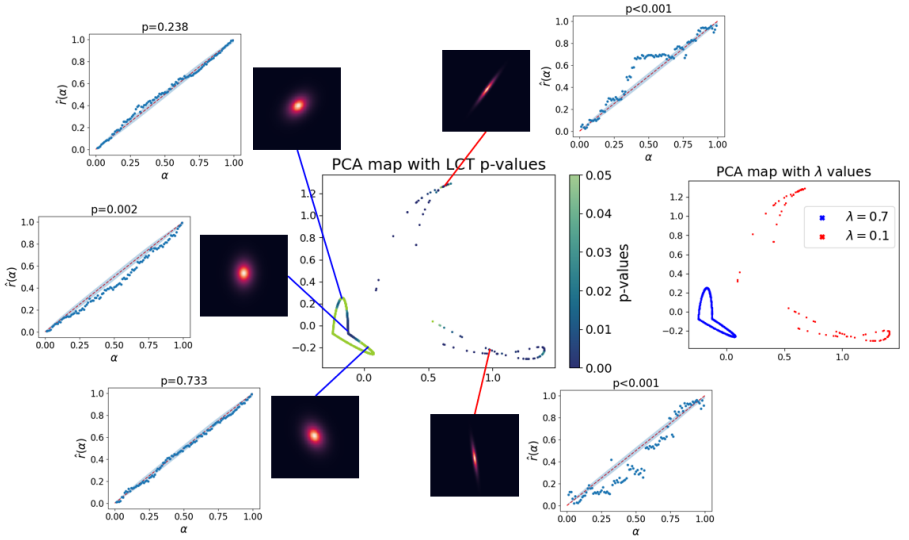


Figure 4.4: Diagnostics for conditional density estimation: Visualization of test galaxy points in $\mathbb{R}^{400}$ using the first two components (center panel, "PCA map with LCT p-values"). ConvMDN's P-values indicate good fit for 90% of spheroidal galaxies ($\lambda = 0.7$) but poor fit for the 10% elongated galaxies ($\lambda = 0.1$). Local P-P plots reveal significant deviations in CDEs for the latter group, highlighting the need for improved density approximations. Figure adapted from Zhao et al. (2021).

4.3 Calibration of Classification Models

Even though proper loss functions encourage calibration, the obtained classifiers are often not calibrated (Gneiting and Katzfuss, 2014; Vaicenavicius et al., 2019). Thus, in practice, it is common to check whether the probabilistic classifier is *calibrated* (Dawid, 1982; Gupta et al., 2020; Murphy and Epstein, 1967). In the context of binary classification, calibration is often defined as follows:

Definition 5 (Marginal calibration)**.** *A probabilistic binary classifier* $g : \mathcal{X} \longrightarrow [0, 1]$ *is marginally calibrated if, and only if,*

$$\mathbb{P}(Y = 1 | g(\mathbf{X})) = g(\mathbf{X}) \text{ (almost surely)}. \tag{4.9}$$

Equation 4.9 is often referred to as *calibration* or *perfect calibration*.

4.3.1 Evaluating Marginal Calibration

There are many ways to evaluate whether a probabilistic binary classifier $g : \mathcal{X} \longrightarrow [0, 1]$ is calibrated in the sense of Definition 5:

Calibration Plot (Reliability Diagram). A calibration plot, also known as reliability diagram (Zadrozny and Elkan, 2002), is a graphical tool that compares the estimated probabilities $g(\mathbf{x})$ with the actual outcomes on a holdout set. The predicted probabilities are partitioned into bins, and within each bin, the average predicted probability is compared with the observed frequency of the positive class. Formally, given a set of estimated probabilities $\{g(\mathbf{x}_i)\}_{i=1}^n$ evaluated on a holdout set, the steps are as follows:

1. Sort the predicted probabilities $\{g(\mathbf{x}_i)\}_{i=1}^n$.
2. Divide the sorted probabilities into J contiguous bins.
3. For each bin B_j, calculate the observed proportion of sample points that fall into B_j:

$$\hat{o}_j = \frac{1}{|B_j|} \sum_{i \in B_j} y_i,$$

and the average estimated probability in B_j:

$$\hat{p}_j = \frac{1}{|B_j|} \sum_{i \in B_j} p_i.$$

4. Create a scatter plot of $(\hat{p}_j, \hat{o}_j)$.

By construction, $\hat{o}_j$ is an estimate of $\mathbb{P}(Y = 1 | g(\mathbf{X}) \in B_j)$. Thus, if g is marginally calibrated, we expect that $\hat{p}_j \approx \hat{o}_j$ for all j. Deviations from the diagonal $\hat{p}_k = \hat{o}_k$ indicate miscalibration.

Expected Calibration Error (ECE). The Expected Calibration Error is a summary statistic derived from the calibration plot that quantifies the difference between predicted probabilities and observed frequencies across bins:

$$\text{ECE} = \sum_{j=1}^{J} \frac{|B_j|}{n} \left| \hat{o}_j - \hat{p}_j \right|$$

A lower ECE indicates better calibration.

Maximum Calibration Error (MCE). The Maximum Calibration Error is a more conservative summary statistic, focusing on the worst-case deviation across all bins:

$$\text{MCE} = \max_{j=1,\ldots,J} \left| \hat{o}_j - \hat{p}_j \right|.$$

Again, low MCE values indicate better calibration.

If one concludes that g is not calibrated, it can be recalibrated. The next section discusses ways to recalibrate g.

4.3.2 Recalibration of Probabilistic Classifiers

Several methods exist to recalibrate g to improve its marginal calibration. Below are a few approaches:

Platt scaling. Let $g : \mathcal{X} \longrightarrow [0, 1]$ be a binary probabilistic classsifer that outputs a score $g(\mathbf{x})$ for each input $\mathbf{x}$. Platt scaling (Platt et al., 1999) fits a logistic regression model to these scores, aiming to find a transformation that better reflects the true probabilities.

The recalibratied probabilities $g'(\mathbf{x})$ is given by:

$$g'(\mathbf{x}) = \frac{1}{1+\exp(\alpha g(\mathbf{x})+\beta)},$$

where α and β are parameters determined by fitting the logistic regression model on a labeled dataset $\{(g(\mathbf{x}_i), y_i)\}_{i=1}^n$.

Histogram binning. Histogram binning (Zadrozny and Elkan, 2001) is a non-parametric version of Platt scalling. The idea is to use the calibration plot (Section 4.3.1) to recalibrate g. Specifically, for each $\mathbf{x} \in \mathcal{X}$, the recalibrated version of g is given by

$$g'(\mathbf{x}) = \hat{p}_j,$$

where j is the bin B_j that contains $\mathbf{x}$.

Isotonic regression. Isotonic regression (Zadrozny and Elkan, 2002) fits a non-decreasing function to the estimated probabilities $g(\mathbf{x})$, ensuring that the recalibrated probabilities are better aligned with the observed frequencies of the positive class. Specifically, given a set of estimated probabilities $\{g(\mathbf{x}_i)\}_{i=1}^n$, isotonic regression recelibrates g by finding the function $g'(\mathbf{x})$ that minimizes the Brier Score:

$$\sum_{i=1}^{n} (\hat{g}(\mathbf{x}_i) - y_i)^2$$

subject to the constraint

$$g'(\mathbf{x}_{i+1}) \geq g'(\mathbf{x}_i) \text{ for all } i.$$

Temperature scaling. Temperature scaling is a simple recalibration technique used primarily with neural networks (Guo et al., 2017). It recalibrates $g(\mathbf{x})$ by reescaling its logit by a constant. Specifically, let

$$\text{logit}(p) = \log \frac{p}{1-p}$$

be the logit of p. Temperature scaling recalibrates g according to

$$\text{logit}(g'(\mathbf{x})) = \frac{1}{T}\text{logit}(g(\mathbf{x})),$$

where T is the temperature parameter, typically chosen by minimizing the negative

log-likelihood or the Brier Score on a validation set. $T > 1$ smooth the predicted probabilities, reducing overconfidence in the predictions. In particular, when $T \longrightarrow \infty$, $g'(\mathbf{x}) = 1/2$. When $T = 1$, no transformation is made.

4.3.3 Limitations of Marginal Calibration

A classifier can be marginally calibrated even if it disregards information in $\mathbf{x}$. This parallels the scenario in regression (Theorem 4). For example, $g(\mathbf{X}) := \mathbb{P}(Y = 1)$ is marginally calibrated. In fact, g is calibrated if and only if there exists a space $\mathcal{Z}$ and a function $\lambda : \mathcal{X} \longrightarrow \mathcal{Z}$ such that

$$\mathbb{P}(Y = 1|\lambda(\mathbf{X})) = g(\mathbf{X}) \quad \text{(almost surely)};$$

see Vaicenavicius et al. 2019, Proposition 1 and Gupta et al. 2020, Proposition 1. However, marginal calibration does not imply that g fully utilizes the information in $\mathbf{x}$. Specifically, it does not ensure that the *Bayes classifier*[2] is recovered, that is, $g(\mathbf{X})$ can be very different from $\mathbb{P}(Y = 1|\mathbf{X})$. Moreover, it does not guarantee fairness; it is possible that $\mathbb{P}(Y = 1|\mathbf{X} \in A, g(\mathbf{X})) \neq g(\mathbf{X})$, where $A \subset \mathcal{X}$ represents a subgroup of interest.

Therefore, we suggest that marginal calibration should be evaluated only after selecting the best probabilistic classifier based on a suitable proper loss function, as discussed in Section 3.1. Marginal calibration alone should not drive classifier selection, as it may not reflect the optimal exploitation of information in $\mathbf{x}$.

4.3.4 Example

In this toy example, we apply several calibration methods to a logistic regression model trained on synthetic data generated as follows:

1. The input features X were drawn from a standard normal distribution:

$$X_i \sim \mathcal{N}(0, 1) \quad \text{for } i = 1, 2, \ldots, n.$$

2. The binary labels Y were generated according to a logistic model with a non-linear transformation of the input features. Specifically, the probability of the

[2]Here, $\mathbb{P}(Y = 1|\mathbf{X})$ is referred to as the Bayes classifier.

positive class was given by:

$$\mathbb{P}(Y_i = 1|x_i) = \frac{\exp\left(x_i^3\right)}{1 + \exp\left(x_i^3\right)}.$$

The labels were then sampled as:

$$Y_i \sim \text{Bernoulli}\left(\mathbb{P}(Y_i = 1 \mid X_i)\right).$$

3. A logistic regression model was trained on the training set to predict the probability of the positive class $\mathbb{P}(Y = 1|x)$.

We used 1000 sample points to train the model, and 2000 to calibrate it, as well as to evaluate the performance of the calibration method. The full details are shown in the Classification Calibration Notebook.

The calibration plots in Figure 4.5 and the metrics in Table 4.1 show that histogram binning is the only method that enhances both proper loss functions and marginal calibration. However, it is important to note that histogram binning inherently achieves perfect calibration because its recalibration is specifically designed to produce a perfect calibration plot. Additionally, while isotonic regression improves calibration, it worsens the loss functions. This happens because the calibration plot is non-monotonic, causing probabilities in the range of approximately 25% to 75% to be mapped to 50%.

Table 4.1: Comparison of Calibration Methods: Expected Calibration Error (ECE), Maximum Calibration Error (MCE), Negative Brier Score, and Log Loss for different calibration methods applied to a logistic regression model. Lower values indicate better performance.

Method	ECE	MCE	Brier Score	Log Loss
Before Calibration	0.0854	0.2723	0.1961	0.5661
Isotonic	0.0266	0.1766	0.2504	0.6939
Platt	0.0958	0.2246	0.1985	0.5747
Histogram Binning	0.0000	0.0000	0.1855	0.5308
Temperature Scaling	0.0810	0.2032	0.1957	0.5670

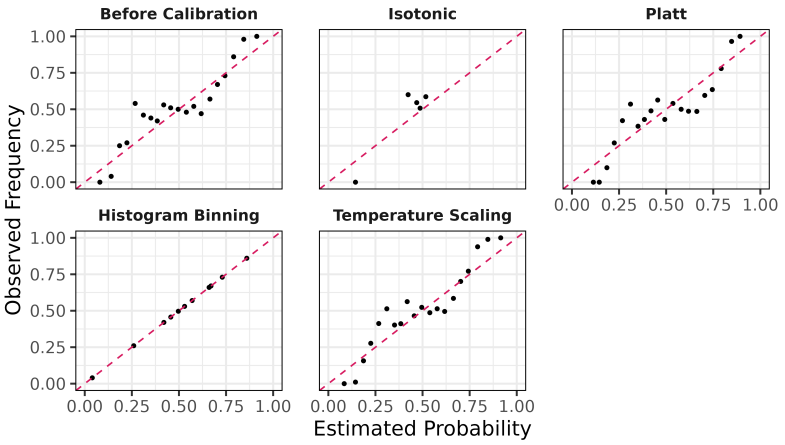


Figure 4.5: Calibration plots for different recalibration methods applied to a logistic regression model. Each plot compares the estimated probabilities against the observed frequencies on a validation set. The diagonal line represents perfect marginal calibration, where estimated probabilities match the observed frequencies.

4.4 Summary

In this chapter, we explored techniques for assessing and improving the accuracy of conditional density estimates and classification models. We began by introducing the concept of Probability Integral Transform (PIT) values, a straightforward diagnostic tool used to evaluate whether a conditional density estimator, $\hat{f}$, is globally consistent with the true density f. While PIT values provide a useful first check – where uniformity suggests a good fit – we highlighted the limitations of this method. Specifically, we demonstrated that even poorly specified models might pass the PIT uniformity test, making it an insufficient criterion for model validation.

To address these shortcomings, we introduced *Conditional PIT values*. Unlike global PIT, which assesses uniformity across the entire dataset, Conditional PIT values focus on the distribution of PIT values given specific covariates. This more granular approach allows us to identify where the model deviates from the true conditional density, offering a clearer path to improvement. We discussed how to use these insights to recalibrate the model, ensuring better alignment with the underlying data structure.

In parallel, we examined calibration in classification models. Calibration is crucial

for ensuring that the predicted probabilities from a classifier are reliable. We defined *marginal calibration*—where the predicted probability of an event closely matches the observed frequency—and presented several methods to assess and improve calibration, including *Platt scaling*, *Histogram binning*, *Isotonic regression*, and *Temperature scaling*. Through examples, we showed how these methods could be applied to recalibrate classifiers, improving both the model's reliability and its predictive accuracy.

A key takeaway from this chapter is that *calibration*—whether in the context of regression or classification—should be considered only after selecting the best model using appropriate loss functions. Calibration alone does not guarantee that a model fully captures the relationships within the data or that it leverages all available information. Therefore, it should be seen as the final step in the modeling process, ensuring that the selected model provides trustworthy predictions.

Chapter 5

From Conditional Densities to Prediction Regions

Color Study. Squares with Concentric Circles. Wassily Kandinsky, 1913, Städtische Galerie im Lenbachhaus, Munich.

> A good forecaster is not smarter than everyone else, he merely has his ignorance better organized.
>
> Unknown Author

Oftentimes, the end goal of a supervised learning model is to provide prediction regions for new sample points. Prediction regions are sets of possible outcomes that reflect the range within which the true outcome is likely to fall, given a new sample point. These regions provide a more complete picture of the model's predictions by offering a range rather than a point estimate, thus incorporating uncertainty into the prediction process.

Formally, a prediction region is defined as follows:

Definition 6 (Prediction region)**.** *A prediction region (or* prediction set*, or* prediction band*) is a function $R : \mathcal{X} \longrightarrow \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ is the set of the parts of $\mathcal{Y}$.*

In this chapter, we will begin by exploring the concept of optimal (also known as oracle) prediction regions derived from different loss functions. This understanding will lay the foundation for developing and assessing various methods for estimating prediction regions in practice. Next, we will examine techniques for estimating these regions, focusing on conformal methods, which offer guaranteed coverage, making them robust tools for uncertainty quantification.

5.1 Optimal Prediction Regions

To define optimal prediction regions, we start by introducing a loss function, denoted as

$$L : \mathcal{P}(\mathcal{Y}) \times \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R},$$

where $L(R; (\mathbf{X}, Y))$ quantifies the cost associated with a prediction region $R \subseteq \mathcal{Y}$ when the input $\mathbf{X}$ and true output Y are observed.

The goal is to find an optimal prediction region that minimizes the overall ex-

pected loss, also known as the risk. The risk function is defined as:

$$\mathcal{R} : (\mathcal{X} \longrightarrow \mathcal{P}(\mathcal{Y})) \longrightarrow \mathbb{R},$$
$$R \longmapsto \mathcal{R}(R) := \mathbb{E}_{\mathbf{X},Y}[L(R;(\mathbf{X},Y))],$$

where $\mathcal{R}(R)$ represents the expected loss or risk associated with a prediction region R over the joint distribution of $\mathbf{X}$ and Y. The expectation is taken with respect to the distribution of both the input $\mathbf{X}$ and the output Y, making the risk function a measure of how well the prediction region R performs on average.

Different choices of the loss function L lead to different forms of optimal prediction regions. For example, some loss functions might penalize regions that are too large or too small, while others might emphasize the inclusion of the true output Y within the region. The next examples will illustrate common types of prediction regions that arise from different loss functions, as shown in Figure 5.1. These examples demonstrate how varying the loss function influences the shape and properties of the optimal prediction region, reflecting different practical considerations. The optimal regions are also known as *oracle regions*, since they require knowledge of $f(y|\mathbf{x})$ or, at the very least, certain properties of it.

Exemple 5.1 (HPD Region). Consider a loss function defined as

$$L(R,(\mathbf{x},y)) = \mathbb{I}(y \notin R) + \lambda \text{Vol}(R),$$

where $\mathbb{I}(y \notin R)$ is 1 if the true outcome y is not in the prediction region R, and 0 otherwise. The term $\text{Vol}(R)$ represents the size of the region, and $\lambda > 0$ balances the trade-off between including y in R and keeping R small. The prediction region that minimizes this loss is the highest predictive density (HPD) region:

$$R(\mathbf{x}) = \{y \in \mathcal{Y} : f(y|\mathbf{x}) > \lambda\},$$

where $f(y|\mathbf{x})$ is the conditional density of y given $\mathbf{x}$. The HPD region includes the most likely values of y and balances accuracy with the region's size.

HPD regions are applicable in both regression and classification settings. In regression, different loss functions can lead to various types of prediction regions.

Exemple 5.2 (Quantile-based Region). Consider the case where $\mathcal{Y} \subset \mathbb{R}$. Fix $\alpha \in (0,1)$ and assume that the prediction region is an interval, $R(\mathbf{x}) = (a(\mathbf{x}), b(\mathbf{x}))$. The optimal

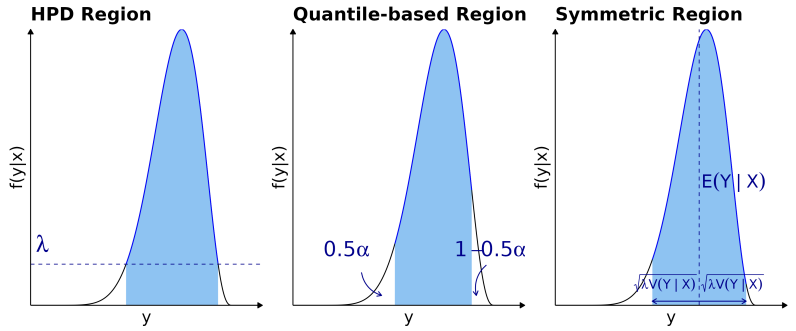


Figure 5.1: Illustration of the optimal prediction regions corresponding to different loss functions: the highest predictive density (HPD) region from Example 5.1, the quantile-based region from Example 5.2, and the symmetric region from Example 5.3. Each plot demonstrates how the choice of loss function influences the shape and properties of the resulting prediction region.

prediction region that minimizes the risk for the loss function

$$L((a,b),(\mathbf{x},y)) = \alpha\frac{b(\mathbf{x})-a(\mathbf{x})}{2} + [(a(\mathbf{x})-y)_+ + (y-b(\mathbf{x}))_+]$$

is the quantile-based interval:

$$R(\mathbf{x}) = (q_{\alpha/2}(\mathbf{x}), q_{1-\alpha/2}(\mathbf{x})),$$

where $q_\gamma(\mathbf{x}) = F^{-1}(\gamma|\mathbf{x})$ represents the γ-th conditional quantile of Y given $\mathbf{x}$. This interval captures the middle α portion of the conditional distribution of Y, providing a balanced region that minimizes both the width of the interval and the penalty for excluding y.

Exemple 5.3 (Symmetric Region). Consider the case where $\mathcal{Y} \subset \mathbb{R}$. Fix $\lambda > 0$, and assume that the prediction region is an interval, $R(\mathbf{x}) = (a(\mathbf{x}), b(\mathbf{x}))$. The prediction region that minimizes the risk for the loss function

$$L((a,b),(\mathbf{x},y)) = \lambda\frac{b(\mathbf{x})-a(\mathbf{x})}{2} + \left[y - \frac{a(\mathbf{x})+b(\mathbf{x})}{2}\right]\frac{2}{b(\mathbf{x})-a(\mathbf{x})}$$

is the symmetric interval:

$$R(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}] \pm \sqrt{\lambda\mathbb{V}[Y|\mathbf{x}]}.$$

This interval is centered around the expected value $\mathbb{E}[Y|\mathbf{x}]$, with a width proportional to the standard deviation $\sqrt{\mathbb{V}[Y|\mathbf{x}]}$, scaled by the factor $\sqrt{\lambda}$. The result is a prediction region that symmetrically balances the interval's width with the distribution's variance.

The three types of prediction regions discussed offer unique strengths and limitations:

HPD Region: These regions are particularly effective for capturing the most probable outcomes, making them well-suited for skewed or multimodal distributions. However, they can result in irregularly shaped regions that are computationally challenging to handle/report.

Quantile-based Region: Quantile-based intervals are straightforward to compute and interpret, requiring only two numbers to represent the interval. However, these regions might lead to wide regions in multimodal cases.

Symmetric Region: Symmetric regions are easy to compute and are particularly effective for symmetric distributions, also needing just two numbers for representation. However, they can become excessively large when applied to skewed distributions.

5.1.1 Tuning Parameters and Coverage

In practice, the tuning parameters involved in the loss functions, such as λ in Example 5.1 are chosen in a way such that the resulting prediction region has the desired coverage. More specifically, typically, for each $\mathbf{x}$, λ is chosen so that, in theory, *conditional coverage* is achieved:

$$\mathbb{P}(Y \in R(\mathbf{X})|\mathbf{X}) = 1 - \alpha \text{ (a.e.)}. \tag{5.1}$$

If $R(\mathbf{x})$ satisfies conditional coverage, *marginal coverage* is also achieved:

$$\mathbb{P}(Y \in R(\mathbf{X})) = 1 - \alpha. \tag{5.2}$$

In the next sections, we explore different methods to approximate optimal regions in practice.

5.2 Plug-in Prediction Regions

One straightforward approach to constructing prediction regions involves leveraging an initial estimate of the conditional density function, $\widehat{f}$, and directly plugging it into the expression for the optimal region. For example, an estimated Highest Predictive Density (HPD) region can be obtained as:

$$\{y : \widehat{f}(y|\mathbf{x}) \geq t_{1-\alpha}(\mathbf{x})\},$$

where the threshold $t_{1-\alpha}(\mathbf{x})$ is chosen such that:

$$\int_{\{y:\widehat{f}(y|\mathbf{x})\geq t_{1-\alpha}(\mathbf{x})\}} \widehat{f}(y|\mathbf{x})\, dy = 1-\alpha.$$

Similarly, prediction regions can be derived without needing full knowledge of $f(y|\mathbf{x})$ by directly estimating critical quantities. For instance, a quantile regressor that provides estimates $\widehat{q}_{\alpha/2}$ and $\widehat{q}_{1-\alpha/2}$ can be used to construct the prediction region:

$$\{y : \widehat{q}_{\alpha/2}(\mathbf{x}) < y < \widehat{q}_{1-\alpha/2}(\mathbf{x})\}. \tag{5.3}$$

While these quantiles can also be derived using $\widehat{f}$, this direct estimation offers a more practical approach in many scenarios.

Plug-in prediction regions are designed primarily to capture the aleatoric uncertainty inherent in the data, which refers to the variability in predictions due to the inherent randomness of the outcome given the features. However, since the quantities such as the conditional density $f(y|\mathbf{x})$ or the quantiles $q_{\alpha/2}(\mathbf{x})$ are only estimated from data, the resulting prediction regions often do not achieve the correct coverage. This is because the estimates may not fully account for the epistemic uncertainty, which arises from the uncertainty in model parameters or structure.

In the following sections, we will explore methods that build upon these estimates and also incorporate epistemic uncertainty, leading to prediction regions that more accurately reflect the true uncertainty and are more likely to achieve the desired coverage.

5.3 Conformal Regions

The main goal in conformal predictions is to use the data to obtain a valid prediction region (that is, a prediction region whose coverage matches its nominal coverage), $R(\mathbf{X}_{n+1})$, under very few assumptions. Indeed, typically, validity depends only the i.i.d. assumption (Angelopoulos, Bates, et al., 2023; Shafer and Vovk, 2008; Vovk et al., 2005).

A widely employed approach for constructing such prediction regions is the split method (Lei et al., 2018; Papadopoulos et al., 2002; Vovk, 2012). In this method, the data is divided into two sets: the training set, $\mathcal{D}_1$, and the calibration set, $\mathcal{D}_2$. We assume that the size of the calibration set is $|\mathcal{D}_2| = n$. Typically, the calibration set can be much smaller than the training set; see Section 5.3.1 for some guidance on how to make such split.

After the data has been split, a non-conformity score $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is trained using $\mathcal{D}_1$. The function $h(\mathbf{x}, y)$ serves to quantify the extent to which a given label value y aligns with the feature values $\mathbf{x}$ of an instance. A high value of $h(\mathbf{x}, y)$ suggests that the appearance of label y in an instance characterized by features $\mathbf{x}$ is improbable. In the context of regression, a few examples of non-conformity scores are:

- **[Regression-split]** $h(\mathbf{x}, y) = |y - \widehat{r}(\mathbf{x})|$, where $\widehat{r}$ is an estimate of the regression of Y on $\mathbf{x}$ (Lei et al., 2018)
- **[Weighted]** $h(\mathbf{x}, y) = \frac{|y - \widehat{r}(\mathbf{x})|}{\hat{\rho}(\mathbf{x})}$, where $\hat{\rho}(\mathbf{x})$ is any estimator of the conditional mean absolute deviation of $|Y - \hat{\mu}(\mathbf{X})||\mathbf{X} = \mathbf{x}$ (Lei et al., 2018)
- **[CD-split and variations]** $h(\mathbf{x}, y) = -\widehat{f}(y|\mathbf{x})$ (Izbicki et al., 2022; Izbicki et al., 2020; Lei and Wasserman, 2014)
- **[HPD-split]** $h(\mathbf{x}, y) = -\int_{\{y' : \hat{f}(y'|\mathbf{x}) \leq \hat{f}(y|\mathbf{x})\}} \hat{f}(y'|\mathbf{x}) dy'$ (Izbicki et al., 2022)
- **[CQR]** $h(\mathbf{x}, y) = \max\{\widehat{q}_{\alpha_1}(\mathbf{x}) - y, y - \widehat{q}_{\alpha_2}(\mathbf{x})\}$, where $\widehat{q}_{\alpha_1}, \widehat{q}_{\alpha_2}, \alpha_1 < \alpha_2$, are quantile estimates (Romano et al., 2019)
- **[CDF-split]** $h(\mathbf{x}, y) = |\widehat{F}(y|\mathbf{x}) - 1/2|$ where $\widehat{F}$ is a CDF estimate (Chernozhukov et al., 2021)

Note that the non-conformity scores involving conditional densities can also be applied to classification tasks (see Section 5.3.5). For non-conformity scores that handle

multivariate responses with mixed data types, refer to Dheur et al. (2024). Additionally, Manokhin (2022) provides a comprehensive list of papers on conformal predictions.

The conformal region has the shape

$$R(\mathbf{x}_{n+1}) = \{y : h(\mathbf{x}_{n+1}, y) \leq t\},$$

which means it consists of all y that are highly conformal to $\mathbf{x}$. The non-conformity score h determines the shape and properties of the prediction region.

For example, regression-split always leads to homoscedastic intervals centered around $\widehat{r}(\mathbf{x})$:

$$R(\mathbf{x}_{n+1}) = [\widehat{r}(\mathbf{x}_{n+1}) - t, \widehat{r}(\mathbf{x}_{n+1}) + t].$$

On the other hand, using $h(\mathbf{x}, y) = -\widehat{f}(y|\mathbf{x})$ can result in regions that are not necessarily intervals; they can be unions of intervals, for instance. CDF-split leads to prediction intervals around the median of $Y|\mathbf{x}$:

$$R(\mathbf{x}_{n+1}) = [\widehat{F}^{-1}(1/2 - t|\mathbf{x}_{n+1}), \widehat{F}^{-1}(1/2 + t|\mathbf{x}_{n+1})].$$

CQR, meanwhile, results in a calibrated version of quantile-based prediction intervals (Equation 5.3):

$$R(\mathbf{x}_{n+1}) = [q_{\alpha_1}(\mathbf{x}_{n+1}) - t, q_{\alpha_2}(\mathbf{x}_{n+1}) + t].$$

Some examples of prediction regions given by various non-conformity scores are given in Figure 5.2.

After selecting the non-conformity score, the next step is to determine the cutoff t. The split conformal method proceeds as follows. First, we compute the random variable $U_i := h(\mathbf{X}_i, Y_i)$ for each sample point $(\mathbf{X}_i, Y_i) \in \mathcal{D}_2$. The cutoff t is then set to be

$$t := U_{\lceil 1-\alpha \rceil},$$

the $\lceil n(1-\alpha) \rceil$ order statistic among $U_1, \ldots, U_n$. The next theorem shows that split conformal regions have the right marginal coverage:

Theorem 8. *(Lei et al., 2018; Papadopoulos et al., 2002; Vovk, 2012) If the data is i.i.d., the prediction region*

$$R(\mathbf{X}_{n+1}) = \left\{y : h(\mathbf{X}_{n+1}, y) \leq U_{\lceil 1-\alpha \rceil}\right\}$$

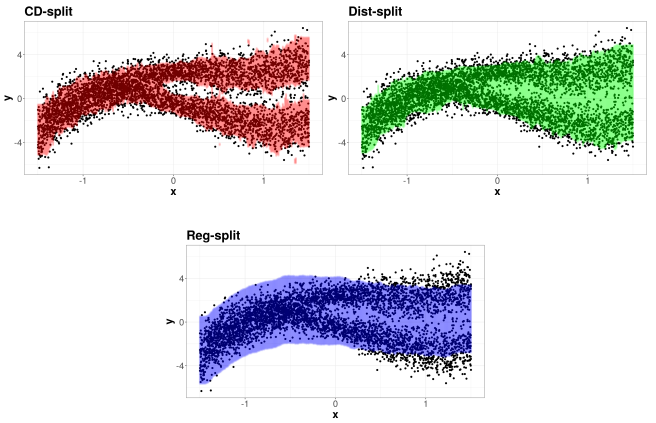


Figure 5.2: Prediction regions given by the following methods: CD-Split, which is based on $\widehat{f}(y|\mathbf{x})$ (Izbicki et al., 2020)), Dist-split, which is based on $\widehat{F}(y|\mathbf{x})$ (Izbicki et al., 2020)), and Regression-split, which is based on $\widehat{r}(\mathbf{x})$ (Lei et al., 2018)).

is a marginally valid prediction region, that is,

$$\mathbb{P}\left(Y_{n+1} \in R(\mathbf{X}_{n+1})\right) \geq 1-\alpha.$$

Proof. Because the data is i.i.d., the rank of U_{n+1} is uniform among $\{1,\ldots,n+1\}$. It follows that

$$\begin{aligned}\mathbb{P}\left(Y_{n+1} \in R(\mathbf{X}_{n+1})\right) &= \mathbb{P}\left(Y_{n+1} \in \left\{y : h(\mathbf{X}_{n+1}, y) \leq U_{\lceil 1-\alpha \rceil}\right\}\right) \\ &= \mathbb{P}\left(h(\mathbf{X}_{n+1}, Y_{n+1}) \leq U_{\lceil 1-\alpha \rceil}\right) = \mathbb{P}\left(U_{n+1} \leq U_{\lceil 1-\alpha \rceil}\right) \geq 1-\alpha.\end{aligned}$$

□

Remarkably, Theorem 8 does not assume that the non-conformity score needs to be correctly specified. For instance, the regression function $r(\mathbf{x})$ does not be well estimated if the score $h(\mathbf{x}, y) = |y - \widehat{r}(\mathbf{x})|$ is used.

If the non-conformity score is continuous, the coverage does not exceed $1-\alpha$ by too much:

Theorem 9. *(Lei et al., 2018) If the data is i.i.d. and $h(\mathbf{X}, Y)$ is continuous, the prediction region*

$$R(\mathbf{X}_{n+1}) = \left\{y : h(\mathbf{X}_{n+1}, y) \leq U_{\lceil 1-\alpha \rceil}\right\}$$

is such that

$$\mathbb{P}(Y_{n+1} \in R(\mathbf{X}_{n+1})) \leq 1 - \alpha + \frac{1}{n+1}.$$

From a computational perspective, after training the non-conformity score, computing prediction bands becomes straightforward. The following R code demonstrates this using the regression-split non-conformity score calculated with a K-Nearest Neighbor approach (Section 2.6.2) with $K = 10$. The conformal calibration of the prediction sets is performed in line 25. The code leads to the estimates shown in Figure 5.3.

```
1  # Load the necessary libraries
2  library(FNN)
3  library(dplyr)
4
5  # Example data
6  n <- 500
7  x <- runif(n)
8  y <- sin(2 * pi * x) + rnorm(n, sd = 0.1)
9  data <- data.frame(x = x, y = y)
10
11 # Split the data into training and calibration sets
12 train_indices <- sample(seq_len(n), size = 0.7 * n)
13 train_data <- data[train_indices, ]
14 calibration_data <- data[-train_indices, ]
15
16 # k-NN regression to define the non-conformity score
17 predictions <- FNN::knn.reg(
18   train = as.matrix(train_data$x),
19   test = as.matrix(calibration_data$x),
20   y = train_data$y,
21   k = 10
22 )
23
24 # Conformal calibration
25 t <- quantile(abs(predictions$pred - calibration_data$y), probs = 0.95)
26
27 # Generate a fine grid for the x-axis
28 grid_data <- data.frame(x_grid = seq(min(data$x), max(data$x), length.
       out = 500))
29
```

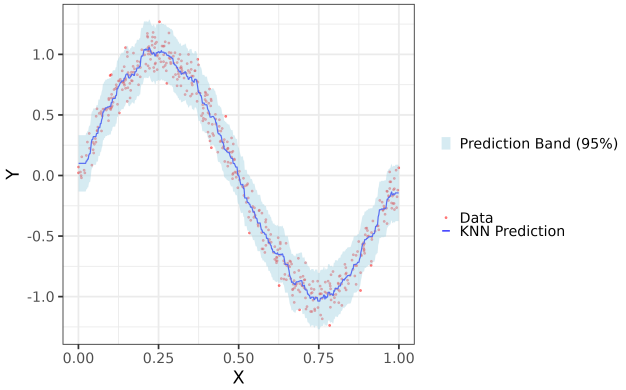


Figure 5.3: Conformal predictions with kNN regression. The light blue shaded area indicates the 95% prediction bands calculated using conformal inference, providing a measure of uncertainty around the predictions.

```r
# Evaluate predictions on the grid
grid_data <- grid_data %>%
  mutate(y_pred = FNN::knn.reg(
    train = as.matrix(train_data$x),
    test = as.matrix(grid_data$x_grid),
    y = train_data$y,
    k = 10
  )$pred)

# prediction bands are then given by (y_pred-t,y_pred+t)
```

In this example, the marginal coverage of the intervals obtained for the specific training and calibration data is 97.10% (see the full code at the Conformal Notebook. Notice that this value does *not* fall below $95\%+1/(n+1) \approx 95.66\%$, as the upper bound in Theorem 9 seems to suggest. This is because the randomness in the probabilities involved in both Theorems 8 and 9 include the randomness over the calibration data. We discuss this in further details in the next section.

5.3.1 A different perspective on conformal regions: tolerance regions

> The highest result of education is tolerance.
>
> Helen Keller

If we examine the derivations of Theorems 8 and 9 more closely, we observe that the probability associated with the marginal coverage in these theorems is integrated over:

- The training sample used to construct the non-conformity score, $\mathcal{D}_1$,
- The calibration sample used to determine the cutoff t, $\mathcal{D}_2$,
- The new data point, $(\mathbf{X}_{n+1}, Y_{n+1})$.

Therefore, the interpretation of Theorem 8 is as follows: if one were to observe an infinite number of training and calibration datasets, and for each dataset assess whether a new observation Y_{n+1} falls within the prediction region $R(\mathbf{X}_{n+1})$, the coverage probability would be at least $1-\alpha$.

However, in practice, we have a single train and a single calibration dataset, and therefore ideally one would like to guarantee that $\mathbb{P}(Y_{n+1} \in R(\mathbf{X}_{n+1})|\mathcal{D}_1, \mathcal{D}_2)$ to be larger than $1-\alpha$. That is, the coverage probability would ideally be larger than $1-\alpha$ given the observed data $\mathcal{D}_1, \mathcal{D}_2$. Although this does not hold, the next theorem shows that there is a high probability that the coverage will be large for a fixed $\mathcal{D}_1, \mathcal{D}_2$:

Theorem 10. *(Bian and Barber, 2023; Vovk, 2012) Let $R(\cdot)$ be a split conformal prediction region with nominal level $1-\alpha$. If the data is i.i.d., for any $\delta \in (0, 0.5]$,*

$$\mathbb{P}\left(\beta(\mathcal{D}_1, \mathcal{D}_2) \geq 1 - \alpha - \sqrt{\frac{\log(1/\delta)}{2n}}\right) \geq 1 - \delta,$$

where $\beta(\mathcal{D}_1, \mathcal{D}_2) = \mathbb{P}(Y_{n+1} \in R(\mathbf{X}_{n+1})|\mathcal{D}_1, \mathcal{D}_2)$ is the random variable that measures the probability of coverage for a given calibration and training sets, and $n = |\mathcal{D}_2|$ is the size of the calibration set.

Theorem 10 shows that, with high probability, the training and calibration set are such that a split conditional region has with marginal coverage not much smaller

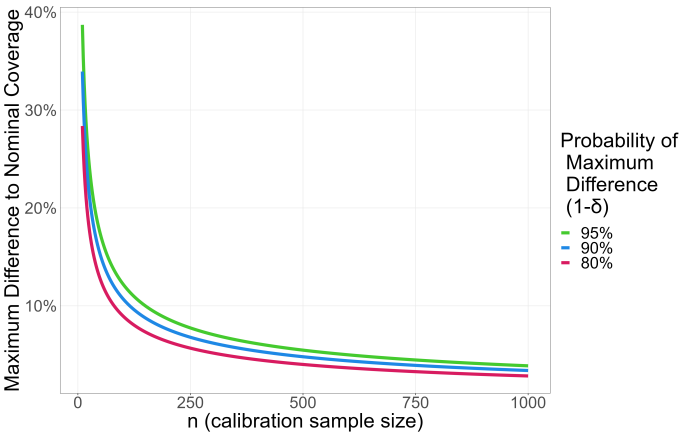


Figure 5.4: Maximum distance of the nominal coverage $1-\alpha$ as a function of the probability δ and the calibration sample size n according to Theorem 10.

than the nominal value $1-\alpha$. Technically, the split conformal approach leads to $(\alpha + \sqrt{\frac{\log(1/\delta)}{2n}}, \delta)-$ tolerance regions (Fraser and Guttman, 1956; Wilks, 1941). See Hulsman (2022) for additional relationships between tolerance regions and conformal methods.

Figure 5.4 shows, for various values of δ and n, the maximum distance of the true coverage $\beta(\mathcal{D}_1, \mathcal{D}_2)$ to $1-\alpha$. This will of course come at the expense that the prediction regions will be larger.

Interestingly, Theorem 10 also shows that if a split conformal method is trained with $\alpha' = \alpha - \sqrt{(2n)^{-1}\log(1/\delta)}$, then it will satisfy the stringer condition that

$$\mathbb{P}(\beta(\mathcal{D}_1, \mathcal{D}_2) \geq 1-\alpha) \geq 1-\delta.$$

Theorem 10 can also be used to determine the size of the calibration set. Specifically, if one desires to have coverage at a distance of at most ϵ from $1-\alpha$ with probability $1-\delta$, one should take

$$n = \frac{\log(1/\delta)}{2\epsilon^2}.$$

5.3.2 Comparison to Prediction Sets from Linear Models

Under the standard linear model $Y|\mathbf{x} \sim N\left(\boldsymbol{\beta}^\intercal \mathbf{x}, \sigma^2\right)$ (see Section 3.3) and assuming the data points are i.i.d., the traditional prediction interval for a new sample point $\mathbf{x}_{n+1}$ is given by (Neter et al., 1996):

$$R(\mathbf{x}_{n+1}) = [\widehat{r}(\mathbf{x}_{n+1}) - t_l, \widehat{r}(\mathbf{x}_{n+1}) + t_l],$$

where $\widehat{r}(\mathbf{x}_{n+1}) = \widehat{\boldsymbol{\beta}}^\intercal \mathbf{x}_{n+1}$, with $\widehat{\boldsymbol{\beta}}$ being the least squares estimate of $\boldsymbol{\beta}$ (Equation 2.9). The interval width is given by:

$$t_l(\mathbf{x}_{n+1}) = t_{(1-\alpha/2, n-d-1)} \times \sqrt{\hat{\sigma}^2 \left(1 + \mathbf{x}_{n+1}^\top (\mathbb{X}^\top \mathbb{X})^{-1} \mathbf{x}_{n+1}\right)},$$

where

$$\widehat{\sigma}^2 = \frac{1}{n-d-1} \sum_{i=1}^{n} \left(y_i - \widehat{\boldsymbol{\beta}}^\intercal \mathbf{x}_i\right)^2.$$

This prediction interval differs from the regression-split conformal interval $R(\mathbf{x}_{n+1}) = [\widehat{r}(\mathbf{x}_{n+1}) - t, \widehat{r}(\mathbf{x}_{n+1}) + t]$, even when the calibration set is large. Unlike regression-split intervals, which have constant length, the width of the standard linear prediction interval depends on $\mathbf{x}_{n+1}$ through $\mathbf{x}_{n+1}^\top (\mathbb{X}^\top \mathbb{X})^{-1} \mathbf{x}_{n+1}$. The interval is wider for points $\mathbf{x}_{n+1}$ that are farther from the bulk of the training data.

For this standard prediction interval to be valid, the assumptions of the linear model must hold. To demonstrate this, we run simulations comparing this interval with conformal intervals obtained via regression-split using the same linear model. The full details are shown in the Linear Prediction Sets Notebook.

In the simulations, features are generated as independent draws from a standard normal distribution with mean zero and variance one. Each sample consists of 150 features, and 300 sample points are used to train the linear model via least squares. We use 100 additional sample points to calibrate the conformal intervals.

The simulation scenarios are as follows:

- **Linear Homoskedastic:**

$$y_i = 2 \cdot x_{i,1} + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2),$$

  where $\sigma = 0.3$.

- **Linear Heteroskedastic:**

$$y_i = 2 \cdot x_{i,1} + \frac{|x_{i,1}|}{2} \cdot \epsilon_i, \quad \epsilon_i \sim t_2,$$

  where t_2 follows a Student's t-distribution with 2 degrees of freedom.

- **Nonlinear Heteroskedastic:**

$$y_i = \sin(5 \cdot x_{i,2}) + \frac{|x_{i,1}|}{2} \cdot \epsilon_i, \quad \epsilon_i \sim t_2.$$

Figure 5.5 compares the prediction sets from conformal methods and linear models across various data-generating schemes. The top panel demonstrates that while linear models achieve the nominal 90% coverage (indicated by the red dashed line) when correctly specified, they tend to overcover when the model is misspecified. In contrast, conformal methods consistently provide valid coverage in all scenarios. The bottom panel shows that in misspecified settings, the linear model produces larger prediction intervals.

5.3.3 Achieving Asymptotic Conditional Coverage

Conformal methods do not control coverage conditional on $\mathbf{x}$ (Equation 5.1). Indeed, it can be shown that exact conditional validity can be obtained only under strong assumptions about the distribution of $(\mathbf{X}, Y)$ (Foygel Barber et al., 2021; Lei and Wasserman, 2014; Vovk, 2012).

However, under further assumptions about how the non-conformity score is computed, some non-conformity scores lead to prediction regions that also have other good properties such as asymptotic conditional validity. We illustrate this on HPD-split (Izbicki et al., 2022), which uses the non-conformity score

$$h(\mathbf{x}, y) = -\int_{\{y' : \hat{f}(y'|\mathbf{x}) \leq \hat{f}(y|\mathbf{x})\}} \hat{f}(y'|\mathbf{x}) dy'.$$

This score is represented by the shaded in Figure 5.6.

First, Y is assumed to belong to a bounded space:

Assumption 8. *$Y \in \mathcal{Y}$, where $\mathcal{Y}$ is bounded.*

Second, $\hat{f}(y|\mathbf{x})$ is assumed to be consistent:

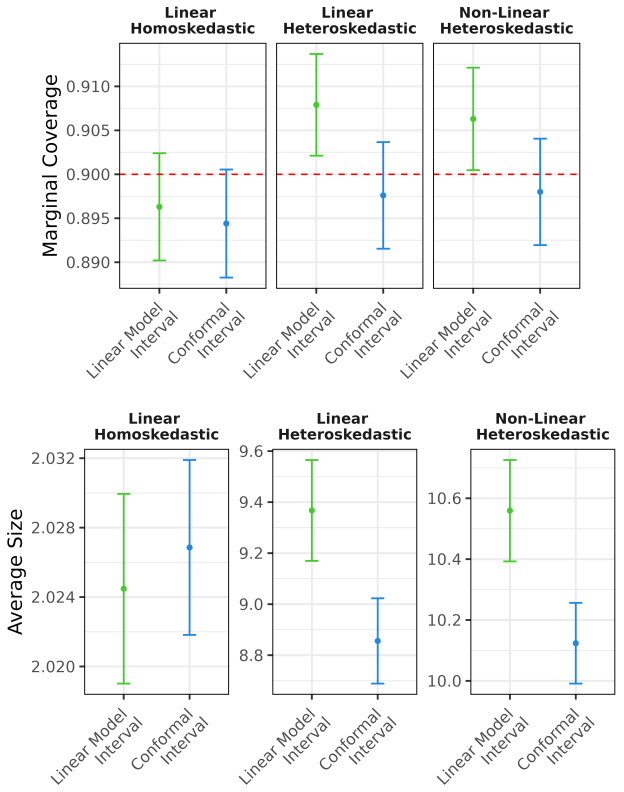


Figure 5.5: Top: Marginal coverage comparison between conformal methods and linear model prediction sets across data-generating schemes. The red dashed line marks 90% nominal coverage. Bottom: Average size comparison. Linear models achieve 90% only when correctly specified, while conformal inference ensures valid coverage even with misspecification.

Assumption 9 (Consistency of $\hat{f}$)**.** *There exist $\eta_n = o(1)$ and $\rho_n = o(1)$ s.t.*

$$\mathbb{P}\left(\mathbb{E}\left[\sup_{y\in\mathcal{Y}}\left(\hat{f}(y|\mathbf{X}) - f(y|\mathbf{X})\right)^2 \middle| \hat{f}\right] \geq \eta_n\right) \leq \rho_n$$

Similarly, the conditional cumulative density function of $f(Y|\mathbf{X})$, which we denote by $H(z|\mathbf{x})$, needs to be assumed to be well behaved: it needs to be smooth (bounded density) and have no plateau close the α quantile:

Assumption 10. *For every $\mathbf{x}$, $H(u|\mathbf{x})$ is continuous, differentiable and $\frac{dH(u|\mathbf{x})}{du} \leq M_1$. Also $\frac{dH(u|\mathbf{x})}{du} \geq M_2 > 0$ in a neighborhood of $q_\alpha(\mathbf{x})$.*

Under the above assumptions HPD-split converges to the $1-\alpha$-level HPD Region (Example 5.1) and satisfies asymptotic conditional coverage:

Theorem 11 (Izbicki et al. (2022))**.** *Under Assumptions 8, 9, 10 and if the the data is i.i.d., the $1-\alpha$-level HPD-split region converges to the $1-\alpha$-level HPD set and there exist random sets, $\mathbf{\Lambda}_n$, such that $\mathbb{P}(\mathbf{X}_{n+1} \in \mathbf{\Lambda}_n | \mathbf{\Lambda}_n) = 1 - o_{\mathbb{P}}(1)$ and*

$$\sup_{\mathbf{x}_{n+1}\in\mathbf{\Lambda}_n} \left|\mathbb{P}(Y_{n+1} \in R(\boldsymbol{X}_{n+1})|\boldsymbol{X}_{n+1} = \mathbf{x}_{n+1}) - (1-\alpha)\right| = o(1).$$

Other scores that also satisfy asymptotic conditional validity are CD-split (Izbicki et al., 2020), Dist-Split (Izbicki et al., 2020), and CQR (Romano et al., 2019).

5.3.3.1 Label-Conditional Conformal Regions

A second, often desired, type of conditional coverage is class-conditional (or label-conditional) coverage, defined as:

$$\mathbb{P}(Y_{n+1} \in R(\mathbf{X}_{n+1})|Y_{n+1}) \geq 1-\alpha \text{ (almost surely).}$$

This criterion ensures that the predictive region $R(\mathbf{X}_{n+1})$ contains the true label with high probability, conditioned on the label itself. Class-conditional coverage becomes especially useful in settings involving dataset shift, where the joint distribution of $(\mathbf{X}, Y)$ differs between the training and target domains (Masserano et al., 2024; Quiñonero-Candela et al., 2022).

In particular, class-conditional coverage is crucial under *prior probability shift*, a specific form of dataset shift where the conditional distribution $\mathbf{X}|Y$ remains stable,

but the class prior $\mathbb{P}(Y = y)$ can vary across domains (Assunção et al., 2023; Forman, 2008; Podkopaev and Ramdas, 2021; Vaz et al., 2019). This is because a model trained to satisfy class-conditional coverage on the training set will continue to maintain this coverage in the target set, even when class distributions shift.

An approach to constructing conformal prediction sets that ensure class-conditional coverage is to select a distinct conformal cutoff t for each class y (Sadinle et al., 2019; Vovk et al., 2005). Specifically, the prediction sets take the form

$$R(\mathbf{x}_{n+1}) = \{y : h(\mathbf{x}_{n+1}, y) \leq t_y\},$$

where h is the conformal score function. Similar to the standard split conformal method, we first compute the random variable $U_i := h(\mathbf{X}_i, Y_i)$ for each sample point $(\mathbf{X}_i, Y_i) \in \mathcal{D}_2$. A separate cutoff t_y is determined for each class y by calculating the $1-\alpha$ quantile of the set $\{U_i : Y_i = y\}$, meaning we only consider the scores of instances with label y.

See Ding et al. (2024) and references therein for other approaches to achieve label-conditional coverage.

5.3.4 Local Conformal Regions

A different approach to get closer to control of conditional coverage is to first create a partition the feature space, say $\mathcal{A}$, and then apply the split conformal approach to each partition element $A \in \mathcal{A}$ separately (Boström and Johansson, 2020; Boström et al., 2021; Foygel Barber et al., 2021; Izbicki et al., 2022; Lei and Wasserman, 2014; Vovk, 2012). This yields methods that control local coverage, meaning that, for every $A \in \mathcal{A}$,

$$\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1}) | \mathbf{X}_{n+1} \in A) \geq 1 - \alpha.$$

The choice of $\mathcal{A}$ is crucial to guarantee that

$$\mathbb{P}(Y \in C(\mathbf{X}) | \mathbf{X} \in A) \approx \mathbb{P}(Y \in R(\mathbf{X}) | \mathbf{X} = \mathbf{x}),$$

that is, the local conformal regions $R(\mathbf{X})$ will not only have local coverage but will also be close to conditional coverage.

There are several ways to choose $\mathcal{A}$. For instance, while Lei and Wasserman, 2014 partitions the feature space by creating a hyper-rectangular mesh, Mondrian-based

approaches (Boström and Johansson, 2020; Boström et al., 2021) define a partition with a predefined Mondrian taxonomy (Vovk et al., 2005). Here, we review `Locart` (Cabezas et al., 2024), which creates $\mathcal{A}$ based on the following theorem:

Theorem 12 (The coarsest partition that controls conditional coverage; adapted from Cabezas et al. 2024)**.** *Let $\mathcal{A}$ be a partition of $\mathcal{X}$ such that, for any $A \in \mathcal{A}$, $\mathbf{x}_1, \mathbf{x}_2 \in A$ if and only if $h(\mathbf{X}, Y)|\mathbf{X} = \mathbf{x}_1 \sim h(\mathbf{X}, Y)|\mathbf{X} = \mathbf{x}_2$, where $h(\mathbf{X}, Y)$ is any non-conformity score. Let $t^*_{1-\alpha}(\mathbf{x})$ be $(1-\alpha)$-quantile of the non-conformity scores. Then,*

1. *If $\mathbf{x}_1, \mathbf{x}_2 \in A$, then $t^*_{1-\alpha}(\mathbf{x}_1) = t^*_{1-\alpha}(\mathbf{x}_2)$ for every $\alpha \in (0, 1)$*
2. *If $\mathcal{J}$ is another partition of $\mathcal{X}$ such that, for any $J \in \mathcal{J}$, $\mathbf{x}_1, \mathbf{x}_2 \in J$ implies that $t^*_{1-\alpha}(\mathbf{x}_1) = t^*_{1-\alpha}(\mathbf{x}_2)$ for every $\alpha \in (0, 1)$, then*

$$\mathbf{x}_1, \mathbf{x}_2 \in J \implies \mathbf{x}_1, \mathbf{x}_2 \in A.$$

Cabezas et al. (2024) attempts to recover the partition described in this theorem. The key idea is to use a regression tree, which naturally partitions the feature space. Now, because regression trees are consistent estimators of conditional distributions (Meinshausen and Ridgeway, 2006), a regression tree that predicts the score $h(\mathbf{X}, Y)$ using $\mathbf{x}$ as an input attempts to recover the partition described by Theorem 12, that is, the partition that groups features according to the conditional distribution of the residuals. This is, therefore, how `Locart` builds the partition $\mathcal{A}$. Concretely, once the non-conformity score h has been defined, `Locart` consists of the following steps:

1. Create the dataset $\{(\mathbf{X}_i, h(\mathbf{X}_i, Y_i))\}$ based on the calibration dataset, and split it into two disjoint sets I_{part} and I_{cut}.
2. Fit a regression tree on $\{(\mathbf{X}_i, h(\mathbf{X}_i, Y_i))\}_{i \in I_{\text{part}}}$ that predicts the non-conformity score h based on the features $\mathbf{X}$. This tree induces a partition $\mathcal{A}$ of the feature space generated by its terminal nodes. Let $\mathcal{T} : \mathcal{X} \to \mathcal{A}$ be the function mapping an element of $\mathcal{X}$ to the partition element it belongs.
3. Estimate $\hat{t}_{1-\alpha}(\mathbf{x}_{n+1})$ for a new instance $\mathbf{x}_{n+1}$ by applying the conformal approach to all instances in $\{(\mathbf{X}_i, h(\mathbf{X}_i, Y_i))\}_{i \in I_{\text{cut}}}$ that fall into the same element of $\mathcal{A}$ as $\mathbf{x}_{n+1}$ does. That is, estimate $\hat{t}_{1-\alpha}(\mathbf{x}_{n+1})$ as

$$\hat{t}_{1-\alpha}(\mathbf{x}_{n+1}) = \hat{q}_{1-\alpha}(A(\mathbf{x}_{n+1})) \tag{5.4}$$

where

$$A(\mathbf{x}_{n+1}) = \{\hat{s}_i : \mathcal{T}(\mathbf{x}_i) = \mathcal{T}(\mathbf{x}_{n+1}), (\mathbf{X}_i, \hat{s}_i) \in \{(\mathbf{X}_i, \hat{s}_i)\}_{i \in I_{\text{cut}}}\}.$$

4. Using $\hat{t}_{1-\alpha}(\mathbf{x}_{n+1})$ from Equation 5.4, define the `Locart` prediction interval as

$$R(\mathbf{x}_{n+1}) = \left\{y : h(\mathbf{x}_{n+1}, y) \leq \hat{t}_{1-\alpha}(\mathbf{x})\right\}.$$

Cabezas et al. (2024) shows that this approach not only leads to local validity, but that under some assumptions it also satisfies asymptotic conditional validity. The paper also extends `Locart` to random forests.

5.3.5 Conformal Sets for Classification

In classification tasks, conformal prediction aims to produce sets of labels that include the true label with a predefined probability, $1-\alpha$ (Angelopoulos et al., 2020; Valle et al., 2023; Vovk et al., 2005). One of the most intuitive non-conformity scores for classification is based on class probability:

$$h(\mathbf{x}, y) = -\widehat{f}(y|\mathbf{x}) = -\widehat{\mathbb{P}}(Y = y|\mathbf{x}), \tag{5.5}$$

which serves as the classification analogue to the CD-split score discussed in Section 5.3. While this score ensures marginal coverage, it does not guarantee conditional coverage, even with large sample sizes. This limitation results in undercoverage for more challenging subgroups and overcoverage for easier ones.

A more robust alternative is adaptive prediction sets (APS; Romano et al. 2020), where the non-conformity score is defined as:

$$h(\mathbf{x}, y) = \sum_{y' \in \mathcal{Y}: \widehat{\mathbb{P}}(Y=y'|\mathbf{x}) > \widehat{\mathbb{P}}(Y=y|\mathbf{x})} \widehat{\mathbb{P}}(Y = y'|\mathbf{x}).$$

This score is analogous to the HPD-split non-conformity score used in regression (Section 5.3), but is specifically designed for classification. Similar to the regression version HPD-split (Section 5.3.3), APS also achieves asymptotic conditional coverage.

Next, we apply APS (HPD-Split) and a localized version of the score in Equation 5.5 (CD-Split; see Izbicki et al. 2022 for details) to the MNIST dataset (LeCun et al.,

1995). The data is divided into three sets: 9% for potential future samples, 70% for estimating $\mathbb{P}(y|\mathbf{x})$, and 21% for calculating split residuals. The conditional density, $\mathbb{P}(Y = y|\mathbf{x})$, is estimated using a convolutional neural network. Figure 5.7 presents six examples of images and their corresponding prediction sets. The top row displays examples where two labels were assigned to each data point. These cases typically appear ambiguous to humans. Both methods yield similar results.

5.3.6 Example: Photometric Redshift Prediction

We apply conformal methods to a key problem in cosmology: estimating a galaxy's redshift (y), a proxy for its distance from Earth, based on photometric features and r-magnitude ($\mathbf{x}$) derived from imaging data. Details about this problem can be found in Chapter 7.

Here, we construct prediction bands for redshifts using the Happy A dataset (Beck et al., 2017), specifically designed for comparing photometric redshift prediction algorithms. Happy A contains 74,950 galaxies from the Sloan Digital Sky Survey DR12. Our analysis employs 64,950 galaxies for training, 5,000 for prediction, and an additional 5,000 for assessing the performance of conformal methods.

Within the training set, we apply random forests to fit the conformity of reg-split (Lei et al., 2018) and quantile-split (Romano et al., 2019). For density-based methods (HPD-split (Izbicki et al., 2022), CD-split+ (Izbicki et al., 2020), and Dist-split (Izbicki et al., 2020)), the non-conformity score is fitted using a Gaussian mixture density neural network (Section 3.5) with three components, one hidden layer, and 500 neurons. Using this mixture model, CD-split and HPD-split generate predictive regions as the union of at most three intervals. All methods adhere to a marginal coverage level of $1 - \alpha = 80\%$.

Table 5.1 details the local coverage and average size of prediction bands in two regions of the feature space: bright galaxies and faint galaxies, classified based on r-magnitude values. While all methods achieve local coverage close to the nominal 80% level for bright galaxies, this is not the case for faint galaxies. Nominal coverage for faint galaxies is only achieved by HPD-split and CD-split+, indicating superior control over conditional coverage in this context. Additionally, the table reveals that, except for reg-split, all methods produce regions of similar size. Reg-split yields smaller regions at the expense of worse local coverage among faint galaxies.

Figure 5.8 presents examples of the prediction regions for each method, with a

horizontal line marking the true redshift for each instance. The size of all prediction regions increases for faint galaxies, which often exhibit multimodal densities (Kügler et al., 2016; Polsterer, 2016; Wittman, 2009). Specifically, in the selected instances of faint galaxies, the true redshift is either near 0.25 or 0.75, but never around 0.5. This multimodal behavior suggests that the central portion of the prediction intervals generated by Dist-split and quantile-split frequently includes unlikely estimates of the true redshift.

Table 5.1: Coverage and average size of the prediction bands for the photometric redshift prediction problem, along with their standard errors.

	Galaxy	HPD-split	CD-split+	Dist-split	Quantile-split	Reg-split
Coverage	Bright	0.800 (0.006)	0.795 (0.006)	0.802 (0.006)	0.808 (0.006)	0.807 (0.006)
	Faint	0.788 (0.018)	0.792 (0.018)	0.746 (0.019)	0.754 (0.019)	0.658 (0.021)
Average Size	Bright	0.050 (0.001)	0.049 (0.001)	0.050 (0.001)	0.051 (0.001)	0.044 (0.000)
	Faint	0.065 (0.001)	0.066 (0.001)	0.064 (0.002)	0.074 (0.002)	0.045 (0.000)

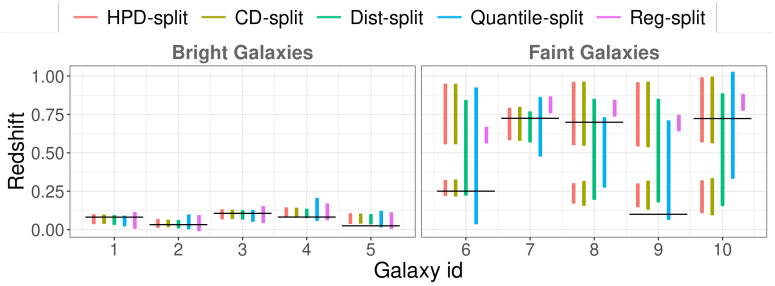


Figure 5.8: Prediction bands obtained for 5 bright and 5 faint galaxies from the test set. Horizontal lines indicate the true redshift of each galaxy.

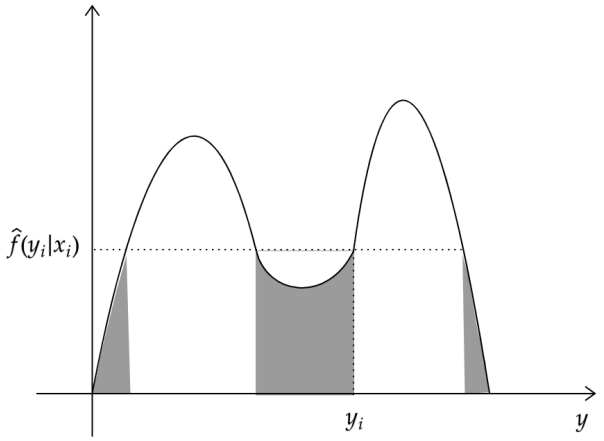


Figure 5.6: The HPD-split score of the sample point $(\mathbf{x}_i, y_i)$ is given by the (negative) of the shaded area on the plot.

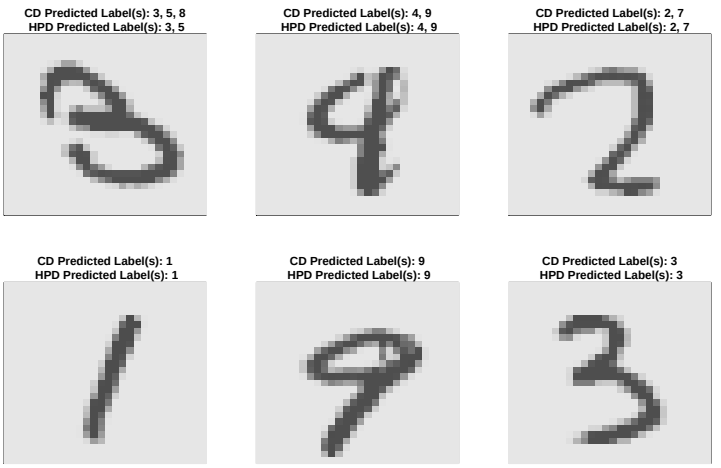


Figure 5.7: Prediction sets generated by CD-Split and APS (HPD-Split) for selected instances from the MNIST dataset.

5.4 Comparing Predictive Intervals: Linear, Bayesian, Conformal Methods, and Plug-in Approaches

In this section, we compare the performance of several methods for constructing prediction intervals, emphasizing the key conceptual differences between them.

The data is generated using a homoscedastic linear Gaussian model, where the response variable y is generated as

$$y = 5x_1 + \epsilon, \quad \epsilon \sim \mathcal{N}(0,1),$$

where x_1 is the first feature of $\mathbf{x}$. The features $\mathbf{X} \in \mathbb{R}^{65}$ are i.i.d. $\mathcal{N}(0,1)$ random variables.

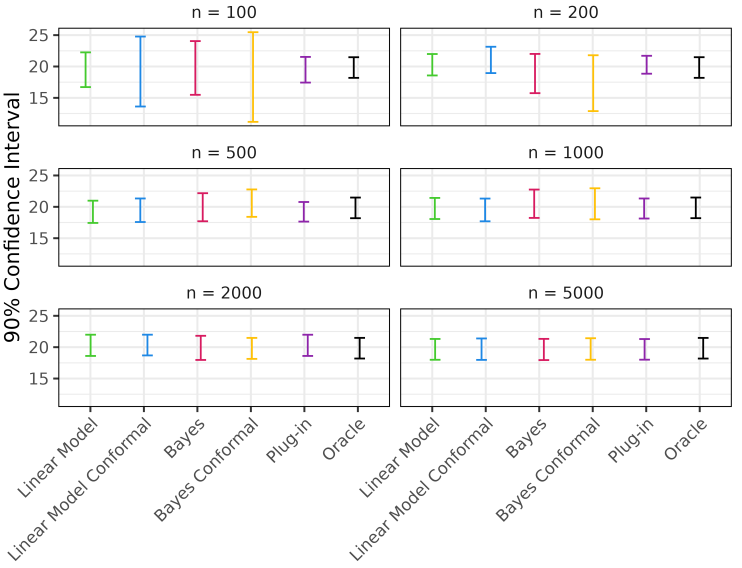
The methods compared are as follows:

- **Linear Model**: A standard linear regression model is fit using ordinary least squares. Prediction intervals are based on the assumption of normally distributed residuals, as detailed in Section 5.3.2.
- **Linear Model with Conformal Prediction**: This method applies the regression-split conformal approach to the predictions of the linear model, yielding prediction sets that provide coverage guarantees without making distributional assumptions.
- **Bayesian Regression (Bayes)**: A Bayesian linear regression model is fit with a Gaussian prior on the regression coefficients. Prediction intervals are derived from the posterior predictive distribution, as described in Section 6.1).
- **Bayesian Regression with Conformal Prediction (Bayes Conformal)**: This method incorporates conformal prediction into the Bayesian linear model using the regression-split conformal approach, ensuring valid coverage without distributional assumptions.
- **Plug-in Method**: The plug-in method models $f(y|\mathbf{x})$ as a Gaussian distribution with mean $\widehat{\boldsymbol{\beta}}^\intercal \mathbf{x}$ and variance $\widehat{\sigma}^2$, where $\widehat{\sigma}^2$ is the maximum likelihood estimate (see Equation 3.5). The prediction interval is then the symmetric plug-in interval based on $\widehat{f}$ (Example 5.3).

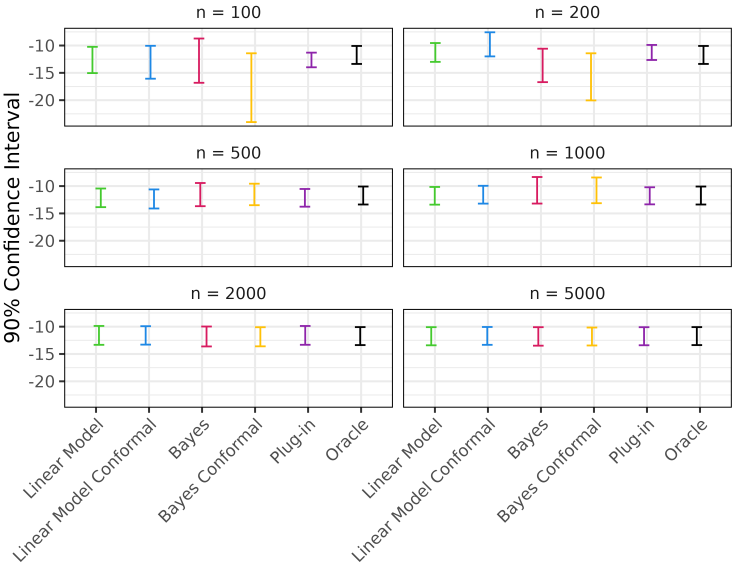
- **Oracle**: These are the oracle symmetric intervals (see Example 5.3) using the true, but unknown in practice, values of $\boldsymbol{\beta}$ and σ^2, assuming that $Y|\mathbf{x} \sim N\left(\boldsymbol{\beta}^\intercal \mathbf{x}, \sigma^2\right)$. In this case, all the oracle regions discussed in Section 5.1 are the same because the Gaussian distribution is symmetric and unimodal.

We simulate data with training sample sizes $n \in \{100, 200, 500, 1000, 2000, 5000\}$, and for each sample size, we fit the models mentioned above. The prediction regions are calibrated to have nominal coverage of 90% (for the Bayesian region, we use the coverage according to the predictive distribution). The code is available at the Predictive Intervals Comparison Notebook.

Figure 5.9 displays the prediction intervals for each method across the different training sample sizes for two new data points. When the training sample size is small, methods that account for epistemic uncertainty – namely the conformal intervals, the linear model, and Bayesian predictive sets – produce wider intervals than the oracle, reflecting their ability to capture the additional uncertainty from limited data. In contrast, the plug-in interval attempts to directly emulate the oracle, often showing similar or even narrower widths, but at the cost of reduced coverage, falling below the nominal level. As the sample size increases, all methods gradually converge toward the oracle's prediction intervals. Further insights into the role of the Bayesian predictive distribution in shaping prediction regions can be found in Section 1.2.



(a) First sample point



(b) Second sample point

Figure 5.9: Prediction intervals for two data points across varying sample sizes. Methods accounting for epistemic uncertainty produce wider intervals for small sample sizes, while all methods converge toward the oracle as sample size increases.

5.5 Summary

In this chapter, we explored various techniques for constructing prediction regions for the label of a new sample point. We began by deriving optimal regions according different loss functions, such as the HPD and the quantile-based regions. We then explored how estimates of the conditional density $f(y|\mathbf{x})$ can be directly used to provide an estimate of this optimal sets. These estimated regions however may not have the correct nominal coverage. To address this challenge, we introduced conformal prediction, a framework designed to generate prediction regions with guaranteed marginal coverage, assuming only that data points are exchangeable. We saw that different non-conformity scores yield regions with different properties. In particular some methods are constructed to ensure that, as the sample size increases, they approximate the optimal regions closely. Additionally, we examined conformal methods that are designed to achieve other properties, such as local conditional coverage. We will see another approach to construct prediction bands in Section 6.1.2.

Chapter 6

Capturing Epistemic Uncertainty through Bayesian and Ensemble Techniques

La Clairvoyance. René Magritte, 1936, Private Collection.

> The greatest obstacle to discovery is not ignorance – it is the illusion of knowledge.
>
> Daniel J. Boorstin

In Chapter 3, we explored how aleatoric uncertainty, the inherent variability in outcomes given the same input, can be captured through the conditional density $f(y|\mathbf{x})$, and we discussed several methods for estimating it. While epistemic uncertainty – our uncertainty about the model itself – can sometimes be directly quantified for specific conditional density estimation (CDE) models like $\widehat{f}(y|\mathbf{x})$, these approaches are often complex and difficult to work with (Calonico et al., 2018; Cheng and Chen, 2019). Instead, Chapter 5 shows that we can address epistemic uncertainty by incorporating it into the construction of predictive sets derived from $\widehat{f}(y|\mathbf{x})$.

In this chapter, we explore how Bayesian models and modern ensemble techniques effectively model epistemic uncertainty around Y, offering a structured approach to quantify and incorporate it alongside aleatoric uncertainty. Additionally, we discuss the use of bootstrap techniques to assess epistemic uncertainty from a frequentist perspective.

For a broader overview of other related techniques, see Nemani et al. (2023).

6.1 Bayesian Models

In a Bayesian context, we need additional notation. As in the frequentist setting, the Bayesian model requires a family of distributions that model the aleatoric uncertainty of $Y|\mathbf{x}$. We will denote this class by

$$\mathcal{F} = \{f(y|\mathbf{x}, \theta) : \theta \in \Theta\}.$$

Θ may represent a nonparametric space (that is, a space that cannot be mapped to $\mathbb{R}^d$). Additionally, the Bayesian model also poses a prior distribution over Θ (or, equivalently, over $\mathcal{F}$) which is named the *prior distribution*, which for simplicity we assume to have density $f(\theta)$ with respect to some dominating measure. The prior distribution encodes the epistemic uncertainty about the data generating model.

Let $D = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ be the training data. As in most traditional

regression models, in this section we assume that $\mathbf{x}_i$'s are fixed. Moreover, we assume that, given θ, the data points are independent, and that they all have the same conditional distribution $f(y|\mathbf{x},\theta)$.

Prediction in Bayesian models is typically obtained by computing the predictive density

$$f(y|\mathbf{x},D) = \int f(y|\mathbf{x},D,\theta)f(\theta|\mathbf{x},D)d\theta, \tag{6.1}$$

where $(\mathbf{x},y)$ represents a new sample point which we assume to be independent of D (given θ). This predictive distribution incorporates both the aleatoric and the epistemic uncertainty.

Under the assumptions that were made, the above expression can be simplified to

$$f(y|\mathbf{x},D) = \int f(y|\mathbf{x},\theta)f(\theta|D)d\theta,$$

$f(\theta|D)$ is the posterior distribution, while $f(y|\mathbf{x},\theta)$ is simply the model for the aleatoric uncertainty. In words, the Bayesian method weights the different uncertainty models $f(y|\mathbf{x},\theta)$ according to their posterior evidence $f(\theta|D)$. Thus, the Bayesian model naturally models the epistemic uncertainty, which in this context is the uncertainty regarding the true value of θ. For a deeper discussion on how these uncertainties are combined, see Section 1.2.

Important Warning! In the Bayesian framework, $f(y|\mathbf{x})$ does not represent the same conditional density discussed in Chapter 3. Instead, it refers to the *a priori* predictive density:

$$f(y|\mathbf{x}) = \int f(y|\mathbf{x},\theta)f(\theta)d\theta.$$

In this context, the $f(y|\mathbf{x})$ we aim to estimate in Chapter 3 corresponds to $f(y|\mathbf{x},\theta)$, where θ represents the true parameter value.

Example: Linear Gaussian Regression. To illustrate these principles, we now consider a practical example: the application of Bayesian methods to Gaussian linear regression. In this case, we take $\mathcal{F}$ to represent a Gaussian linear regression model (as in Section 3.3):

$$Y|\mathbf{x},\boldsymbol{\beta} \sim N\left(\boldsymbol{\beta}^\intercal\mathbf{x},\sigma^2\right),$$

where for simplicity we assume σ^2 is known, and use the following prior distribution:

$$\boldsymbol{\beta} \sim N\left(\mathbf{0}, \Sigma_d\right),$$

where Σ_d is a $d \times d$ covariance matrix. This leads to the posterior distribution

$$\boldsymbol{\beta}|D \sim N\left(\frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\mathbb{X}^\intercal\mathbb{X} + \Sigma_d^{-1}\right)^{-1}\mathbb{X}^\intercal\mathbb{Y}, \left(\frac{1}{\sigma^2}\mathbb{X}^\intercal\mathbb{X} + \Sigma_d^{-1}\right)^{-1}\right),$$

where $\mathbb{X}$ and $\mathbb{Y}$ are defined in Section 3.3. Moreover, the predictive distribution, $f(y|\mathbf{x}, D)$, is given by

$$Y|\mathbf{x}, D \sim N\left(\left[\frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\mathbb{X}^\intercal\mathbb{X} + \Sigma_d^{-1}\right)^{-1}\mathbb{X}^\intercal\mathbb{Y}\right]^\intercal\mathbf{x}, \mathbf{x}^\intercal\left(\frac{1}{\sigma^2}\mathbb{X}^\intercal\mathbb{X} + \Sigma_d^{-1}\right)^{-1}\mathbf{x} + \sigma^2\right),$$

Many of the models that will be introduced in the following sections are particular cases of the Bayesian approach to uncertainty quantification.

6.1.1 Aleatoric vs. Epistemic Uncertainty Revisited

The Bayesian framework offers a clear decomposition of uncertainty into distinct sources. If we quantify a model's uncertainty by its variance, the law of total variance gives the following breakdown:

$$\mathbb{V}\left[Y|\mathbf{x}, D\right] = \mathbb{E}_{\theta\sim f(\theta|D)}\left[\mathbb{V}\left[Y|\mathbf{x}, \theta\right]\right] + \mathbb{V}_{\theta\sim f(\theta|D)}\left[\mathbb{E}\left[Y|\mathbf{x}, \theta\right]\right].$$

This decomposition shows that the total uncertainty is the sum of the aleatoric uncertainty (the first term on the right-hand side, which represents the average intrinsic variability of the outcome Y) and epistemic uncertainty (the second term, capturing the variation in the prediction due to uncertainty about θ).

As the training sample size increases, the predictive distribution $f(y|\mathbf{x}, D)$ tends to converge to the true conditional density $f(y|\mathbf{x}, \theta)$ (Schervish, 2012). This happens because, with sufficient data, the uncertainty due to model parameters (epistemic uncertainty) decreases, leaving only the inherent uncertainty of $Y|\mathbf{x}$ (aleatoric uncertainty). Figure 6.1 illustrates this behavior for a Gaussian example. For small sample

sizes, $f(y|\mathbf{x}, D)$ is wider than $f(y|\mathbf{x}, \theta)$, reflecting greater epistemic uncertainty. However, as the sample size increases, the predictive distribution narrows and approaches the true conditional density, indicating that only aleatoric uncertainty remains. The code to construct this plot is available at the Bayesian Density Notebook).

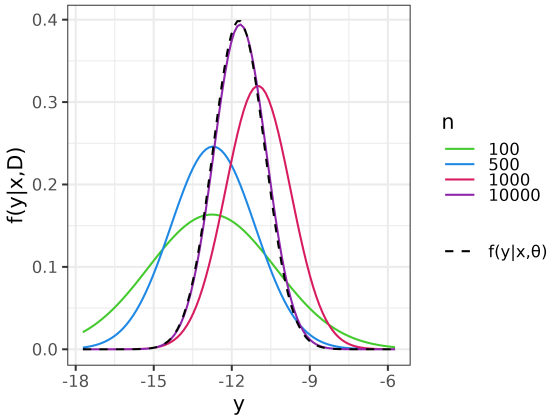


Figure 6.1: Colored lines represent the predictive distribution $f(y|\mathbf{x}, D)$ for datasets D with different sample sizes n. The black dashed line is the true distribution $f(y|x, \theta)$, where θ is the true mean and variance. As n increases, the predictive distribution converges to $f(y|x, \theta)$.

Since $f(y|\mathbf{x}, D)$ converges to $f(y|\mathbf{x}, \theta)$ as n grows, it can be used as an approximation of the true conditional density. This allows prediction regions to be constructed using $f(y|\mathbf{x}, D)$, similar to the oracle regions discussed in Section 5.1. However, these regions are usually wider than the plug-in regions from Section 5.2, as they account for both sources of uncertainty. In contrast, the conditional density estimation techniques described in Chapter 3 capture only aleatoric uncertainty, as seen in the numerical results from Section 5.4.

In the next section, we explore how plug-in regions based on $f(y|\mathbf{x}, D)$ can also be derived from a rigorous Bayesian decision-theoretic perspective.

6.1.2 Bayesian-Optimal Prediction Regions

The Bayesian framework to statistical inference offers a different approach to construct predictive regions. From a Bayesian decision-theoretic perspective, the

Bayes risk of a predictive region R is defined as

$$\mathcal{R}^{\mathrm{B}}(R) := \mathbb{E}[L(R;(\mathbf{X},Y))|D,\mathbf{X}].$$

The expectation is taken with respect to the predictive distribution of the new label Y, conditional on its features $\mathbf{X}$ and the entire training set D (see Section 6.1, particularly Equation 6.1, for a detailed explanation of the predictive distribution).

The optimal regions corresponding to the losses discussed in Examples 5.1–5.3 are:

- **[HPD Region]**
$$R(\mathbf{x}) = \{y \in \mathcal{Y} : f(y|\mathbf{x}, D) > \lambda\},$$
- **[Quantile-based Region]**
$$R(\mathbf{x}) = (q_{\alpha/2}(\mathbf{x}; D), q_{1-\alpha/2}(\mathbf{x}; D)),$$
where $q_\alpha(\mathbf{x}; D)$ denotes the α quantile of the predictive distribution $Y|\mathbf{x}, D$,
- **[Symmetric Region]**
$$R(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}, D] \pm \sqrt{\lambda \mathbb{V}[Y|\mathbf{x}, D]}.$$

In essence, the Bayesian counterpart to oracle prediction sets replaces the unknown quantity $f(y|\mathbf{x})$ with the predictive distribution $f(y|\mathbf{x}, D)$, which is known up to computational considerations. Thus, Bayesian regions can be computed directly.

6.2 Gaussian Process Regression

Gaussian processes can be used to describe the epistemic uncertainty about a regression a function under a Bayesian framework. They are flexible and can be applied to model nonlinear regressions. Next, we explore two complementary but equivalent viewpoints on Gaussian processes: the feature space perspective (Section 6.2.1) and the kernel perspective (Section 6.2.2).

6.2.1 Feature Space Perspective

Let $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}^p$ be a fixed map from the feature space to $\mathbb{R}^p$. For instance, the map

$$\Phi(\mathbf{x}) = (x_1, \ldots, x_d, x_1x_1, x_1x_2, \ldots, x_dx_{d-1}, x_dx_d)$$

contains all two-order interactions of the original inputs. Φ is called a *feature map*. In its simplest form, a Gaussian process fits a Bayesian linear regression model (as that described in Section 6.1) to $\Phi(\mathbf{x})$. That is, the model assumes that

$$Y|\mathbf{x}, \boldsymbol{\beta} \sim N\left(\boldsymbol{\beta}^\intercal \Phi\left(\mathbf{x}\right), \sigma^2\right)$$

and use the following prior distribution:

$$\boldsymbol{\beta} \sim N\left(\mathbf{0}, \Sigma_p\right),$$

where Σ_p is a $p \times p$ covariance matrix. The computations of the posterior and predictive distributions are the same as those derived for linear regression (Section 6.1), but with $\mathbf{x}$ being replaced by $\Phi(\mathbf{x})$. Other assumptions about the likelihood function can also be made, and σ^2 may also be estimated from data (Gramacy, 2020).

Performing the feature map explicitly, however, can be time and memory-consuming. Fortunately, Gaussian processes can also be implemented using the kernel trick (Theodoridis and Koutroumbas, 2006). We describe this approach in what follows.

6.2.2 Kernel Perspective

Notice that the prior distribution over $\boldsymbol{\beta}$'s in the last section induces a distribution over the space of all regression functions. For instance, for any $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the properties of the Gaussian distribution imply that the regression function evaluated at those points, $(\boldsymbol{\beta}^\intercal \Phi\left(\mathbf{x}_1\right), \ldots, \boldsymbol{\beta}^\intercal \Phi\left(\mathbf{x}_n\right))$, has a multivariate Gaussian distribution *a priori*. The kernel perspective of Gaussian processes builds on this by directly placing a Gaussian distribution in the space of all regression functions.

Concretely, the kernel perspective views a Gaussian process as a distribution over functions (namely, the regression functions). Formally, a Gaussian process is a collection of random variables such that every finite collection of those random variables has a multivariate normal distribution (Williams and Rasmussen, 2006). Here we consider Gaussian processes over $\mathcal{F}$, the collection of all functions from the

original feature space $\mathbb{R}^d$ to $\mathbb{R}$. In this case, a Gaussian process is a distribution over $\mathcal{F}$ that is characterized by its mean function, $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and covariance function $K(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))\,(f(\mathbf{x}') - m(\mathbf{x}'))\right]$, and is denoted by

$$f(\mathbf{x}) \sim \text{GP}\left(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')\right).$$

If $f(\mathbf{x}) \sim \text{GP}\left(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')\right)$ then, for any $\mathbf{x}_1, \ldots, \mathbf{x}_n$,

$$(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) \sim N(\mu, \Sigma),$$

where $\mu = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n))$ and Σ is a $n \times n$ covariance matrix whose (i, j) entrance is given by $K(\mathbf{x}_i, \mathbf{x}_j)$.

In the context of regression analysis, a Gaussian process is used to describe the epistemic uncertainty over the collection of regression functions $\mathcal{F}$ by using $\text{GP}\left(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')\right)$ as a prior distribution over $\mathcal{F}$. The data consists of the points $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, and it is typically assumed that

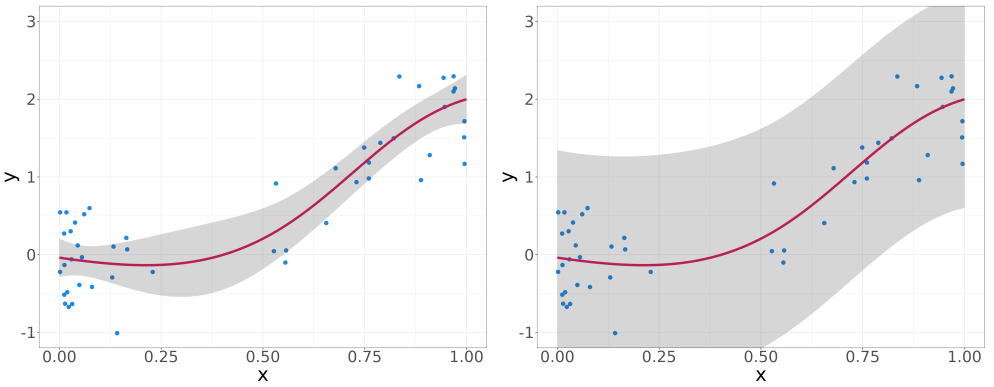
$$Y_i | \mathbf{x}, f \sim N\left(f(\mathbf{x}_i), \sigma^2\right)$$

(see Gramacy 2020 for other assumptions and extensions to when σ^2 is unkown). This prior is then updated after the data has been observed. It turns out that the posterior distribution is also a Gaussian process, as is the predictive distribution.

Each kernel corresponds to a different feature space in the formulation of Section 6.2.1. Some of these feature maps may even have infinite components, which would make the perspective in Section 6.2.1 impossible to implement in practice. Indeed, in this case, the approach is fully nonparametric. Also, each feature space corresponds to a different kernel. For more details about this connection, see Williams and Rasmussen (2006).

Figure 6.2 presents a scatter plot of the data, the estimated regression line, the 95% credible interval around $r(\mathbf{x})$ from the posterior distribution $f(\mathbf{x})|\mathbf{x}, D$ (left panel), and the 95% prediction intervals from the predictive distribution of $Y|\mathbf{x}, D$ (right panel) for a simulated example. The code to construct this plot is available at the Gaussian Process Notebook). The intervals widen in areas with less training data, indicating increased (epistemic) uncertainty about the regression function. This is also reflected on the width of the prediction sets.

The choice of prior, determined by the kernel in a Gaussian Process, plays a critical



(a) Epistemic uncertainty around $r(\mathbf{x})$ based on the posterior distribution of $f(\mathbf{x})|\mathbf{x}, D$

(b) Prediction sets based on the predictive distribution of $Y|\mathbf{x}, D$

Figure 6.2: Gaussian Process Regression: The plot illustrates the original data points in blue, the estimated regression line in red, and the credible/prediction bands as the shaded area.

role in shaping the regression estimates and the corresponding credible and prediction intervals. Figure 6.3 illustrates how different prior distributions, reflected in the selection of kernel bandwidth, affect the example in Figure 1.3. A smaller bandwidth, which reflects a narrower prior, leads to a smoother regression function that is less responsive to local variations in the data and maintains relatively low uncertainty, even in regions with sparse data. On the other hand, a larger bandwidth, representing a more diffuse prior, imposes a stronger smoothness assumption, resulting in a less smooth regression estimate with wider uncertainty bands, especially in areas with fewer data points. In practice, hyperparameters from the prior distribution are often estimated using an empirical Bayes approach, where the marginal likelihood of the data, also referred to as Type-II maximum likelihood, is maximized (Williams and Rasmussen, 2006). The code to construct this plot is available at the Gaussian Process Kernels).

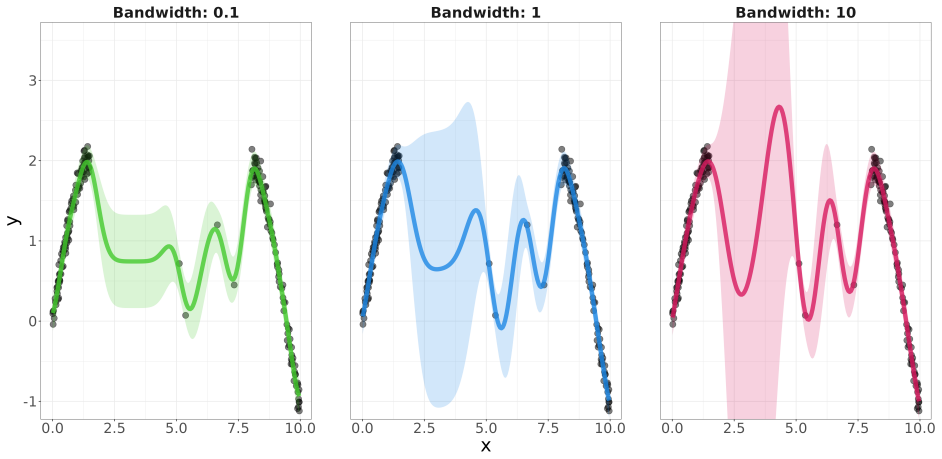


Figure 6.3: Gaussian Process Regression with Varying Bandwidths: The three panels compare the predictions from Gaussian process models with different prior assumptions (small, moderate, and large bandwidths). The shaded areas represent 95% credible intervals for the regression function (that is, the epistemic uncertainty around $r(\mathbf{x})$), illustrating how the prior influences both the flexibility of the model and the uncertainty regions, particularly in areas with sparse data.

6.3 Bayesian Additive Regression Trees (BART)

In its simplest form, BART (Chipman et al., 2012) assumes that

$$Y = \sum_{b=1}^{B} g_b(\mathbf{x}) + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$. Each $g_b(\mathbf{x})$ is assumed to be constant across hyperrectangles defined over the feature space. Specifically, $g_b(\mathbf{x})$ takes the form of a regression tree. Let T_b represent the topology of the regression tree associated with g_b, and let $\boldsymbol{\mu}_b = \{\mu_{b,1}, \ldots, \mu_{b,|T_b|}\}$ be the outputs of this tree associated with each of the $|T_b|$ leaves of T_b. We define

$$g_b(\mathbf{x}) := g(\mathbf{x}; T_b; \boldsymbol{\mu}_b) := \mu_{b,i},$$

where $\mu_{b,i}$ is the output of T_b associated with $\mathbf{x}$.

The BART model thus assumes that the regression function is a sum of the outputs

of each regression tree,

$$\mathbb{E}[Y|\mathbf{x}] = \sum_{b=1}^{B} g_b(\mathbf{x}) = \sum_{b=1}^{B} g(\mathbf{x}; T_b; \boldsymbol{\mu}_b).$$

The parameters of the model are therefore $\theta = (T_1, \boldsymbol{\mu}_1, \ldots, T_B, \boldsymbol{\mu}_B, \sigma)$. The prior for θ is chosen such that the following conditional independences hold:

$$f(\theta) = f(\sigma) \prod_{b=1}^{B} f(T_b) f(\boldsymbol{\mu}_b | T_b),$$

and

$$f(\boldsymbol{\mu}_b | T_b) = \prod_{i=1}^{|T_b|} f(\mu_{b,i} | T_b).$$

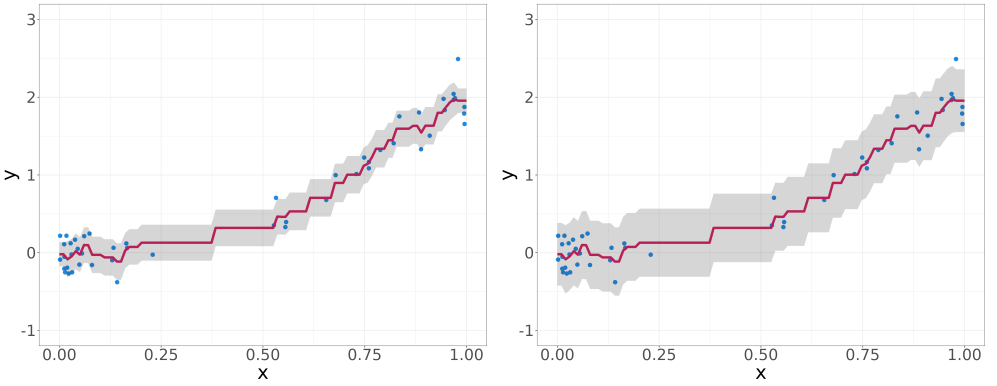
Each of these components is chosen as follows:

- The prior distribution $f(T_j)$ is defined following Chipman et al. (1998). Concretely, the probability that a node at depth d is not a leaf is given by $\alpha(1+d)^{-\beta}$, where $\alpha \in (0,1)$ and $\beta \geq 0$ are hyperparameters. Additionally, the prior distribution for selecting the splitting variable at each node is uniform, as is the prior distribution for the threshold used to split that variable.
- The prior distribution for each $\mu_{b,i}$ is assumed to be Gaussian. For details on the selection of its hyperparameters, refer to Chipman et al. (1998).
- The prior distribution for σ^2 is taken to be the inverse chi-square distribution. Again, see Chipman et al. (1998) for guidance on selecting its hyperparameter.

Chipman et al. (1998) shows how MCMC can be performed to sample from the posterior distribution induced by this prior and model.

Figure 6.4 presents a scatter plot of the data, the estimated regression line, the 95% credible interval around $r(\mathbf{x})$ from the posterior distribution $f(\mathbf{x})|\mathbf{x}, D$ (left panel), and the 95% prediction intervals from the predictive distribution of $Y|\mathbf{x}, D$ (right panel) for the same simulated example as Figure 6.2. The model was fitted to this data using the *BART* package in R (Sparapani et al., 2021); see the full code at the BART Notebook.

See Hill et al. (2020) and Tan and Roy (2019) and references therein for extensions of this model. These extensions, for instance, allow for different distributions for ϵ.



(a) Epistemic uncertainty around $r(\mathbf{x})$ based on the posterior distribution of $f(\mathbf{x})|\mathbf{x}, D$

(b) Prediction sets based on the predictive distribution of $Y|\mathbf{x}, D$

Figure 6.4: Bayesian Additive Regression Trees: The plot illustrates the original data points in blue, the estimated regression line in red, and the credible/prediction bands as the shaded area.

6.4 Monte Carlo Dropout

Monte Carlo dropout is a technique originally designed to avoid overfitting of a neural network (Srivastava et al., 2014), but that can also be used to measure the uncertainty around point predictions in such networks. This section provides a succinct overview of this technique.

Recall that, in a regression context, a feed-forward neural network outputs an estimate with the shape (Section 2.6.6)

$$g(\mathbf{x}) = a \circ \mathbb{H}_H \dots \tilde{a} \circ \mathbb{H}_1 \cdot \tilde{a} \circ \mathbb{H}_0 \cdot \tilde{\mathbf{x}},$$

where $\tilde{\mathbf{x}} = (1, \mathbf{x})$ and $\tilde{a}(\mathbf{y}) = (1, a(\mathbf{y}))$. In a network with dropout, the output (at training time) is instead given by

$$g(\mathbf{x}) = a \circ \mathbb{H}_H \left(\mathbf{b}_H \odot \dots \tilde{a} \circ \mathbb{H}_1 \cdot \left(\mathbf{b}_1 \odot \tilde{a} \circ \mathbb{H}_0 \cdot \left(\mathbf{b}_0 \odot \tilde{\mathbf{x}}\right)\right)\right),$$

where $\odot$ denotes the Hadamard product, and each $\mathbf{b}_i$ is a vector in which the first component is one, and the remaining ones are independent Bernoulli random variables with some predefined parameter p. A different set of $\mathbf{b}_i$'s is drawn at each training

iteration. In words, dropout randomly sets the activation function of each node to be zero with probability p. Intuitively, dropout decreases how much a network depends on each parameter, thus preventing overfitting. Indeed, it can be shown that it effectively performs stochastic gradient descent on a regularized version of the loss function (Wager et al., 2013).

If used during prediction time, dropout induces a distribution over the possible outputs of a fitted network. This randomness can be used to quantify the uncertainty around the predictions. Specifically, consider a fixed network with estimated parameters $\mathbb{H}_0, \ldots, \mathbb{H}_H$. Let θ denote the network parameters that determine its output (that is, the output is a function of $\mathbf{x}$ and θ). The randomness on the Bernoulli random variables in dropout $\mathbf{b}_i$'s induces a distribution over θ, say $q(\theta)$. It has been shown that, under some conditions, $q(\theta)$ approximates the posterior distribution over θ given by a specific Bayesian deep Gaussian process (Gal and Ghahramani, 2016). More precisely, $q(\theta)$ is the distribution that has smallest Kullback-Leibler divergence to the true posterior $p(\theta|D)$, where D represents the training data, among a specific family of tractable distributions over θ. Thus, if a model for the aleatoric uncertainty, $Y|\mathbf{x}, \theta$, is available, we can approximate the Bayesian predictive distribution via

$$f(y|\mathbf{x}, D) \approx \int f(y|\mathbf{x}, \theta) q(\theta) d\theta.$$

In particular, we can effectively sample from the predictive distribution by:

(i) Sampling θ from $q(\theta)$. This can be effectively done by using Monte Carlo dropout on the network.

(ii) Sampling from $Y|\mathbf{x}, \theta$ using the model for the aleatoric uncertainty.

See also Kendall and Gal (2017) and Valdenegro-Toro and Mori (2022) for variations and further details of this idea, and Folgoc et al. (2021) for a criticism of this approximation.

6.4.1 Batch normalization

Batch normalization, initially devised make training of neural networks faster and more stable (Ioffe and Szegedy, 2015), has also been discovered to serve as an implicit regularizer for standard network solutions (Luo et al., 2018), akin to the effects observed with Monte Carlo dropout (Section 6.4). Moreover, research

indicates its utility in quantifying uncertainty surrounding point predictions within these networks (Teye et al., 2018). This section shows a concise summary of the technique.

During training, batch normalization renormalizes the input of each node, say h, according to

$$\tilde{h} = \frac{h - \mu_B}{\sigma_B},$$

where μ_B and σ_B represent the mean and standard deviation of batch size being used, $B \subset \{1, \ldots, n\}$. Because the batch being used during training changes, this process induces a randomness in the output of the network. Specifically, let θ denote the network parameters that determine its output (that is, the output is a function of $\mathbf{x}$ and θ). The randomness on B induces a distribution over θ, say $q(\theta)$. It has been shown that, under some conditions, $q(\theta)$ approximates the posterior distribution over θ given by a specific Bayesian process (Teye et al., 2018). More precisely, $q(\theta)$ is the distribution that has smallest Kullback-Leibler divergence to the true posterior $p(\theta|D)$, where D represents the training data, among a specific family of tractable distributions over θ. Thus, as in Monte Carlo droput, if a model for the aleatoric uncertainty, $Y|\mathbf{x}, \theta$, is available, we can approximate the Bayesian predictive distribution via

$$f(y|\mathbf{x}, D) \approx \int f(y|\mathbf{x}, \theta) q(\theta) d\theta.$$

6.5 Deep ensembles

Deep ensembles (Lakshminarayanan et al., 2017) are inspired by Random Forests (Section 2.6.3). In this approach, one trains B neural networks that model the aleatoric uncertainty $f(y|\mathbf{x})$. However, instead of training these networks on bootstrap samples of the original dataset, the full dataset is always used. Instead, the initial weights for each network are independently selected at random, and the dataset is randomly shuffled for each training iteration. Moreover, the authors suggest the use of adversarial training to smooth the estimated densities (Goodfellow et al., 2014; Szegedy et al., 2013).

Each network gives a different estimate of $f(y|\mathbf{x})$, say $\widehat{f}_b(y|\mathbf{x})$. The estimated distribution that measures both the aleatoric and the epistemic uncertainty is then

defined to be the mixture

$$\frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(y|\mathbf{x}).$$

Hoffmann and Elster (2021) show that this corresponds to an approximation of the Bayesian predictive density $\widehat{f}(y|\mathbf{x}, D)$ for a specific prior. See Rahaman et al. (2021) for criticisms to this approach.

6.6 The Bootstrap

The bootstrap (Efron and Tibshirani, 1994) is a flexible tool for constructing frequentist confidence sets. In the context of regression models, it can be used to quantify epistemic uncertainty about the regression function. Specifically, for a fixed $\mathbf{x} \in \mathcal{X}$, the goal of the bootstrap is to create an interval, $(L_{\mathbf{x}}(D), U_{\mathbf{x}}(D))$, such that

$$\mathbb{P}(L_{\mathbf{x}}(D) \leq r(\mathbf{x}) \leq U_{\mathbf{x}}(D)) \approx 1 - \alpha,$$

where $D = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ is the training data, and $r(\mathbf{x})$ is the true regression function.

There are several variants of the bootstrap method to achieve this goal (see, e.g., Shalizi 2013; Wasserman 2006 and references therein). A simple version, applicable when a method for estimating $r(\mathbf{x})$ is available, proceeds as follows:

1. **Resampling:** Generate B bootstrap samples, $D_1^*, D_2^*, \ldots, D_B^*$, by randomly sampling with replacement from the original dataset D. Each bootstrap sample D_b^* contains n points.

2. **Estimate the Regression Function:** For each bootstrap sample D_b^*, compute the estimate $\hat{r}_b(\mathbf{x})$ of the regression function using the same method applied to the original data. This results in B estimates: $\hat{r}_1(\mathbf{x}), \hat{r}_2(\mathbf{x}), \ldots, \hat{r}_B(\mathbf{x})$.

3. **Construct Confidence Intervals:** For each $\mathbf{x}$, construct a confidence interval for $r(\mathbf{x})$ by taking the $\alpha/2$ and $1 - \alpha/2$ quantiles of the distribution of $\hat{r}_b(\mathbf{x})$ across the B bootstrap samples.

In addition to confidence sets, the bootstrap can also be used to create *prediction sets* by resampling from the original data, as discussed in Davison and Hinkley (1997).

This allows for the capture of both aleatoric and epistemic uncertainties. However, the bootstrap requires several assumptions to be valid, and its performance is asymptotic in the sample size n (Hall, 1992).

Figure 6.5 demonstrates the application of the bootstrap procedure to construct confidence intervals around a regression function; the code to obtain this plot is avaiable at the Bootstrap Notebook. Three polynomial models (as in Example 2.1), with degrees 5, 8, and 10, were fitted to a dataset containing dense regions at the extremes of the x range and sparse data in the middle. The solid lines represent the predicted mean function for each model, while the shaded areas depict the 95% bootstrap confidence intervals. The varying width of the confidence intervals across the x values illustrates the epistemic uncertainty captured by the bootstrap—the intervals are wider in regions with sparse data, where uncertainty is higher. Notably, the degree of uncertainty is also influenced by the choice of model, with each polynomial producing different levels of uncertainty.

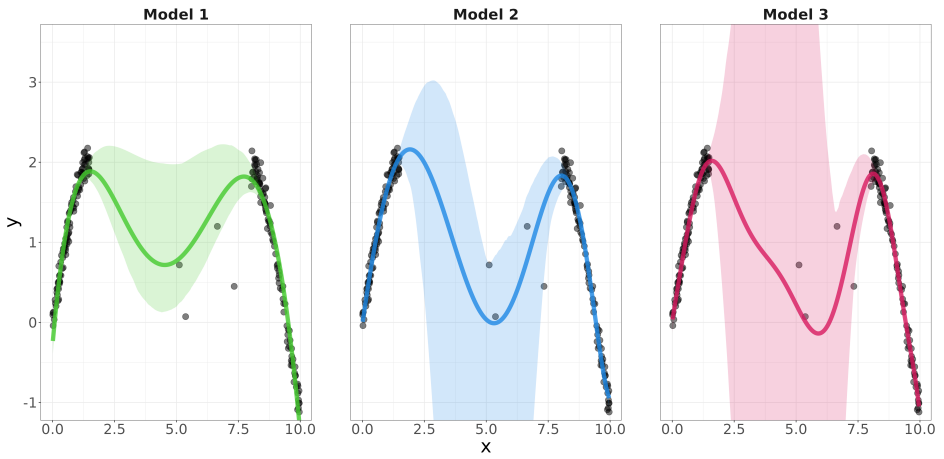


Figure 6.5: Bootstrap predictions for three polynomial regression models. The solid lines represent the mean predicted regression function, while the shaded areas indicate 95% bootstrap confidence intervals. Wider intervals reflect increased epistemic uncertainty, especially in areas with sparse data.

6.7 Summary

In this chapter, we discussed several techniques for quantifying both aleatoric and epistemic uncertainties in machine learning models.

Bayesian models offer a structured way to incorporate both types of uncertainty through the posterior predictive distribution, $f(y|\mathbf{x}, D)$. We demonstrated that using $f(y|\mathbf{x}, D)$ in oracle sets is sound from a Bayesian decision-theoretic perspective.

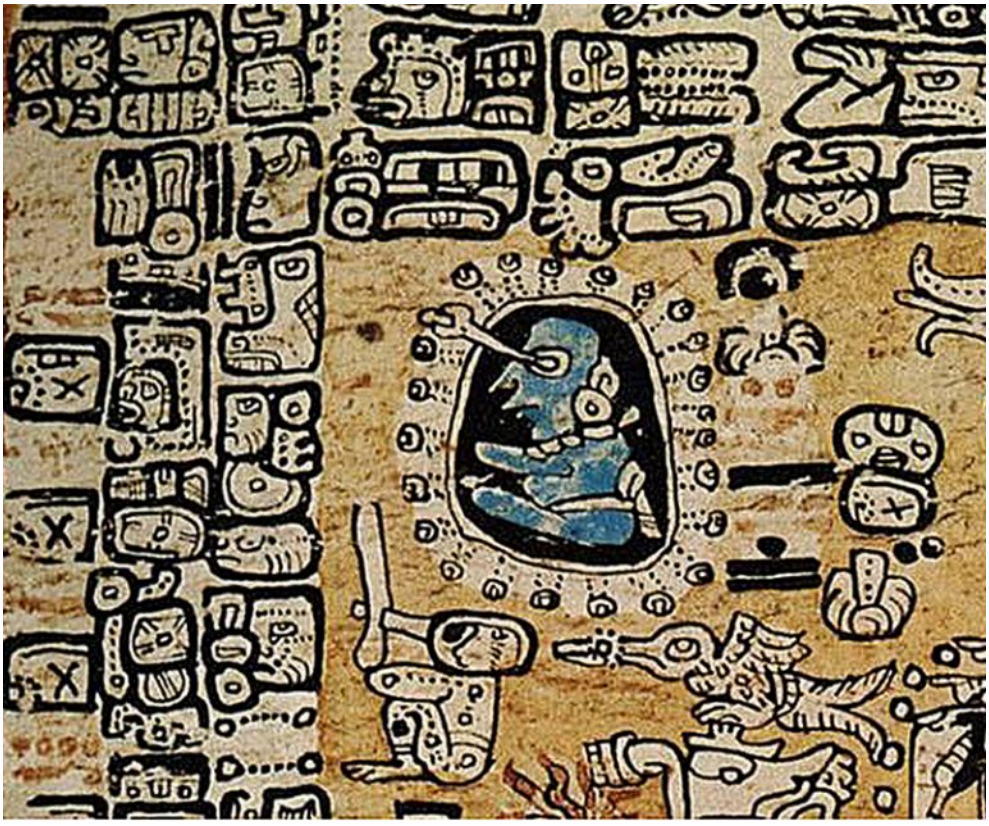
We explored different models that produce variations of $f(y|\mathbf{x}, D)$. Gaussian processes provide a flexible, nonparametric approach to capturing uncertainty in complex, nonlinear settings. BART, with its tree-based structure, offers a more interpretable model, making it particularly useful when transparency is important. Techniques like Monte Carlo Dropout and Batch Normalization, initially developed to improve neural network training, repurpose their randomness to estimate uncertainty and approximate Bayesian models. Finally, Deep Ensembles leverage multiple neural networks with random initialization to capture model uncertainty. Finally, bootstrap methods provide a frequentist approach to measuring epistemic uncertainty by refitting models on variations of the original dataset.

Part II

Applications

Chapter 7

Photometric Redshift Prediction



Mayan astronomer depicted in the Madrid Codex, page 34; Mexico, 15th century.

.

> To confine our attention to terrestrial matters would be to limit the human spirit.
>
> Stephen Hawking

This chapter provides applications of UQ techniques to the problem of photometric redshift estimation, a crucial task in astrophysics. Redshift, which measures how much the wavelength of light from a galaxy has been stretched due to the expansion of the universe, is essential for determining the distance to galaxies and for inferring cosmological parameters. While spectroscopy provides precise redshift measurements, it is resource-intensive and impractical for large-scale sky surveys. Consequently, photometry is used as a faster alternative. This technique records the radiation from astronomical objects through approximately 5-10 broad-band filters, providing quicker measurements that are less detailed than those obtained from spectroscopy.

In photometric redshift estimation, the goal is to predict the redshift, denoted as y, of an object based on its observed photometric covariates $\mathbf{x}$. This prediction is made using a dataset of galaxies with spectroscopically confirmed redshifts. Due to degeneracies – situations where galaxies at different distances appear to have similar colors in their photometric data because their light passes through similar filters – and intricate observational noise, probability densities in the form of $f(y|\mathbf{x})$ provide a more suitable description of the relationship between $\mathbf{x}$ and z than traditional regression $\mathbb{E}[Y|\mathbf{x}]$ (Almosallam et al., 2016; Beck et al., 2016; Benitez, 2000; Dalmasso et al., 2020b; Dey et al., 2021; Malz and Hogg, 2022; Mandelbaum et al., 2008; Zhou et al., 2021).

In Section 7.1 we apply some of the techniques presented in the last chapters to estimate the redshift of galaxies simulated by the photo-z collaboration from the Vera C. Rubin Observatory. Then, in Section 7.2, we predict the redshift from quasares observed by the Southern Photometric Local Universe Survey.

7.1 Vera C. Rubin Observatory

In this section, we leverage data generated from the Rubin Observatory's Legacy Survey of Space and Time (LSST)-Dark Energy Science Collaboration photo-z data challenge (Schmidt et al., 2020) to conduct a comparative analysis of $f(y|\mathbf{x})$ estimates. Additionally, we demonstrate the application of our recalibration procedure outlined in Section 4.2.2 using this dataset.

For our calibration set, we utilize the "training set" detailed in Schmidt et al. (2020), comprising approximately 44,000 instances. The Buzzard-highres-v1.0 simulation, based on the Chinchilla-400 N-body dark matter simulation, provides a mock catalogue of 238 million galaxies. Although the redshift range extends from 0 to 8.7, a color correction issue limits it to 0-2.0. The dataset, designed to simulate LSST observations, includes photometry in six bands (ugrizy) with added Gaussian noise. Our test set contains 399,356 galaxies.

We also adopt the same test set as described in Schmidt et al. (2020). Our recalibration process involves applying `Cal-PIT` (Section 4.2.2) to reshape `trainZ`, an estimate of the marginal distribution of redshifts. Despite `trainZ` being a naive estimate devoid of specific information about individual redshifts, Schmidt et al. (2020) has demonstrated its efficacy across various commonly used metrics assessing marginal coverage. This is a consequence of Theorem 4, which shows that the marginal distribution $f(y)$ leads to uniform PIT values.

We train $r^{\widehat{f}}$ on the calibration data and use `Cal-PIT` to recalibrate the CDEs of our validation and test sets. We assess the quality of our recalibrated CDEs with the L^2 loss (Section 3.1).

In Figure 7.1 (top), we present diagnostic ALPs applied to the original estimate, $\widehat{f}(y)$, for select galaxies in the test set. Despite the *marginal* PIT distribution closely aligning with the identity line, the ALPs reveal that $\widehat{f}(y)$ falls short as a reliable estimate. The local CDF of PIT for these instances exhibits significant deviations from the identity line, emphasizing the limitations of the estimate. Figure 7.1 (center) shows that multimodal CDEs can be recovered, even when the input CDE before calibration is unimodal.

Given that the true distributions are unknown, our ability to evaluate how reasonable such estimates are relies on indirect methods. The bottom section of Figure 7.1 presents a straightforward yet insightful illustration, confirming the meaningfulness of CDEs from `Cal-PIT`. In this figure, we juxtapose the CDEs with the actual redshift

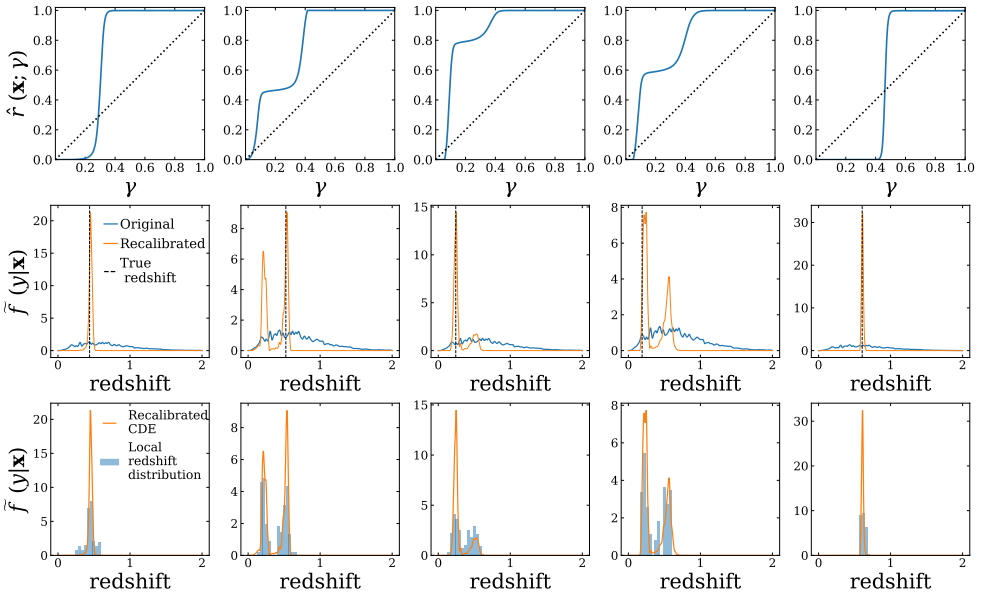


Figure 7.1: *Top:* Diagnostic ALPs for five galaxies before recalibration. *Center:* Photo-z CDEs for the corresponding galaxies before and after using `Cal-PIT` along with their true redshifts. `Cal-PIT` recovers bimodal CDEs even if our initial estimate was not bimodal. *Bottom:* Comparison of the CDEs with the distribution of true redshifts of other galaxies having similar imaging properties. We observe that the histograms show bimodal distributions only when the inferred CDEs are bimodal.

distribution of other galaxies possessing similar imaging data. The identification of these counterparts involves searching for galaxies in the training set whose colors and magnitudes (normalized by subtracting the mean and dividing by the standard deviation for each feature) fall within a Euclidean distance of 0.5 units from our selected galaxies. Figure 7.1 (bottom) exhibits their redshift distribution as an inverse-distance weighted histogram alongside their predicted CDEs. Notably, we observe bimodal histograms corresponding to bimodal CDEs and unimodal histograms aligning with unimodal CDEs, consistent with astronomers' expectations.

Furthermore, the performance of `Cal-PIT` is quantified in Table 7.1, showcasing a lower L^2 loss compared to any of the methods in the LSST-DESC data challenge (Schmidt et al., 2020). Interestingly, it approaches the performance level of FlexCode, and yields a substantial improvement over trainZ in terms of the L^2 loss, decreasing from -0.84 to -10.71 after recalibration.

Table 7.1: Comparison with methods benchmarked in the LSST-DESC Photo-z Data Challenge (Schmidt et al., 2020). In terms of L_2 loss, `Cal-PIT` performs better than all the other methods compared, including `FlexCode`.

Photo-z Algorithm	L^2 Loss
`ANNz2` (Sadeh et al., 2016)	-6.88
`BPZ` (Benitez, 2000)	-7.82
`Delight` (Leistedt and Hogg, 2017)	-8.33
`EAZY` (Brammer et al., 2008)	-7.07
`FlexCode` (Izbicki and Lee, 2017)	**-10.60**
`GPz` (Almosallam et al., 2016)	-9.93
`LePhare` (Arnouts et al., 1999)	-1.66
`METAPhoR` (Cavuoti et al., 2017)	-6.28
`CMNN` (Graham et al., 2018)	-10.43
`SkyNet` (Graff et al., 2014)	-7.89
`TPZ` (Carrasco Kind and Brunner, 2013)	-9.55
`trainZ` (Schmidt et al., 2020)	-0.83
`Cal-PIT`	**-10.71**

7.2 Southern Photometric Local Universe Survey (S-PLUS)

S-PLUS (Southern Photometric Local Universe Survey; Mendes de Oliveira et al. 2019) is an extensive sky survey aiming to cover approximately 9300 square degrees of the Southern Sky using an optical system composed of twelve filters. These filters include five broad bands (ugriz), similar to those used in the Sloan Digital Sky Survey (SDSS), as well as seven narrow bands. In addition to its intrinsic data, S-PLUS integrates photometric data from other surveys such as GALEX and WISE, extending the wavelength range and further improving photometric redshift predictions.

In this section, based on Nakazono et al. (2024), we evaluate the value of adding narrow bands to the photometric set of covariates in the prediction of quasar redshifts. Specifically, we compare two sets of features:

- **broad+GALEX+WISE**: u−r, g−r, r−i, r−z, FUV−r, NUV−r, r−W1, r−W2
- **broad+GALEX+WISE+narrow**: the above eight broad-band colors extended with the same seven narrow-band colors as in the first case.

We use 33,151 labeled quasars. Further details about the analysis can be found in Nakazono et al. (2024).

We estimate the distribution over redshifts for each quasar using FlexCode (Sec-

tion 3.4) and a Bayesian Mixture Density Network (BMDN). The latter integrates a Mixture Density Network (Section 3.5) with Bayesian Neural Networks (Bishop, 1997), estimating both epistemic and aleatoric uncertainties.

The L_2 loss function values for FlexCode are -2.88 (with narrow bands) and -1.32 (without narrow bands). For BMDN, these values are -2.85 (with narrow bands) and -1.46 (without narrow bands). Thus, narrow bands improve both models. Moreover, the FlexCode model trained with narrow bands produced the best photo-z's PDF, as evidenced by having the lowest L_2 loss.

Figure 7.2 evaluates the models by plotting the histogram of the PIT statistic (Section 4.1) for BMDN and FlexCode models, both trained with and without narrow bands. The figure indicates that all probability density functions (PDFs) are under-dispersed, suggesting that these models could be improved by exploring alternative methods.

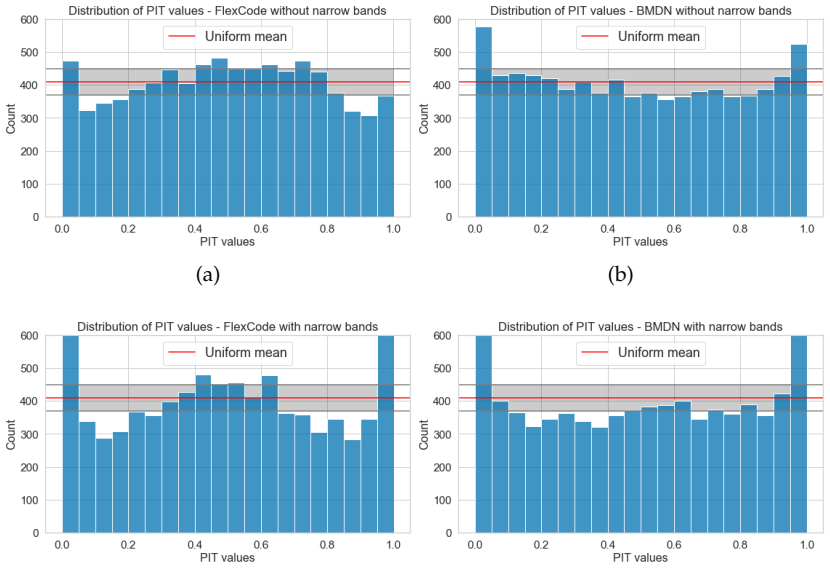


Figure 7.2: Distrution of PIT statistic for: (a) FlexCode trained without narrow bands (broad+GALEX+WISE); (b) BMDN trained without narrow bands; (c) FlexCode trained with narrow bands (broad+GALEX+WISE+narrow); and (d) BMDN trained with narrow bands. The PDFs are well-calibrated if the PIT distribution is close to a uniform distribution (red line).

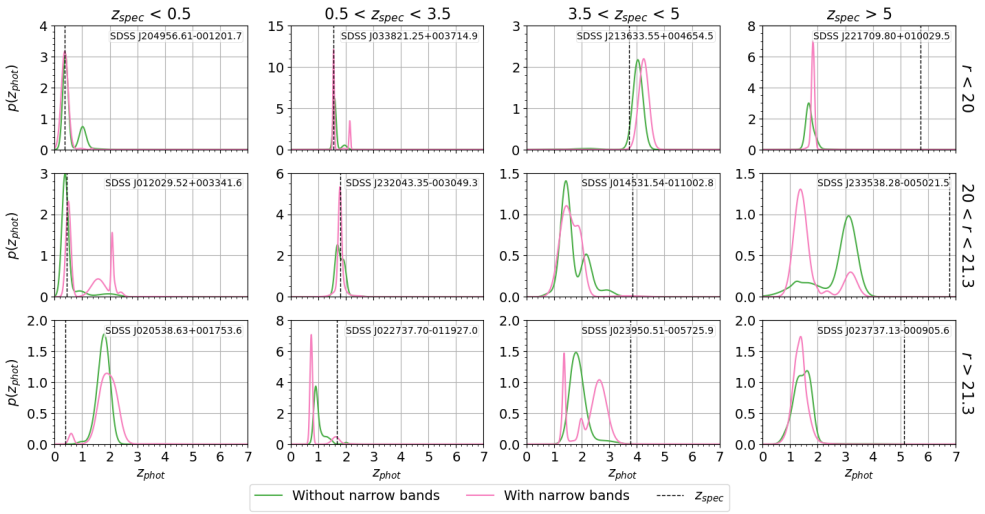


Figure 7.3: Density estimates from BMDN for the redshift of 12 quasar samples from the testing set. The spectroscopic redshift is indicated by the dashed vertical line. Green curves represent the model trained with broad+GALEX+WISE, while pink curves represent the model trained with broad+GALEX+WISE+narrow.

Figures 7.3 and 7.4 show examples of estimated PDFs for both sets of features and both methods. The figures demonstrate that the primary peaks of the PDFs are close to the true redshift for quasars with redshift $z_{\rm spec} < 3.5$ and $r < 21.3$ for both methods, which is expected since these ranges include most of the quasars in the spectroscopic sample. FlexCode shows sharper PDFs when narrow bands are included, indicating more precise redshift estimates. For BMDN, this precision improvement is notable only for specific objects. Redshift estimates for quasars with redshift $z > 5$ are highly underestimated, likely due to potential misclassification in the SDSS pipeline. The object SDSS J022737.70-011927.0 exemplifies color-redshift degeneracy, as the PDFs from FlexCode and BMDN indicate that the measured colors for this quasar correspond to approximately two distinct redshift estimates. Interestingly, BMDN assigns a lower probability to the true redshift value, while FlexCode assigns it a higher probability.

Figure 7.5 shows the estimated feature importances for FlexCode (Section 3.4.1). The top two most important features for quasar photo-z estimation are r − W1 and u − r. These features measure the overall shape of the SED over a wide wavelength range.

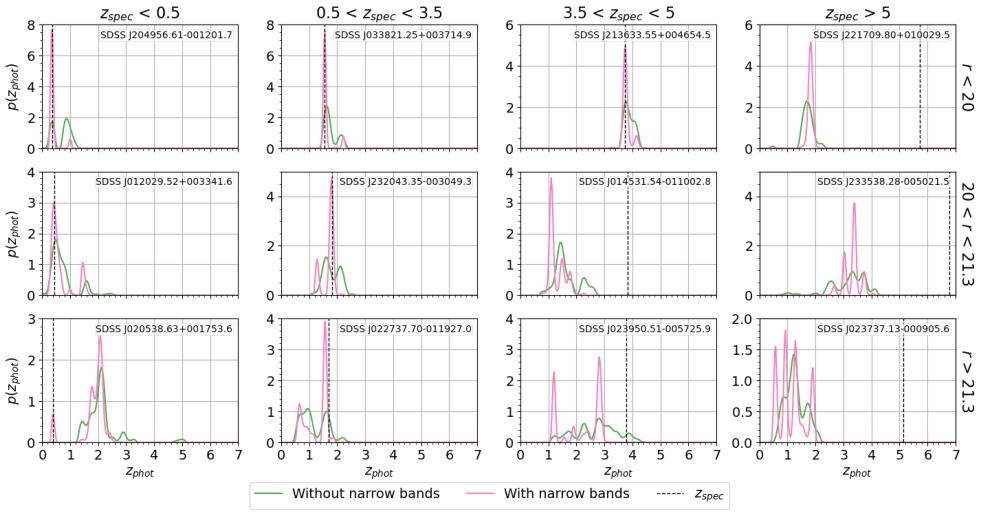


Figure 7.4: Density estimates from FlexCode for the redshift of 12 quasar samples from the testing set. The spectroscopic redshift is indicated by the dashed vertical line. Green curves represent the model trained with broad+GALEX+WISE, while pink curves represent the model trained with broad+GALEX+WISE+narrow.

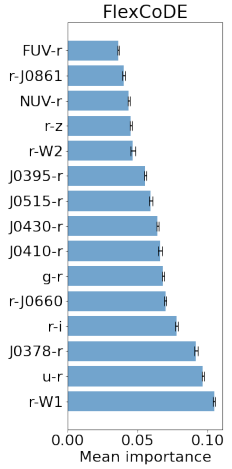


Figure 7.5: Average feature importances for FlexCoDE.

Chapter 8

Dengue Nowcasting

The Plague of Flies. James Tissot, circa 1896-1902, Jewish Museum, New York.

> An ounce of prevention is worth a pound of cure.
>
> Benjamin Franklin

Dengue remains a significant public health challenge, especially in tropical regions where the *Aedes aegypti* mosquito is common. The disease affects millions each year, with symptoms that can range from mild to severe. Since there is no specific treatment for dengue, predicting and managing outbreaks quickly is crucial to reduce its impact on communities (Lancet, 2024).

Traditional dengue surveillance systems are essential but often suffer from delays in reporting, which can slow down the response to outbreaks (Bastos et al., 2019). These delays make it harder for health authorities to act quickly, which is why nowcasting—predicting the current situation using real-time data—is so important. Nowcasting helps provide the most up-to-date information, allowing for better decision-making and more effective control measures.

However, prediction alone is insufficient. Equally important is quantifying the uncertainty in these predictions. This chapter demonstrates how Dengue-tracker (https://diseasesurveillance.github.io/) incorporates uncertainty quantification (UQ) into dengue nowcasting (Xiao et al., 2024). By including UQ, the system not only provides estimates of dengue cases but also prediction bands that indicate the possible range of outcomes. These prediction bands are crucial for helping health authorities assess risks and make informed decisions (Codeço et al., 2016).

8.1 Prediction Model

The core idea behind Dengue-tracker is that during a dengue outbreak, the number of searches for dengue-related terms in search engines increases. Thus, we use Google Trends data as covariates to estimate the actual dengue cases, even when official reports are delayed.

To model this relationship, we selected specific search terms that are highly correlated with dengue outbreaks. These terms are “Dengue" and “Sintomas Dengue" (Dengue Symptoms). The choice of these terms was guided by historical analysis, where we observed consistent spikes in search volumes corresponding with reported

dengue cases.

Our nowcasting model can be formally described as follows. Let $Y_{l,t}$ represent the count of new dengue cases in location l during week t. Assume we are currently in week $t = t_0$. Due to reporting delays, $Y_{l,t}$ may be unreliable for weeks close to t_0. To correct these estimates, we assume that $Y_{l,t}$ is reliable for $t \leq t_0 - K$ (where we set $K = 4$ in our case, based on empirical evidence that suggests that most reporting delays typically do not exceed four weeks in most locations).

To predict $Y_{l,t}$ for recent weeks, we utilize Google Trends search data, denoted as $X_{k,l,t}$, where $X_{k,l,t}$ represents the volume of searches on Google Trends for term k in location l and week t. Our model is specified as:

$$Y_{l,t} = \beta_{0,l} + \sum_{k=1}^{K} \beta_{k,l} \cdot X_{k,l,t} + \epsilon_{l,t},$$

where $\epsilon_{l,t}$ is an error term with zero mean and zero variance. This linear model suggests that dengue cases in a particular location and week are approximately proportional to the search activity for relevant terms.

To account for regional variations, we fit the model independently for each location l, using all available data up to $t \leq t_0 - K$. We employ standard least squares estimation (Equation 3.4) to obtain the coefficients $\widehat{\beta}_{0,l}$ and $\widehat{\beta}_{k,l}$, which are then used to predict $Y_{l,t}$ for more recent weeks:

$$Y_{l,t} = \widehat{\beta}_{0,l} + \sum_{k=1}^{K} \widehat{\beta}_{k,l} \cdot X_{k,l,t}.$$

8.2 Uncertainty Quantification for the Number of Cases

To account for the uncertainty in the predictions provided by the Dengue-tracker, we evaluate three different methods for building 90% prediction intervals:

- **Linear Model:** We use the standard prediction intervals derived from the Gaussian linear model used to generate the point predictions. These intervals are detailed in Section 5.3.2.
- **Linear Quantile Regression:** We apply linear quantile regression (Section 3.10) to model the conditional quantiles of Y_t given X_t. This method estimates the $\alpha/2$ and $1 - \alpha/2$ quantiles of $f(y_t|\mathbf{x}_t)$ using all available data up to $t_0 - K$.

- **Conformalized Quantile Regression (CQR):** We enhance the quantile regression approach by applying split-Conformalized Quantile Regression (CQR; Romano et al. 2019). This method uses the conformal score $h(\mathbf{x}, y) = \max\{\widehat{q}_{\alpha/2}(\mathbf{x}) - y, y - \widehat{q}_{1-\alpha/2}(\mathbf{x})\}$, where $\widehat{q}_{\alpha/2}$ and $\widehat{q}_{1-\alpha/2}$ are the estimated quantiles. Although CQR is primarily designed for i.i.d. data, split-conformal has shown robust performance in practice, provided that $f(y_t|\mathbf{x}_t)$ is stationary and exhibits weak dependence (Oliveira et al., 2022).

8.3 Results

The official data used in this analysis comes from the Notifiable Diseases Information System (SINAN; Sistema de Informação de Agravos de Notificação (SINAN) 2024), a critical component of Brazil's public health infrastructure managed by the Brazilian Ministry of Health. SINAN collects, processes, and disseminates data on notifiable diseases, including dengue, across the country. For our analysis, we specifically model state-level data to capture regional variations in dengue incidence. While SINAN provides a comprehensive dataset that is crucial for monitoring disease trends and guiding public health responses, the inherent delays in reporting and data entry can create a lag in the availability of real-time information. This lag highlights the importance of nowcasting techniques to supplement and enhance the insights derived from SINAN data.

Figure 8.1 presents the estimated curves for dengue cases in three Brazilian states: Rio de Janeiro, Pernambuco and São Paulo. We fit a different model independently for each state. Each plot includes the 90% prediction bands obtained via CQR alongside the officially reported case numbers. We compare our approach to InfoDengue, which uses a Bayesian hierarchical model to nowcast events by analyzing temporal, spatial, and delay-related variability by evaluating weekly updating patterns (Codeço et al., 2016). The results demonstrate that the model provides accurate, timely estimates of dengue cases.

Table 8.1 presents the mean and standard error of these coverage probabilities for the three UQ models. The conformal model consistently shows coverage probabilities closer to the nominal 90% value across most states compared to the standard linear model. However, it does not deviate much from the quantile regression approach.

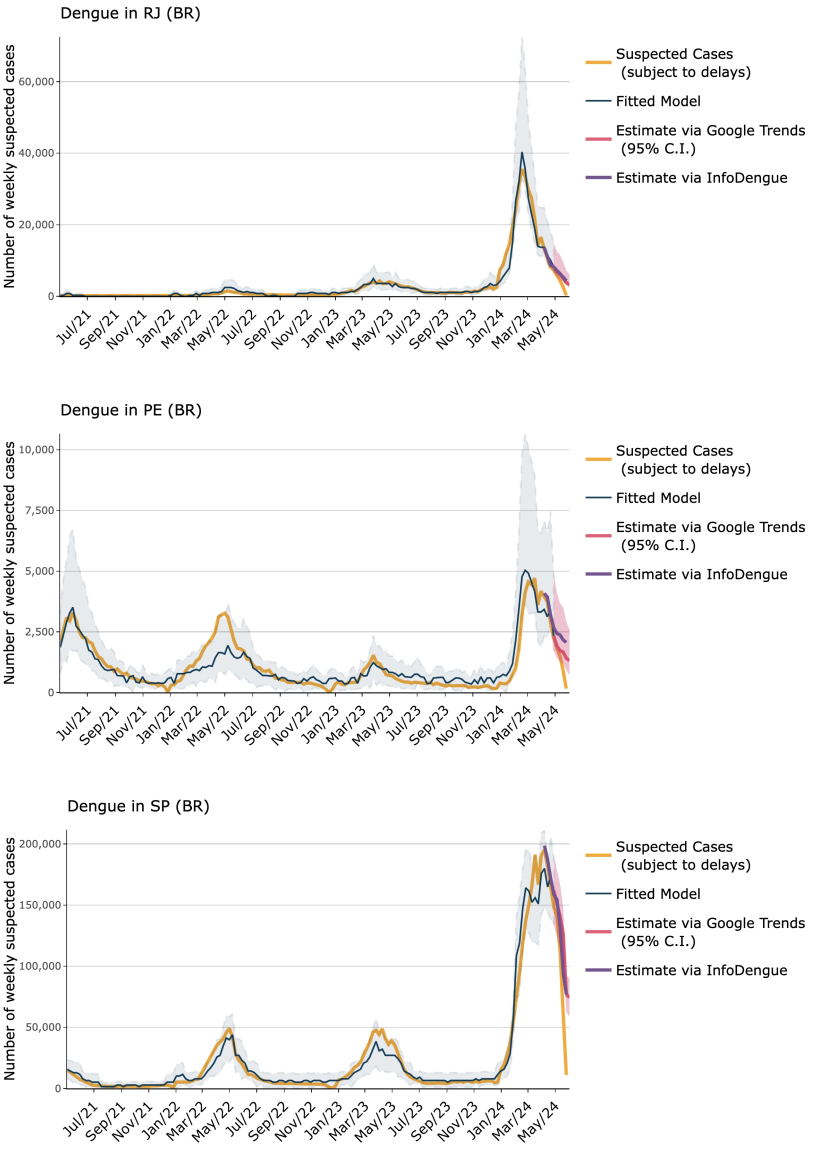


Figure 8.1: Estimated dengue case curves for Rio de Janeiro, Pernambuco and São Paulo. The plots feature 90% prediction bands obtained via CQR and the official reported case numbers, illustrating the model's accuracy in estimating dengue cases without delays.

Table 8.1: Mean and standard error (in parentheses) of coverage probabilities for different models across Brazilian states. Nominal coverage is 90%.

State	Linear Model	Linear QR	Conformal
AC	0.5 (<0.001)	0.875 (0.006)	0.875 (0.006)
AL	0 (<0.001)	1 (<0.001)	1 (<0.001)
AM	0.751 (0.012)	0.875 (0.006)	0.875 (0.006)
AP	0.25 (<0.001)	0.875 (0.006)	0.875 (0.006)
BA	0.625 (0.006)	1 (<0.001)	1 (<0.001)
BR	0 (<0.001)	1 (<0.001)	1 (<0.001)
CE	0.125 (0.006)	0.125 (0.006)	0.125 (0.006)
DF	0.249 (0.012)	1 (<0.001)	1 (<0.001)
ES	0.125 (0.006)	0.5 (<0.001)	0.5 (<0.001)
GO	0.625 (0.006)	0.75 (<0.001)	0.75 (<0.001)
MA	0.75 (<0.001)	1 (<0.001)	1 (<0.001)
MG	0.625 (0.006)	1 (<0.001)	1 (<0.001)
MS	0.375 (0.006)	0.75 (<0.001)	0.75 (<0.001)
MT	0.375 (0.006)	1 (<0.001)	1 (<0.001)
PA	0.25 (<0.001)	0.875 (0.006)	0.875 (0.006)
PB	0.125 (0.006)	1 (<0.001)	1 (<0.001)
PE	0.125 (0.006)	0.25 (<0.001)	0.25 (<0.001)
PI	0.125 (0.006)	1 (<0.001)	1 (<0.001)
PR	0.875 (0.006)	1 (<0.001)	1 (<0.001)
RJ	0.125 (0.006)	1 (<0.001)	1 (<0.001)
RN	0 (<0.001)	0.25 (<0.001)	0.25 (<0.001)
RO	0.125 (0.006)	1 (<0.001)	1 (<0.001)
RR	0.75 (<0.001)	0.75 (<0.001)	0.75 (<0.001)
RS	0.125 (0.006)	1 (<0.001)	1 (<0.001)
SC	0.625 (0.006)	1 (<0.001)	1 (<0.001)
SE	0.375 (0.006)	1 (<0.001)	1 (<0.001)
SP	0 (<0.001)	1 (<0.001)	1 (<0.001)
TO	1 (<0.001)	1 (<0.001)	1 (<0.001)

Chapter 9

Likelihood-free Inference (LFI)

Las Meninas. Diego Velázquez, 1656, Museo del Prado, Madrid.

.

> Truth is much too complicated to allow anything but approximations.
>
> John von Neumann

In many statistical inference problems within the sciences, establishing the relationship between the parameters of interest and observable data is often intricate. However, it is often feasible to create computational forward models that simulate realistic data. This models take the parameters values θ as inputs, and output observable data $\mathbf{x}$ from that model at θ. Examples of this occur in genetics (Beaumont, 2010; Estoup et al., 2012), astronomy (Cameron and Pettitt, 2012; Ho et al., 2019; Weyant et al., 2013), high-energy physics (Kieseler et al., 2022), and many other science fields. In such scenarios, the complexity of the data generation process often hinders the derivation of an analytically precise form for the likelihood function. Thus, one cannot use standard Bayesian or frequentist tools as no tractable form for the posterior distribution or the likelihood function is available.

These challenges have sparked a recent surge in interest in "likelihood-free inference" methods, often called Simulation-Based Inference (SBI), or Simulator-Based Inference. This category includes approaches such as "Approximate Bayesian Computation" (ABC) (Marin et al., 2012; Marin et al., 2016), amortized likelihood estimators (Fasiolo et al., 2018; Gutmann and Corander, 2016; Izbicki et al., 2014; Järvenpää et al., 2021; Lueckmann et al., 2019; Papamakarios et al., 2019; Picchini et al., 2020; Wood, 2010), and amortized posterior estimators (Greenberg et al., 2019; Lueckmann et al., 2017; Papamakarios and Murray, 2016). See Cranmer et al. (2020) for an overview.

Here, we first described the approach by Izbicki et al. (2019), which uses FlexCode to estimate posterior distributions (Section 9.1). We discuss how such an approach is useful, and what needs to be adapted. Then, we discuss the LF2I (Likelihood-Free Frequentist Inference) framework introduced by Dalmasso et al. (2020a) and Dalmasso et al. (2021) (Section 9.3).

In this chapter, we assume we have a parametric model F_θ, $\theta \in \Theta$, that models observable data $\mathbf{X} \in \mathcal{X}$, that is, $\mathbf{X} \sim F_\theta$. Our goal is to recover θ that generated the true observed data $\mathbf{x}_o$. We assume we are able to generate from F_θ for any given θ, even if the likelihood function induced by F_θ is not analytically tractable. We assume

Θ is continuous, even though several of the methods discussed here can be used if it is discrete.

9.1 Approximate Bayesian Computation via Conditional Density Estimation (ABC-CDE)

Given a prior distribution over Θ, $f(\theta)$, the posterior distribution $f(\theta|\mathbf{x})$ is simply the conditional density of θ on $\mathbf{x}$. Examining LFI through a *conditional density estimation* (CDE) framework provides a systematic approach to tackle the following three challenges:

(i) how to efficiently estimate the posterior density $f(\theta|\mathbf{x}_o)$, where $\mathbf{x}_o$ is the observed sample; in particular, in settings with complex, high-dimensional data and costly simulations,

(ii) selecting appropriate tuning parameters and evaluating the performance of ABC and related methods solely based on simulations and observed data; that is, without knowing the true posterior distribution, and

(iii) how to best choose summary statistics for ABC and related methods when given a very large number of candidate summary statistics.

9.1.1 Estimating the Posterior Density via CDE

One way to estimate $f(\theta|\mathbf{x}_o)$ via CDE methods is to generate an i.i.d. sample $\mathcal{T} = \{(\theta_i, \mathbf{x}_i)\}$ by sampling $\theta \sim f(\theta)$ and then $\mathbf{x} \sim f(\mathbf{x}|\theta)$ for each pair. Applying the chosen CDE method to $\mathcal{T}$ yields an estimated density $\widehat{f}(\theta|\mathbf{x})$. However, this naive approach may yield suboptimal results because some $\mathbf{x}$ values are distant from the observed data $\mathbf{x}_o$. In ABC applications, the focus is solely on estimating $f(\theta|\mathbf{x})$ for $\mathbf{x}_o$, not all possible $\mathbf{x}$.

To address this issue, one can instead estimate $f(\theta|\mathbf{x})$ by training on a set $\mathcal{T}$ comprising samples $\mathbf{x}$ near $\mathbf{x}_o$. This training set is created by using a basic ABC rejection sampling algorithm (see Algorithm 1) with a fixed distance function $d(\mathbf{x}, \mathbf{x}_o)$ (possibly based on summary statistics) and tolerance level ϵ.

We apply our conditional density estimator to the new training set $\mathcal{T}$ and evaluate it at $\mathbf{x} = \mathbf{x}_o$. This procedure can be regarded as an ABC post-processing technique

Algorithm 1: Training set for CDE via Rejection ABC

Input: Tolerance level ϵ, number of desired samples B, distance function d, sample $\mathbf{x}_0$

Output: Training set $\mathcal{T}$ which approximates the joint distribution of $(\theta, \mathbf{X})$ in a neighborhood of $\mathbf{x}_0$

$\mathcal{T} \leftarrow \{\}$;
while $|\mathcal{T}| < B$ **do**
  Sample $\theta \sim f(\theta)$;
  Sample $\mathbf{x} \sim f(\mathbf{x}|\theta)$;
  if $d(\mathbf{x}, \mathbf{x}_0) < \epsilon$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\theta, \mathbf{x})\}$;
  end
end
return $\mathcal{T}$;

(Marin et al., 2012): the first ABC approximation to the posterior is obtained via the sample $\theta_1, \ldots, \theta_B$, which can be seen as a sample from $f(\theta|d(\mathbf{X}, \mathbf{x}_o) < \epsilon)$. We take the results of the ABC sampler and estimate the conditional density exactly at the point $\mathbf{x} = \mathbf{x}_o$ using other forms of conditional density estimation.

In what follows, we assume for simplicity that we are interested in estimating the posterior distribution of a single parameter $\theta \in \mathbb{R}$, even if there are several parameters in the problem.[1]

9.1.2 Method Selection: Comparing Different Estimators of the Posterior

Surrogate Loss. Ultimately, we need to be able to decide which approach is best for approximating $f(\theta|\mathbf{x}_o)$ without knowledge of the true posterior. Ideally, we seek an estimator $\widehat{f}(\theta|\mathbf{x}_o)$ minimizing the L_2 loss at $\mathbf{x}_o$:

$$L_{\mathbf{x}_o}(\widehat{f}, f) = \int (\widehat{f}(\theta|\mathbf{x}_o) - f(\theta|\mathbf{x}_o))^2 d\theta. \tag{9.1}$$

However, since $L_{\mathbf{x}_o}$ relies on the true $f(\theta|\mathbf{x}_o)$, which is unknown, method selection

[1]Most inference problems can be expressed as the computation of unidimensional quantities. Say one is interested in estimating m functions of parameters of the model $\boldsymbol{\theta}$, $g_1, \ldots, g_m$. One can then (i) use ABC to obtain a single simulation set $\mathcal{T} = \{(\boldsymbol{\theta}_1, \mathbf{x}_1), \ldots, (\boldsymbol{\theta}_B, \mathbf{x}_B)\}$ (ii) for each function g_i, compute $\mathcal{T}^{g_i} = \{(g_i(\boldsymbol{\theta}_1), \mathbf{x}_1), \ldots, (g_i(\boldsymbol{\theta}_B), \mathbf{x}_B)\}$, and (iii) fit a (univariate) conditional density estimator to $\mathcal{T}^{g_i}$ to estimate $f(g_i(\boldsymbol{\theta})|\mathbf{x}_o)$.

becomes challenging. To address this, we instead use a *surrogate* loss function:

$$L^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f) = \int\int (\widehat{f}(\theta|\mathbf{x}) - f(\theta|\mathbf{x}))^2 \frac{f(\mathbf{x})\mathbb{I}(d(\mathbf{x},\mathbf{x}_o) < \epsilon)}{\mathbb{P}(d(\mathbf{X},\mathbf{x}_o) < \epsilon)} d\theta d\mathbf{x},$$

This surrogate loss ensures a close fit within an ϵ-neighborhood of $\mathbf{x}_o$. The denominator $\mathbb{P}(d(\mathbf{X},\mathbf{x}_o) < \epsilon)$ is a constant, making $\frac{f(\mathbf{x})\mathbb{I}(d(\mathbf{x},\mathbf{x}_o)<\epsilon)}{\mathbb{P}(d(\mathbf{X},\mathbf{x}_o)<\epsilon)}$ a proper density in $\mathbf{x}$.

The advantage of the above definition is that we can directly *estimate* $L^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f)$ from the ABC posterior sample. Indeed, $L^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f)$ can be written as

$$\mathbb{E}_{\mathbf{X}'}\left[\int \widehat{f}^2(\theta|\mathbf{X})d\theta\right] - 2\mathbb{E}_{(\theta',\mathbf{X}')}\left[\widehat{f}(\theta|\mathbf{X})\right] + K_f, \tag{9.2}$$

where $(\theta', \mathbf{X}')$ is a random vector with distribution induced by a sample generated according to the ABC rejection procedure in Algorithm 1; and K_f is a constant that does not depend on the estimator $\widehat{f}(\theta|\mathbf{x}_o)$. Thus, given a sample of size B' of the ABC algorithm, $(\theta'_1, \mathbf{x}'_1), \ldots, (\theta'_B, \mathbf{x}'_B)$, we can estimate $L^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f)$ (up to the constant K_f) via

$$\widehat{L}^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f) = \frac{1}{B'}\sum_{k=1}^{B'}\int \widehat{f}^2(\theta|\mathbf{x}'_k)d\theta - 2\frac{1}{B'}\sum_{k=1}^{B'}\widehat{f}(\theta'_k|\mathbf{x}'_k). \tag{9.3}$$

When given a set of estimators $\mathcal{F} = \{\widehat{f}_1, \ldots, \widehat{f}_m\}$, we select the method with the smallest estimated surrogate loss,

$$\widehat{f} := \arg\min_{\widehat{f}\in\mathcal{F}} \widehat{L}^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f).$$

Next, we investigate the conditions under which the estimated surrogate loss is close to the true loss; the proofs for all results can be found in Izbicki et al. (2019). The following theorem states that, if $(\widehat{f}(\theta|\mathbf{x}) - f(\theta|\mathbf{x}))^2$ is a smooth function of $\mathbf{x}$, then the (exact) surrogate loss $L^{\epsilon}_{\mathbf{x}_o}$ is close to $L_{\mathbf{x}_o}$ for small values of ϵ.

Theorem 13. *Assume that, for every $\theta \in \Theta$, $g_\theta(\mathbf{x}) := (\widehat{f}(\theta|\mathbf{x}) - f(\theta|\mathbf{x}))^2$ satisfies the Hölder condition of order β with a constant K_θ[2] such that $K_H := \int K_\theta d\theta < \infty$. Then $|L^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f) - L_{\mathbf{x}_o}(\widehat{f}, f)| \leq K_H\epsilon^\beta = O(\epsilon^\beta)$*

The next theorem shows that the estimator $\widehat{L}^{\epsilon}_{\mathbf{x}_o}$ in Equation 9.3 does indeed converge to the true loss $L_{\mathbf{x}_o}(\widehat{f}, f)$.

[2]That is, there exists a constant K_θ such that for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ $|g_\theta(\mathbf{x}) - g_\theta(\mathbf{y})| \leq K_\theta(d(\mathbf{x},\mathbf{y}))^\beta$.

Theorem 14. *Let K_f be as in Equation 9.2. Under the assumptions of Theorem 13,* $|\widehat{L}^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f) + K_f - L_{\mathbf{x}_o}(\widehat{f}, f)| = O(\epsilon^{\beta}) + O_P(1/\sqrt{B'})$

Under some additional conditions, it is also possible to guarantee that not only the estimated surrogate loss is close to the true loss, but that the result holds uniformly for a finite class of estimators of the posterior distribution. This is formally stated in the following theorem.

Theorem 15. *Let $\mathcal{F} = \{\widehat{f}_1, \ldots, \widehat{f}_m\}$ be a set of estimators of $f(\theta|\mathbf{x}_o)$. Assume there exists M such that $|\widehat{f}_i(\theta|\mathbf{x})| \leq M$ for every $\mathbf{x}$, θ, and $i = 1, \ldots, m$.* [3] *Moreover, assume that for every $\theta \in \Theta$, $g_{i,\theta}(\mathbf{x}) := (\widehat{f}_i(\theta|\mathbf{x}) - f(\theta|\mathbf{x}))^2$ satisfies the Hölder condition of order β with constants K_θ such that $K_H := \int K_\theta d\theta < \infty$. Then, for every $\nu > 0$,*

$$\mathbb{P}\left(\max_{\widehat{f} \in \mathcal{F}} |\widehat{L}^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f) + K_f - L_{\mathbf{x}_o}(\widehat{f}, f)| \geq K_\epsilon \epsilon^{\beta} + \nu\right) \leq 2me^{-\frac{B'\nu^2}{2(M^2+2M)^2}}.$$

Finally, the next corollary shows that the procedure we propose in this section, with high probability, picks an estimate of the posterior density that has a true loss that is close to the true loss of the best method in $\mathcal{F}$.

Corollary 3. *Let $\widehat{f}^* := \arg\min_{\widehat{f} \in \mathcal{F}} \widehat{L}^{\epsilon}_{\mathbf{x}_o}(\widehat{f}, f)$ be the best estimator in $\mathcal{F}$ according to the estimated surrogate loss, and let $f^* = \arg\min_{\widehat{f} \in \mathcal{F}} L_{\mathbf{x}_o}(\widehat{f}, f)$ be the best estimator in $\mathcal{F}$ according to the true loss. Then, under the assumptions from Theorem 15, with probability at least $1 - 2me^{-\frac{B'\nu^2}{2(M^2+2M)^2}}$, $L_{\mathbf{x}_o}(\widehat{f}^*, f) \leq L_{\mathbf{x}_o}(f^*, f) + 2(K_H\epsilon^{\beta} + \nu)$.*

9.1.3 Summary Statistics Selection

In a conventional ABC framework, the variable $\mathbf{x}$ is not composed of the raw data; instead, it comprises a multitude of summary statistics. The standard ABC approach encounters challenges when employing all these statistics simultaneously, particularly when certain statistics contribute minimal information regarding the model parameters (Blum, 2010).

We can use FlexCode (Section 3.4) as a method for either (i) directly estimating $f(\theta|\mathbf{x}_o)$ in scenarios involving an extensive array of summary statistics, or (ii)

[3] Such assumptions hold if the $\widehat{f}_i$'s are obtained via FlexCode with bounded basis functions (e.g., Fourier basis) or a kernel density estimator on the ABC samples.

assigning an importance measure to each summary statistic. This measure can subsequently be employed for variable selection in Approximate Bayesian Computation (ABC) and related procedures.

Two versions of FlexCode prove particularly advantageous for these purposes: FlexCode-SAM[4] and FlexCode-RF[5]. These versions of FlexCode possess the ability to dynamically adapt to the number of relevant covariates. Consequently, both FlexCode-SAM and FlexCode-RF can autonomously identify the pertinent summary statistics crucial for estimating the posterior distribution of θ. This eliminates the need for explicit pre-estimation summary statistic selection; instead, FlexCode-SAM or FlexCode-RF automatically discard irrelevant covariates.

In a broader context, FlexCode offers a tool for calculating an importance measure for summary statistics, applicable not only within the FlexCode framework but also in other procedures (recall Section 3.4.1). In the context of LFI, one can infer the relevance of the j-th summary statistic *in posterior estimation* from its relevance in estimating the I first *regression* functions in FlexCode – even if we do not use FlexCode for estimating the posterior using the technique discussed in Equation 3.9. We can use these values to select variables for estimating $f(\theta|\mathbf{x}_o)$ via other ABC methods. For example, one approach is to choose all summary statistics such that $u_j > t$, where the threshold value t is defined by the user.

9.2 Experiments

9.2.1 Examples with Known Posteriors

We begin our analysis with examples featuring well-known and computable posterior distributions:

1. **Mean of Gaussian with Known Variance:** $X_1, \ldots, X_{20}|\mu \overset{iid}{\sim} \mathcal{N}(\mu, 1)$, $\mu \sim \mathcal{N}(0, \sigma_0^2)$. Experiments are repeated for σ_0 in a grid of ten values from 0.5 to 100.

2. **Precision of Gaussian with Unknown Precision:** $X_1, \ldots, X_{20}|(\mu, \tau) \overset{iid}{\sim} \mathcal{N}(\mu, 1/\tau)$, $(\mu, \tau) \sim$ Normal-Gamma$(0, 1, \alpha_0, \beta_0)$. Experiments are conducted

[4]FlexCode with expansion coefficients estimated through Sparse Additive Models (Ravikumar et al., 2009)

[5]FlexCode with expansion coefficients estimated via Random Forests

with α_0 and β_0 to ensure $\mathbb{E}[\tau] = 1$ and $\sqrt{\mathbb{V}[\tau]}$ spans a grid of ten values from 0.1 to 5.

In these examples, observed data $\mathbf{x}_o$ are drawn from a $N(0,1)$ distribution. We run each experiment 200 times, that is, with 200 different values of $\mathbf{x}_o$. The training set $\mathcal{T}$, which is used to build conditional density estimators, is constructed according to Algorithm 1 with $B = 10,000$ and a tolerance level ϵ that corresponds to an acceptance rate of 1%. For the distance function $d(\mathbf{x}, \mathbf{x}_o)$, we choose the Euclidean distance between minimal sufficient statistics normalized to have mean zero and variance 1; these statistics are $\bar{x}$ for scenario 1 and $(\bar{x}, s)$ for scenario 2.

We conduct a comparative analysis of the following methods:

- ABC: Utilizing the rejection ABC method with minimal sufficient statistics. This involves applying a kernel density estimator to the θ coordinate of $\mathcal{T}$, with the bandwidth determined through cross-validation.
- FlexCode_Raw-NN* Employing the FlexCode estimator with Nearest Neighbors regression.
- FlexCode_Raw-Series: Utilizing the FlexCode estimator with Spectral Series regression as described by (Lee and Izbicki, 2016).
- FlexCode_Raw-RF: Applying the FlexCode estimator with Random Forest regression.

The three FlexCode estimators (indicated by "Raw") are directly applied to the sorted values of the original covariates $X_{(1)}, \ldots, X_{(20)}$, without using minimal sufficient statistics or other summary statistics. To evaluate the performance of each method, we calculate the true loss $L_{\mathbf{x}_o}$ for each $\mathbf{x}_o$. Additionally, we estimate the surrogate loss $L^{\epsilon}_{\mathbf{x}_o}$ using Equation 9.3 with an additional sample of size $B' = 10,000$ from Algorithm 1.

9.2.1.1 CDE and Method Selection

Figures 9.1 and 9.2,(left) show the effectiveness of methods in estimating posterior density for Settings 1-2. Panels (a) depict the proportion of instances where each method yields optimal results based on true loss from Equation 9.1. Generally, FlexCode-based methods outperform ABC, especially with larger prior variances.

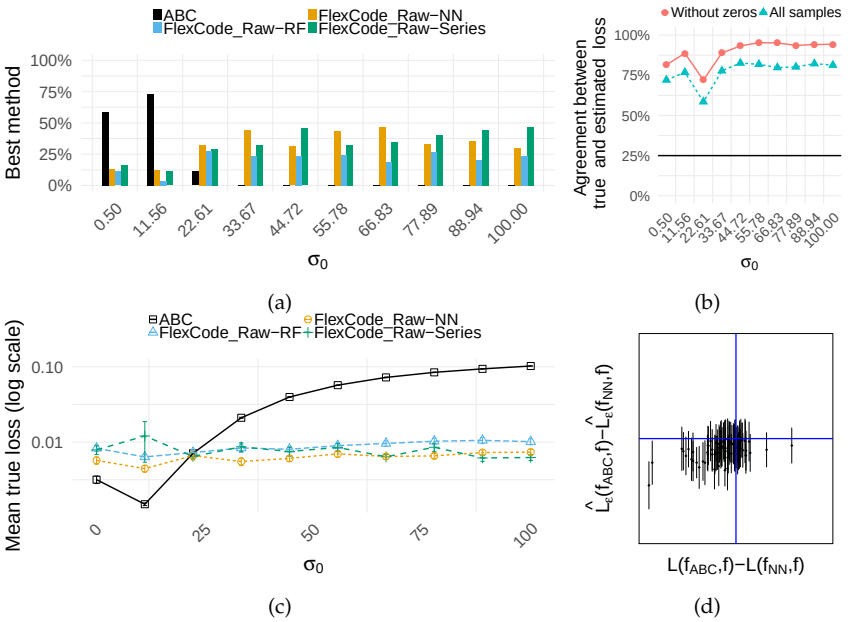


Figure 9.1: CDE and method selection for scenario 1. *Left:* Rejection ABC excels with small σ_0, while NN and Series FlexCode perform better for moderate and large σ_0. *Right:* Surrogate loss estimation guides method choice; horizontal line in (b) represents random selection.

FlexCode with Nearest Neighbors regression excels in this context, while FlexCode with expansion coefficients via Spectral Series regression is also competitive, as supported by Panels (c) displaying average true loss with standard errors.

Figures 9.1 and 9.2, right, summarizes our method selection algorithm's performance. Panels (b) depict the agreement between the true loss $L_{\mathbf{x}_o}(\widehat{f}, f)$ and the estimated loss (Equation 9.3) in method selection. Two algorithm variations are presented: one including all data (triangles) and another excluding cases with inconclusive confidence intervals (circles). The baseline represents random method selection. The plots consistently show similar conclusions between true and estimated losses. Additional scatterplots (Panels d) illustrate differences between true and estimated losses for specific settings, confirming agreement between surrogate and true losses in identifying the best method for posterior density estimation.

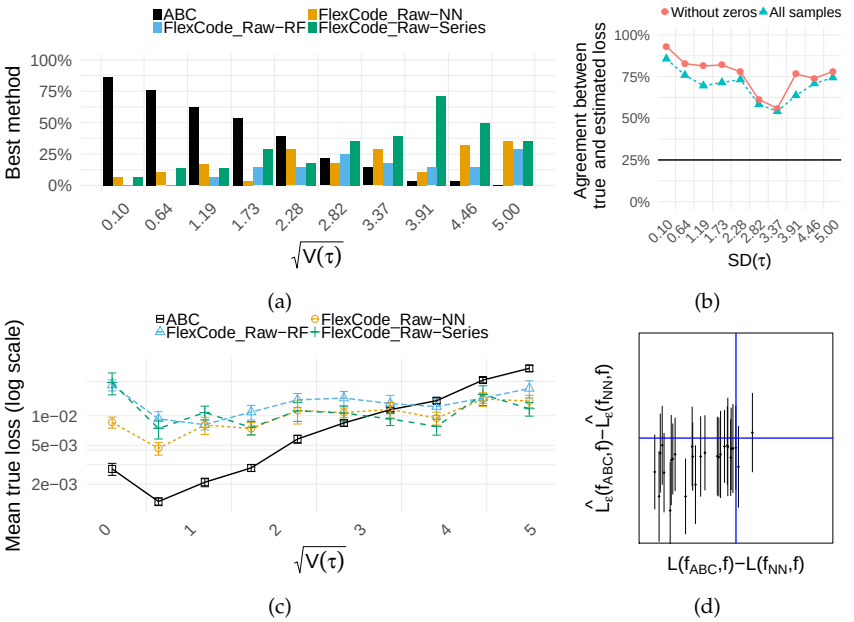


Figure 9.2: CDE and method selection for scenario 2. *Left:* Rejection ABC excels with small σ_0, while NN and Series FlexCode perform better for moderate and large σ_0. *Right:* Surrogate loss estimation guides method choice; horizontal line in (b) represents random selection.

9.2.1.2 Summary Statistic Selection

In this section, we assess the performance of FlexCode-RF for summary statistics selection (Sec. 9.1.3). For this purpose, the following summary statistics were used:

1. Mean: average of the data points; $\frac{1}{n}\sum_{i=1}^{n} x_i$
2. Median: median of the data points; median$\{x_i\}_{i=1,\ldots,n}$
3. Mean 1: average of the first half of the data points; $\frac{1}{n/2}\sum_{i=1}^{n/2} x_i$
4. Mean 2: average of the second half of the data points; $\frac{n/2+1}{n}\sum_{i=n/2+1}^{n} x_i$
5. SD: standard deviation of the data points; $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2}$
6. IQR: interquartile range of the data points, that is:

$$\text{quantile}_{75\%}\{x_i\}_{i=1,\ldots,n} - \text{quantile}_{25\%}\{x_i\}_{i=1,\ldots,n}$$

7. Quartile 1: first quantile of the data points; $\text{quantile}_{25\%}\{x_i\}_{i=1,\ldots,n}$
8. Independent random variables $\sim N(0,1)$ with a range of 43.

Figures 9.3 and 9.4 present results from applying FlexCode-RF and ABC across different scenarios using various summary statistics. Panels (a) depict the true loss as the number of statistics increases. FlexCode-RF remains robust to the inclusion of irrelevant statistics, while standard ABC's performance deteriorates as noise or nuisance statistics are added. Panels (b) display the average importance of each statistic, as defined by Equation 3.9, where $u_{i,j}$ represents the mean decrease in the Gini index. FlexCode-RF assigns high importance to sufficient statistics or those strongly correlated with them. For instance, in Figure 9.3, location measures receive higher importance, whereas in Figure 9.4, dispersion measures are more significant. FlexCode-RF consistently assigns zero importance to random noise statistics, indicating that the summary statistic selection method effectively identifies relevant statistics for accurate posterior estimation, $f(\theta|\mathbf{x}_o)$.

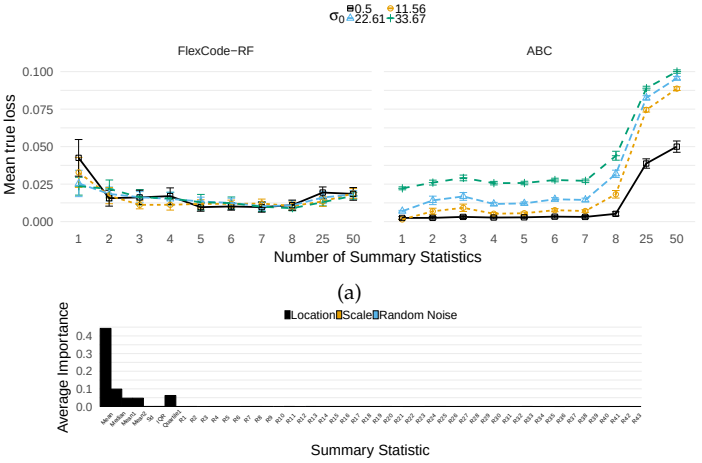


Figure 9.3: Summary statistics selection in scenario 1 (mean of a Gaussian with known variance). ABC sensitivity to noise is evident in (a) (entries 8-51), with FlexCode-RF showing robustness. FlexCode-RF identifies key location statistics in (b) (entries 1-5).

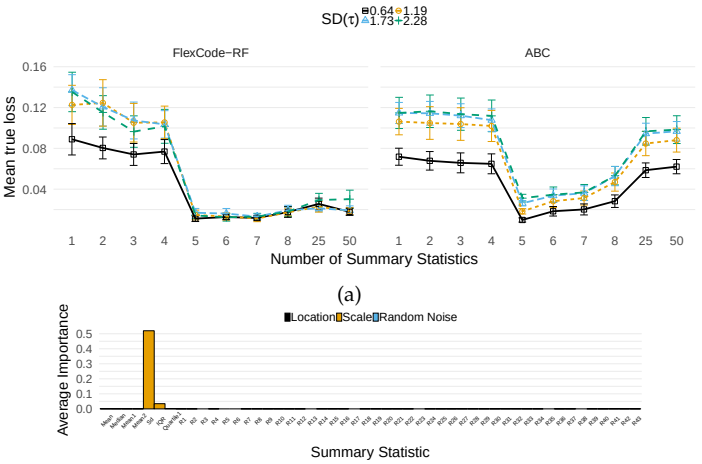


Figure 9.4: Summary statistics selection in scenario 2 (precision of a Gaussian with unknown precision). ABC sensitivity to noise is evident in (a) (entries 8-51), with FlexCode-RF showing robustness. FlexCode-RF identifies key location statistics in (b) (entries 1-5) and emphasizes dispersion statistics (e.g., entry 5).

9.2.2 Application: Estimating a Galaxy's Dark Matter Density Profile

We extend our analysis to more complex simulations under the ΛCDM framework, concentrating on the dark matter distribution in galaxies using the Navarro-Frenk-White (NFW) profile (Navarro, 1996). Our goal is to constrain the NFW parameters for the Sculptor dwarf spheroidal galaxy, starting with the critical energy E_c (Strigari et al. 2017, Equation 15), while keeping other parameters fixed at standard values.

The observed data $\mathbf{x}_0$ include velocities and positions of 200 stars, simulated based on the NFW model (Liu et al., n.d.). We employ ABC, defining the distance metric as the L^2 norm between kernel density estimates of the joint distribution of velocity and position. This distance metric is applied consistently across FlexCode-NN and FlexCode-Series. For functional data, we use FlexCode-Functional, which employs functional kernel regression for coefficient estimation (Ferraty and Vieu, 2006).

To evaluate the performance of the CDE methods, we generate 1000 test observations, with each ABC sample containing 1000 accepted observations (acceptance rate of 0.1). The prior for E_c is uniformly distributed over $U(0.01, 1.0)$.

Figure 9.5 illustrates the true losses for various methods. The left panel shows that FlexCode estimators outperform ABC for certain scenarios with low true E_c. The right panel, which compares estimated surrogate losses, supports similar findings, demonstrating that the surrogate loss serves as a practical substitute for the unavailable true loss.

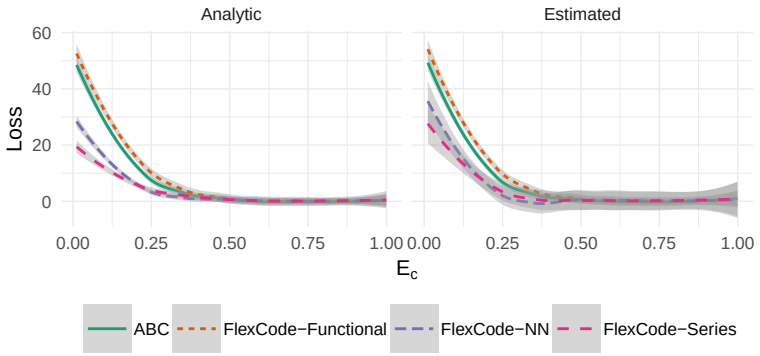


Figure 9.5: *Left:* True loss in various parameter regions. *Right:* The estimated surrogate loss (here shifted with a constant for easier comparison) can be used to identify improvements upon the ABC estimates.

9.3 Likelihood Free Frequentist Inference (LF2I)

The goal of Likelihood Free Frequentist Inference (LF2I; Dalmasso et al. 2020a; Dalmasso et al. 2021) is to construct a set $R(\mathbf{X})$ such that

$$\mathbb{P}_{\mathbf{X}|\theta}\left(\theta \in R(\mathbf{X})\right) = 1 - \alpha, \tag{9.4}$$

where $\alpha \in (0,1)$, regardless of the true value of the unknown parameter $\theta \in \Theta$. That is, $R(\mathbf{X})$ needs to be a frequentist set – it needs to control conditional coverage. In this section, we describe how LF2I achieves this. The full process is illustrated in Figure 9.6.

9.3.1 Confidence Sets via Neyman Inversion

The starting point of LF2I is Neyman confidence sets, which are based on Neyman's inversion of hypothesis test (Neyman, 1937b). Specifically, Neyman's confidence sets have the shape

$$R(\mathbf{X}) := \{\theta \in \Theta \,|\, \tau(\mathbf{X};\theta) \geq C_\theta\}, \tag{9.5}$$

where $\tau(\mathbf{X};\theta)$ measures how plausible it is that $\mathbf{X}$ was generated from θ (formally, it is a test statistic for null hypothesis that the data was generated from θ; see Section

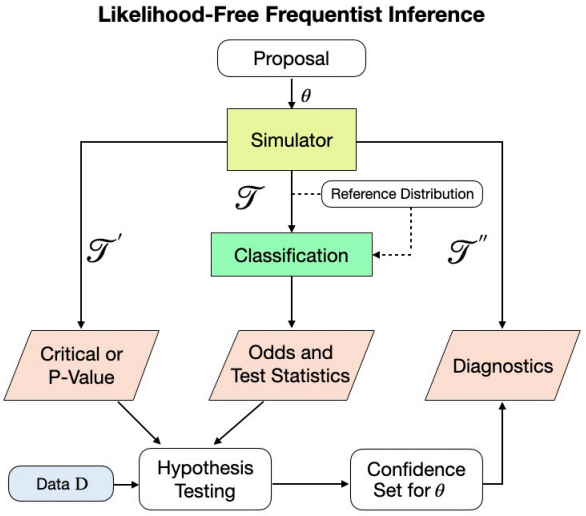


Figure 9.6: The three-branch modular framework for likelihood-free frequentist inference (LF2I).

9.3.2), and C_θ is chosen so that conditional coverage holds (Equation 9.4). From a hypothesis test perspective, C_θ is the rejection cutoff that leads to type I error control. In other words, $R(\mathbf{x})$ contains plausible values for θ (according to the metric τ) given data $\mathbf{x}$.

The challenges of using Neyman's confidence sets in an LFI context are that:

- Most standard techniques for constructing τ require knowledge of an analytical expression for the likelihood function. This is the case of the Likelihood Ratio Test, for instance.
- The standard derivation of C_θ either uses asymptotic derivations that only work if the sample size is large and for certain test statistics, or uses Monte Carlo simulations that are each fixed θ, which is too expensive for larger dimensional spaces.

In what follows, we discuss how to address these challenges using some of the Uncertainty Quantification techniques we have introduced before. We also discuss how to check whether the confidence regions given by LF2I have the correct conditional coverage. Code to implement LF2I in Python can be found at https://github.com/lee-group-cmu/lf2i.

9.3.2 Choosing a Test Statistic

Any test statistic τ can be used in LF2I as long as it can be computed. In this section, we show some examples of statistics that can be used. Some of these statistics are influenced by Bayesian methods, although we apply them in a frequentist context.

9.3.2.1 Likelihood-Based Statistics

Many traditional test statistics rely on the likelihood function $f(\mathbf{x}|\theta)$. Here, we will discuss two of them, and then discussed how they can be used in a LFI setting.

The first one is the likelihood ratio (LR) statistic, which tests $H_0 : \theta = \theta_0$ using

$$\lambda(\mathbf{x};\theta_0) = \frac{f(\mathbf{x}|\theta_0)}{\sup_{\theta\in\Theta} f(\mathbf{x}|\theta)}. \tag{9.6}$$

The second one is the Bayes Factor (BF), which essentially changes the maximization in the LR statistic by an integration with respect to a prior distribution $f(\theta)$:

$$\beta(\mathbf{x};\theta_0) \equiv \frac{f(\mathbf{x}|H_0)}{f(\mathbf{x}|H_1)} = \frac{f(\mathbf{x}|\theta_0)}{\int_\Theta f(\mathbf{x}|\theta)df(\theta)}. \tag{9.7}$$

In an LFI setting, $f(\mathbf{x}|\theta)$ cannot be evaluated, and therefore these statistics cannot be computed. Thus, they need to be estimated. In order to do that, consider a proposal distribution $\pi(\theta)$ over the parameter space. We can use the forward simulator to generate $(\theta_1, \mathbf{X}_1), \ldots, (\theta_B, \mathbf{X}_B)$, where each $(\theta, \mathbf{X}) \sim \pi(\theta)f(\mathbf{x}|\theta)$. By construction, the conditional density of $\mathbf{X}|\theta$ is $f(\mathbf{x}|\theta)$, and thus this quantity can be estimated using the conditional density estimators discussed in Chapter 3.

Dalmasso et al. (2020a) and Dalmasso et al. (2021) in particular estimate the likelihood function using the ratio trick (Section 3.7). Notice that the notation of that section corresponds to the choice $y := \mathbf{x}$ and $\mathbf{x} =: \theta$. Dalmasso et al. (2021) takes $g(\mathbf{x}|\theta)$ in Equation 3.13 to be $g(\mathbf{x})$, that is, a distribution that does not depend on θ. In this way,

$$\mathbb{O}(\mathbf{x};\theta) = \frac{f(\mathbf{x}|\theta)}{g(\mathbf{x})},$$

that is, $\mathbb{O}(\mathbf{x};\theta)$ is, up to $g(\mathbf{x})$, the likelihood function. Now, because $g(\mathbf{x})$ is constant in θ, this implies that the $\mathbb{O}(\mathbf{x};\theta)$ is a version of the likelihood function, and therefore can be used in the LR statistic or the BF without the needs of applying the correction $g(\mathbf{x}|\theta)$ from Equation 3.14 (Izbicki et al., 2014). In other words, the LR statistic can be

estimated by

$$\widehat{\lambda}(\mathbf{x};\theta_0) = \frac{\widehat{\mathbb{O}}(\mathbf{x};\theta_0)}{\sup_{\theta\in\Theta}\widehat{\mathbb{O}}(\mathbf{x};\theta)},$$

and the Bayes Factor by

$$\widehat{\beta}(\mathbf{x};\theta_0) = \frac{\widehat{\mathbb{O}}(\mathbf{x};\theta_0)}{\int_{\theta\in\Theta}\widehat{\mathbb{O}}(\mathbf{x};\theta)d\pi(\theta)},$$

where

$$\widehat{\mathbb{O}}(\mathbf{x};\theta) := \frac{\widehat{\mathbb{P}}(Y=1|\mathbf{x},\theta)}{\widehat{\mathbb{P}}(Y=0|\mathbf{x},\theta)}.$$

Moreover, if $g(\mathbf{x}) := \int f(\mathbf{x}|\theta)d\pi(\theta) = f(\mathbf{x})$ (that is, if g is chosen to be the marginal distribution of $\mathbf{x}$ with respect to the prior $\pi(\theta)$), the odds becomes the Bayes factor itself, and thus, in this special case,

$$\widehat{\beta}(\mathbf{x};\theta_0) = \widehat{\mathbb{O}}(\mathbf{x};\theta_0).$$

See Cranmer et al. (2020), Heinrich (2022), and Thomas et al. (2022) for variations of these statistics and how to estimate them based on data from the simulator.

9.3.2.2 Waldo

An alternative test statistic that can be used to construct confidence sets is the Waldo statistic (Masserano et al., 2023), which is given by

$$\omega(\mathbf{x};\theta_0) = (\mathbb{E}[\theta|\mathbf{x}] - \theta_0)^t \mathbb{V}[\theta|\mathbf{x}]^{-1}(\mathbb{E}[\theta|\mathbf{x}] - \theta_0), \tag{9.8}$$

where $\mathbb{E}[\theta|\mathbf{x}]$ and $\mathbb{V}[\theta|\mathbf{x}]$ represent the posterior mean and variance of θ given data $\mathbf{x}$. Waldo is a Bayesian version of the Wald test statistic (Wald, 1943).

Similar to the likelihood-based statistics, Waldo cannot be directly computed in an LF2I setting. However, it can be approximated by

$$\widehat{\omega}(\mathbf{x};\theta_0) = (\widehat{\mathbb{E}}[\theta|\mathbf{x}] - \theta_0)^t \widehat{\mathbb{V}}[\theta|\mathbf{x}]^{-1}(\widehat{\mathbb{E}}[\theta|\mathbf{x}] - \theta_0), \tag{9.9}$$

where $\widehat{\mathbb{E}}[\theta|\mathbf{x}]$ and $\widehat{\mathbb{V}}[\theta|\mathbf{x}]$ are estimates of $\mathbb{E}[\theta|\mathbf{x}]$ and $\mathbb{V}[\theta|\mathbf{x}]$ obtained using the data from the simulator, $(\theta_1, \mathbf{X}_1), \ldots, (\theta_B, \mathbf{X}_B) \sim \pi(\theta)f(\mathbf{x}|\theta)$. These can be obtained in two ways:

- Using an estimate of the posterior distribution, $\widehat{f}(\theta|\mathbf{x})$, which can be obtained using e.g. the ABC-CDE approach discussed in Section 9.1. For instance, if $\theta \in \mathbb{R}$, this corresponds to taking $\mathbb{E}[\theta|\mathbf{x}] := \int \theta \widehat{f}(\theta|\mathbf{x}) d\theta$ and $\mathbb{V}[\theta|\mathbf{x}] := \int \theta^2 \widehat{f}(\theta|\mathbf{x}) d\theta - \mathbb{E}^2[\theta|\mathbf{x}]$.

- Directly estimating both $\mathbb{E}[\theta|\mathbf{x}]$ and $\mathbb{V}[\theta|\mathbf{x}]$ via regression/supervised learning. Concretely, $\mathbb{E}[\theta|\mathbf{x}]$ is the regression of θ on $\mathbf{x}$, and $\mathbb{V}[\theta|\mathbf{x}] = \mathbb{E}[\theta^2|\mathbf{x}] - \mathbb{E}^2[\theta|\mathbf{x}]$ is the difference between the regression of θ^2 on $\mathbf{x}$ and the first regression (squared) that was already estimated.

Masserano et al. (2023) shows that the Waldo statistic leads to larger power/tighter (frequentist) confidence sets than the Wald statistic in regions where $\pi(\theta)$ places large mass.

9.3.3 Calibrating the Cutoffs

After selecting the test statistic, denoted as $\tau(\mathbf{X};\theta)$, the next step is to determine C_θ for constructing the confidence set described in Equation 9.5. To ensure that this set achieves a confidence level of $1-\alpha$, it is necessary to define C_θ as follows:

$$C_\theta = F^{-1}_{\tau(\mathbf{X};\theta)|\theta}(\alpha),$$

where $F_{\tau(\mathbf{X};\theta)|\theta}$ represents the cumulative distribution function of the random variable $\tau(\mathbf{X};\theta)$ conditioned on the parameter θ. In other words, C_θ is the α conditional quantile of $\tau(\mathbf{X};\theta)$ on θ. Again, in an LFI setting we do not know this distribution, but we can estimate it. There are a few ways of doing it:

- Use the simulator to create a Monte Carlo sample $\mathbf{X}_1, \ldots, \mathbf{X}_B$, all sampled at the same θ. Then, compute the α quantile of the values $\{\tau(\mathbf{X}_1;\theta), \ldots, \tau(\mathbf{X}_B;\theta)\}$. Notice however that, to compute confidence sets, it is necessary to obtain C_θ for a finely sampled grid of θ values. This approach can therefore be expensive, especially when dealing with a parameter space of large dimensions.

- Use quantile regression to regress $\tau(\mathbf{X};\theta)$ on θ. That is, we (i) use the simulator to obtain a training set $(\theta_1, \mathbf{X}_1), \ldots, (\theta_B, \mathbf{X}_B)$, (ii) compute the transformed data $(\theta_1, \tau(\mathbf{X}_1;\theta_1)), \ldots, (\theta_B, \tau(\mathbf{X}_B;\theta_B))$, and (iii) use e.g. one of the techniques of Section 3.10 to α-quantile regress $\tau(\mathbf{X};\theta)$ on θ. This is the approach proposed by Dalmasso et al. (2020a) and Dalmasso et al. (2021). Alternatively, one may

estimate the full conditional distribution of $\tau(\mathbf{X};\theta)$ on θ using for instance the techniques from Chapter 3, and then derive the estimated α-quantile.

9.3.4 Evaluating Coverage

Once the test statistic is selected and cutoffs are determined, we construct a confidence set denoted as $R(\mathbf{X})$, defined as follows:

$$R(\mathbf{X}) := \left\{ \theta \in \Theta \,\middle|\, \tau(\mathbf{X};\theta) \geq \widehat{C}_\theta \right\}.$$

Notice that $R(\mathbf{X})$ may not be a valid confidence set for every $\theta \in \Theta$ due to the fact that $\widehat{C}_\theta$ is an estimate of C_θ. In other words, the constructed confidence set may not achieve the correct coverage. To assess the adequacy of such estimates, LF2I examines their quality by estimating the coverage probability as a function of θ. This evaluation helps ensure that the confidence set provides reliable coverage across the parameter space Θ.

In order to check coverage, LF2I uses the fact that

$$\mathbb{P}_{\mathbf{X}|\theta}\left(\theta \in R(\mathbf{X})\right) = \mathbb{E}\left[Z|\theta\right],$$

where $Z := \mathbb{I}(\theta \in R(\mathbf{X}))$ and $(\theta, \mathbf{X})$ is drawn from the simulator. In other words, the coverage probability is the regression of Z on θ. Thus, LF2I (i) uses the simulator to obtain a training set $(\theta_1, \mathbf{X}_1), \ldots, (\theta_B, \mathbf{X}_B)$, (ii) compute the transformed data $(\theta_1, Z_1)), \ldots, (\theta_B, Z_B)$, where $Z_i = \mathbb{I}(\theta_i \in R(\mathbf{X}_i))$ and (iii) use e.g. one of the techniques of Chapter 2 to regress Z on θ (equivalently, fit a probabilistic classifier). By checking whether such estimates are far from $1-\alpha$ for all θ's, one can determine the adequacy of coverage for $R(\mathbf{X})$ and pinpoint regions that require improvement for specific θ values. Subsequently, adjustments can be made to reestimate C_θ in those identified areas.

9.3.5 Nuisance Parameters

The parameters θ can often be decomposed into $\theta = (\mu, \nu)$, where $\mu \in M$ are parameters of interest and $\nu \in N$ are nuisance. In this case, one only cares about constructing confidence sets for μ.

In this setting, Neyman confidence sets then have the shape

$$R(\mathbf{X}) := \{\mu \in M \,|\, \tau(\mathbf{X}; \mu) \geq C_\mu\},$$

where $\tau(\mathbf{X}; \mu)$ measures how plausible it is that $\mathbf{X}$ was generated from μ. We now discuss how to design both $\tau(\mathbf{X}; \mu)$ and C_μ.

One way to create a test statistic $\tau(\mathbf{X}; \mu)$ is to use the techniques from Section 9.3.2 to design a statistic that tests the hypothesis

$$H_0 : \mu = \mu_0 \text{ and } \nu = \nu_0.$$

Denote this statistic by $\tau(\mathbf{X}; (\mu_0, \nu_0))$. This statistic can then be integrated over ν to produce $\tau(\mathbf{X}; \mu)$ in a Bayesian context: $\tau(\mathbf{X}; \mu) = \int \tau(\mathbf{X}; (\mu, \nu)) dP(\nu)$. Alternatively, in a frequentist vain, one may take its maximum value: $\tau(\mathbf{X}; \mu) = \sup_{\nu \in N} \tau(\mathbf{X}; (\mu, \nu))$. Furthermore, certain statistical models are inherently equipped to manage nuisance parameters without additional adjustments. This is the case of Waldo (Section 9.3.2.2):

$$\omega(\mathbf{x}; \mu_0) = (\mathbb{E}[\mu|\mathbf{x}] - \mu_0)^t \mathbb{V}[\mu|\mathbf{x}]^{-1} (\mathbb{E}[\mu|\mathbf{x}] - \mu_0).$$

Once the test statistic is defined, the cutoffs C_μ need to be chosen. One way of doing that in order to guarantee that $R(\mathbf{X})$ has coverage $1 - \alpha$ is to set C_μ to be

$$C_\mu = \inf_{\nu \in N} F^{-1}_{\tau(\mathbf{X};\mu)|\mu,\nu}(\alpha),$$

where $F_{\tau(\mathbf{X};\mu)|\mu,\nu}$ represents the cumulative distribution function of the random variable $\tau(\mathbf{X}; \mu)$ conditioned on all parameters (μ, ν). Of course, such cumulative distribution must be estimated, which can be done in the same way as described in Section 9.3.3.

Alternately, one may also integrate $F^{-1}_{\tau(\mathbf{X};\mu)|\mu,\nu}(\alpha)$ with respect to ν, although this procedure does not have coverage guarantees (Dalmasso et al., 2021). See also Masserano et al. (2024) for a different way to handle nuisance parameters that also controls coverage, and Dalmasso et al. (2021) for alternative approximations such as profiling.

9.3.6 Example: Muon energy estimation

In this section, we aim to estimate muon energy using a high-granularity calorimeter within a particle collider experiment. Each data point is represented as a 3D image, $\mathbf{x}$, with a dimensionality of approximately 50,000. The sample consists of $n = 1$ with a total of 886,716 3D inputs, each corresponding to a scalar muon energy θ. These data are generated through simulations using GEANT4 (Agostinelli et al., 2003), a simulator known for accurately modeling Standard Model dynamics. The dataset can be accessed at Kieseler et al., 2021.

The primary objective is to determine whether a high-granularity calorimeter provides tighter constraints on muon energy compared to detectors that measure only total energy. We investigate three levels of energy measurement: (i) a 1D input representing the sum of calorimeter cells for muons with energy $E > 0.1$ GeV; (ii) 28 custom features extracted from spatial and energy information of calorimeter cells (see Kieseler et al., 2022); and (iii) the full calorimeter data, $\mathbf{x} \in \mathbb{R}^{51,200}$. For each data point, LF2I confidence sets are constructed using the estimated Bayes Factor (Equation 9.7). A convolutional neural network classifier, as described in Kieseler et al., 2022, is used to estimate the odds function for the full calorimeter data, while critical values are obtained via quantile gradient boosted trees. For the 1D and 28D datasets, a gradient boosting classifier is employed to learn the odds function. Around 83% of the data ($B = 738,930$) is used for training, with 14% allocated to estimate critical values ($B' = 123,155$). For comparison, SMC-ABC (Sisson et al., 2007), another likelihood-free inference (LFI) algorithm, is applied to all simulated datasets, utilizing the same total of $B + B' = 862,085$ samples. The remaining data ($B'' = 24,631$) is reserved for validation and diagnostics.

Figure 9.7 (center) shows that LF2I with the Bayes Factor test statistic achieves nominal coverage of 68.3% across all datasets. In contrast, SMC-ABC produces overly conservative credible intervals, leading to over-coverage. In terms of interval length (constraining power), Figure 9.7 (right) indicates that SMC-ABC intervals are significantly wider than LF2I confidence sets for both the 1D and 28D datasets. Due to computational limitations, running SMC-ABC on the full $51,200$-dimensional calorimeter data was not feasible. Notably, the information in the data directly influences the size of LF2I confidence sets: the full calorimeter results in noticeably smaller intervals and higher constraining power.

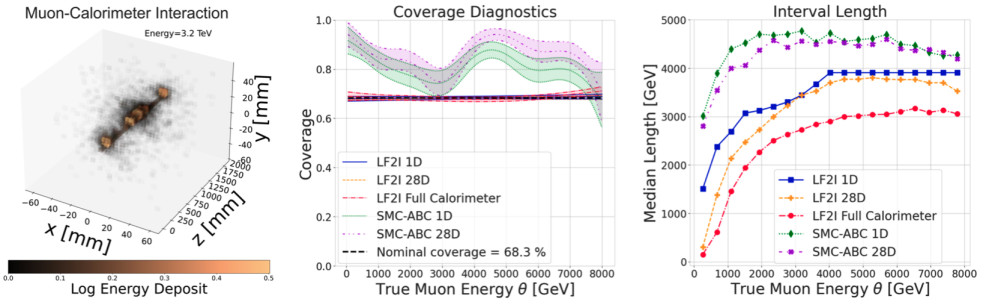


Figure 9.7: LF2I outperforms SMC-ABC with nominal coverage and tighter intervals. **Left:** Example muon data at $\theta \approx 3.2$ TeV in a $32 \times 32 \times 50$ cell calorimeter. **Center:** LF2I (blue, orange, red) maintains 68.3% coverage, while SMC-ABC (green, purple) consistently over-covers. **Right:** Median interval lengths. SMC-ABC, despite computational intensity, lacks constraining power. Full calorimeter data for SMC-ABC is omitted due to computational constraints.

9.4 Summary

This chapter explored Likelihood-Free Inference in statistical models where the likelihood function is intractable. We focused on two methodologies: Approximate Bayesian Computation via Conditional Density Estimation (ABC-CDE) and Likelihood-Free Frequentist Inference (LF2I).

ABC-CDE was presented as a reliable approach for estimating posterior distributions using conditional density estimation, providing an efficient solution in complex models from a Bayesian perspective. LF2I was introduced as a framework for constructing frequentist confidence sets that ensure proper coverage, regardless of the true parameter values. Its modular design allows for the integration of various test statistics and calibration techniques, making it adaptable to a wide range of applications.

Chapter 10

Optimizing Construction Schedules: Mitigating Weather-Related Delays

Frustrated Workers. Designed with the help of IA.

.

> The best way to predict the future is to create it.
>
> Peter Drucker

Construction delays can stem from various factors, including the incompetence of involved parties, material and equipment shortages, and adverse weather conditions (Durdyev et al., 2020; Sanni-Anibire and Egbu, 2022; Sepasgozar et al., 2019). These delays negatively impact both owners and contractors, leading to contractual penalties and reduced resource productivity. Among these factors, weather events stand out as a significant source of uncertainty due to their direct effect on productivity (Assaf and Al-Hejji, 2006; Schuldt et al., 2021). Modeling delays caused by weather is particularly challenging, as these events vary not only in time and location but also in the specific weather conditions required for different construction tasks. Consequently, effective project management must accurately account for these conditions for each task to minimize potential negative impacts on productivity and deadlines.

This chapter illustrates how the tools presented in previous chapters can estimate the uncertainty in a project's completion time. Specifically, it aims to estimate the probability distribution of Z, the project's duration in days, relying exclusively on historical meteorological data. We denote such distribution as $f(z|i, y)$, where j is the day of year and y the year of project implementation.

The construction time will depend on a construction schedule. This schedule outlines the tasks needed to produce the deliverables, including their sequence, durations, and resource requirements (Edition, 2008). Figure 10.1 provides an example of a fictitious schedule with tasks carried out sequentially, without accounting for weather-related delays. We assume this schedule is known.

We also assume we know the necessary climatic conditions for each task must be determined. Table 10.1, adapted from Ballesteros-Pérez et al. (2017), shows the climatic thresholds that determine when specific tasks cannot be performed. For example, "earthworks" cannot be done if the average daily temperature is below 0ºC. In this study, it is assumed that climatic limits are assumed for the performance of tasks and that their impact does not extend to subsequent days.

Having these components, we are now ready to describe our estimator of $f(z|i, y)$.

Table 10.1: Weather limits for type of tasks: earthworks (E), concrete (C), formwork (F), steelwork (S), outdoor painting (O), and pavements (P). “x" denotes conditions under which the task cannot be performed.

Weather variable	E	C	F	S	O	P
Minimum temperature ≤ 0ºC		x				x
Average temperature ≤ 0ºC	x			x	x	
Maximum temperature ≥ 40ºC		x		x		x
Precipitation ≥ 1 mm					x	x
Precipitation ≥ 10 mm	x	x			x	x
Precipitation ≥ 30 mm	x	x		x	x	x
Wind gusts $\geq$30 knots		x	x	x	x	

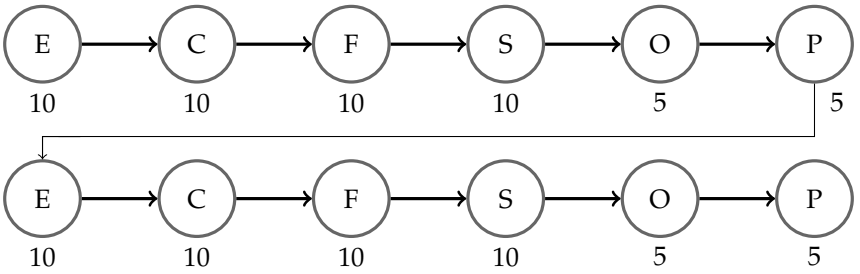


Figure 10.1: Construction schedule represented by a finite network of arcs, where the nodes represent the activities of earthworks (E), concrete (C), formwork (F), steelwork (S), outdoor painting (O), and pavements (P). For each task, a planned duration is associated below the node.

10.1 Estimating the distribution

Consider a project composed of a set of p tasks, represented by $t = \{1, \ldots, p\}$. The schedule of Figure 10.1 contains $p = 12$ tasks. Although the tasks do not overlap in this schedule, the method can be easily applied when they do intersect. The project execution time, Z, is defined as the number of days until all p tasks are completed, taking delays into account. The objective is to model the distribution of Z by considering the execution time of each task, including delays due to climate factors.

To achieve this, let the random variable $X_{i,y}^{(t)}$ indicate whether task t was executed on day i of year y considering climatic factors:

$$X_{i,y}^{(t)} = \begin{cases} 1 & \text{if task } t \text{ was executed on day } i \text{ of year } y, \\ 0 & \text{otherwise.} \end{cases}$$

Our model assumes $X_{i,y}^{(t)} \sim \text{Bernoulli}(\theta_i^{(t)})$, where $\theta_i^{(t)}$ is the probability of completing task t on the i-day of a given year, with $X_{i,y}^{(t)}$'s independent for each fixed t.

Notice that, given a starting day and year, Z is a deterministic function of $X_{i,y}^{(t)}$'s. Therefore, after $\boldsymbol{\theta^{(t)}} = (\theta_1^{(t)}, \ldots, \theta_{365}^{(t)})$ have been estimated, Z can be sampled from this model by using Monte Carlo simulation to sample $X_{i,y}^{(t)}$'s with the estimated parameters. This method easily incorporates task dependencies. For multiple tasks on the same day, each task is checked independently.

The Monte Carlo samples are implicitly defining an estimate $\hat{f}(z|i, y)$, which can then be used to approximate α-quantiles and the mean of Z, as well as to compute prediction sets or any of the other tools developed in previous chapters.

The parameters $\boldsymbol{\theta^{(t)}}$ can estimated using historical weather data in various ways. For instance, we can use a KNN-like estimate:

$$\widehat{\theta}_i^{(t)} = \frac{1}{N} \sum_{y=1}^{Y} \sum_{m \in \mathcal{N}(l)} \sum_{j \in \mathcal{N}(i)} Y_{t,m,j,y}, \tag{10.1}$$

where $\mathcal{N}(l)$ is the set of K_l locations closest to the location where the construction will be performed, $\mathcal{N}(i)$ is the set of K_i days closest to i,[1] and $Y_{t,l,i,y}$ is the indicator

[1]Notice that "day of the year" is a circular variable, so instead of the Euclidean distance we use e.g.

that task t could be performed on location l on the i-th day of year y. For other approaches to estimate this parameter, see Comito et al. (2024).

10.2 Model selection

Suppose we temporally split our dataset into two parts: training and validation. Using the training set, we create I estimates for $f(z|j,y)$: $f_1(z|j,y),\ldots,f_I(z|j,y)$, where j is the day of the year and y the year of project implementation. Each estimate can come from different configurations of tuning parameters of a given model or entirely different models.

Our goal in this section is to select the model that best approximates the true f. To achieve this, we first create variations of the validation datasets, denoted by $(Z_1,I_1,Y_1),\ldots,(Z_m,I_m,Y_m)$, using various temporal splits of the original validation dataset. Specifically, we randomly choose different starting dates (I_k,Y_k). The Z_k values are then obtained by applying the construction schedule to data after the chosen start date, reflecting actual observed weather conditions.

Given a loss function L, we can estimate the risk of an estimate $\widehat{f}$ via

$$\widehat{R}(f,\widehat{f}) = \frac{1}{m}\sum_{k=1}^{m} L(Z_k,\widehat{f}(z|I_k,Y_k)).$$

Next, we discuss choices of loss functions.

10.2.1 CDE loss

Inspired by the L^2 loss for continuous problems (Equation 3.1), we consider the loss function

$$L(p,\widehat{p}) = \sum_z \left(p(z|i,y) - \widehat{p}(z|i,y)\right)^2,$$

where L depends on the true distribution $p(z)$, making direct calculation infeasible. However, L can be decomposed as follows:

$$L(p,\widehat{p}) = \sum_z \widehat{p}^2(z|i,y) - 2\widehat{p}(z|i,y) + K,$$

$d(i,j) = \min\{|i-j|, 365-|i-j|\}$.

where K is a constant independent of $\widehat{p}$. Consequently, K does not need to be calculated for ranking models based on L.

10.2.2 Weighted pinball loss

If we are primarily interested in accurately estimating certain quantiles of $p(z|j,y)$, we can utilize the pinball loss (Section 3.10.1).

Here, we extend the standard pinball loss by combining multiple α values to assess model uncertainty. This is done using a weighting function $w(\alpha)$ to assign weights to each α. The weighted loss $L^w(Z,\widehat{p})$ is defined as:

$$L^w(Z,\widehat{p}) = \int w(\alpha) L_\alpha(Z, Q(\alpha))\, d\alpha$$

where $Q(\alpha)$ is the α-quantile of $\widehat{p}$ and L_α is the α-pinball loss (Equation 3.15).

The weighting function $w(\alpha)$ can be tailored to emphasize different parts of the distribution based on the specific application. For example, in construction projects, accurately estimating the upper tails of $p(z)$ is critical to ensure the observed value lies within the density with high probability. Therefore, it is appropriate to weight the right tail more heavily, such as with $w(\alpha) \propto \alpha^2$, making higher α values more significant. Alternatively, other weights can shift the focus to different quantiles. For instance, the left tail can be emphasized using $w(\alpha) \propto (1-\alpha)^2$, or both tails can be equally weighted with $w(\alpha) \propto (\alpha - 0.5)^2$.

10.3 Example

To illustrate the application of the model, we used daily meteorological data from the UK, following criteria similar to those of Ballesteros-Pérez et al. (2017). The MIDAS dataset (*Met Office MIDAS Open: UK Land Surface Stations Data (1853-current)* 2019) includes daily records of precipitation, temperature, and wind speed from 1985 to 2020. We applied comparable inclusion and exclusion criteria for weather stations, imputing missing values with a 7-day moving average if at least three observations were available. Stations with significant data gaps in the validation and test sets were excluded, resulting in the selection of 68 stations for analysis. The model was trained on years 1985-2012, and the years 2013-2016 were used to select its tuning parameters.

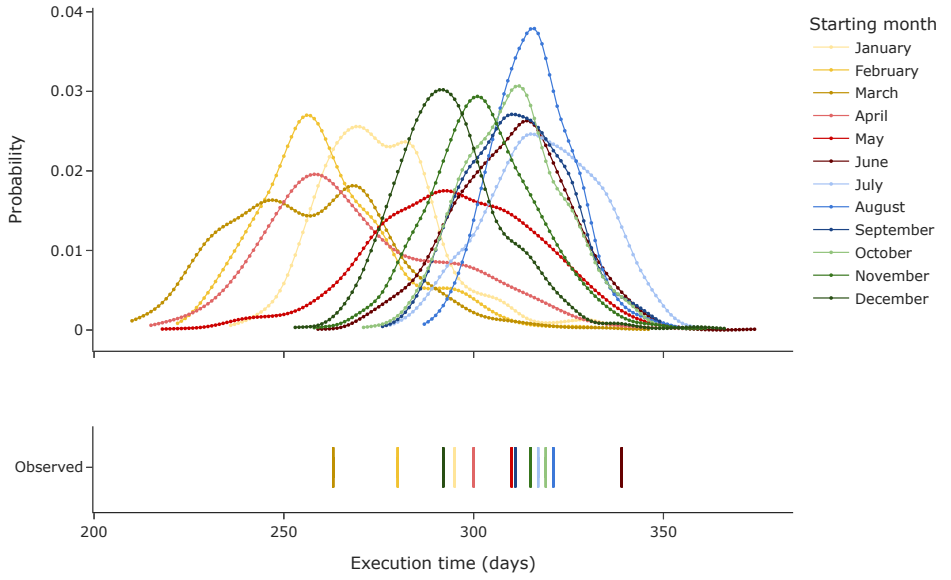


Figure 10.2: Illustrative application of the method using real-world data: the model estimates the distribution of project durations as a function of its starting date, providing valuable insights for optimal project scheduling. The observed lines at the bottom represent actual project durations, demonstrating the high accuracy of the model's estimates.

We used the schedule from Figure 10.1 and the climatic limits from Table 10.1. The project duration was fixed at 160 days, disregarding regional climatic conditions. We estimated the probability of workable days using only days with complete data for all selected variables.

Figure 10.2 demonstrates a byproduct of our estimates $\widehat{f}(z|i,y)$ as a function of its proposing starting day i for a specific location (station 908). The results highlight the model's effectiveness in providing accurate and actionable insights for project scheduling. The comparison between the model's estimates and the actual project durations, represented by the observed lines at the bottom, underscores the method's precision and reliability.

Chapter 11

Closing Thoughts

[UNDER CONSTRUCTION]

11.1 Summary of Key Concepts

This work has explored a variety of methods and tools for Uncertainty Quantification in machine learning, focusing on how different types of uncertainty – aleatoric and epistemic – manifest in both theoretical and applied settings. Aleatoric uncertainty refers to the inherent variability in the data, which cannot be reduced even with an infinite amount of information. It is driven by randomness in the system being modeled and is captured by the conditional distributions of Y given $\mathbf{x}$, which is the $f(y|\mathbf{x})$ in regression and $\mathbb{P}(Y = y|\mathbf{x})$ in classification. In contrast, epistemic uncertainty arises from a lack of knowledge of the true data generating process $f(y|\mathbf{x})$ due to insufficient data. While aleatoric uncertainty can only be reduced by gathering more information (i.e., additional features) for the sample points, epistemic uncertainty, in principle, can be diminished as more data is collected, assuming the model effectively captures the true underlying process.

We saw several approaches to quantify aleatoric uncertainty by estimating conditional densities. In classification problems, probabilistic classifiers automatically estimate $f(y|\mathbf{x})$ as long as a proper loss function is used to train and choose parameters. For regression problems, we saw techniques that include parametric models, kernel-based approaches, and flexible nonparametric models like FlexCode and normalizing flows. After the conditional density has been estimated, we saw that prediction sets can be constructed to account for both aleatoric and epistemic uncertainties. These sets can be derived directly from the conditional density estimates, or from quantities such as quantiles, means, and variances. Notably, conformal methods provide a flexible and increasingly popular framework for constructing prediction sets with guaranteed marginal coverage as long as the data is i.i.d.

Although conformal methods are straightforward to apply and can be adapted to

a wide range of models, they do not guarantee conditional coverage. As a result, they may over- or under-cover certain subgroups of the data. To mitigate this limitation, we explored enhancements to conformal prediction that improve conditional coverage properties, ensuring better reliability across various regions of the feature space.

An alternative approach to modeling both aleatoric and epistemic uncertainty is through Bayesian methods, which inherently combine these uncertainties. Indeed, the posterior predictive distribution $f(y|\mathbf{x}, D)$, where D represents the training data, captures aleatoric uncertainty via the model's conditional distribution and epistemic uncertainty through the posterior over model parameters. This distribution can then be directly used to construct Bayesian prediction sets. These sets can also be further refined with conformal prediction techniques to ensure valid coverage. However, Bayesian models often rely on computationally intensive methods such as Markov Chain Monte Carlo (MCMC), limiting scalability for large or complex datasets.

Neural networks, when combined with techniques such as dropout and deep ensembles, provide an alternative practical tool for managing epistemic uncertainty. Although these methods may initially seem ad-hoc, they can be understood as approximations to Bayesian solutions under specific priors. For instance, Monte Carlo dropout approximates Bayesian inference by sampling from the model's uncertainty during both training and prediction. Deep ensembles, which aggregate predictions from multiple neural networks with different initializations, offer another way to capture model uncertainty. However, it remains uncertain whether these approximations are sufficiently accurate or if the chosen priors are appropriate, especially in complex models. This is crucial since the prior selection directly influences the estimation of epistemic uncertainty.

We also saw that the bootstrap as a tool to evaluate epistemic uncertanties under a frequentist perspective. The bootstrap can be applied to any model, but its coverage guarantees are asymptotic and require assumptions to be met.

Calibration is another critical aspect of UQ. It ensures that probabilistic predictions reflect actual observed frequencies, making it essential in both regression and classification tasks. However, calibration alone is not sufficient. While many models may be calibrated, some can be dull and fail to fully utilize the predictive power of the available data. For this reason, calibration should only be applied after selecting the best model using appropriate proper loss functions, such as the Brier score, cross-entropy loss or the L_2 loss. In this context, calibration serves as a fine-tuning step, rather than as the primary criterion for model selection.

Here's an improved version of the text, avoiding personal references:

11.2 Dataset/Distributional Shifts and Ontological Uncertainties

In real-world applications, the distribution of training data frequently differs from that observed during deployment. This shift can compromise the reliability of uncertainty quantification if not adequately addressed. Techniques like domain adaptation, transfer learning, and importance weighting have been developed to adjust models and improve alignment with new data distributions (Gibbs and Candes, 2021; Izbicki et al., 2017; Kasa et al., 2024; Masserano et al., 2024; Tibshirani et al., 2019). However, these methods often rely on strong assumptions for effective correction or may only optimize against specific criteria, such as marginal validity (Gibbs and Candès, 2024).

Despite these technical advancements, there remain limits to what can be anticipated, leading to uncertainties that lie beyond the model's assumptions—referred to as ontological uncertainty, or "unknown unknowns" (Frank et al., n.d.; Gansch and Adee, 2020; Zhang et al., 2020). Unlike *known* dataset shifts, which can be mitigated through adaptation, ontological uncertainty arises from unforeseen changes—such as new regulations, disruptive technologies, or rare global events—that fall outside the model's predictive scope.

Index

.

Bibliography

Afrasiabi, M., Mohammadi, M., Rastegar, M., Stankovic, L., Afrasiabi, S., & Khazaei, M. (2020). Deep-based conditional probability density function forecasting of residential loads. *IEEE Transactions on Smart Grid*, *11*(4), 3646–3657.

Agostinelli, S., Allison, J., Amako, K. a., Apostolakis, J., Araujo, H., Arce, P., Asai, M., Axen, D., Banerjee, S., Barrand, G., et al. (2003). Geant4—a simulation toolkit. *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *506*(3), 250–303.

Algeri, S., & Zhang, X. (2022). Exhaustive goodness of fit via smoothed inference and graphics. *Journal of Computational and Graphical Statistics*, *31*(2), 378–389.

Almosallam, I. A., Jarvis, M. J., & Roberts, S. J. (2016). GPZ: non-stationary sparse Gaussian processes for heteroscedastic uncertainty estimation in photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, *462*(1), 726–739. https://doi.org/10.1093/mnras/stw1618

Angelopoulos, A., Bates, S., Malik, J., & Jordan, M. I. (2020). Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*.

Angelopoulos, A. N., Bates, S., et al. (2023). Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, *16*(4), 494–591.

Arnouts, S., Cristiani, S., Moscardini, L., Matarrese, S., Lucchin, F., Fontana, A., & Giallongo, E. (1999). Measuring and modelling the redshift evolution of clustering: the Hubble Deep Field North. *Monthly Notices of the Royal Astronomical Society*, *310*(2), 540–556. https://doi.org/10.1046/j.1365-8711.1999.02978.x

Assaf, S. A., & Al-Hejji, S. (2006). Causes of delay in large construction projects. *International Journal of Project Management*, *24*(4), 349–357.

Assunção, G. O., Izbicki, R., & Prates, M. O. (2023). Is augmentation effective to improve prediction in imbalanced text datasets? *arXiv preprint arXiv:2304.10283*.

.

Ballesteros-Pérez, P., Rojas-Céspedes, Y. A., Hughes, W., Kabiri, S., Pellicer, E., Mora-Melià, D., & del Campo-Hitschfeld, M. L. (2017). Weather-wise: A weather-aware planning tool for improving construction productivity and dealing with claims. *Automation in construction*, *84*, 81–95.

Bastos, L. S., Economou, T., Gomes, M. F., Villela, D. A., Coelho, F. C., Cruz, O. G., Stoner, O., Bailey, T., & Codeço, C. T. (2019). A modelling approach for correcting reporting delays in disease surveillance data. *Statistics in medicine*, *38*(22), 4363–4377.

Beaumont, M. A. (2010). Approximate bayesian computation in evolution and ecology. *Annual review of ecology, evolution, and systematics*, *41*, 379–406.

Beck, R., Lin, C.-A., Ishida, E., Gieseke, F., de Souza, R., Costa-Duarte, M., Hattab, M., Krone-Martins, A., & Collaboration, C. (2017). On the realistic validation of photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, *468*(4), 4323–4339.

Beck, R., Dobos, L., Budavári, T., Szalay, A. S., & Csabai, I. (2016). Photometric redshifts for the SDSS Data Release 12. *Monthly Notices of the Royal Astronomical Society*, *460*(2), 1371–1381. https://doi.org/10.1093/mnras/stw1009

Bejani, M. M., & Ghatee, M. (2021). A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, *54*(8), 6391–6438.

Benedetti, J. K. (1977). On the nonparametric estimation of regression functions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 248–253.

Benitez, N. (2000). Bayesian Photometric Redshift Estimation. *The Astrophysical Journal*, *536*(2), 571–583. https://doi.org/10.1086/308947

Bertin, K., Lacour, C., & Rivoirard, V. (2016). Adaptive pointwise estimation of conditional density function. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, *52*(2), 939–980.

Bian, M., & Barber, R. F. (2023). Training-conditional coverage for distribution-free predictive inference. *Electronic Journal of Statistics*, *17*(2), 2044–2066.

Bickel, P. J., & Li, B. (2007). Local polynomial regression on unknown manifolds. In *Ims lecture notes–monograph series, complex datasets and inverse problems* (pp. 177–186). Institute of Mathematical Statisitcs.

Bishop, C. M. (1994). Mixture density networks.

Bishop, C. M. (1997). Bayesian neural networks. *J. Braz. Comput. Soc.*, *4*(1).

Blum, M. G. (2010). Approximate bayesian computation: A nonparametric perspective. *Journal of the American Statistical Association*, *105*(491), 1178–1187.

Bordoloi, R., Lilly, S. J., & Amara, A. (2010). Photo-z performance for precision cosmology. *Monthly Notices of the Royal Astronomical Society*, *406*(2), 881–895. https://doi.org/10.1111/j.1365-2966.2010.16765.x

Boström, H., & Johansson, U. (2020). Mondrian conformal regressors. *Conformal and Probabilistic Prediction and Applications*, 114–133.

Boström, H., Johansson, U., & Löfström, T. (2021). Mondrian conformal predictive distributions. *Conformal and Probabilistic Prediction and Applications*, 24–38.

Bowman, A. W. (1985). A comparative study of some kernel-based nonparametric density estimators. *Journal of Statistical Computation and Simulation*, *21*(3-4), 313–327.

Brammer, G. B., van Dokkum, P. G., & Coppi, P. (2008). EAZY: A Fast, Public Photometric Redshift Code. *The Astrophysical Journal*, *686*(2), 1503–1513. https://doi.org/10.1086/591786

Breiman, L. (2001a). Statistical modeling: The two cultures. *Statistical Science*, *16*(3), 199–231.

Breiman, L. (2001b). Random forests. *Machine Learning*, *45*(1), 5–32.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, *78*(1), 1–3.

Cabezas, L. M., Otto, M. P., Izbicki, R., & Stern, R. B. (2024). Regression trees for fast and adaptive prediction intervals. *Information Sciences*, 121369.

Calonico, S., Cattaneo, M. D., & Farrell, M. H. (2018). On the effect of bias estimation on coverage accuracy in nonparametric inference. *Journal of the American Statistical Association*, *113*(522), 767–779.

Cameron, E., & Pettitt, A. (2012). Approximate bayesian computation for astronomical model analysis: A case study in galaxy demographics and morphological transformation at high redshift. *Monthly Notices of the Royal Astronomical Society*, *425*(1), 44–65.

Carrasco Kind, M., & Brunner, R. J. (2013). TPZ: photometric redshift PDFs and ancillary information by using prediction trees and random forests. *Monthly Notices of the Royal Astronomical Society*, *432*(2), 1483–1501. https://doi.org/10.1093/mnras/stt574

Cavuoti, S., Amaro, V., Brescia, M., Vellucci, C., Tortora, C., & Longo, G. (2017). METAPHOR: a machine-learning-based method for the probability density

estimation of photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, *465*(2), 1959–1973. https://doi.org/10.1093/mnras/stw2930

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Cheng, G., & Chen, Y.-C. (2019). Nonparametric inference via bootstrapping the debiased estimator. *Electronic Journal of Statistics*, *13*, 2194–2256.

Cheng, K. F., & Chu, C.-K. (2004). Semiparametric density estimation under a two-sample density ratio model. *Bernoulli*, *10*(4), 583–604.

Chernozhukov, V., Wüthrich, K., & Zhu, Y. (2021). Distributional conformal prediction. *Proceedings of the National Academy of Sciences*, *118*(48).

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian cart model search. *Journal of the American Statistical Association*, *93*(443), 935–948.

Chipman, H. A., George, E. I., & McCulloch, R. E. (2012). Bart: Bayesian additive regression trees. *Annals of Applied Statistics*, *6*(1), 266–298.

Codeço, C. T., Cruz, O. G., Riback, T. I., Degener, C. M., Gomes, M. F., Villela, D., Bastos, L., Camargo, S., Saraceni, V., Lemos, M. C. F., et al. (2016). Infodengue: A nowcasting system for the surveillance of dengue fever transmission. *BioRxiv*, 046193.

Comito, M. B., Izbicki, R., do Canto Hubert Junior, P., & Moura, F. (2024). Improving decision-making in construction: Nonparametric modeling of weather-induced delays [in prep.].

Cook, S. R., Gelman, A., & Rubin, D. B. (2006). Validation of software for bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, *15*(3), 675–692.

Cranmer, K., Brehmer, J., & Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, *117*(48), 30055–30062.

Cranmer, K., Pavez, J., & Louppe, G. (2015). Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*.

Dalmasso, N., Izbicki, R., & Lee, A. (2020a). Confidence sets and hypothesis testing in a likelihood-free inference setting. *International Conference on Machine Learning*, 2323–2334.

Dalmasso, N., Masserano, L., Zhao, D., Izbicki, R., & Lee, A. B. (2021). Likelihood-free frequentist inference: Bridging classical statistics and machine learning in simulator-based inference. *arXiv preprint arXiv:2107.03920*.

Dalmasso, N., Pospisil, T., Lee, A. B., Izbicki, R., Freeman, P. E., & Malz, A. I. (2020b). Conditional density estimation tools in python and r with applications to photometric redshifts and likelihood-free cosmological inference. *Astronomy and Computing*, 100362.

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap methods and their application*. Cambridge university press.

Dawid, A. P. (1982). The well-calibrated bayesian. *Journal of the American Statistical Association*, *77*(379), 605–610.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, *39*(1), 1–22.

Dey, B., Andrews, B. H., Newman, J. A., Mao, Y.-Y., Rau, M. M., & Zhou, R. (2021). Photometric Redshifts from SDSS Images with an Interpretable Deep Capsule Network. *arXiv e-prints*, Article arXiv:2112.03939, arXiv:2112.03939.

Dey, B., Zhao, D., Newman, J. A., Andrews, B. H., Izbicki, R., & Lee, A. B. (2022). Calibrated predictive distributions via diagnostics for conditional coverage. *arXiv preprint arXiv:2205.14568*.

Dheur, V., Bosser, T., Izbicki, R., & Ben Taieb, S. (2024). Distribution-free conformal joint prediction regions for neural marked temporal point processes. *Machine Learning*, 1–48.

Di Marzio, M., Fensore, S., Panzera, A., & Taylor, C. C. (2016). A note on nonparametric estimation of circular conditional densities. *Journal of Statistical Computation and Simulation*, 1–10.

Ding, T., Angelopoulos, A., Bates, S., Jordan, M., & Tibshirani, R. J. (2024). Class-conditional conformal prediction with many classes. *Advances in Neural Information Processing Systems*, *36*.

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*.

Durdyev, S., Hosseini, M. R., Baroudi, B., & Roshanaei, M. (2020). Causes of delay in residential building construction projects: A case study of azerbaijan. *Journal of Engineering, Design and Technology*, *18*(3), 659–672.

Edition, F. (2008). *A guide to the project management body of knowledge (pmbok guide)*. Project Management Institute.

Efromovich, S. (2010). Dimension reduction and adaptation in conditional density estimation. *Journal of the American Statistical Association*, *105*(490), 761–774.

.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. Chapman; Hall/CRC.

England, J. R., & Viscarra Rossel, R. A. (2018). Proximal sensing for soil carbon accounting. *Soil*, *4*(2), 101–122.

Estoup, A., Lombaert, E., MARIN, J.-M., Guillemaud, T., Pudlo, P., Robert, C. P., & CORNUET, J.-M. (2012). Estimation of demo-genetic model probabilities with approximate bayesian computation using linear discriminant analysis on summary statistics. *Molecular ecology resources*, *12*(5), 846–855.

Fan, J., Yao, Q., & Tong, H. (1996). Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, *83*(1), 189–206.

Fasiolo, M., Wood, S. N., Hartig, F., & Bravington, M. V. (2018). An extended empirical saddlepoint approximation for intractable likelihoods. *Electron. J. Statist.*, *12*(1), 1544–1578. https://doi.org/10.1214/18-EJS1433

Ferraty, F., & Vieu, P. (2006). *Nonparametric functional data analysis: Theory and practice*. Springer Science & Business Media.

Folgoc, L. L., Baltatzis, V., Desai, S., Devaraj, A., Ellis, S., Manzanera, O. E. M., Nair, A., Qiu, H., Schnabel, J., & Glocker, B. (2021). Is mc dropout bayesian? *arXiv preprint arXiv:2110.04286*.

Forman, G. (2008). Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, *17*, 164–206.

Foygel Barber, R., Candes, E. J., Ramdas, A., & Tibshirani, R. J. (2021). The limits of distribution-free conditional predictive inference. *Information and Inference: A Journal of the IMA*, *10*(2), 455–482.

Frank, M., Fuchs, C., & Zeller-Plumhoff, B. (n.d.). Helmholtz uq dictionary [Accessed: 2024-09-09]. https://dictionary.helmholtz-uq.de/content/landing_page.html

Fraser, D. A., & Guttman, I. (1956). Tolerance regions. *The Annals of Mathematical Statistics*, *27*(1), 162–179.

Fröhlich, A., Ramos, T., Cabello, G., Buzatto, I., Izbicki, R., & Tiezzi, D. (2024). Personalizedus: Interpretable breast cancer risk assessment with local coverage uncertainty quantification. *arXiv preprint arXiv:2408.15458*.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning*, 1050–1059.

Gansch, R., & Adee, A. (2020). System theoretic view on uncertainties. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1345–1350.

Ghosh, A., & Bera, A. K. (2002). Neyman's smooth test and its applications in econometrics. In *Handbook of applied econometrics and statistical inference* (pp. 199–252). CRC Press.

Gibbons, J. D., & Chakraborti, S. (2014). *Nonparametric statistical inference: Revised and expanded*. CRC press.

Gibbs, I., & Candes, E. (2021). Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, *34*, 1660–1672.

Gibbs, I., & Candès, E. J. (2024). Conformal inference for online prediction with arbitrary distribution shifts. *Journal of Machine Learning Research*, *25*(162), 1–36.

Gneiting, T., & Katzfuss, M. (2014). Probabilistic forecasting. *Annual Review of Statistics and Its Application*, *1*, 125–151.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Graff, P., Feroz, F., Hobson, M. P., & Lasenby, A. (2014). SKYNET: an efficient and robust neural network training tool for machine learning in astronomy. *Monthly Notices of the Royal Astronomical Society*, *441*(2), 1741–1759. https://doi.org/10.1093/mnras/stu642

Graham, M. L., Connolly, A. J., Ivezić, Ž., Schmidt, S. J., Jones, R. L., Jurić, M., Daniel, S. F., & Yoachim, P. (2018). Photometric Redshifts with the LSST: Evaluating Survey Observing Strategies. *The Astronomical Journal*, *155*(1), Article 1, 1. https://doi.org/10.3847/1538-3881/aa99d4

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman; Hall/CRC.

Greenberg, D., Nonnenmacher, M., & Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 2404–2414). PMLR.

Grivol, G., Izbicki, R., Okuno, A. A., & Stern, R. B. (2023). Flexible conditional density estimation for time series. *arXiv preprint arXiv:2301.09671*.

.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1321–1330.

Gupta, C., Podkopaev, A., & Ramdas, A. (2020). Distribution-free binary classification: Prediction sets, confidence intervals and calibration. *Advances in Neural Information Processing Systems*, *33*, 3711–3723.

Gutmann, M. U., & Corander, J. (2016). Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, *17*(125), 1–47.

Gutmann, M. U., Dutta, R., Kaski, S., & Corander, J. (2018). Likelihood-free inference via classification. *Statistics and Computing*, *28*, 411–425.

Györfi, L., Kohler, M., Krzyzak, A., Walk, H., et al. (2002). *A distribution-free theory of nonparametric regression* (Vol. 1). Springer.

Hall, P., Racine, J. S., & Li, Q. (2004). Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, *99*, 1015–1026.

Hall, P. (1987). On kullback-leibler loss and density estimation. *The Annals of Statistics*, 1491–1519.

Hall, P. (1992). On bootstrap confidence intervals in nonparametric regression. *The Annals of Statistics*, 695–711.

Heinrich, L. (2022). Learning optimal test statistics in the presence of nuisance parameters. *arXiv preprint arXiv:2203.13079*.

Hill, J., Linero, A., & Murray, J. (2020). Bayesian additive regression trees: A review and look forward. *Annual Review of Statistics and Its Application*, *7*, 251–278.

Ho, M., Rau, M. M., Ntampaka, M., Farahi, A., Trac, H., & Póczos, B. (2019). A robust and efficient deep learning method for dynamical mass measurements of galaxy clusters. *The Astrophysical Journal*, *887*(1), 25.

Hoffmann, L., & Elster, C. (2021). Deep ensembles from a bayesian perspective. *arXiv preprint arXiv:2105.13283*.

Horta, A., Malone, B., Stockmann, U., Minasny, B., Bishop, T., McBratney, A., Pallasser, R., & Pozza, L. (2015). Potential of integrated field spectroscopy and spatial analysis for enhanced assessment of soil contamination: A prospective review. *Geoderma*, *241*, 180–209.

Huang, C.-W., Krueger, D., Lacoste, A., & Courville, A. (2018). Neural autoregressive flows. *International Conference on Machine Learning*, 2078–2087.

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, *110*, 457–506.

Hulsman, R. (2022). Distribution-free finite-sample guarantees and split conformal prediction. *arXiv preprint arXiv:2210.14735*.

Hyndman, R. J., Bashtannyk, D. M., & Grunwald, G. K. (1996). Estimating and visualizing conditional densities. *Journal of Computational & Graphical Statistics*, *5*, 315–336.

Ichimura, T., & Fukuda, D. (2010). A fast algorithm for computing least-squares cross-validations for nonparametric conditional kernel density functions. *Computational Statistics Data Analysis*, *54*(12), 3404–3410.

Inácio, M. d. A., & Izbicki, R. (2018). Conditional density estimation using fourier series and neural networks. *Anais do VI Symposium on Knowledge Discovery, Mining and Learning*, 41–48.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448–456.

Izbicki, R., & Lee, A. (2016). Conditional density estimation in a high-dimensional regression setting. *Journal of Computational and Graphical Statistics, to appear*.

Izbicki, R., Lee, A., & Schafer, C. (2014). High-dimensional density ratio estimation with extensions to approximate likelihood computation. *Journal of Machine Learning Research (AISTATS Track)*, 420–429.

Izbicki, R., & Lee, A. B. (2017). Converting high-dimensional regression to high-dimensional conditional density estimation. *Electronic Journal of Statistics*, *11*(2), 2800–2831.

Izbicki, R., Lee, A. B., & Freeman, P. E. (2017). Photo-z estimation: An example of nonparametric conditional density estimation under selection bias. *The Annals of Applied Statistics*, *11*(2), 698–724.

Izbicki, R., Lee, A. B., & Pospisil, T. (2019). ABC–CDE: Toward approximate bayesian computation with complex high-dimensional data and limited simulations. *Journal of Computational and Graphical Statistics*, *28*(3), 481–492.

Izbicki, R., Shimizu, G., & Stern, R. B. (2022). CD-split and HPD-split: Efficient conformal regions in high dimensions. *Journal of Machine Learning Research*.

.

Izbicki, R., Shimizu, G. T., & Stern, R. B. (2020). Distribution-free conditional predictive bands using density estimators. *Proceedings of Machine Learning Research (AISTATS Track)*.

Järvenpää, M., Gutmann, M. U., Vehtari, A., & Marttinen, P. (2021). Parallel Gaussian process surrogate Bayesian inference with noisy likelihood evaluations. *Bayesian Anal.*, *16*(1), 147–178. https://doi.org/10.1214/20-BA1200

Kasa, K., Zhang, Z., Yang, H., & Taylor, G. W. (2024). Adapting conformal prediction to distribution shifts without labels. *arXiv preprint arXiv:2406.01416*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 3146–3154.

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, *30*.

Kieseler, J., Strong, G. C., Chiandotto, F., Dorigo, T., & Layer, L. (2022). Calorimetric measurement of multi-TeV muons via deep regression. *The European Physical Journal C*, *82*(1), 1–26.

Kieseler, J., Strong, G. C., Chiandotto, F., Dorigo, T., & Layer, L. (2021). Preprocessed dataset for "calorimetric measurement of multi-tev muons via deep regression". *URL https://doi. org/10.5281/zenodo*, *5163817*.

Koenker, R., & Bassett Jr, G. (1978). Regression quantiles. *Econometrica: journal of the Econometric Society*, 33–50.

Kpotufe, S., & Dasgupta, S. (2012). A tree-based regressor that adapts to intrinsic dimension. *Journal of Computer and System Sciences*, *78*(5), 1496–1515.

Kpotufe, S. (2011). K-nn regression adapts to local intrinsic dimension. *Advances in Neural Information Processing Systems*, 729–737.

Kügler, S. D., Gianniotis, N., & Polsterer, K. L. (2016). A spectral model for multimodal redshift estimation. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8.

Lafferty, J., & Wasserman, L. (2008). Rodeo: Sparse, greedy nonparametric regression. *The Annals of Statistics*, *36*(1), 28–63.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, *30*.

Lancet, T. (2024). Dengue: The threat to health now and in the future.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. *International conference on artificial neural networks*, *60*, 53–60.

Lee, A. B., & Izbicki, R. (2016). A spectral series approach to high-dimensional nonparametric regression. *Electronic Journal of Statistics*, *10*(1), 423–463.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., & Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, *113*(523), 1094–1111.

Lei, J., & Wasserman, L. (2014). Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *76*(1), 71–96.

Leistedt, B., & Hogg, D. W. (2017). Data-driven, Interpretable Photometric Redshifts Trained on Heterogeneous and Unrepresentative Data. *The Astrophysical Journal*, *838*(1), Article 5, 5. https://doi.org/10.3847/1538-4357/aa6332

Liu, M.-S., Fragkiadaki, K., & Walker, M. (n.d.). Likelihood-free inference of fornax dark matter density profile.

Lueckmann, J.-M., Bassetto, G., Karaletsos, T., & Macke, J. H. (2019). Likelihood-free inference with emulator networks. *Symposium on Advances in Approximate Bayesian Inference*, 32–53.

Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., & Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 1289–1299). Curran Associates, Inc. http://papers.nips.cc/paper/6728-flexible-statistical-inference-for-mechanistic-models-of-neural-dynamics.pdf

Luo, P., Wang, X., Shao, W., & Peng, Z. (2018). Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*.

Ma, X., He, X., & Shi, X. (2016). A variant of k nearest neighbor quantile regression. *Journal of Applied Statistics*, *43*(3), 526–537.

Mallat, S. (1999). *A wavelet tour of signal processing*. Academic press.

Malz, A. I., & Hogg, D. W. (2022). How to Obtain the Redshift Distribution from Probabilistic Redshift Estimates. *The Astrophysical Journal*, *928*(2), Article 127, 127. https://doi.org/10.3847/1538-4357/ac062f

.

Mandelbaum, R., Seljak, U., Hirata, C. M., Bardelli, S., Bolzonella, M., Bongiorno, A., Carollo, M., Contini, T., Cunha, C. E., Garilli, B., Iovino, A., Kampczyk, P., Kneib, J. .-., Knobel, C., Koo, D. C., Lamareille, F., Le Fèvre, O., Le Borgne, J. .-., Lilly, S. J., . . . Tasca, L. (2008). Precision photometric redshift calibration for galaxy-galaxy weak lensing. *Monthly Notices of the Royal Astronomical Society*, *386*(2), 781–806. https://doi.org/10.1111/j.1365-2966.2008.12947.x

Manning, C. D. (2009). *An introduction to information retrieval*. Cambridge university press.

Manokhin, V. (2022). *Awesome conformal prediction* (Version v1.0.0) ["If you use Awesome Conformal Prediction, please cite it as below."]. Zenodo. https://doi.org/10.5281/zenodo.6467205

Marin, J.-M., Pudlo, P., Robert, C. P., & Ryder, R. J. (2012). Approximate bayesian computational methods. *Statistics and computing*, *22*(6), 1167–1180.

Marin, J.-M., Raynal, L., Pudlo, P., Ribatet, M., & Robert, C. (2016). ABC random forests for Bayesian parameter inference. *Bioinformatics (Oxford, England)*, *35*. https://doi.org/10.1093/bioinformatics/bty867

Marques, L., & Berenson, D. (2024). Quantifying aleatoric and epistemic dynamics uncertainty via local conformal calibration. *arXiv preprint arXiv:2409.08249*.

Masserano, L., Dorigo, T., Izbicki, R., Kuusela, M., & Lee, A. (2023). Simulator-based inference with waldo: Confidence regions by leveraging prediction algorithms and posterior estimators for inverse problems. *International Conference on Artificial Intelligence and Statistics*, 2960–2974.

Masserano, L., Shen, A., Doro, M., Dorigo, T., Izbicki, R., & Lee, A. B. (2024). Classification under nuisance parameters and generalized label shift in likelihood-free inference. *arXiv preprint arXiv:2402.05330*.

McAllister, R. T., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., & Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Meier, L., Van de Geer, S., & Bühlmann, P. (2009). High-dimensional additive modeling. *The Annals of Statistics*, *37*(6B), 3779–3821.

Meinshausen, N., & Ridgeway, G. (2006). Quantile regression forests. *Journal of machine learning research*, *7*(6).

Mendes de Oliveira, C., Ribeiro, T., Schoenell, W., Kanaan, A., Overzier, R. A., Molino, A., Sampedro, L., Coelho, P., Barbosa, C. E., Cortesi, A., Costa-Duarte, M. V., Herpich, F. R., Hernandez-Jimenez, J. A., Placco, V. M., Xavier, H. S., Abramo, L. R., Saito, R. K., Chies-Santos, A. L., Ederoclite, A., ... Zaritsky, D. (2019). The Southern Photometric Local Universe Survey (S-PLUS): improved SEDs, morphologies, and redshifts with 12 optical filters. *Monthly Notices of the Royal Astronomical Society*, *489*(1), 241–267. https://doi.org/10.1093/mnras/stz1985

Met office midas open: Uk land surface stations data (1853-current). (2019).

Mian, Z., Deng, X., Dong, X., Tian, Y., Cao, T., Chen, K., & Al Jaber, T. (2024). A literature review of fault diagnosis based on ensemble learning. *Engineering Applications of Artificial Intelligence*, *127*, 107357.

Murphy, A. H., & Epstein, E. S. (1967). Verification of probabilistic predictions: A brief review. *Journal of Applied Meteorology and Climatology*, *6*(5), 748–755.

Nakazono, L., R Valença, R., Soares, G., Izbicki, R., Ivezić, Ž., R Lima, E., T Hirata, N., Sodré Jr, L., Overzier, R., Almeida-Fernandes, F., et al. (2024). The quasar catalogue for s-plus dr4 (qucats) and the estimation of photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, *531*(1), 327–339.

Navarro, J. F. (1996). The structure of cold dark matter halos. *Symposium-international astronomical union*, *171*, 255–258.

Nelder, J. A., & Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, *135*(3), 370–384.

Nemani, V., Biggio, L., Huan, X., Hu, Z., Fink, O., Tran, A., Wang, Y., Zhang, X., & Hu, C. (2023). Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial. *Mechanical Systems and Signal Processing*, *205*, 110796.

Neter, J., Kutner, M. H., Nachtsheim, C. J., & Wasserman, W. (1996). Applied linear statistical models.

Neyman, J. (1937a). » smooth test» for goodness of fit. *Scandinavian Actuarial Journal*, *1937*(3-4), 149–199.

Neyman, J. (1937b). Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, *236*(767), 333–380.

Okuno, A. A., & Polo, F. M. (2021). Ocde: Odds conditional density estimator. *arXiv preprint arXiv:2107.04118*.

.

Oliveira, R. I., Orenstein, P., Ramos, T., & Romano, J. V. (2022). Split conformal prediction for dependent data. *arXiv preprint arXiv:2203.15885*.

Padarian, J., Minasny, B., & McBratney, A. (2022). Assessing the uncertainty of deep learning soil spectral models using monte carlo dropout. *Geoderma*, *425*, 116063.

Papadopoulos, H., Proedrou, K., Vovk, V., & Gammerman, A. (2002). Inductive confidence machines for regression. *European Conference on Machine Learning*, 345–356.

Papamakarios, G., & Murray, I. (2016). Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in neural information processing systems*, *29*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, *22*(1), 2617–2680.

Papamakarios, G., Pavlakou, T., & Murray, I. (2017). Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, *30*.

Papamakarios, G., Sterratt, D., & Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *The 22nd International Conference on Artificial Intelligence and Statistics*, 837–848.

Picchini, U., Simola, U., & Corander, J. (2020). Adaptive MCMC for synthetic likelihoods and correlated synthetic likelihoods. *arXiv preprint arXiv:2004.04558*.

Platt, J., et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, *10*(3), 61–74.

Podkopaev, A., & Ramdas, A. (2021). Distribution-free uncertainty quantification for classification under label shift. *Uncertainty in artificial intelligence*, 844–853.

Polsterer, K. L. (2016). Dealing with uncertain multimodal photometric redshift estimations. *Proceedings of the International Astronomical Union*, *12*(S325), 156–165.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. *Advances in neural information processing systems*, 6638–6648.

Qin, J. (1998). Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, *85*(3), 619–630.

Quandt, R. E. (1958). The estimation of the parameters of a linear regression system obeying two separate regimes. *Journal of the american statistical association*, *53*(284), 873–880.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2022). *Dataset shift in machine learning*. Mit Press.

Rahaman, R., et al. (2021). Uncertainty quantification and deep ensembles. *Advances in Neural Information Processing Systems*, *34*, 20063–20075.

Ravikumar, P., Lafferty, J., Liu, H., & Wasserman, L. (2009). Sparse additive models. *Journal of the Royal Statistical Society, Series B*, *71*(5), 1009–1030.

Ray, E. L., & Reich, N. G. (2018). Prediction of infectious disease epidemics via weighted density ensembles. *PLoS computational biology*, *14*(2), e1005910.

Reis, V. C., & Izbicki, R. (2023). Spectral smooth tests for goodness-of-fit. *arXiv preprint arXiv:2308.06601*.

Romano, Y., Patterson, E., & Candès, E. (2019). Conformalized quantile regression. In *Advances in neural information processing systems* (pp. 3543–3553). Curran Associates, Inc.

Romano, Y., Sesia, M., & Candes, E. (2020). Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, *33*, 3581–3591.

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

Rosenblatt, M. (1969). Conditional probability density and regression estimators. In P. Krishnaiah (Ed.), *Multivariate analysis ii*.

Rowe, B. T., Jarvis, M., Mandelbaum, R., Bernstein, G. M., Bosch, J., Simet, M., Meyers, J. E., Kacprzak, T., Nakajima, R., Zuntz, J., et al. (2015). Galsim: The modular galaxy image simulation toolkit. *Astronomy and Computing*, *10*, 121–150.

Sadeh, I., Abdalla, F. B., & Lahav, O. (2016). ANNz2: Photometric Redshift and Probability Distribution Function Estimation using Machine Learning. *Publications of the Astronomical Society of the Pacific*, *128*(968), 104502. https://doi.org/10.1088/1538-3873/128/968/104502

Sadinle, M., Lei, J., & Wasserman, L. (2019). Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, *114*(525), 223–234.

Sanni-Anibire, M. O., & Egbu, C. (2022). Causes of delay in construction projects: A review of literature. *Built Environment Project and Asset Management*, *12*(1), 28–46.

.

Schervish, M. J. (2012). *Theory of statistics*. Springer Science & Business Media.

Schervish, M. J., & DeGroot, M. H. (2014). *Probability and statistics* (Vol. 563). Pearson Education London, UK:

Schmidt, S. J., Malz, A. I., Soo, J. Y. H., Almosallam, I. A., Brescia, M., Cavuoti, S., Cohen-Tanugi, J., Connolly, A. J., DeRose, J., Freeman, P. E., Graham, M. L., Iyer, K. G., Jarvis, M. J., Kalmbach, J. B., Kovacs, E., Lee, A. B., Longo, G., Morrison, C. B., Newman, J. A., ... LSST Dark Energy Science Collaboration. (2020). Evaluation of probabilistic photometric redshift estimation approaches for The Rubin Observatory Legacy Survey of Space and Time (LSST). *Monthly Notices of the Royal Astronomical Society*, *499*(2), 1587–1606. https://doi.org/10.1093/mnras/staa2799

Schuldt, S., Moerschell, J., Kim, J.-W., & Fang, D. (2021). Weather and construction: A preliminary study on the impact of weather events on productivity. *Journal of Construction Engineering and Management*, *147*(6), 04021058.

Searle, R., McBratney, A., Grundy, M., Kidd, D., Malone, B., Arrouays, D., Stockman, U., Zund, P., Wilson, P., Wilford, J., et al. (2021). Digital soil mapping and assessment for australia and beyond: A propitious future. *Geoderma Regional*, *24*, e00359.

Selvan, R., Faye, F., Middleton, J., & Pai, A. (2020). Uncertainty quantification in medical image segmentation with normalizing flows. *Machine Learning in Medical Imaging: 11th International Workshop, MLMI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Proceedings 11*, 80–90.

Sepasgozar, A., Samad M Eshtehardian, Shirowzhan, S., Nadoushani, Z. M., & Foroozanfa, M. (2019). Delay causes and emerging digital tools: A novel model of delay analysis and risk mitigation. *International Journal of Project Management*, *37*(7), 941–961.

Shafer, G., & Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, *9*(3).

Shalizi, C. (2013). Advanced data analysis from an elementary point of view.

Shiga, M., Tangkaratt, V., & Sugiyama, M. (2015a). Direct conditional probability density estimation with sparse feature selection. *Machine Learning*, *100*(2-3), 161–182.

Shiga, M., Tangkaratt, V., & Sugiyama, M. (2015b). Direct conditional probability density estimation with sparse feature selection. *Machine Learning*, *100*(2), 161–182. https://doi.org/10.1007/s10994-014-5472-x

Sisson, S. A., Fan, Y., & Tanaka, M. M. (2007). Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, *104*(6), 1760–1765.

Sistema de Informação de Agravos de Notificação (SINAN). (2024). Dengue, chikungunya e zika - sinan portal [Accessed: 2024-09-03]. https://portalsinan.saude.gov.br/sinan-dengue-chikungunya

Sparapani, R., Spanbauer, C., & McCulloch, R. (2021). Nonparametric machine learning and efficient computation with bayesian additive regression trees: The bart r package. *Journal of Statistical Software*, *97*, 1–66.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stone, C. J. (1977). Consistent nonparametric regression. *The annals of statistics*, 595–620.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, *36*(2), 111–133.

Strigari, L. E., Frenk, C. S., & White, S. D. (2017). Dynamical models for the sculptor dwarf spheroidal in a Λcdm universe. *The Astrophysical Journal*, *838*(2), 123.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., & Okanohara, D. (2010a). Conditional density estimation via least-squares density ratio estimation. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 781–788.

Sugiyama, M., Suzuki, T., & Kanamori, T. (2010b). Density ratio estimation: A comprehensive review (statistical experiment and its related topics). *Kulib Repository*, *1703*, 10–31.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Takeuchi, I., Nomura, K., & Kanamori, T. (2009). Nonparametric conditional density estimation using piecewise-linear solution path of kernel quantile regression. *Neural Computation*, *21*(2), 533–559.

Tan, Y. V., & Roy, J. (2019). Bayesian additive regression trees and the general bart model. *Statistics in medicine*, *38*(25), 5048–5069.

Tanaka, M., Coupon, J., Hsieh, B.-C., Mineo, S., Nishizawa, A. J., Speagle, J., Furusawa, H., Miyazaki, S., & Murayama, H. (2018). Photometric redshifts for

hyper suprime-cam subaru strategic program data release 1. *Publications of the Astronomical Society of Japan*, *70*(SP1). https://doi.org/10.1093/pasj/psx077

Tatiana, B., Didier, C., David, R., & Derek, Y. (2009). Mixtools: An r package for analyzing finite mixture models. *Journal of Statistical Software*, *32*(6), 1–29.

Teye, M., Azizpour, H., & Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. *International Conference on Machine Learning*, 4907–4916.

Theodoridis, S., & Koutroumbas, K. (2006). *Pattern recognition*. Elsevier.

Thomas, O., Dutta, R., Corander, J., Kaski, S., & Gutmann, M. U. (2022). Likelihood-free inference by ratio estimation. *Bayesian Analysis*, *17*(1), 1–31.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Tibshirani, R. J., Foygel Barber, R., Candes, E., & Ramdas, A. (2019). Conformal prediction under covariate shift. *Advances in neural information processing systems*, *32*.

Vaicenavicius, J., Widmann, D., Andersson, C., Lindsten, F., Roll, J., & Schön, T. (2019). Evaluating model calibration in classification. *The 22nd International Conference on Artificial Intelligence and Statistics*, 3459–3467.

Valdenegro-Toro, M., & Mori, D. S. (2022). A deeper look into aleatoric and epistemic uncertainty disentanglement. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1508–1516.

Valle, D., Izbicki, R., & Leite, R. V. (2023). Quantifying uncertainty in land-use land-cover classification using conformal statistics. *Remote Sensing of Environment*, *295*, 113682.

Vaz, A. F., Izbicki, R., & Stern, R. B. (2019). Quantification under prior probability shift: The ratio estimator and its extensions. *Journal of Machine Learning Research*, *20*(79), 1–33.

Vovk, V. (2012). Conditional validity of inductive conformal predictors. *Asian conference on machine learning*, 475–490.

Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world* (Vol. 29). Springer.

Wager, S., Wang, S., & Liang, P. S. (2013). Dropout training as adaptive regularization. *Advances in neural information processing systems*, *26*.

Wald, A. (1943). Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical society*, *54*(3), 426–482.

Wasserman, L. (2006). *All of nonparametric statistics*. Springer Science & Business Media.

Wehenkel, A., & Louppe, G. (2019). Unconstrained monotonic neural networks. *Advances in neural information processing systems*, *32*.

Weyant, A., Schafer, C., & Wood-Vasey, W. M. (2013). Likelihood-free cosmological inference with type ia supernovae: Approximate bayesian computation for a complete treatment of uncertainty. *The Astrophysical Journal*, *764*(2), 116.

Wilks, S. S. (1941). Determination of sample sizes for setting tolerance limits. *The Annals of Mathematical Statistics*, *12*(1), 91–96.

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning* (Vol. 2). MIT press Cambridge, MA.

Wittman, D. (2009). What lies beneath: Using p (z) to reduce systematic photometric redshift errors. *The Astrophysical Journal Letters*, *700*(2), L174.

Wood, S. (2010). Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, *466*, 1102–4. https://doi.org/10.1038/nature09319

Wu, J. T., Leung, K., & Leung, G. M. (2020). Nowcasting and forecasting the potential domestic and international spread of the 2019-ncov outbreak originating in wuhan, china: A modelling study. *The lancet*, *395*(10225), 689–697.

Xiao, Y., Soares, G., Bastos, L., Izbicki, R., & Moraga, P. (2024). Dengue nowcasting in brazil by combining official surveillance data and google trends information. *medRxiv*. https://doi.org/10.1101/2024.09.02.24312934

Yang, Y., & Tokdar, S. T. (2015). Minimax-optimal nonparametric regression in high dimensions. *The Annals of Statistics*, *43*(2), 652–674.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *Icml*, *1*, 609–616.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 694–699.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. *arXiv preprint arXiv:2106.11342*.

Zhang, H., Zimmerman, J., Nettleton, D., & Nordman, D. J. (2020). Random forest prediction intervals. *The American Statistician*.

.

Zhao, D., Dalmasso, N., Izbicki, R., & Lee, A. B. (2021). Diagnostics for conditional density models and bayesian inference algorithms. *Uncertainty in Artificial Intelligence*, 1830–1840.

Zhou, R., Newman, J. A., Mao, Y.-Y., Meisner, A., Moustakas, J., Myers, A. D., Prakash, A., Zentner, A. R., Brooks, D., Duan, Y., Landriau, M., Levi, M. E., Prada, F., & Tarle, G. (2021). The clustering of DESI-like luminous red galaxies using photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, *501*(3), 3309–3331. https://doi.org/10.1093/mnras/staa3764