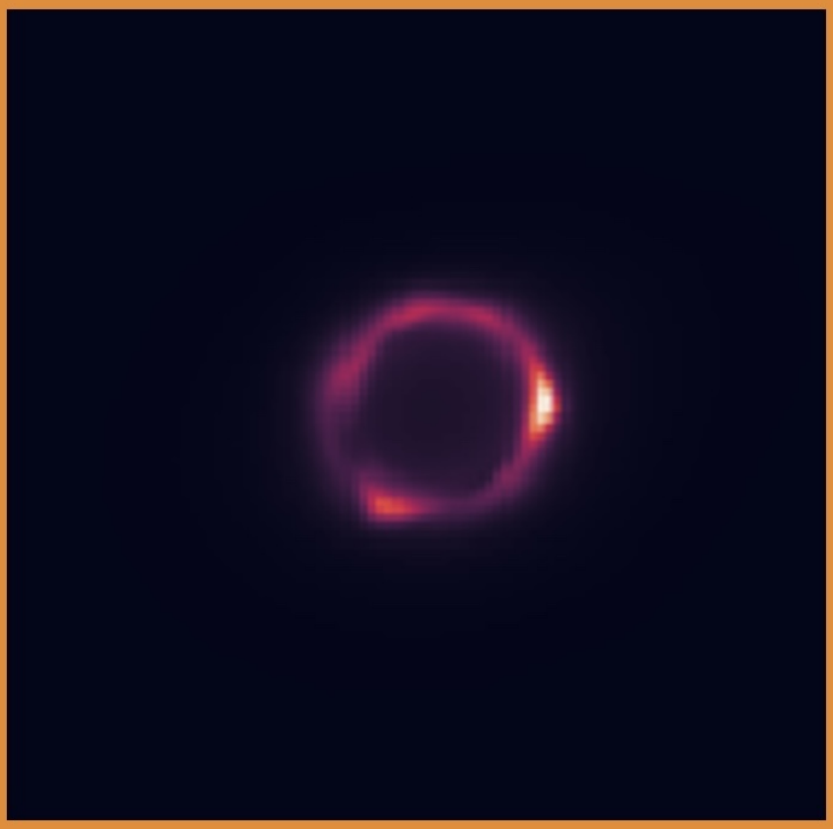
MSc thesis in Applied Mathematics

Point Prediction Uncertainty

Chenhao Zhang

2025



MSc thesis in Applied Mathematics

Point prediction uncertainty

Chenhao Zhang

January 2025

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Applied Mathematics

The work in this thesis was carried out in the:

ING Bank

Delft University of Technology

Supervisors: Dr. ir. J. Bierkens
Dr. Max Baak
Ralph Urlus

Abstract

This thesis investigates point prediction uncertainty in machine learning, focusing on its decomposition into variance and bias components. We explore the sources of these components and develop methods for their quantification.

We utilize the general bias-variance decomposition framework to analyze and quantify uncertainty in various predictive tasks using proper scoring rules and Bregman divergences.

Two approaches for quantifying variance uncertainty are investigated: Bregman Information and Beta/Dirichlet scores. Bregman Information, derived from the theoretical framework, captures the variance component but has limitations in sensitivity. Beta/Dirichlet scores offer better interpretability and performance near the decision boundary but discard some of the information.

To explore bias uncertainty, we leverage out-of-distribution (OOD) detection methods, particularly GradNorm and TracIn. GradNorm utilizes information from the gradient space to identify OOD samples, while TracIn employs a follow-up training approach to quantify the information gain required to correct potential prediction errors.

Through the experiments on synthetic and real-world datasets, we demonstrate the effectiveness of these methods in quantifying uncertainty. We analyze the impact of various factors on the performance of these methods, highlighting the need for careful hyperparameter tuning and showing their limitations.

The code for the methods and experiments is publicly available on GitHub.

This research contributes to the understanding and quantification of point prediction uncertainty in machine learning. The methods explored offer tools for identifying and mitigating uncertainty in machine learning models.

Acknowledgements

As I complete my Master of Science in Applied Mathematics, I find myself filled with gratitude for the individuals and institutions that have played important roles in shaping this academic journey.

First and foremost, I extend my heartfelt appreciation to my esteemed supervisors, Dr. ir. J. Bierkens and Dr. Max Baak, for their guidance, encouragement, and invaluable insights throughout this research endeavor. Their expertise and support, has been instrumental in refining the ideas presented in this thesis.

I would also like to express my sincere gratitude to Ralph Urlus, my daily supervisor, for his patient guidance and insightful feedback. His attention to detail and his willingness to share his expertise have been valuable in ensuring the rigor and clarity of this research.

I would also like to express my sincere gratitude to the Delft University of Technology for providing a stimulating and intellectually enriching environment for my graduate studies. Having the opportunity to learn from excellent faculty and to participate in a vibrant academic learning has been a very rewarding experience.

I am deeply grateful to ING Bank and my colleagues for their generous support of this research project. Their commitment to fostering academic exploration and their recognition of the importance of uncertainty quantification in real-world applications have made this research possible.

To my family and friends, thank you for your unwavering love, encouragement, and understanding throughout my studies. Your support has been my constant source of motivation, and your belief in my abilities has propelled me forward during challenging times.

With heartfelt gratitude, I acknowledge the contributions of each individual and institution that has made this academic journey a resounding success.

Contents

List of Figures

1 Introduction

In the financial industry, decisions often revolve around individual cases, carrying significant consequences for both the institution and the individuals involved. For instance, when evaluating loan applications, banks utilize models to assess creditworthiness and determine whether to approve or reject a loan. However, the model's overall uncertainty, which reflects its average performance across a population of applicants, does not necessarily reveal the uncertainty associated with a specific individual's prediction. Even a highly accurate model can make erroneous predictions for certain individuals, leading to potentially adverse outcomes.

This highlights the critical need for understanding and quantifying point prediction uncertainty – the uncertainty associated with individual predictions made by a model. In the context of loan applications, point prediction uncertainty can provide valuable insights into the reliability of the model's assessment for a particular applicant, enabling a more nuanced and informed decision-making process.

Understanding and quantifying point prediction uncertainties is the central topic of this thesis. In this chapter, we introduce the concept of pointwise uncertainty and delve into the main research questions that guide this work. We explore the challenges associated with assessing and interpreting uncertainty in individual predictions and lay the foundation for the subsequent chapters, which delve into specific methods and analyses aimed at addressing these challenges.

1.1 Uncertainty in Predictive Modeling

Uncertainty is an intrinsic element of predictive modeling, arising from various sources and influencing the reliability of predictions. Broadly, uncertainty in machine learning can be categorized into two main types: aleatoric and epistemic uncertainty.

Aleatoric Uncertainty: This type of uncertainty, also known as statistical uncertainty, stems from the inherent randomness in the data-generating process [2]. It represents the irreducible noise or stochasticity that cannot be eliminated, even with perfect knowledge of the underlying system or an infinitely large dataset. In essence, aleatoric uncertainty reflects the inherent unpredictability of the phenomenon being modeled, as depicted on the left side of Figure 1.1. Even with many training samples, there is inherent variability in the data that the model cannot perfectly capture [1].

Epistemic Uncertainty: This type of uncertainty, also referred to as systematic uncertainty, arises from limitations in our knowledge or understanding of the system. It can stem from factors such as insufficient data, model misspecification, or incomplete feature representation. Crucially, epistemic uncertainty can be reduced by acquiring more data, improving the model's structure, or incorporating more relevant features. The right side of Figure 1.1 illustrates this concept through the example of out-of-distribution(OOD) error, where the model makes larger errors when predicting in areas with few training samples.

While these two types of uncertainty are often used to characterize model-wise uncertainty, which pertains to the overall uncertainty of the model across the entire data space, our research focuses on point-wise

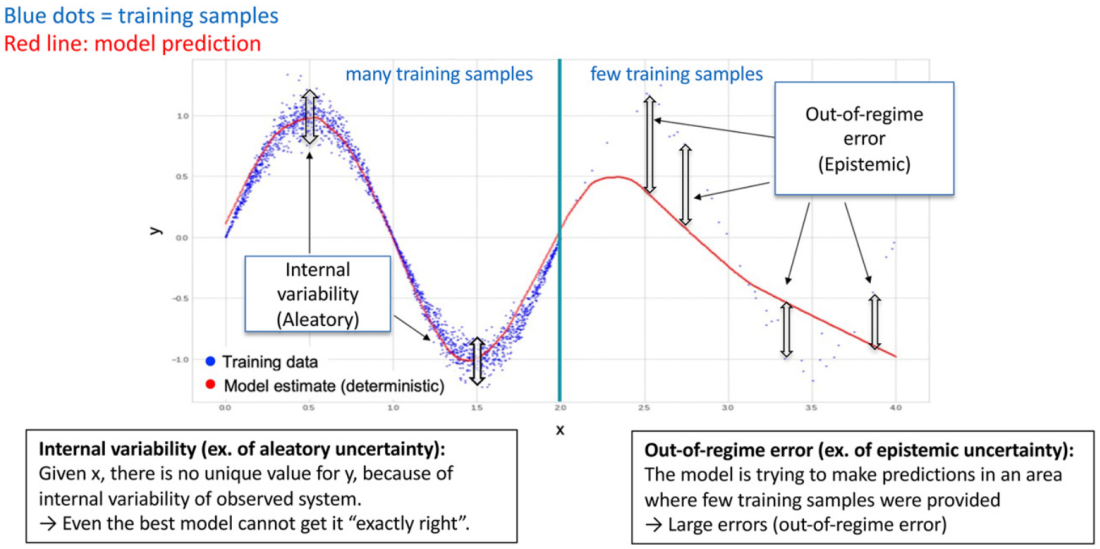


Figure 1.1: Example of Aleatoric and Epistemic Uncertainty [1].

uncertainty, which is the uncertainty associated with individual predictions at specific data points. Specifically, this thesis delves into point-wise uncertainty in the context of classification problems, where the goal is to assign data points to discrete categories, rather than regression problems.

1.2 Pointwise Uncertainty

To formalize this understanding, let's delve deeper into the process of training a predictive model. We begin with a learner – an algorithm that takes training data as input and produces a classifier. This classifier is a function mapping from the feature space $\mathscr{X}$ to probabilities over the label space $\mathscr{Y}$. The training data, comprising feature-label pairs $(x,y) \in \mathscr{X} \times \mathscr{Y}$, is drawn from an underlying distribution P.

Given a data point $x \in \mathscr{X}$, the classifier outputs a probability vector, representing the model's belief about the likelihood of different labels $y \in \mathscr{Y}$ for that point, for convenient we may assume there are in total c labels here. However, the output of the learner is not deterministic. It is influenced by various sources of randomness:

- **Randomness in the training data (D):** The specific instances used for training can vary due to sampling variability or data collection procedures.
- **Randomness within the learner itself:** This includes factors like random parameter initialization (θ) and all the other randomness in the learner (u), such as the learner can be affected by hardware [3].

Consequently, a single realization of the learner at one data point x can be represented as the conditional probability distribution $P(y \mid x; D, \theta, u) = P(y \mid x; \Theta)$, where $\Theta = (D, \theta, u)$, which differs in each realization. This highlights that the predicted probability vector $p_{x,\Theta} = (P(y_1 \mid x; \Theta), \ldots, P(y_c \mid x; \Theta))$ of the prediction is not a fixed value but rather a random vector from a distribution of possible probability vectors, denoted as P_x. This distribution P_x is typically unknown and reflects the variability in the model's

predictions due to the randomness inherent in the training process.

The true probability distribution P_x^t for the data point x is determined by the conditional distribution of the underlying distribution P at that point, i.e. $P_x^t = P(y \mid x)$. Let's denote the corresponding probability vector as p_x^t. Therefore, the uncertainty in the model's prediction at x is related to the discrepancy between the distribution P_x and the true probability vector p_x^t.

In practice, what we obtain from the classifier is a single sample $p \sim P_x$. In order to evaluate p, we need to find out property of P_x itself, as well as typically the the relation between P_x and p_x^t.

To further illustrate these concepts, let's consider the example shown in Figure 1.2. This figure depicts points drawn from two Gaussian distributions with the same variance and different means. The three lines represent possible decision boundaries learned by different classifiers trained on varying subsets of the data.

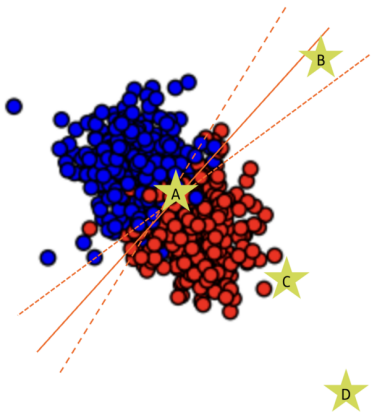


Figure 1.2: Example of Point-wise Uncertainty.

This example give us a better under standing of the two key components of point-wise uncertainty:

- **Variance - Randomness in the training procedure:** Points A and B in Figure 1.2 are likely to have different predictions depending on the specific classifier used. This variability in predictions is caused by the randomness in the training process, such as the selection of training data and the initialization of model parameters. This variability corresponds to the variance component of uncertainty.
- **Bias - Lack of information at a specific point:** While points C and D have relatively stable predictions across different classifiers, we intuitively trust the prediction for point C more than that for point D. This is because point C is closer to the observed data, while point D lies in a less informative region. This lack of information at point D contributes to the bias component, reflecting the potential for systematic errors in the model's prediction. This can be a serious problem when dealing with the softmax output, as a model can be uncertain in its predictions even with a high softmax output [4].

Point B exemplifies a combination of both variance and bias. It is both unstable, due to its proximity to the decision boundary, and untrustworthy, due to its distance from the training data.

It's important to note that in real-world scenarios with multidimensional feature spaces, data points can exhibit high variance and bias uncertainties even without being obvious outliers. A data point might

lie in a sparse region of the feature space, meaning it is surrounded by few other data points, yet still be considered an inlier based on its feature values. Such points are particularly susceptible to large variance and bias uncertainties, as the model may have limited information to make accurate and reliable predictions in those regions. Capturing and quantifying these uncertainties for such inliers within sparse regions is also a goal of this thesis.

Thus, it is natural to decompose and analyze pointwise uncertainty into these two key components: variance and bias, to gain a deeper understanding of the sources of prediction errors and develop more reliable and trustworthy predictive models.

Research Questions

In this study, to restrict the scope of the problem, we will make the following assumptions:

1. The ground truth distribution P, from which the training data is drawn, remains constant throughout the training and evaluation process. In other words, we assume there is no distribution shift over underlying distribution P as defined in the beginning of this section. This assumption allows us to isolate the uncertainty arising from the learner and the training process itself, rather than changes in the underlying data distribution.

2. The learner is assumed to be consistent, meaning that as the amount of training data increases, the predictions p of the learner converge to the ground truth p^t_x. This ensures that the observed uncertainty is not primarily due to instability in the learner's predictions with increasing data. Furthermore, this assumption implies that with sufficient training data, the learner will neither underfit nor overfit the data, striking a balance between capturing the underlying patterns and avoiding excessive memorization of the training data.

With these assumptions in mind, our research aims to address the following key questions:

- **Decomposition of Uncertainty:** Can we effectively decompose the point-wise uncertainty into two distinct components: one caused by the variance in the model's predictions due to randomness in the training process, and the other caused by the bias in the model's predictions due to systematic errors or lack of information?

- **Quantification of Uncertainty:** Can we develop and apply methods to quantify both the variance and bias components of point-wise uncertainty? This quantification will enable a more precise and nuanced understanding of the sources of uncertainty, facilitating better-informed decision-making in various applications. Which means we also aim to develop methods that are applicable to a wide range of learners[1], ideally achieving model agnosticism, so that they can be readily applied to various machine learning models without requiring specific adaptations or assumptions about their architecture or training process.

By addressing these research questions, we aim to contribute to a deeper understanding of point-wise uncertainty and its implications for predictive modeling. The insights gained from this research can aid in the development of more reliable and trustworthy prediction systems, enabling better decision-making in various domains.

[1] For simplicity, the evaluations have primarily used neural networks.

1.3 Datasets

To investigate the practical implications of the assumptions outlined above and address the research questions posed, we conduct experiments on a diverse collection of datasets. These datasets encompass two main categories: synthetic toy generated datasets and real-world datasets. This dual approach allows us to explore point-wise uncertainty under both controlled and realistic conditions, providing a comprehensive understanding of the factors influencing uncertainty in predictive models.

1.3.1 Toy Generators

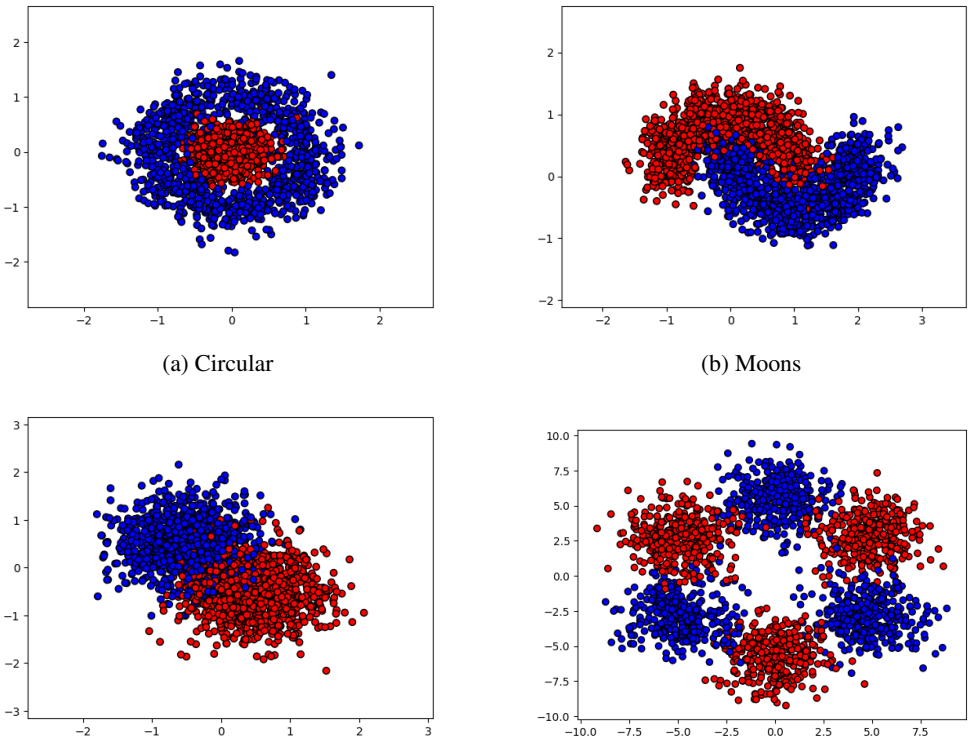
We begin by introducing a collection of two-dimensional, synthetic toy generators, shown in Figure 1.3, designed to simulate various data distributions with varying degrees of complexity. These generators offer a controlled environment for investigating the behavior of uncertainty measures under specific conditions. The generators include:

- **Gaussian Blobs**: This generator, being the most classic setting, produces two Gaussian clusters with varying degrees of overlap. This allows for the exploration of uncertainty in linearly and non-linearly separable cases. The advantage is that it provides a simple and interpretable setting for analyzing uncertainty, with a clear separation between clusters (depending on the overlap parameter) that allows for easy visualization and understanding of the model's behavior.
- **Moons**: With its more curved decision boundary than Gaussian Blobs, this generator produces two interleaving half-circles, creating a non-linearly separable dataset. The advantage is that it introduces a non-linear decision boundary, requiring models to learn more complex patterns. This is valuable for studying the behavior of more sophisticated models and understanding uncertainty in non-linear settings.
- **Circular**: This generator produces two concentric circles, providing a challenging scenario with a closed decision boundary for models that are not capable of capturing circular decision boundaries. The advantage is that it presents a unique challenge with its closed decision boundary. It requires models to learn a specific type of non-linearity, which can be difficult for some algorithms. This is useful for identifying models that struggle with specific types of decision boundaries.
- **Ring Blobs**: This generator produces multiple Gaussian blobs arranged in a ring-like pattern, creating a complex and multi-modal dataset. This is a more difficult task which can help our methods get a more convinced result. The advantage is that it creates a multi-modal and complex dataset with multiple Gaussian blobs arranged in a ring-like pattern. This complex scenario is valuable for evaluating the robustness and flexibility of uncertainty quantification methods.

By utilizing these toy generators, we can systematically examine the impact of data distribution, model complexity, and other factors on the estimated uncertainty. This analysis will provide valuable insights into the strengths and limitations of different uncertainty quantification techniques and their practical implications for real-world applications.

1.3.2 Real Datasets

In addition to synthetic toy datasets, we utilize real-world datasets to evaluate the practical applicability of our proposed methods for quantifying point prediction uncertainty. These datasets provide a more realistic setting for assessing the performance and robustness of our techniques.



(a) Circular

(b) Moons

(c) GaussianBlobs

(d) Ringblobs

Figure 1.3: Toy generator datasets.

One of the real-world datasets we employ is the Titanic dataset. This dataset contains information about the passengers onboard the Titanic, including their demographics, travel details, and survival status. We chose the Titanic dataset for several reasons:

- **Simplicity:** It is a relatively small and simple dataset, making it computationally manageable and allowing for a clear interpretation of the results.
- **Interpretable features:** The features in the dataset are easy to understand and have clear real-world meanings, facilitating the analysis and interpretation of the uncertainty estimates.
- **Well-studied:** The Titanic dataset is a classic and well-studied dataset in the machine learning community, providing a benchmark for comparison with other methods and facilitating the verification of our results.

The simplicity and interpretability of the Titanic dataset make it an ideal testbed for our methods. It allows us to verify the performance of our techniques in a controlled environment and gain insights into their behavior in a real-world setting. The well-defined features and clear prediction task enable us to speculate on the expected performance of our methods and analyze the results in a meaningful way.

By evaluating our methods on both synthetic and real-world datasets, we aim to demonstrate their robustness and generalizability across different data distributions and complexities. This comprehensive evaluation provides a strong foundation for the practical application of our techniques in various domains.

1.4 Structure of the Thesis

This thesis is structured to provide a comprehensive exploration of point prediction uncertainty. It starts with foundational concepts and culminates in a detailed analysis of specific uncertainty components. Chapter 2 delves into existing research on uncertainty quantification, focusing on techniques, applications, and decomposing uncertainty into variance and bias. Chapter 3 establishes the mathematical framework for decomposing uncertainty, including the general bias-variance decomposition and its application to exponential families. Chapter 4 explores the quantification and analysis of variance uncertainty, including the Bregman Information estimator and the Beta/Dirichlet estimator, along with experimental evaluations. Chapter 5 investigates bias uncertainty through the lens of out-of-distribution detection, employing methods like GradNorm and TracIn, along with experimental evaluations. Chapter 6 summarizes the key findings, discusses their implications, and addresses the limitations of the presented methods, suggesting potential avenues for future research. Finally, Chapter 7 synthesizes the key findings and contributions of the thesis, highlighting their impact on uncertainty quantification and bias mitigation in deep learning. This structured approach guides the reader through the complexities of point prediction uncertainty, providing a clear and comprehensive understanding of the topic.

2 Related Work

This chapter explores the growing field of uncertainty quantification (UQ) in deep learning, with a specific focus on its application to neural networks and the decomposition of point-wise uncertainty into variance and bias components. We examine various techniques for estimating and analyzing uncertainty in neural network predictions, discuss the practical implications and challenges associated with these methods, and review relevant literature on the sources and types of uncertainty in machine learning.

2.1 Uncertainty in models

Uncertainty quantification is essential for building trust in neural network models and enabling informed decision-making, especially in risk-sensitive applications. Various approaches have been proposed for quantifying uncertainty in neural networks. Bayesian Neural Networks (BNNs) offer a principled way to represent uncertainty by treating model parameters as random variables and inferring their posterior distributions. This allows for capturing both aleatoric and epistemic uncertainty. Seminal works by MacKay and Neal laid the foundation for Bayesian learning in neural networks, addressing challenges in setting network parameters, defining prior distributions, and performing computationally demanding integrations [5, 6].

Other techniques for uncertainty estimation in neural networks include Monte Carlo Dropout, which applies dropout during inference to create a pseudo-ensemble of networks, and ensemble methods, which combine predictions from multiple neural networks trained with different initializations or hyperparameters [7]. Deep Gaussian Processes (DGPs) extend the concept of Gaussian Processes to deep architectures, offering a flexible approach for uncertainty estimation [8].

Chen et al. introduce a confidence scoring mechanism for deep neural networks that uses a whitebox meta-model with linear classifier probes [9]. This approach outperforms multiple baselines in a filtering task, demonstrating the importance of confidence scoring in bridging the gap between experimental and real-world applications. The use of area under the receiver operating characteristic curve (AUROC) as a primary metric underscores the focus on the filtering aspect of confidence scoring, which aligns with the emphasis on point prediction uncertainty in this thesis.

2.2 Point Prediction Uncertainty

While many UQ methods focus on global-wise uncertainty, our research specifically addresses point prediction uncertainty, which is the uncertainty associated with individual predictions. Bouthillier et al. highlight the importance of considering variance in machine learning benchmarks when evaluating point prediction uncertainty, identifying sources of variance such as data sampling, data augmentation, model initialization, and hyperparameter optimization [3].

Several factors contribute to point prediction uncertainty [3, 4], including:

- **Data uncertainty**: This arises from noise or inherent randomness in the data. Measurement errors, missing values, and inherent stochasticity in the underlying process can all contribute to data uncertainty.
- **Model uncertainty**: This stems from limitations in the model's capacity or its ability to capture the true underlying relationship in the data. Choosing an appropriate model architecture and complexity is crucial for minimizing model uncertainty.
- **Training process uncertainty**: This is caused by randomness in the training process, such as data sampling and parameter initialization. Different random seeds or variations in the training data can lead to different model parameters and predictions.

These factors can significantly impact the performance and uncertainty of individual predictions, emphasizing the need for robust evaluation methods that account for these sources of variation. Their work underscores the importance of not just focusing on average model performance but also considering the variability in predictions across different trials and experimental conditions, which is crucial for understanding point prediction uncertainty.

2.3 Variance and Bias Decomposition

Understanding the sources of uncertainty is crucial for developing effective UQ methods. Abdar et al. categorize uncertainty into aleatoric uncertainty, inherent in the data, and epistemic uncertainty, arising from limitations in the model's knowledge [2]. These types of uncertainty typically focus on characterizing global-level uncertainty, but our research delves into pointwise uncertainty, which is more effectively captured through the lens of variance and bias decomposition.

A key aspect of our research is the decomposition of point prediction uncertainty into its variance and bias components. This decomposition, as explored by Gruber and Buettner, provides valuable insights into the different sources of prediction errors and guides the development of more robust and reliable models [10]. Their work introduces a general bias-variance decomposition framework for strictly proper scoring rules, enabling the analysis of uncertainty in a broader class of prediction tasks beyond traditional settings. We will follow this framework, providing a detailed proof of this decomposition and demonstrating its application in quantifying point prediction uncertainty.

Various techniques have been proposed for estimating and analyzing variance. Classical approaches such as cross-validation, particularly k-fold cross-validation, provide a robust way to estimate variance by partitioning the data into subsets, training the model on different combinations of these subsets, and evaluating its performance on the held-out [11]. The variability in performance across these different data splits gives an indication of the model's variance.

In addition to cross-validation, bootstrap methods involve repeatedly resampling the training data with replacement and retraining the model to estimate the variability in its predictions [2, 10]. This allows for assessing the stability of the model's predictions and quantifying the uncertainty due to variations in the training data, which we are going to use in Chapter 4.

Another approach is to use Deep Ensembles, which involve training multiple deep neural networks with different architectures and initialization parameters [2, 10]. By analyzing the diversity in predictions across these different models, Deep Ensembles provide insights into the sources of variance and can help improve the robustness of the overall prediction.

2.4 Out-of-distribution Detection

Out-of-distribution (OOD) detection is a critical task in machine learning that focuses on identifying inputs that differ significantly from the training data distribution [12]. These inputs often pose challenges for deep learning models, as their predictions on OOD data can be unreliable and unpredictable.

Uncertainty quantification helps the OOD detection, as the model's epistemic uncertainty estimates can provide valuable information for identifying OOD inputs [13]. Inversely, OOD detection plays a crucial role in improving uncertainty quantification in machine learning models. By identifying instances where a model encounters data significantly different from its training set, OOD detection helps refine and enhance the estimation of uncertainty associated with the model's predictions. This is because encountering OOD data often leads to higher uncertainty in the model's output, signaling potential unreliability [14]. OOD detection mechanisms can identify these instances, effectively highlighting potentially unreliable predictions and prompting further investigation or human intervention [15]. This is particularly important in safety-critical applications where incorrect predictions can have significant consequences.

Furthermore, OOD detection can be used to improve uncertainty estimation techniques themselves. By incorporating OOD detection into uncertainty estimation frameworks, models can provide more accurate and nuanced uncertainty estimates. For example, in semantic segmentation tasks, OOD detection can help identify pixels that are likely to be misclassified due to being out-of-distribution, leading to better uncertainty estimates for those pixels [16].

In essence, OOD detection acts as a critical feedback mechanism for uncertainty quantification. By identifying and flagging potential areas of high uncertainty, it helps refine and improve the overall uncertainty estimation process, leading to more reliable and trustworthy AI systems. This relationship is particularly relevant in the context of uncertainty caused by bias, as the uncertainty associated with individual predictions can be used to assess the likelihood of an input being OOD.

Two methods employed for OOD detection are TracIn and GradNorm. TracIn, proposed by Pruthi et al., offers a unique approach to OOD detection by calculating the influence of each training example on a prediction. It achieves this by tracing how the loss on a test point changes during the training process when a specific training example is included or excluded [17]. This method is informative with regards to OOD detection because OOD inputs, being dissimilar to the training data, are expected to exhibit lower influence scores across the training dataset. This is because the model's prediction on an OOD input is less likely to be strongly supported by any specific training examples. Consequently, these lower influence scores can serve as indicators of potential OOD inputs, effectively contributing to uncertainty quantification by highlighting predictions with weaker connections to the training data. GradNorm, introduced by Huang et al. [18], detects OOD inputs by analyzing the "gradient space," which can be understood as the multi-dimensional space where the gradients of the model's parameters reside. This method specifically utilizes the vector norm of gradients backpropagated from the KL divergence between the softmax output and a uniform probability distribution, leveraging the idea that gradients for in-distribution and OOD inputs will exhibit distinct characteristics in this space. We shall use these techniques in Chapter 5.

3 Uncertainty Decomposition

In this chapter, we delve into the concept of uncertainty decomposition, a critical aspect of understanding and mitigating the limitations of predictive models. We will explore how uncertainty of the prediction of the classifier can be dissected into its constituent components, providing valuable insights into the sources and nature of prediction errors. A key framework for this analysis is the bias-variance decomposition, which has been widely studied and applied in various machine learning contexts [19, 20].

We will further explore a generalized bias-variance decomposition introduced by Gruber and Buettner (2023), which extends this framework to a broader class of predictive tasks using the concept of proper scores [10]. Proper scores play a crucial role in evaluating and quantifying the uncertainty of probabilistic models. They incentivize models to output calibrated probabilities, capturing both accuracy and confidence in predictions.

By decomposing proper scores, we can gain a deeper understanding of how uncertainty manifests itself in different parts of the model. Specifically, we can analyze how the bias and variance terms contribute to the overall uncertainty. The bias term reflects the model's systematic errors or its inability to capture the true underlying relationship in the data. On the other hand, the variance term measures the model's sensitivity to fluctuations in the training data, indicating its tendency to overfit.

This generalized decomposition offers a more comprehensive view of uncertainty, applicable to various tasks beyond classification, including probabilistic modeling. It provides a powerful tool for analyzing and addressing the limitations of predictive models, leading to more robust and reliable predictions.

A crucial aspect of this generalized framework is its ability to quantify uncertainty through the lens of Bregman information. This information-theoretic measure, derived from Bregman divergences, acts as a powerful proxy for model variance, enabling a more nuanced understanding of prediction variability. By employing Bregman information, we can move beyond traditional uncertainty metrics, such as confidence scores, which can be unreliable under domain drift. This approach is particularly relevant in modern deep learning, where models often encounter data distributions that differ significantly from their training sets.

Building upon the work of Gruber and Buettner, this chapter provides a detailed walkthrough of their proposed decomposition. We elaborate on their essential steps and offer a comprehensive exposition of the underlying mathematical principles. Specifically, we follow Theorem 9, their proposed decomposition, but modify the proof strategy to highlight the connection between the decomposition and the Bregman divergence. This alternative proof offers a new perspective on the decomposition and clarifies its relationship with the broader concept of Bregman divergences. Furthermore, we provide a concrete example using the Brier score to illustrate the meaning of the theorem in a more clear way.

For Proposition 15, we follow the original proof provided by Gruber and Buettner but expand upon it with additional details and explanations. This aims to enhance the readability and accessibility of the proof, making it easier for readers to follow the logical steps and grasp the underlying mathematical concepts.

By providing a thorough and accessible treatment of this framework, combining detailed explanations with illustrative examples, we hope to facilitate its broader application in uncertainty quantification and contribute to a deeper understanding of the sources of uncertainty in machine learning models.

3.1 Preliminaries

To lay the groundwork for our exploration of uncertainty decomposition, we begin by establishing some essential mathematical preliminaries. This section introduces key definitions and theorems related to convex analysis, proper scoring rules, and Bregman divergences. These concepts form the bedrock upon which the general bias-variance decomposition framework is built. A solid understanding of these foundational elements will enable us to delve deeper into the intricacies of uncertainty analysis and appreciate the elegance and power of the generalized decomposition approach.

We begin with the fundamental concept of convexity, which plays a crucial role in defining and understanding convex functions.

Definition 1 (Convex set)**.** *A subset of an affine space over the $\mathbb{R}$ is convex if, given any two points in the subset, the subset contains the whole line segment that joins them.*

Definition 2 (Convex class of probability measures)**.** *To say that a set $\mathscr{P}$ of probability measures on $(\Omega, \mathscr{A})$ is convex (more standard than saying 'a convex class') just means that for all $P, Q \in \mathscr{P}$, and all $\lambda \in [0,1]$, the probability measure $P + (1-\lambda)Q$ is a member of $\mathscr{P}$. While $P + (1-\lambda)Q$ is defined pointwise: for any $A \in \mathscr{A}$,*

$$(P + (1-\lambda)Q)(A) := P(A) + (1-\lambda)Q(A).$$

With the concept of convex sets in hand, we can now define score and entropy within this context. These concepts are crucial for evaluating the quality of probabilistic predictions.

Definition 3 (Score and entropy)**.** *Let Ω be a general sample space, $\mathscr{A}$ be a σ-algebra of subsets of Ω. Let $\mathscr{P}$ be a convex class of probability measures on $(\Omega, \mathscr{A})$ we define the score on it as $S : \mathscr{P} \to \mathscr{F}(\Omega, \bar{\mathbb{R}})$, where $\mathscr{F}(\Omega, \bar{\mathbb{R}})$ is the extended real-valued functions on Ω. Then S can also be considered as $S : \mathscr{P} \to \mathscr{F}(\mathscr{P}, \bar{\mathbb{R}})$, by defining*

$$S(P) \cdot Q = \int S(P) \cdot \omega \mathrm{d}Q(\omega)$$

We then define the associated entropy $G : \mathscr{P} \to \mathbb{R}$ as $G(P) = S(P) \cdot P$.

Following we are going to give examples of the score and entropy. Consider a m classes classification task, the classifier would output a probability vector $p = (p_1, \ldots, p_m)$, where p_i represent the probability of the the data point belongs to i class. Then

Example 1 (Brier score)**.** *The Brier score over the sample space R^m is defined as follow*

$$S(p) \cdot i = -\sum_{j=1}^{m} (\delta_{ij} - p_j)^2 = 2p_i - \sum_{j=1}^{m} p_j^2 - 1,$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. After this we have the Brier score over the probability measure

$$S(p) \cdot q = \sum_{i=1}^{m} q_i \cdot (S(p) \cdot i)$$

then the associated entropy

$$\begin{aligned} G(p) &= S(p) \cdot p \\ &= \sum_{i=1}^{m} p_i \cdot (S(p) \cdot i) \\ &= \sum_{i=1}^{m} p_i \cdot (2p_i - \sum_{j=1}^{m} p_j^2 - 1) \\ &= \sum_{i=1}^{m} 2p_i^2 - \sum_{j=1}^{m} p_j^2 - 1 \\ &= \sum_{i=1}^{m} p_i^2 - 1 \end{aligned}$$

Example 2 (Log score)**.** *The log score over the sample space R^m is defined as follow*

$$S(p)\dot{i} = \ln p_i,$$

the log score over the probability measure

$$S(p) \cdot q = \sum_{i=1}^{m} q_i \ln p_i$$

the associated entropy

$$S(p) \cdot p = \sum_{i=1}^{m} p_i \ln p_i$$

The concept of a proper score is central to our analysis. Proper scores incentivize models to output calibrated probabilities, which are essential for reliable uncertainty estimation.

Definition 4 (Proper)**.** *A score S is proper if $S(P) \cdot Q \leq S(Q) \cdot Q$ for all $P, Q \in \mathscr{P}$, and strictly proper if the equality implies $P = Q$.*

Convex functions and their properties play a crucial role in the theory of proper scoring rules and uncertainty decomposition.

Definition 5 (Convex)**.** *A function $G : \mathscr{P} \to \mathbb{R}$ is convex if and only if*

$$G((1-\lambda)P_0 + \lambda P_1) \leq (1-\lambda)G(P_0) + \lambda G(P_1)$$

for all $\lambda \in (0,1)$, $P_0, P_1 \in \mathscr{P}$; Strictly convex if and only if equality implies $P_0 = P_1$.

The notion of a subgradient is essential for understanding the properties of convex functions and their relationship to proper scores.

Definition 6 (Subgradient)**.** *A subgradient $x' \in \mathscr{L}(\mathscr{P})$, where $\mathscr{L}(\mathscr{P}) = \{f \mid \int |f| \mathrm{d}P < \infty, P \in \mathscr{P}\}$, at point $x \in U \subset \mathscr{P}$ of a function $\phi : U \to \mathbb{R}$ fullfills the property $\phi(y) \geq \phi(x) + x' \cdot (y - x)$ for all $y \in U$. A function ϕ' which maps to a subgradient of $\phi(x)$ for all $x \in U$ is called a selection of subgradients or, if it is unambiguous in the context, just subgradient of ϕ.*

Lemma 1 provides a useful inequality for strictly convex functions, which will be used in later proofs.

Lemma 1. *[21] For the strictly convex functions the subgradients satisfy*

$$\phi(y) > \phi(x) + x' \cdot (y - x)$$

for all $y \neq x$.

Proof. By the definition of subgradient at x, for $y \neq x$ it holds that $\phi(y) \geq \phi(x) + x' \cdot (y - x)$. We assume there is a $y \neq x$ with $\phi(y) = \phi(x) + x' \cdot (y - x)$. By strictly convex we know that for $\lambda \in [0,1]$

$$\phi(\lambda x + (1-\lambda)y) < \lambda\phi(x) + (1-\lambda)\phi(y) = \phi(x) + (1-\lambda)x'(y-x)$$

For the point $\lambda x + (1-\lambda)y$ apply the subgradient condition, we have

$$\phi(\lambda x + (1-\lambda)y) \geq \phi(x) + x'(\lambda x + (1-\lambda)y - x) = \phi(x) + (1-\lambda)x'(y-x)$$

which show the contradiction. Thus $\phi(y) > \phi(x) + x' \cdot (y - x)$ holds for all $y \neq x$. □

The Convex Support Theorem is a fundamental result in functional analysis that is the extension of the supporting hyperplane theorem in a infinite dimension setting.

Theorem 2 (Convex Support Theorem)**.** *Any convex function from a real vector space X into $\mathbb{R}$ is the pointwise maximum of the affine functions that lie below it. That is, if $p : X \to \mathbb{R}$ is convex, then for each $x_0 \in X$ there exists some affine function $f : X \to \mathbb{R}$ that satisfies $f(x) \leq p(x)$ for all $x \in X$ and $f(x_0) = p(x_0)$ [22].*

Lemma 3 establishes a connection between the convexity of a function and the existence of subgradients, providing a key tool for analyzing convex functions on a set of distributions.

Lemma 3. *G is convex on a convex set of distributions U if and only if there exists $G' : U \to \mathscr{L}(U)$ maps U into the U-integrable functions, such that for all $P \in U$, we have $G(Q) \geq G(P) + G'(P)(Q - P)$ for all $Q \in U$, and strictly convex if and only if the equality implies $P = Q$.*

Proof. "⇐": For all $P \in U, Q \in U$, let $R = (1-\lambda)Q + \lambda P$, by the assumption we have

$$(1-\lambda)G(Q) \geq (1-\lambda)G(R) + (1-\lambda)G'(R)(Q-R) \tag{3.1}$$
$$\lambda G(P) \geq \lambda G(R) + \lambda G'(R)(P-R) \tag{3.2}$$

add the above two equations we have

$$(1-\lambda)G(Q) + \lambda G(P) \geq G(R) = G((1-\lambda)Q + \lambda P) \tag{3.3}$$

which shows G is convex.
For the strictly convex case, the equality in (3.3) can only be achieved when there equality in both (3.1) and (3.2). By assumption, for (3.1) and (3.2), the equality implies $R = P$ and $R = Q$, which also implies $P = Q$, by the definition of convex, the equality in (3.3) implies $P = Q$ only if G is strictly convex.

"⇒": By Theorem 2, we have that there is an affine function $f_P : U \to \mathbb{R}$ that satisfies $f_P(Q) \leq G(Q)$ for all $Q \in U$, and $f_P(P) = G(P)$. By the equality, subtract $f_P(P)$ and $G(P)$ on each side, the inequality still holds

$$G(Q) - G(P) \geq f_P(Q) - f_P(P).$$

By the affinity of the function f_P, we have

$$G(Q) \geq G(P) + f_P(Q - P)$$

The equality implies $P = Q$ if and only if for $Q \neq P$ implies strictly greater. We assume that exists a $Q \neq P$ such that the equality holds, and we want to show a contradiction. By the strictly convexity, for $\lambda \in (0,1)$ we have

$$\begin{aligned} G(\lambda P + (1-\lambda)Q) &< \lambda G(P) + (1-\lambda)G(Q) \\ &= G(Q) + (1-\lambda) f_P(Q-P) \end{aligned}$$

and for $\lambda P + (1-\lambda)Q$, we applied subgradient condition at P

$$\begin{aligned} G(\lambda P + (1-\lambda)Q) &\geq G(P) + f_P(\lambda P + (1-\lambda)Q - P) \\ &= G(Q) + (1-\lambda) f_P(Q-P) \end{aligned}$$

where there is a contradiction. It shows $Q \neq P$ implies strictly greater, which implies the equality implies $P = Q$. □

Theorem 4 establishes a fundamental link between proper scores and convex entropy functions, highlighting the importance of convexity in the theory of proper scoring rules.

Theorem 4. *The score S is (strictly) proper if and only if the associated entropy G is (strictly) convex and has S as a subgradient.[23]*

Proof. "⇒": Since S is proper, we have $S(Q) \cdot Q \geq S(P) \cdot Q$ for all P. By adding and subtracting the same term $S(P) \cdot P$ on the RHS, we have $G(Q) \geq G(P) + S(P) \cdot (Q-P)$, which implies S is a subgradient of G by the definition of subgradient. And according to Lemma 3, G is convex. If S is strict proper, by the definition of strict proper the equality implies $P = Q$, also according to Lemma 3, G is strictly convex.
"⇐": Since G is convex and has S as a subgradient, and by the subgradient condition, we have

$$\begin{aligned} & G(Q) - G(P) - S(P) \cdot (Q-P) \geq 0 \\ \Longleftrightarrow & S(Q) \cdot Q - S(P) \cdot P + S(P) \cdot P - S(P) \cdot Q \geq 0 \\ \Longleftrightarrow & S(Q) \cdot Q - S(P) \cdot Q \geq 0 \end{aligned}$$

which shows S is proper. And If G is strictly convex $\Longleftrightarrow$ the equality in the first inequality implies $P = Q$ $\Longleftrightarrow$ the equality in the last inequality implies $P = Q \Longleftrightarrow S$ is strictly proper. □

3.2 Bias variance decomposition

Building on the preliminary knowledge, we now introduce the concept of convex conjugates and Bregman divergences, which are essential for understanding the general bias-variance decomposition. These tools allow us to analyze uncertainty in a broader class of predictive tasks, extending beyond traditional classification and regression settings

Definition 7 (Convex conjugate)**.** *Given a vector space X with dual vector space X^*, pairing $\langle x, x^* \rangle = x^*(x)$ for $x \in X$ and $x^* \in X^*$ and a function $\phi : X \to \bar{\mathbb{R}}$, the convex conjugate $\phi^* : X^* \to \bar{\mathbb{R}}$ of ϕ is defined as $\phi^*(x^*) = \sup_{x \in X} \langle x, x^* \rangle - \phi(x)$*

we will give an example of convex conjugate

Example 3 (convex conjugate of Brier score entropy)**.** *As introduced earlier in example 1, the Brier score entropy* $G(p) = \sum_{j=1}^{m} p_j^2 - 1$*, then*

$$\begin{aligned} G^*(p^*) &= \sup_{p \in \mathscr{P}} \langle p, p^* \rangle - G(P) \\ &= \sup_{\sum_{j=1}^{m} p_j = 1} \sum_{j=1}^{m} p^*(j) p_j - \sum_{j=1}^{m} p_j^2 + 1 \end{aligned}$$

By using the method of Lagrange multipliers we have that

$$\begin{aligned} p_j &= \frac{1}{2}\left(p^*(j) - \frac{1}{m} \sum_{i=1}^{m} p^*(i) + \frac{2}{m} \right) \\ &= \frac{1}{2}\left(p^*(j) - \bar{p^*} \right) + \frac{1}{m} \end{aligned}$$

where $\bar{p^*} = \frac{1}{m} \sum_{i=1}^{m} p^*(i)$*, substituting the p value into the formula* $\sum_{j=1}^{m} p^*(j) p_j - \sum_{j=1}^{m} p_j^2 + 1$*, we have*

$$G^*(p^*) = \frac{1}{4} \sum_{j=1}^{m} (p^*(j))^2 - \frac{1}{m} \left(\frac{m}{2} \bar{p^*} - 1 \right)^2 + 1$$

Lemma 5 establishes a connection between the convex conjugate of a function and its subgradient, providing a valuable tool for analyzing the properties of convex functions.

Lemma 5. *Let* P^* *be a subgradient at point* P *of a convex function* $G : U \to \mathbb{R}$*, then it holds that*

$$G^*(P^*) = P^* \cdot P - G(P).$$

Proof. By the definition of subgradient, we have

$$\begin{aligned} & G(Q) \geq G(P) + P^* \cdot (Q - P) \\ \Rightarrow & P^* \cdot Q - G(Q) \leq P^* \cdot P - G(P), \text{ for all } Q \in \mathscr{P} \end{aligned}$$

Combined with the definition of convex conjugate, we have

$$G^*(P^*) = \sup_{Q \in \mathscr{P}} \{ P^* \cdot Q - G(Q) \} = P^* \cdot P - G(P)$$

□

For strictly convex functions, Lemma 6 shows that their subgradients are injective, and the inverse of a subgradient is itself a subgradient of the convex conjugate.

Lemma 6. *If* $G : U \to \mathbb{R}$ *is strictly convex, then any subgradient* G' *of* G *is an injective function and its inverse* $(G')^{-1}$ *exists on* $G'(U) := \{ G'(P) \mid P \in U \}$*. Further,* $(G')^{-1}$ *is a subgradient of convex conjugate* G^* *[10].*

Proof. For $P \neq Q$, by the definition of subgradient,

$$\begin{aligned} G(Q) &> G(P) + G'(P)(Q - P) \\ G(P) &> G(Q) + G'(Q)(P - Q) \end{aligned}$$

which can be write as

$$G(Q) - G(P) > G'(P) \cdot (Q - P)$$
$$G(P) - G(Q) > -G'(Q) \cdot (Q - P)$$

add the above two inequalities, we have

$$0 > (G'(P) - G'(Q)) \cdot (Q - P)$$

which shows $G'(P) \neq G'(Q)$, in this way it is an an injective function, so the inverse exists.
Next, we want to show $(G')^{-1}$ is a subgradient of G^* on $G'(U)$, which is equivalent to show

$$G^*(Q^*) \geq G^*(P^*) + (G')^{-1}(P^*) \cdot (Q^* - P^*) \tag{3.4}$$

where $Q^*, P^* \in G'(U)$. Since $(G')^{-1}$ exists, we may assume that $Q^* = G'(Q)$ and $P^* = G'(P)$, then we can write inequality (3.4) as

$$\sup_{Y \in \mathscr{D}} \{Q^* \cdot Y - G(Y)\} \geq \sup_{X \in \mathscr{D}} \{Q^* \cdot X - G(X)\} + Q^* \cdot P - P^* \cdot P$$

by Lemma 5, we have

$$\begin{aligned} & Q^* \cdot Q - G(Q) \geq P^* \cdot P - G(P) + Q^* \cdot P - P^* \cdot P \\ \Longleftrightarrow & G(P) \geq G(Q) + Q^* \cdot (P - Q) \end{aligned}$$

since Q^* is the subgradient of G at Q, the equation holds. □

Definition 8 defines the Bregman divergence, a measure of the dissimilarity between two points in a space defined by a convex function. This concept is central to Lemma 7, which demonstrates a relationship between Bregman divergences in the original space and those in the dual space defined by the convex conjugate. This lemma is crucial for deriving the general bias-variance decomposition presented in Theorem 10.

Definition 8 (Bregman divergence)**.** *Given a convex function ϕ and its subgradient ϕ' we define the Bregman divergence as*

$$d_{\phi,\phi'}(x,y) = \phi(y) - \phi(x) - \phi'(x) \cdot (y - x)$$

Lemma 7. *For strict convex function $G : U \to \mathbb{R}$ with subgradient S, the Bregman divergences $d_{G,S}$ and $d_{G^*,S^{-1}}$, we have*

$$d_{G,S}(P,Q) = d_{G^*,S^{-1}}(S(Q), S(P)).$$

Proof. By the definition of Bregman divergence, the LHS can be written as

$$d_{G,S}(P,Q) = G(Q) - G(P) - S(P) \cdot (Q - P)$$

in the same way and by the Lemma 5 and 6, the RHS can be written as

$$\begin{aligned} & d_{G^*,S^{-1}}(S(Q), S(P)) \\ =& G^*(S(P)) - G^*(S(Q)) - (S(P) - S(Q)) \cdot S^{-1}(S(Q)) \\ =& (S(P) \cdot P - G(P)) - (S(Q) \cdot Q - G(Q)) - (S(P) \cdot Q - S(Q) \cdot Q) \\ =& G(Q) - G(P) - S(P) \cdot (Q - P) \end{aligned}$$

now we show that LHS = RHS. □

Lemma 8 and 9 provide further results on the properties of convex conjugates and Bregman divergences, which are necessary for proving the general bias-variance decomposition theorem.

Lemma 8. *If Q^* is a random variable with values in $\mathscr{L}(\mathscr{P})$, such that $\mathbb{E}[\phi^*(Q^*)]$ is finite and $\mathbb{E}[Q^*] \in \mathscr{L}(\mathscr{P})$, then $\phi^*(\mathbb{E}[Q^*]) \in \mathbb{R}$ is finite for any convex function $\phi : U \to \mathbb{R}$*

Proof. By Jensen's inequality

$$\infty > \mathbb{E}[\phi^*(Q^*)] \geq \phi^*(\mathbb{E}[Q^*])$$

and since $\mathbb{E}[Q^*] \in \mathscr{L}(\mathscr{P})$, for any $P \in U$

$$\begin{aligned} -\infty &< \mathbb{E}[Q^*] \cdot P - \phi(P) \\ &\leq \sup_{Q \in \mathscr{P}} \{\mathbb{E}[Q^*] \cdot Q - \phi(Q)\} \\ &= \phi^*(\mathbb{E}[Q^*]) \end{aligned}$$

□

Lemma 9. *Let $p^* \in S(U)$, Q^* be a random variable with realizations in $S(U)$, such that $\mathbb{E}[G^*(Q^*)]$ exists and $\mathbb{E}[Q^*] \in \mathscr{L}(\mathscr{P})$, then*

$$\mathbb{E}[d_{G^*,S^{-1}}(p^*, Q^*)] = d_{G^*,S^{-1}}(p^*, \mathbb{E}[Q^*]) + \mathbb{B}_{G^*}[Q^*]$$

where $\mathbb{B}_{G^*}[Q^*] = \mathbb{E}[G^*(Q^*)] - G^*(\mathbb{E}[Q^*])$.

Proof. By Lemma 8, and the convexity of convex conjugate, we have $G^*(\mathbb{E}[Q^*])$ is finite. By the definition of Bregman divergence

$$\begin{aligned} &\mathbb{E}[d_{G^*,S^{-1}}(p^*, Q^*)] \\ =&\mathbb{E}[G^*(Q^*) - G^*(p^*) - (Q^* - p^*) \cdot S^{-1}(p^*)] \\ =&\mathbb{E}[G^*(Q^*) - G^*(p^*) - (Q^* - p^*) \cdot S^{-1}(p^*)] + G^*(\mathbb{E}[Q^*]) - G^*(\mathbb{E}[Q^*]) \\ =&G^*(\mathbb{E}[Q^*]) - G^*(p^*) - (\mathbb{E}[Q^*] - p^*) \cdot S^{-1}(p^*) + \mathbb{E}[G^*(Q^*)] - G^*(\mathbb{E}[Q^*]) \\ =&d_{G^*,S^{-1}}(p^*, \mathbb{E}[Q^*]) + \mathbb{B}_{G^*}[Q^*] \end{aligned}$$

□

Note: By Lemma 7 we only show that $d_{G^*,S^{-1}} : S(U) \times S(U) \to \mathbb{R}$ is well-defined, bur generally $\mathbb{E}[Q^*] \notin S(U)$. Let $\partial G(U) := \bigcup_{P \in U} \partial G(P)$ be the set of all subgradients of G. From Lemma 8 follows that for all $C \in Conv(\partial G(U))$ the convex conjugate $G^*(C)$ is finite(by choosing Q^* as binary random variables). Consequently, the divergence $d_{G^*,S^{-1}} : S(U) \times Conv(\partial G(U)) \to \mathbb{R}$ is finite, and we can always extend the first domain by extending S^{-1}. Then $d_{G^*,S^{-1}} : Conv(\partial G(U)) \times Conv(\partial G(U)) \to \mathbb{R}$ is a Bregman divergence [10].

Theorem 10 is the central result of this section, providing a general bias-variance decomposition for proper scores. This decomposition offers a comprehensive view of uncertainty in predictive models, applicable to various tasks beyond classification and regression.

Theorem 10 (Bias-Variance decomposition)**.** *For a strictly proper score S with associated negative entropy G, an estimated prediction $\hat{f}$, and the true distribution Q, we have*

$$\mathbb{E}[-S(\hat{f})\cdot Q] = -G(Q) + \mathbb{B}_{G^*}[S(\hat{f})] - d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]). \tag{3.5}$$

which decomposes the score function into error, variance and bias terms [10].

Proof. To show the result, we only need to show

$$\begin{aligned}
&\mathbb{E}[-S(\hat{f})\cdot Q] = -G(Q) + \mathbb{B}_{G^*}[S(\hat{f})] - d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) \\
\Longleftrightarrow &G(Q) - \mathbb{E}[S(\hat{f})\cdot Q] = d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) - \mathbb{B}_{G^*}[S(\hat{f})] \\
\Longleftrightarrow &\mathbb{E}[G(Q) - S(\hat{f})\cdot Q] = d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) - \mathbb{B}_{G^*}[S(\hat{f})] \\
\Longleftrightarrow &\mathbb{E}[G(Q) - S(\hat{f})\cdot \hat{f} + S(\hat{f})\cdot \hat{f} - S(\hat{f})\cdot Q] = d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) - \mathbb{B}_{G^*}[S(\hat{f})] \\
\Longleftrightarrow &\mathbb{E}[G(Q) - G(\hat{f}) + S(\hat{f})\cdot(\hat{f} - Q)] = d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) - \mathbb{B}_{G^*}[S(\hat{f})] \\
\Longleftrightarrow &\mathbb{E}[d_{G,S}(\hat{f},Q)] = d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) - \mathbb{B}_{G^*}[S(\hat{f})]
\end{aligned}$$

By Theorem 4 and Lemma 6 we know that S is the subgradient of G and S^{-1} is the subgradient of G^*, then apply Lemma 7, we show

$$\mathbb{E}[d_{G,S}(\hat{f},Q)] = \mathbb{E}[d_{G^*,S^{-1}}(S(Q),S(\hat{f}))]$$

And finally we apply the Lemma 9 to the RHS,

$$\mathbb{E}[d_{G^*,S^{-1}}(S(Q),S(\hat{f}))] = d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})]) - \mathbb{B}_{G^*}[S(\hat{f})].$$

We notice that it is essentially the decomposition of the Bregman divergence in the the conjugate form. □

This theorem provides a general decomposition of the expected score (which reflects the model's performance) into three components:

- Irreducible Error: The term $-G(Q)$ represents the inherent uncertainty or irreducible error associated with the true distribution Q. This term is independent of the model and cannot be reduced, even with a perfect model.
- Variance: The term $\mathbb{B}_{G^*}[S(f)]$ represents the generalized variance of the model's predictions. It measures how much the predictions vary or fluctuate due to variations in the training data. A high variance indicates that the model is sensitive to the specific training data, leading to higher uncertainty.
- Bias: The term $d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})])$ represents the bias of the model. It measures the systematic error or the difference between the expected prediction and the true distribution. A high bias indicates that the model is making systematic errors and is not capturing the true underlying relationship in the data.

In essence, Theorem 10 decomposes the uncertainty of a predictive model into its bias and variance components, along with the irreducible error. This decomposition provides valuable insights into the sources of prediction errors and can guide the development of more robust and reliable models.

In order to get a better understanding of this decomposition, we use the Brier score as the example.

Example 4 (Brier score decomposition)**.** *We will focus on the variance term* $\mathbb{B}_{G^*}[S(\hat{f})]$ *and the bias term* $d_{G^*,S^{-1}}(S(Q),\mathbb{E}[S(\hat{f})])$*. We may assume, in this case,* $\hat{f} = p = (p_1,\dots,p_m)$ *and the true distribution* $Q = q = (q_1,\dots,q_m)$*.*

$$\mathbb{B}_{G^*}[S(p)] = \mathbb{E}[G^*(S(p))] - G^*(\mathbb{E}[S(p)])$$

let's first calculate the first term

$$\begin{aligned}
G^*(S(p)) =& \frac{1}{4}\sum_{i=1}^{m}(S(p)\cdot i)^2 - \frac{1}{m}\left(\frac{m}{2}\bar{S(p)} - 1\right)^2 + 1 \\
=& \frac{1}{4}\sum_{i=1}^{m}\left(4p_i^2 - 4p_i + 1 - 4p_i\sum_{j=1}^{m}p_j^2 + 2\sum_{j=1}^{m}p_j^2 + \left(\sum_{j=1}^{m}p_j^2\right)^2\right) \\
& - \frac{1}{m}\left[-\frac{m}{2}\left(\sum_{j=1}^{m}p_j^2 + 1\right)\right] + 1 \\
=& \sum_{j=1}^{m}p_i^2 - 1 + \frac{m}{4} - \sum_{j=1}^{m}p_j^2 + \frac{m}{2}\sum_{j=1}^{m}p_j^2 + \frac{m}{4}\left(\sum_{j=1}^{m}p_j^2\right)^2 \\
& - \left(\frac{m}{4}\left(\sum_{j=1}^{m}p_j^2\right)^2 + \frac{m}{2}\sum_{j=1}^{m}p_j^2 + \frac{m}{4}\right) + 1 \\
=& 0
\end{aligned}$$

which implies that the first term is 0*.And for the second term*

$$\begin{aligned}
G^*(\mathbb{E}[S(p)]) =& \frac{1}{4}\sum_{i=1}^{m}(\mathbb{E}[S(p)\cdot i])^2 - \frac{1}{m}\left(\frac{m}{2}\mathbb{E}[\bar{S(p)}] - 1\right)^2 + 1 \\
=& \frac{1}{4}\sum_{i=1}^{m}\left(4(\mathbb{E}[p_i])^2 - 4\mathbb{E}[p_i] + 1 - 4\mathbb{E}[p_i]\sum_{j=1}^{m}\mathbb{E}[p_j^2] + 2\sum_{j=1}^{m}\mathbb{E}[p_j^2] + \left(\sum_{j=1}^{m}\mathbb{E}[p_j^2]\right)^2\right) \\
& - \frac{1}{m}\left[-\frac{m}{2}\left(\sum_{j=1}^{m}\mathbb{E}[p_j^2] + 1\right)\right] + 1 \\
=& \sum_{j=1}^{m}(\mathbb{E}[p_i])^2 - 1 + \frac{m}{4} - \sum_{j=1}^{m}\mathbb{E}[p_j^2] + \frac{m}{2}\sum_{j=1}^{m}\mathbb{E}[p_j^2] + \frac{m}{4}\left(\sum_{j=1}^{m}\mathbb{E}[p_j^2]\right)^2 \\
& - \left(\frac{m}{4}\left(\sum_{j=1}^{m}\mathbb{E}[p_j^2]\right)^2 + \frac{m}{2}\sum_{j=1}^{m}\mathbb{E}[p_j^2] + \frac{m}{4}\right) + 1 \\
=& \sum_{i=1}^{m}\left\{(\mathbb{E}[p_j])^2 - \mathbb{E}[p_j^2]\right\}
\end{aligned}$$

where we can easily notice that it is the sum of the variance of predicting class probability. For the bias term we have

$$\begin{aligned}
& d_{G^*,S^{-1}}(S(q),\mathbb{E}[S(p)]) \\
=& G^*(\mathbb{E}[S(p)]) - G^*(\mathbb{E}[S(q)]) - S^{-1}(S(q))(\mathbb{E}[S(p)] - S(q))
\end{aligned}$$

the first term has already been calculated above

$$G^*(\mathbb{E}[S(p)]) = \sum_{i=1}^{m} \left\{ (\mathbb{E}[p_j])^2 - \mathbb{E}[p_j^2] \right\}.$$

The second term, in the same way of the previous first term equals 0*. So now we focus on the last term*

$$\begin{aligned} S^{-1}(S(q))(\mathbb{E}[S(p)] - S(q)) =& (\mathbb{E}[S(p)\cdot q] - S(q)\cdot q) \\ =& \mathbb{E}[\sum_{i=1}^{m} q_i (S(p)\cdot i)] - S(q)\cdot q) \\ =& \mathbb{E}[\sum_{i=1}^{m} q_i (2p_i - \sum_{j=1}^{m} p_j^2 - 1)] - S(q)\cdot q) \\ =& \mathbb{E}[\sum_{i=1}^{m} 2q_i p_i - \sum_{j=1}^{m} p_j^2 - 1] - (\sum_{j=1}^{m} q_j^2 - 1) \\ =& 2\sum_{i=1}^{m} \mathbb{E}[p_i] q_i - \sum_{i=1}^{m} \mathbb{E}[p_i^2] - \sum_{j=1}^{m} q_j^2 \end{aligned}$$

Here we have

$$\begin{aligned} & d_{G^*,S^{-1}}(S(q), \mathbb{E}[S(p)]) \\ =& G^*(\mathbb{E}[S(p)]) - G^*(\mathbb{E}[S(q)]) - S^{-1}(S(q))(\mathbb{E}[S(p)] - S(q)) \\ =& \sum_{i=1}^{m} \left\{ (\mathbb{E}[p_j])^2 - \mathbb{E}[p_j^2] \right\} - 0 - \left(2\sum_{i=1}^{m} \mathbb{E}[p_i] q_i - \sum_{i=1}^{m} \mathbb{E}[p_i^2] - \sum_{j=1}^{m} q_j^2 \right) \\ =& \sum_{i=1}^{m} \left\{ (\mathbb{E}[p_j])^2 - 2\sum_{i=1}^{m} \mathbb{E}[p_i] q_i + q_i^2 \right\} \\ =& \sum_{i=1}^{m} \left\{ (\mathbb{E}[p_j] - q_i)^2 \right\}. \end{aligned}$$

Similar to the variance term, it is the the sum of the bias between the predicting class probability and the true class probability.

This example demonstrates that in the specific case of the Brier score, the variance and bias terms derived from the general decomposition in Theorem 9 correspond precisely to the classical statistical notions of variance and bias. The variance term captures the variability in the predicted probabilities, while the bias term reflects the discrepancy between the average predicted probabilities and the true probabilities. This correspondence provides valuable intuition for understanding how the Bregman Information and Bregman Divergence, used in the general decomposition, can be interpreted as generalized measures of variance and bias, respectively.

3.3 Exponential families

Having established the general bias-variance decomposition in Theorem 10, we now turn our attention to a specific case that holds significant relevance in machine learning: exponential families. Exponential families encompass a wide array of distributions commonly employed in various learning tasks. By

tailoring the decomposition to this family, we can gain deeper insights into the interplay of bias and variance within these widely used models.

Furthermore, this analysis with the exponential family will lay the groundwork for a specialized decomposition of the classification log-likelihood, a crucial aspect of understanding uncertainty in classification problems. The connection between the general case and this specific scenario will be elucidated, demonstrating the versatility and applicability of the core principles presented in Theorem 10.

Definition 9 introduces the concept of exponential families, a broad class of probability distributions that includes many common distributions used in machine learning. This definition is relevant for Proposition 15, which provides a specific bias-variance decomposition for exponential families.

Definition 9 (Exponential family)**.** *An exponential family is a parametric set of probability distributions,which have probability density function in form of*

$$p_\theta(x) = \exp(\langle \theta, T(x) \rangle - A(\theta))h(x)$$

we call $\theta \in \Theta$ the natural parameter of the convex parameter space $\Theta \subset \mathbb{R}^d$, and A the log normalizer.

Lemma 11 provides important properties of the log normalizer in exponential families, which are used in the proof of Proposition 15.

Lemma 11. *For the random variable Y follow an exponential family distribution, we have that*

$$\nabla A(\theta) = \mathbb{E}[T(Y)],$$

and

$$\nabla^2 A(\theta) = Var[T(Y)],$$

which implies A is a convex function.

Proof. For the exponential family, we have

$$\int_\Omega h(x) \exp(\langle \theta, T(x) \rangle - A(\theta)) \mathrm{d}\mu(x)$$

which implies

$$A(\theta) = \ln \int_\Omega h(x) \exp(\langle \theta, T(x) \rangle) \mathrm{d}\mu(x).$$

Then we take the derivative of $A(\theta)$

$$\begin{aligned}
\nabla A(\theta) =& \frac{\partial A(\theta)}{\partial \theta} \\
=& \frac{\partial}{\partial \theta} \left(\ln \int_\Omega h(x) \exp(\langle \theta, T(x) \rangle) \mathrm{d}\mu(x) \right) \\
=& \frac{\int_\Omega h(x) \exp(\langle \theta, T(x) \rangle) T(x) \mathrm{d}\mu(x)}{\exp(A(\theta))} \\
=& \int_\Omega h(x) \exp(\langle \theta, T(x) \rangle - A(\theta)) T(x) \mathrm{d}\mu(x) \\
=& \int_\Omega p_\theta(x) T(x) \mathrm{d}\mu(x) \\
=& \mathbb{E}[T(Y)]
\end{aligned}$$

for the second order derivative

$$\begin{aligned}\nabla^2 A(\theta) &= \int_\Omega h(x)\exp(\langle\theta, T(x)\rangle - A(\theta))T(x)\,(T(x)-\nabla A(\theta))\,\mathrm{d}\mu(x)\\ &= \int_\Omega p_\theta(x)T(x)\,(T(x)-\nabla A(\theta))\,\mathrm{d}\mu(x)\\ &= \int_\Omega p_\theta(x)T^2(x)\mathrm{d}\mu(x) - \nabla A(\theta)\int_\Omega p_\theta(x)T(x)\mathrm{d}\mu(x)\\ &= \mathbb{E}[T^2(x)] - \mathbb{E}^2[T(x)]\\ &= \mathrm{Var}[T(x)] \succeq 0\end{aligned}$$

which shows A is convex. □

Lemma 12 derives the convex conjugate of the negative Shannon entropy, a key concept in information theory. This lemma is used in the proof of Proposition 15.

Lemma 12 (Convex conjugate of the negative Shannon entropy)**.** *The Convex conjugate of the negative Shannon entropy $H : \mathscr{P} \to \mathbb{R}$, where $H(P) = \ln p \cdot P = \int_\Omega \ln p \cdot \mathrm{d}P$, is $H^*(P^*) = \ln \int_\Omega e^{P^*}\mathrm{d}\mu$, where $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$ is the Radon-Nikodym derivative of a distribution P with base measure μ of the related measure space $(\Omega, \mathscr{F}, \mu)$. We assume the set of distributions $\mathscr{P}$ consists of distributions with the same base measure.*

Proof. The convex conjugate of H is defined as

$$\begin{aligned}H^*(P^*) &= sup_{Q\in\mathscr{P}}\{P^*\cdot Q - H(Q)\}\\ &= sup_{Q\in\mathscr{P}}\left\{\int_\Omega P^*(x)\mathrm{d}Q(x) - \int_\Omega ln\frac{\mathrm{d}Q}{\mathrm{d}\mu}(x)\mathrm{d}Q(x)\right\}\\ &= sup_{Q\in\mathscr{P}}\left\{\int_\Omega P^*(x)q(x)\mathrm{d}\mu(x) - \int_\Omega q(x)lnq(x)\mathrm{d}\mu(x)\right\}\end{aligned}$$

where $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$, for the term inside the supreme

$$\begin{aligned}&exp\left(\int_\Omega P^*(x)q(x)\mathrm{d}\mu(x) - \int_\Omega q(x)\ln q(x)\mathrm{d}\mu(x)\right)\\ =&exp\left(\int_\Omega (P^*(x) - \ln q(x))q(x)\mathrm{d}\mu(x)\right)\\ \le&\int_\Omega exp(P^*(x) - \ln q(x))q(x)\mathrm{d}\mu(x)\\ =&\int_\Omega e^{P^*}\mathrm{d}\mu\end{aligned}$$

where the inequality is from Jensen's inequality, and the equality holds when $P^*(x) - lnq(x)$ is a constant, i.e. $q = e^{P^*-C}$. Then take logarithm on the both side, we have

$$\int_\Omega P^*(x)q(x)\mathrm{d}\mu(x) - \int_\Omega q(x)\ln q(x)\mathrm{d}\mu(x) \le \ln\int_\Omega e^{P^*}\mathrm{d}\mu$$

where we prove the result. □

Theorem 13 and Corollary 14 provide further results on convex conjugates and subgradients, which are necessary for proving Proposition 15.

Theorem 13. *[24] If a convex function f is lower semi-continuous , then $f^{**} = f$.*

Corollary 14. *[24] If a convex function f is lower semi-continuous , then*

$$y \in \partial f(x) \Longleftrightarrow x \in \partial f^*(y) \Longleftrightarrow x^\top y = f(x) + f^*(y).$$

Proposition 15 offers a specific bias-variance decomposition for exponential families, providing a more concrete application of the general decomposition in Theorem 10.

Proposition 15 (Bias-Variance decomposition for exponential family)**.** *For an exponential family density/mass function $p_{\hat{\theta}} = \exp(\langle \hat{\theta}, T(x)\rangle - A(\hat{\theta})) \cdot h(x)$, we have the log likelihood decomposition*

$$\mathbb{E}[-\ln p_{\hat{\theta}}(Y)] = n(\theta) + \mathbb{B}_A\left[\hat{\theta}\right] + d_A\left(\theta, \mathbb{E}\left[\hat{\theta}\right]\right) \tag{3.6}$$

where $n(\theta) = -A^*(\nabla A(\theta)) - \mathbb{E}[T(Y)]$ and $Y \sim p(\theta)$ [10].

Proof. The proof is similar to the proof of Proposition 3.4 in the paper from Gruber and Buettner [10], but provide more detailed explanation.
First, we require the more general formulations for the log score, the negative Shannon entropy by $S(P) = \ln \frac{\mathrm{d}P}{\mathrm{d}\mu}$, $H(P) = \ln \frac{\mathrm{d}P}{\mathrm{d}\mu}\mathrm{d}P$, by Lemma 12 we have the convex conjugate of the negative Shannon entropy $H^*(P^*) = \ln \int_\Omega e^{P^*}\mathrm{d}\mu$.

For an exponential family, we have

$$\frac{\mathrm{d}P_\theta}{\mathrm{d}\mu}(x) = p_\theta(x) = exp(\langle \theta, T(x)\rangle - A(\theta))h(x)$$

which shows

$$\frac{\mathrm{d}P_\theta}{\mathrm{d}\mu} = p_\theta = exp(\langle \theta, T\rangle - A(\theta))h \in \mathscr{L}(P)$$

which also introduces the notation. For the H^* of the log score, it follows

$$\begin{aligned}
& H^*\left(\mathbb{E}\left[S(P_{\hat{\theta}})\right]\right) \\
=& H^*\left(\mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right]\right) \\
=& \ln \int \exp\left(\left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] - \ln h\right)\mathrm{d}\mu \\
=& \ln \int \frac{exp\left(\left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle\right)h}{\exp\left(\mathbb{E}\left[A\left(\hat{\theta}\right)\right]\right)}\mathrm{d}\mu \\
=& \ln \frac{1}{\exp\left(\mathbb{E}\left[A\left(\hat{\theta}\right)\right]\right)} + \ln \int exp\left(\left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle\right)h\mathrm{d}\mu \\
=& \ln \int \exp\left(\left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle\right)h\mathrm{d}\mu - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] \\
=& \ln \int \frac{\exp\left(\left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle\right)h}{\exp\left(A\left(\mathbb{E}\left[\hat{\theta}\right]\right)\right)} \exp\left(A\left(\mathbb{E}\left[\hat{\theta}\right]\right)\right)\mathrm{d}\mu - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] \\
=& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) \cdot \ln \int \exp\left(\left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle - A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \ln h\right)\mathrm{d}\mu \\
& - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] \\
=& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \mathbb{E}\left[A\left(\hat{\theta}\right)\right]
\end{aligned}$$

we also know that

$$\mathbb{B}_{H^*}\left[S(P_{\hat{\theta}})\right] = \mathbb{E}\left[H^*\left(S(P_{\hat{\theta}})\right)\right] - H^*\left(\mathbb{E}\left[S(P_{\hat{\theta}})\right]\right)$$

the first term on the RHS is 0

$$\begin{aligned}\mathbb{E}\left[H^*\left(S(P_{\hat{\theta}})\right)\right] =& \mathbb{E}\left[H^*\left(\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right)\right] \\ =& \mathbb{E}\left[\ln \int \exp\left(\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right) \mathrm{d}\mu\right] = 0.\end{aligned}$$

So far we have

$$\mathbb{B}_{H^*}\left[S(P_{\hat{\theta}})\right] = -H^*\left(\mathbb{E}\left[S(P_{\hat{\theta}})\right]\right) = \mathbb{E}\left[A\left(\hat{\theta}\right)\right] - A\left(\mathbb{E}\left[\hat{\theta}\right]\right) = \mathbb{B}_A\left[\hat{\theta}\right]$$

since log score is a strictly proper score, by Theorem 4, the log score is a subgradient of Shannon entropy, which is strictly convex, and then by Lemma 6 the inverse S^{-1} exists.

For a random variable $Y \sim Q$, we have

$$\begin{aligned}\int_\Omega \mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right] \mathrm{d}Q &= \int_\Omega \left\langle \mathbb{E}\left[\hat{\theta}\right], T\right\rangle - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] - \ln h \, \mathrm{d}Q \\ &= \left\langle \mathbb{E}\left[\hat{\theta}\right], \mathbb{E}[T(Y)]\right\rangle - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] - \mathbb{E}[\ln h(Y)]\end{aligned} \tag{3.7}$$

For log normalizer A For the bias term, we first assume a general $Y \sim Q$

$$\begin{aligned}& d_{H^*,S^{-1}}\left(\ln \frac{\mathrm{d}Q}{\mathrm{d}\mu}, \mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right]\right) \\ \overset{\text{def}}{=}& \underbrace{H^*\left(\mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right]\right)}_{=A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \mathbb{E}\left[A\left(\hat{\theta}\right)\right]} - \underbrace{H^*\left(\ln \frac{\mathrm{d}Q}{\mathrm{d}\mu}\right)}_{=0} - \left(\mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right] - \ln \frac{\mathrm{d}Q}{\mathrm{d}\mu}\right) \cdot \underbrace{S^{-1}\left(\ln \frac{\mathrm{d}Q}{\mathrm{d}\mu}\right)}_{=Q} \\ =& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \mathbb{E}\left[A\left(\hat{\theta}\right)\right] - \int_\Omega \mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right] \mathrm{d}Q + H(Q) \\ \overset{\text{Eq}(3.7)}{=}& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \left\langle \mathbb{E}\left[\hat{\theta}\right], \mathbb{E}[T(Y)]\right\rangle + \mathbb{E}[\ln h(Y)] + H(Q) \\ \overset{\text{Le.11}}{=}& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \left\langle \mathbb{E}\left[\hat{\theta}\right], \mathbb{E}[T(Y)]\right\rangle + \mathbb{E}[\ln h(Y)] + H(Q) + A^*(\mathbb{E}[T(Y)]) \\ & - A^*(\mathbb{E}[T(Y)]) \\ \overset{\text{Le.5}}{=}& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \left\langle \mathbb{E}\left[\hat{\theta}\right], \mathbb{E}[T(Y)]\right\rangle + \mathbb{E}[\ln h(Y)] + H(Q) \\ & + \langle \nabla A^*(\mathbb{E}[T(Y)]), \mathbb{E}[T(Y)]\rangle - A^{**}(\nabla A^*(\mathbb{E}[T(Y)])) - A^*(\mathbb{E}[T(Y)]) \\ \overset{\text{Thm.13}}{=}& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - \left\langle \mathbb{E}\left[\hat{\theta}\right], \mathbb{E}[T(Y)]\right\rangle + \mathbb{E}[\ln h(Y)] + H(Q) \\ & + \langle \nabla A^*(\mathbb{E}[T(Y)]), \mathbb{E}[T(Y)]\rangle - A(\nabla A^*(\mathbb{E}[T(Y)])) - A^*(\mathbb{E}[T(Y)]) \\ \overset{\text{Cor.14}}{=}& A\left(\mathbb{E}\left[\hat{\theta}\right]\right) - A(\nabla A^*(\mathbb{E}[T(Y)])) \\ & - \left\langle \mathbb{E}\left[\hat{\theta}\right] - \nabla A^*(\mathbb{E}[T(Y)]), \nabla A(\nabla A^*(\mathbb{E}[T(Y)]))\right\rangle \\ & + \mathbb{E}[\ln h(Y)] + H(Q) - A^*(\mathbb{E}[T(Y)]) \\ \overset{\text{def}}{=}& d_A\left(\nabla A^*(\mathbb{E}[T(Y)]), \mathbb{E}\left[\hat{\theta}\right]\right) + \mathbb{E}[\ln h(Y)] + H(Q) - A^*(\mathbb{E}[T(Y)]).\end{aligned}$$

Now, if $Y \sim P_\theta$, the distribution from the respective exponential family with natural parameter θ. Then by Lemma 11 and Lemma 6, we have $\theta = \nabla A^*(\mathbb{E}[T(Y)])$, by the definition of log score and Theorem 10

$$\begin{aligned}
\mathbb{E}[-\ln p_{\hat{\theta}}(Y)] =& -H(Q) + \mathbb{B}_{H^*}\left[S(P_{\hat{\theta}})\right] + d_{H^*,S^{-1}}\left(\ln \frac{\mathrm{d}Q}{\mathrm{d}\mu}, \mathbb{E}\left[\ln \frac{\mathrm{d}P_{\hat{\theta}}}{\mathrm{d}\mu}\right]\right) \\
=& -A^*(\nabla A(\nabla A^*(\mathbb{E}[T(Y)]))) - \mathbb{E}[T(Y)] + \mathbb{B}_A\left[\hat{\theta}\right] \\
&+ d_A\left(\nabla A^*(\mathbb{E}[T(Y)]), \mathbb{E}\left[\hat{\theta}\right]\right) \\
=& n(\theta) + \mathbb{B}_A\left[\hat{\theta}\right] + d_A\left(\theta, \mathbb{E}\left[\hat{\theta}\right]\right)
\end{aligned}$$

which prove the result. □

3.4 Summary

This chapter provided a comprehensive exploration of uncertainty decomposition in the context of predictive models. We focused on a powerful framework known as the general bias-variance decomposition, which leverages proper scoring rules and Bregman divergences to analyze and quantify uncertainty.

We established the necessary mathematical preliminaries, including definitions and theorems related to convex analysis, proper scoring rules, and Bregman divergences. These concepts formed the foundation for understanding the general bias-variance decomposition.

The core result of this chapter was Theorem 10, which provided a general decomposition of the expected score into three components: irreducible error, variance, and bias. This decomposition offered a comprehensive view of uncertainty, applicable to various tasks beyond classification and regression.

Furthermore, we explored a specific case of the decomposition tailored to exponential families. This analysis provided valuable insights into the behavior of the general decomposition and its practical implications for commonly used distributions in machine learning.

By understanding the underlying principles of uncertainty decomposition and the role of proper scores, we can gain a deeper appreciation for the sources of prediction errors and achieve more robust and reliable predictions.

4 Variance Uncertainty

In this chapter, we delve into the intricacies of variance uncertainty, a critical component of the general bias-variance decomposition. As established in the previous chapter, variance uncertainty arises from the model's sensitivity to fluctuations in the training data, leading to overfitting and instability in predictions. We explore practical approaches to quantify and analyze variance uncertainty, drawing upon the theoretical foundations laid out earlier.

Our primary focus is on leveraging Corollary 16, which provides a specific bias-variance decomposition for classification tasks. This corollary offers a direct pathway to compute the Bregman information, a key measure of model variance derived from Bregman divergences. By calculating the Bregman information, we can gain a deeper understanding of the variance component of uncertainty and its impact on predictive performance.

Furthermore, we introduce a revised approach to estimate variance uncertainty using Beta and Dirichlet distributions. This method offers a practical and intuitive way to quantify the variability in model predictions, particularly in classification settings. By examining the spread and shape of these distributions, we can gain valuable insights into the stability and confidence of the model's predictions.

Through a combination of theoretical analysis and practical techniques, this chapter aims to equip you with the tools and knowledge to effectively assess and address variance uncertainty in your machine learning models.

4.1 Bregman Information

In this section we focus on the practical application of the general bias-variance decomposition in the context of classification tasks. Specifically, we employ the negative log-likelihood as the scoring rule, which aligns with our primary interest in classification problems. This choice allows us to derive a concrete and readily applicable measure of variance uncertainty: the Bregman Information.

To establish a clear connection with the theoretical framework, we commence by proving Corollary 16. This corollary provides a tailored bias-variance decomposition for classification scenarios, enabling us to directly compute the Bregman Information from the model's predictions. This information-theoretic quantity, rooted in Bregman divergences, serves as a robust proxy for model variance, capturing the sensitivity of predictions to fluctuations in the training data.

By focusing on the Bregman Information, we gain a practical tool to assess and understand the variance component of uncertainty in classification models. This understanding is crucial for developing more stable and reliable predictions.

4.1.1 Bregman Information Estimator for Classification

Corollary 16 (Bias-Variance decomposition for classification)**.** *For logit prediction, unnormalized scores output by the classifier before applying any activation function like softmax, $\hat{z} \in \mathbb{R}^k$ and target $Y \sim Q$ with*

k classes, we have

$$\underbrace{\mathbb{E}[-\ln(sm(\hat{z})\cdot Y)]}_{\text{Negative log likelihood}} = \underbrace{H(Q)}_{\text{Noise}} + \underbrace{\mathbb{B}_{LSE}[\hat{z}]}_{\text{"Variance"}} + \underbrace{d_{LSE}\left(sm^{-1}(Q), \mathbb{E}[\hat{z}]\right)}_{\text{"Bias"}} \tag{4.1}$$

with the LogSumExp function $LSE(x_1,\dots,x_n) = \ln\sum_{i=1}^{n} e^{x_i}$*, the softmax function* $sm = \nabla LSE$*, and Shannon entropy* H*.*

Proof. The proof is similar to the proof of Corollary 3.6 in the paper from Gruber and Buettner [10], but provide more detailed explanation.
For the logit prediction we can consider a categorical distribution with k classes with the parameters $\{p_1,\dots,p_k\}$, which probability mass function can be written as

$$p(x) = \sum_{i=1}^{k} p_i \mathbf{1}_{[x=i]}$$

Let $\eta = (\eta_1,\dots,\eta_{k-1})^\top$ be the natural parameters, where $\eta_i = \ln\frac{p_i}{p_k}$. we find out that the distribution is from the exponential family

$$p(x) = \exp\left(\eta^\top T(x) - A(\eta)\right)$$

where $T(x) = (\mathbf{1}_{[x=1]},\dots,\mathbf{1}_{[x=k-1]})^\top$ and $A(\eta) = \ln\left(1+\sum_{i=1}^{k-1}\exp\eta_i\right)$. As a result, we can apply Prop. 15. In order to do that we will first find out the convex conjugate for A.

$$A^*(\eta^*) = \sup_{\eta\in\mathbb{R}^{k-1}} \eta^{*\top}\eta - A(\eta)$$

by the fact that the derivative over η within the sup should be 0, we have

$$\eta^* = \nabla A(\eta) = \frac{1}{1+\sum_i^{k-1}\exp\eta_i}(\exp\eta_1,\dots,\exp\eta_{k-1})^\top$$

which implies $\eta_i^* \geq 0$ and $\sum_{i=1}^{k-1}\eta_i^* = 1 - \frac{1}{1+\sum_i^{k-1}\exp\eta_i} = 1 - \frac{1}{\exp A(\eta)}$, we may substitute the expression for η_i in terms of η_i^*

$$\eta_i^* = \frac{\exp\eta_i}{1+\sum_i^{k-1}\exp\eta_i} = \frac{\exp\eta_i}{\exp A(\eta)} \iff \eta_i = \ln\eta_i^* + A(\eta)$$

Then by $\sum_{i=1}^{k-1}\eta_i^* = 1 - \frac{1}{\exp A(\eta)}$, we have

$$\begin{aligned}
A^*(\eta^*) =& \eta^{*\top}\eta - A(\eta) \\
=& \sum_{i=1}^{k-1}\eta_i^*\eta_i - A(\eta) \\
=& \sum_{i=1}^{k-1}\eta_i^*(\ln\eta_i^* + A(\eta)) - A(\eta) \\
=& \sum_{i=1}^{k-1}\eta_i^*\ln\eta_i^* + A(\eta)\sum_{i=1}^{k-1}\eta_i^* - A(\eta) \\
=& \sum_{i=1}^{k-1}\eta_i^*\ln\eta_i^* + A(\eta)\sum_{i=1}^{k-1}\eta_i^* - A(\eta) \\
=& \sum_{i=1}^{k-1}\eta_i^*\ln\eta_i^* + \left(1 - \sum_{i=1}^{k-1}\eta_i^*\right)(-A(\eta)) \\
=& \sum_{i=1}^{k-1}\eta_i^*\ln\eta_i^* + \left(1 - \sum_{i=1}^{k-1}\eta_i^*\right)\ln\left(1 - \sum_{i=1}^{k-1}\eta_i^*\right)
\end{aligned}$$

here we have $\nabla A^*(\eta^*) = \left(\ln\frac{\eta_1^*}{1-\sum_{i=1}^{k-1}\eta_i^*}, \ldots, \ln\frac{\eta_{k-1}^*}{1-\sum_{i=1}^{k-1}\eta_i^*}\right)^\top$.
Further, we define an equivalence class

$$\begin{aligned}
[(\eta, 0)] :=& \left\{z \in \mathbb{R}^k \mid z_1 = \eta_1 + z_k, \ldots, z_{k-1} = \eta_{k-1} + z_k\right\} \\
=& \left\{z \in \mathbb{R}^k \mid \mathrm{sm}\left((\eta_1, \ldots, \eta_{k-1}, 0)^\top\right) = \mathrm{sm}(z)\right\}
\end{aligned}$$

which shows all elements in the equivalent class have the same softmax output. For any $\eta, \hat{\eta} \in \mathbb{R}^{k-1}$, and $z \in [(\eta, 0)]$, $\hat{z} \in [(\hat{\eta}, 0)]$, it holds that

$$\begin{aligned}
\mathbb{B}[\hat{\eta}] =& \mathbb{E}\left[A\left(\hat{\theta}\right)\right] - A\left(\mathbb{E}\left[\hat{\theta}\right]\right) \\
=& \mathbb{E}\left[\ln\left(1 + \sum_{i=1}^{k-1}\exp\hat{\eta}_i\right)\right] - \ln\left(1 + \sum_{i=1}^{k-1}\exp\mathbb{E}\left[\hat{\eta}_i\right]\right) \\
=& \mathbb{E}\left[\ln\left(1 + \sum_{i=1}^{k-1}\exp\hat{\eta}_i\right)\right] - \ln\left(1 + \sum_{i=1}^{k-1}\exp\mathbb{E}\left[\hat{\eta}_i\right]\right) + \mathbb{E}[\hat{z}_k] - \mathbb{E}[\hat{z}_k] \\
=& \mathbb{E}\left[\ln\left(1 + \sum_{i=1}^{k-1}\exp\hat{\eta}_i\right)\right] + \mathbb{E}[\ln\exp\hat{z}_k] - \ln\left(1 + \sum_{i=1}^{k-1}\exp\mathbb{E}\left[\hat{\eta}_i\right]\right) \\
& - \ln\exp\mathbb{E}[\hat{z}_k] \\
=& \mathbb{E}\left[\ln\left(\exp\hat{z}_k + \sum_{i=1}^{k-1}\exp\left(\hat{\eta}_i + \hat{z}_k\right)\right)\right] - \ln\left(\exp\mathbb{E}[\hat{z}_k] + \sum_{i=1}^{k-1}\exp\mathbb{E}\left[\hat{\eta}_i + \hat{z}_k\right]\right) \\
=& \mathbb{E}\left[\ln\sum_{i=1}^{k}\exp\hat{z}_i\right] - \ln\sum_{i=1}^{k}\exp\mathbb{E}\left[\hat{z}_i\right] \\
=& \mathbb{B}_{\mathrm{LSE}}[\hat{z}]
\end{aligned}$$

and

$$\begin{aligned}
d_A(\eta, \mathbb{E}[\hat{\eta}]) =& A(\mathbb{E}[\hat{\eta}]) - A(\eta) - \nabla A(\eta) \cdot (\mathbb{E}[\hat{\eta}] - \eta) \\
=& \ln\left(1 + \sum_{i=1}^{k-1} \exp \mathbb{E}[\hat{\eta}_i]\right) - \ln\left(1 + \sum_{i=1}^{k-1} \exp \eta_i\right) \\
& - \sum_{i=1}^{k-1} \frac{\exp \eta_i}{1 + \sum_{j=1}^{k-1} \exp \eta_j} (\mathbb{E}[\hat{\eta}_i] - \eta_i) \\
=& \ln\left(1 + \sum_{i=1}^{k-1} \exp \mathbb{E}[\hat{z}_i - \hat{z}_k]\right) - \ln \sum_{i=1}^{k} \exp z_i \\
& - \sum_{i=1}^{k-1} \frac{\exp z_i}{\sum_{j=1}^{k} \exp z_j} (\mathbb{E}[\hat{z}_i - \hat{z}_k] - z_i) \\
=& \ln \sum_{i=1}^{k} \exp \mathbb{E}[\hat{z}_i] - \mathbb{E}[\hat{z}_k] - \ln \sum_{i=1}^{k} \exp z_i - \sum_{i=1}^{k} \mathrm{sm}_i(z)(\mathbb{E}[\hat{z}_i] - z_i) \\
& + \underbrace{\sum_{i=1}^{k} \mathrm{sm}_i(z)\, \mathbb{E}[\hat{z}_k]}_{=1} \\
=& \ln \sum_{i=1}^{k} \exp \mathbb{E}[\hat{z}_i] - \ln \sum_{i=1}^{k} \exp z_i - \sum_{i=1}^{k} \mathrm{sm}_i(z)(\mathbb{E}[\hat{z}_i] - z_i) \\
=& \mathrm{LSE}(\mathbb{E}[\hat{z}]) - \mathrm{LSE}(z) - \langle \nabla \mathrm{LSE}(z), \mathbb{E}[\hat{z}] - z \rangle \\
=& d_{\mathrm{LSE}}(z, \mathbb{E}[\hat{z}]).
\end{aligned}$$

For $i \in \{1, \ldots, k\}$, we use the probability mass function $Q_i := \frac{\mathrm{d}Q}{\mathrm{d}\mu}(i)$ of the distribution Q with counting measure μ. Then the noise term follows

$$\begin{aligned}
-A^*(\nabla A(\eta)) =& -A^*\left((Q_1, \ldots, Q_{k-1})^\top\right) \\
=& -\sum_{i=1}^{k-1} Q_i \ln Q_i - \left(1 - \sum_{i=1}^{k-1} Q_i\right) \ln\left(1 - \sum_{i=1}^{k-1} Q_i\right) \\
=& H(Q)
\end{aligned}$$

Let $\mathrm{sm}^{-1}(p) := \left(\ln \frac{p_1}{p_k}, \ldots, \ln \frac{p_{k-1}}{p_k}, 0\right)^\top$ for a probability vector p. Further, let Q have the natural parameter vector η, which gives $Q = \mathrm{sm}(z)$ for $z \in [(\eta, 0)]$ and $\mathrm{sm}^{-1}(Q) \in [(\eta, 0)]$. Using the previous calculation and Prop. 15, then by $Y \sim Q$ and $z \in [(\eta, 0)]$, $\hat{z} \in [(\hat{\eta}, 0)]$

$$\begin{aligned}
\mathbb{E}[-\ln(\mathrm{sm}(\hat{z}) \cdot Y)] &= \mathbb{E}[-\ln p_{\hat{\eta}}(Y)] \\
&\overset{Prop.15}{=} -A^*(\nabla A(\eta)) - \mathbb{E}[T(Y)] + \mathbb{B}_A[\hat{\eta}] + d_A(\eta, \mathbb{E}[\hat{\eta}]) \\
&= H(\mathrm{sm}(z)) - \mathbb{E}[\ln 1] + d_{\mathrm{LSE}}(z, \mathbb{E}[\hat{z}]) + \mathbb{B}_{\mathrm{LSE}}[\hat{z}] \\
&= H(Q) + d_{\mathrm{LSE}}(\mathrm{sm}^{-1}(Q), \mathbb{E}[\hat{z}]) + \mathbb{B}_{\mathrm{LSE}}[\hat{z}]
\end{aligned}$$

□

Let's delve into the practical estimation of the Bregman Information term derived in Corollary 16. Focusing on the term $\mathbb{B}_{\mathrm{LSE}}[\hat{z}] = \mathbb{E}\left[\ln \sum_{i=1}^{k} \exp \hat{z}\right] - \ln \sum_{i=1}^{k} \exp \mathbb{E}\left[\hat{z}\right]$, a straightforward estimator can be obtained by generate n classifiers from the learner, which using the input of randomly generated training datasets and random parameter initialization in each time. This yields the following estimator for Bregman Information:

$$\hat{\mathbb{B}}_{\mathrm{LSE}}[\hat{z}] = \frac{1}{n} \sum_{j=1}^{n} \left(\ln \sum_{i=1}^{k} \exp \hat{z}_i^{(j)} \right) - \ln \sum_{j=1}^{k} \exp \left(\frac{1}{n} \sum_{i=1}^{n} \hat{z}_i^{(j)} \right)$$

where $\hat{z}^{(j)} = \left(\hat{z}_1^{(j)}, \ldots, \hat{z}_k^{(j)}\right)$ represents the logit prediction of the j-th classifier.

To generate these n models, the most straightforward approach involves regenerating data from the underlying distribution n times and training a separate model on each of these datasets. However, in practical settings, we may not have the luxury of generating or collecting multiple datasets.

Instead, we can leverage the bootstrap method to create n distinct training sets from a single dataset. The bootstrap involves randomly sampling data points with replacement from the original dataset to create new datasets of the same size. This allows us to generate multiple models from a single dataset, enabling the estimation of Bregman Information even when access to multiple datasets is limited.

4.1.2 Experimental Evaluation

To illustrate the behavior and effectiveness of the Bregman Information estimator, we conduct several experiments on the toy datasets introduced in the previous chapter. These experiments will showcase how Bregman Information captures variance uncertainty under various data distributions and model complexities.

To provide a comprehensive analysis, we conduct experiments on a variety of datasets and classifiers with different characteristics, including Moons, Gaussian Blobs, Circular, and Ring Blobs as discussed in Section 1.3. Furthermore, we explore the impact of various parameter settings, such as sample size, model complexity, and noise level, on the estimated Bregman Information. This allows us to gain a deeper understanding of how the estimator behaves under different conditions and its effectiveness in capturing variance uncertainty across diverse Learnes.

Overview

We begin by examining the performance of the Bregman Information estimator on the Circular dataset using a variety of classifiers, including 5 Nearest Neighbors, RBF SVM, Gaussian Process, Decision Tree, Random Forest, XGBoost, Naive Bayes, and Neural Net [10].

Figure 4.1 displays the output probability score(the predictions after the softmax function) landscape maps (first row) and the corresponding Bregman Information heatmaps (second row) for various classifiers trained on the Circular dataset. The Bregman Information is estimated by resampling from the data generator, showcasing the variance uncertainty associated with each classifier across the data space.

In most cases, the Bregman Information highlights the decision boundary area, where the probability score changes most abruptly. This is consistent with our expectation, as the decision boundary represents the region of highest uncertainty, where the model is most sensitive to variations in the input data. The different classifiers exhibit varying degrees of smoothness in their Bregman Information heatmaps, reflecting the differences in their decision boundaries and their sensitivity to data variations. The Bregman Information effectively captures the variance uncertainty associated with each classifier, providing a visual representation of the model's confidence in its predictions across the data space.

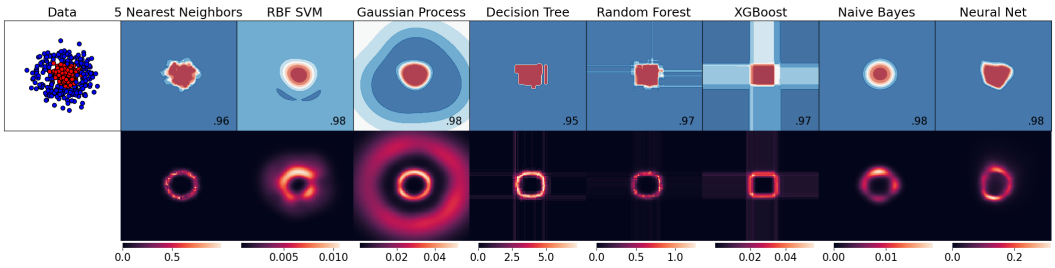


Figure 4.1: Bregman Information for Various Classifiers on the Circular Dataset. This figure displays the output probability score landscape maps (first row) and the corresponding Bregman Information heatmaps (second row) for various classifiers trained on the Circular dataset, the high Bregman infromation score represents the high variance uncertainty. The classifiers include 5 Nearest Neighbors, RBF SVM, Gaussian Process, Decision Tree, Random Forest, XGBoost, Naive Bayes, and Neural Net. The Bregman Information is estimated by resampling from the data generator with training set size of 500 and being resampled 64 times, showcasing the variance uncertainty associated with each classifier across the data space.

It's noteworthy that the ranges of Bregman Information vary significantly across different models, highlighting the strong dependency of this metric on the specific classifier used. For instance, the Gaussian Process model is the only one that approaches a probability score of 0.5 for out-of-distribution (OOD) data, indicating maximum uncertainty. In contrast, most other models assign a high probability score to the blue class for OOD data, effectively classifying OOD points as belonging to the blue class with high confidence. The XGBoost model exhibits noticeable artifacts in its Bregman Information heatmap, likely due to its tree-based structure and sensitivity to data splits.

The different classifiers exhibit varying degrees of smoothness in their Bregman Information heatmaps, reflecting the differences in their decision boundaries and their sensitivity to data variations. The Bregman Information effectively captures the variance uncertainty associated with each classifier, providing a visual representation of the model's confidence in its predictions across the data space.

Sensitivity Analysis

Having gained an overview of the Bregman Information estimator's behavior, we now delve into a more detailed analysis of its performance under various settings. Specifically, we investigate the impact of the following factors:

- Bootstrapping: We investigate the effectiveness of bootstrapping for generating multiple models and its impact on the Bregman Information estimation.
- Resampling Times: We examine the sensitivity of the Bregman Information estimator to the number of times resampling is performed. This helps us determine the appropriate number of resamples needed for reliable uncertainty estimation.
- Class Separation: We examine how the Bregman Information changes as the separation between classes is varied. This helps us understand how the estimator captures uncertainty in situations with varying degrees of class overlap.
- Class Variance: We explore the effect of changing the variance of each class on the estimated Bregman Information. This provides insights into the estimator's sensitivity to the spread of data within each class.

- Training Set Size: We analyze how the size of the training set influences the Bregman Information. This helps us understand the estimator's reliance on the amount of available data.

By systematically exploring these factors, we aim to gain a comprehensive understanding of the Bregman Information estimator's strengths and limitations, providing guidance for its effective application in practice.

Evaluating Bootstrap

To assess the effectiveness of bootstrapping for estimating Bregman Information, we conduct an experiment comparing the results obtained using bootstrapping with those obtained by resampling from the data generator. We select four representative models: 5 Nearest Neighbors, Gaussian Process, XGBoost, and Neural Net.

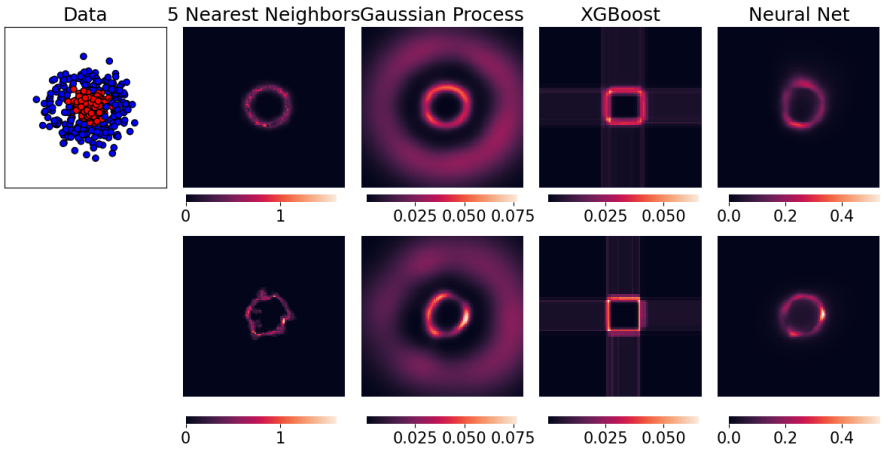


Figure 4.2: Comparison of Bregman Information Estimation using Resampling and Bootstrapping.

Figure 4.2 displays the Bregman Information heatmaps for the four selected models, with the first row showing the results obtained by resampling from the toy generator and the second row showing the results obtained using bootstrapping. All resampling and bootstrapping procedures are performed 64 times. The training set consists of 500 data points.

We observe that bootstrapping performs relatively well in approximating the Bregman Information obtained by resampling from the generator. While the bootstrapped estimates are not as smooth as the true Bregman Information, they still effectively capture the overall patterns and scales of variance uncertainty. As the training set size increases, we expect the bootstrapped estimates to improve further, converging towards the true Bregman Information.

Evaluating Class Separation

To analyze the impact of class separation on the Bregman Information estimator, we generate Circular datasets with varying separation levels (sep) ranging from 0 to 2. We maintain a consistent training set size of 500 points and apply it to a neural network.

Figure 4.3 presents the Bregman Information heatmaps for different class separation levels. As the separation increases from 0 to 2, we observe that the area of high uncertainty, represented by higher Bregman Information values, expands around the decision boundary. This is intuitive, as larger class separation leads to a wider region where the model is less certain about its predictions.

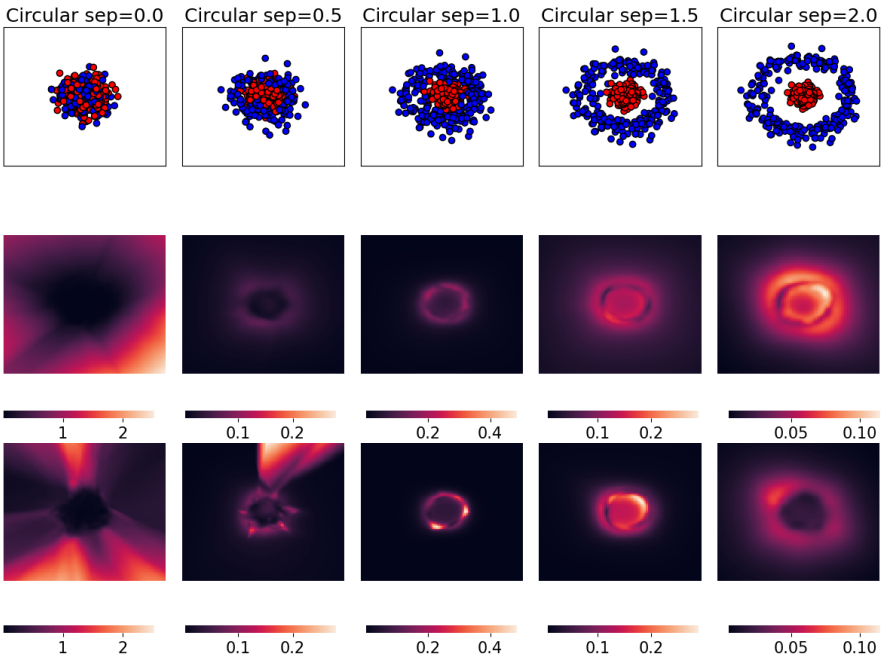


Figure 4.3: Bregman Information for Varying Class Separations.

In the extreme case where the separation is 0, the classes are entirely inseparable, and the model's predictions become essentially random. Consequently, the Bregman Information appears almost uniform across the data space, reflecting the lack of discernible patterns and the high degree of uncertainty in the model's predictions.

Comparing the resampled and bootstrapped plots, we notice some key differences:

- **Performance with limited overlap:** The bootstrapped estimator appears to perform better when there is minimal overlap between the classes. This is likely because bootstrapping relies on resampling from the existing training data, which can be more effective when the classes are well-separated.
- **Impact of training set size:** The performance of the bootstrapped estimator improves with larger training sets, by comparing 0.5 class separation in Figure 4.3 and Figure 4.8. This is because bootstrapping relies on having sufficient data to generate diverse resamples, which can be challenging with smaller datasets.

These observations suggest that the choice between resampling and bootstrapping for estimating Bregman Information depends on the specific characteristics of the dataset and the desired trade-off between accuracy and computational efficiency.

Evaluating Class Variance

We now turn our attention to the impact of class variance on the Bregman Information estimator. We generate Circular datasets with varying variance levels within each class, ranging from 0.2 to 0.6, while maintaining a constant training set size of 500 points and apply it to a neural network.

Figure 4.4 presents the Bregman Information heatmaps for different class variance levels. As the variance

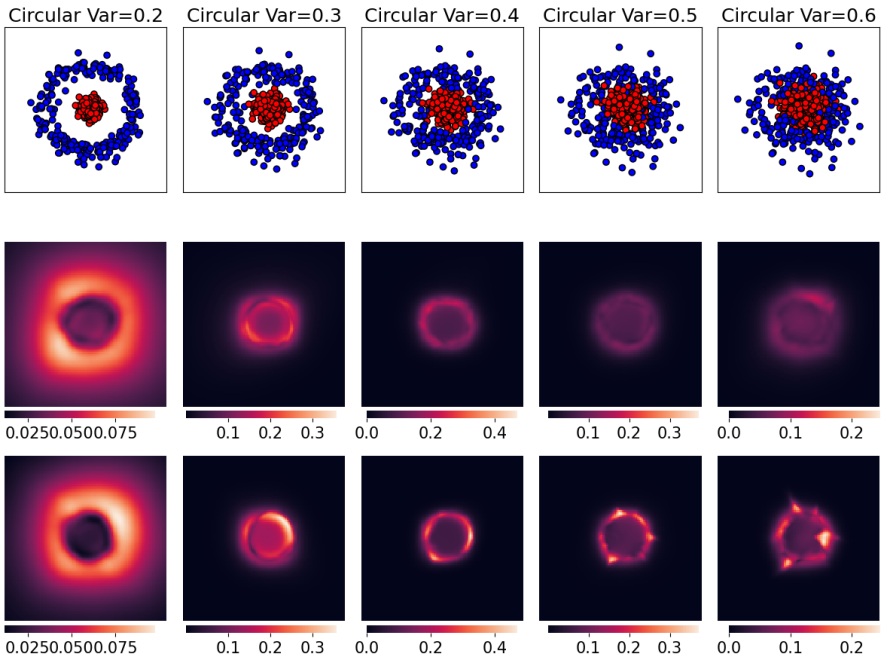


Figure 4.4: Bregman Information for Varying Class Variances.

increases, we observe a noticeable change in the patterns of uncertainty captured by the Bregman Information estimator. With lower variance (0.2 and 0.3), the Bregman Information primarily highlights the decision boundary, indicating that the model's uncertainty is concentrated in the region where the classes are closest to each other. However, as the variance increases (0.4 to 0.6), the high uncertainty regions expand outwards, covering a larger area within each class. This suggests that the model becomes less certain about its predictions even in regions where the data points are further from the decision boundary.

This behavior is consistent with the intuition that higher variance within classes leads to greater overlap and ambiguity between the classes, making it more challenging for the model to accurately classify data points, even those seemingly far from the decision boundary. The Bregman Information estimator effectively captures this increased uncertainty, providing a visual representation of how the model's confidence degrades as the data becomes more dispersed within each class.

Evaluating Training Set Size

Finally, we investigate the impact of training set size on the Bregman Information estimator. We generate Circular datasets with a large number of data points, and analyze the corresponding Bregman Information heatmaps.

Figure 4.5 displays the Bregman Information heatmaps for training set size 10000. As the training set size increases, we observe a distinct pattern emerging in the heatmaps.

Initially, with smaller training sets, the Bregman Information primarily highlights the decision boundary, similar to the patterns observed in previous experiments. However, as the training set size grows, a dark ring appears between the brighter rings in the heatmap. This dark ring corresponds to the region where the probability scores stabilize around 0.5.

This observation suggests that with larger training sets, the model becomes more confident in its pre-

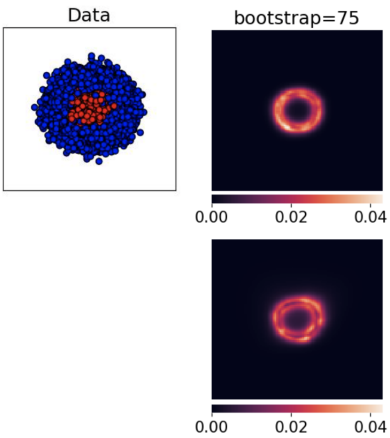


Figure 4.5: Bregman Information for Training Set Size 10000.

dictions near the decision boundary, even though the predicted labels may still fluctuate between the two classes. The Bregman Information estimator captures this phenomenon by showing a decrease in variance uncertainty in the region where the probability scores converge to 0.5.

Bregman Information score on Titanic dataset

In this experiment, we evaluate the effectiveness of Bregman Information for quantifying variance uncertainty on the Titanic dataset. This dataset provides information about the passengers onboard the Titanic, including their demographics, travel details, and survival status. We focus on a classification task where the goal is to predict the ticket class ("Pclass") of a passenger based on the following features:

- Survived: 0 or 1, indicating whether the passenger survived (1) or not (0).
- Sex: 1 for female, 0 for male.
- Age: The age of the passenger in years.
- SibSp: The number of siblings and spouses the passenger had onboard.
- Parch: The number of parents and children the passenger had onboard.
- Embarked: A one-hot encoding representing the port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton). We make them 3 dummy variables.

The target variable, "Pclass," represents the ticket class, with 1 indicating first class, 2 indicating second class, and 3 indicating third class. We masked out passengers younger than 16 years old from the training data and then applied the Bregman Information estimator to quantify the uncertainty associated with these masked data points.

From Figure 4.6, we can observe that both the training data and the masked data points exhibit the Bregman Information scores in a range of roughly from 0 to 0.2. Furthermore, there is no significant difference in the distributions of the scores between the two groups. This suggests that the Bregman Information estimator may not be sensitive enough to detect the uncertainty associated with the masked data points in this high-dimensional setting.

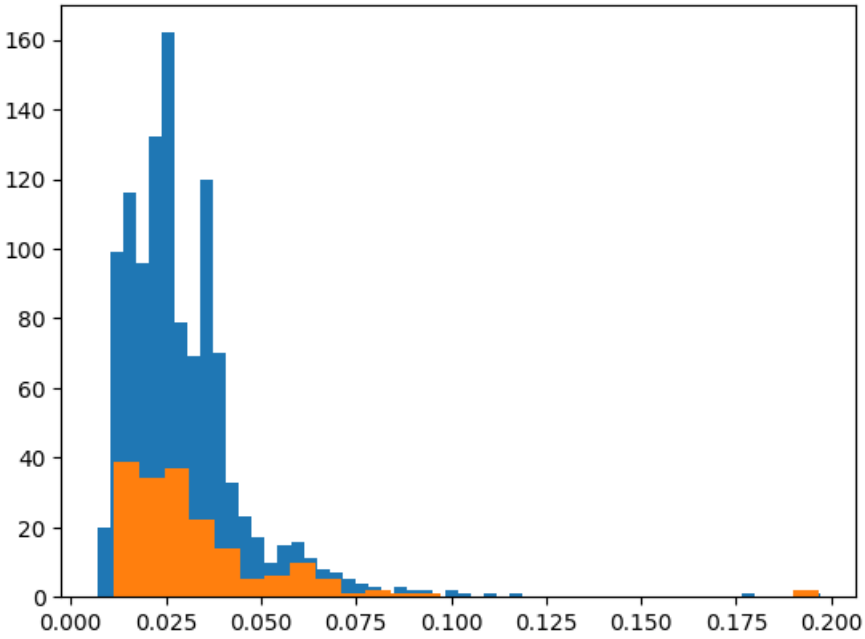


Figure 4.6: Bregman Information for Titanic dataset. This figure shows the distribution of Bregman Information scores for the training data (blue) and the masked data points (orange). The x-axis represents the Bregman Information score, and the y-axis represents the number of data points with that score.

However, this lack of differentiation does not necessarily imply that the method is ineffective. It is possible that the decision boundary in this high-dimensional space is not clearly defined or easily captured by the Bregman Information score. Further investigation is needed to understand the limitations of the method in high-dimensional settings and explore potential modifications to improve its sensitivity and performance.

4.2 Beta and Dirichlet Distribution

While Bregman Information provides a valuable theoretical framework for quantifying variance uncertainty, it also presents certain limitations in practice. As shown in Figure 4.5, a key drawback is that Bregman Information, as defined in the context of classification, solely captures the variance of the probability score generated by the classifier, but overlooks the variance in the corresponding predicted label.

To illustrate this, consider a simple decision rule: if the predicted probability score is greater than 0.5, the assigned label is 1; otherwise, it's 0. Let's compare two learners with two classifiers each:

- Learner 1: classifier 1 predicts 0.49, and classifier 2 predicts 0.51.
- Learner 2: classifier 1 predicts 0.82, and classifier 2 predicts 0.95.

In this context, Bregman Information would indicate higher variance uncertainty for Learner 2 due to the larger spread in probability scores. However, Learner 1 exhibits a more critical difference: the models disagree on the assigned label. This highlights that variance in the probability score alone does not fully

capture the uncertainty relevant to classification. We may be more tolerant of score variance when it is far from the decision boundary (0.5 in this example).

To address this limitation and provide a more nuanced perspective on variance uncertainty, we turn to an alternative approach based on Beta and Dirichlet distributions. These distributions are closely related to the multinomial distribution, which is commonly used to model the probabilities of different outcomes in classification problems. By leveraging the properties of Beta and Dirichlet distributions, we can directly model the variance of the predicted labels, providing a more intuitive and relevant measure of uncertainty in classification settings.

In both binary and multi-class classification cases, we utilize these distributions to model the probability scores generated by the model at each data point. For binary classification, the Beta distribution captures the variability in the predicted probability across different bootstrap samples or models. For multi-class problems, the Dirichlet distribution, a generalization of the Beta distribution, captures the variability in the probability vector generated by the model. By analyzing the spread and shape of these distributions, we can assess the model's confidence and the degree of uncertainty associated with its predictions.

This section delve into the details of using Beta and Dirichlet distributions to estimate variance uncertainty. We explore their theoretical foundations, practical implementation, and interpretation in the context of classification tasks. This approach offers a complementary perspective to Bregman Information, providing a more computationally efficient and intuitive way to quantify and analyze variance uncertainty, with a direct focus on the variance of the predicted labels.

4.2.1 Beta and Dirichlet Estimator

To address the limitations of Bregman Information and capture the variance of predicted labels directly, we introduce an alternative approach based on Beta and Dirichlet distributions. These distributions offer a practical and intuitive way to model the variability in model predictions for both binary and multi-class classification tasks.

Binary Classification:

In binary classification, we often employ a threshold t to determine the predicted label based on the probability score s_k generated by the model for a data point x_k:

$$label_k = \mathbf{1}[s_k \geq t]$$

Due to the randomness in the sampling process or model training, the probability score s_k can be considered a random variable drawn from a distribution $\mathscr{S}_k$, which is a distribution on $[0,1]$:

$$s_k \sim \mathscr{S}_k$$

We define p_k as the probability that the score s_k exceeds the threshold t:

$$p_k^t = P(s_k \geq t)$$

Generate n models in the same way in section 4.1.1 and pick a data point x_k, the number of models that predict a label of 1 for x_k, denoted by L, follows a binomial distribution: $L = \#(label_k^{(n)} = 1) \sim Binom(n, p_k^t)$. Since p_k^t is unknown, we aim to estimate it using a Bayesian approach.

We introduce a prior distribution for p_k^t, denoted by $\lambda_k \sim Beta(\alpha, \beta)$. The posterior distribution of p_k^t given the observed labels L can be derived as follows:

$$\begin{aligned}\pi(p_k^t \mid L = m) &\propto P(L = m \mid p_k^t) \cdot \pi(p_k^t) \\ &\propto (p_k^t)^m (1 - p_k^t)^{n-m} (p_k^t)^{\alpha-1} (1 - p_k^t)^{\beta-1} \\ &= (p_k^t)^{m+\alpha-1} (1 - p_k^t)^{n-m+\beta-1}\end{aligned}$$

This shows that the posterior distribution of λ_k given $L = m$ is also a Beta distribution: $(\lambda_k \mid L = m) \sim Beta(\alpha + m, n - m + \beta)$. We can then estimate p_k^t by the expected value of the posterior distribution: $\hat{p}_k^t = \mathbb{E}[\lambda_k \mid L = m] = \frac{\alpha+m}{\alpha+\beta+n}$.

To evaluate the prediction uncertainty at x_k, we use the following measure:

$$\sigma_k^t = \min[\hat{p}_k^t, 1 - \hat{p}_k^t] = Prob(\text{incorrect label for } x_k)$$

This measure reflects the probability of assigning the wrong label to x_k, providing an estimate of the model's uncertainty at that point.

Multi-class Classification:

For multi-class classification with c classes, we extend this approach using the Dirichlet distribution. At each point x_k, the model generates a probability score vector $\mathbf{s}^k = (s_1^k, \ldots, s_c^k)$. In multi-class setting, since there can be a situation that the probabilities of several classes all greater than the threshold, we abandon the threshold here. Then the predicted label is determined by the index of the highest probability score:

$$label_k = \sum_{i=1}^{c} i \cdot \mathbf{1}[s_i^k = \max_j \{s_j^k\}]$$

Similar to the binary case, we assume that the probability score vector $\mathbf{s}^k$ follows a distribution $\mathscr{S}^k$, which is a distribution on $[0,1]^C$:

$$\mathbf{s}^k \sim \mathscr{S}^k$$

We define the probability vector $P^k = (p_1^k, \ldots, p_c^k)$, where p_i^k represents the probability that the i-th class has the highest probability score:

$$p_i^k = P(s_i^k = \max_j \{s_j^k\})$$

Generate n models in the same way in section 4.1.1 and pick a data point x_k, the label counts for each class at x_k can be represented by the vector $L = (l_1, \ldots, l_c)$, where $l_i = \#(label_k^{(n)} = i)$. This vector follows a multinomial distribution. To estimate the unknown probabilities P^k, we introduce a Dirichlet prior: $\lambda^k \sim Dirichlet(\alpha)$, where $\alpha = (\alpha_1, \ldots, \alpha_c)$.

The posterior distribution of λ^k given the observed label counts L is then:

$$\begin{aligned}\pi(P^k \mid L = \tilde{L}) &\propto P(L = \tilde{L} \mid P^k) \cdot \pi(P^k) \\ &\propto \prod_{i=1}^{c} (p_i^k)^{\tilde{l}_i} \cdot \prod_{i=1}^{c} (p_i^k)^{\alpha_i - 1} \\ &= \prod_{i=1}^{c} (p_i^k)^{\tilde{l}_i + \alpha_i - 1}\end{aligned}$$

This demonstrates that the posterior distribution is also a Dirichlet distribution: $(\lambda^k \mid L = \tilde{L}) \sim Dirichlet(\alpha + \tilde{L})$. We can estimate the probabilities p_i^k using the expected value of the posterior distribution: $\hat{p}_i^k = \frac{\alpha_i + \tilde{l}_i}{\sum_j \{\alpha_j + \tilde{l}_j\}}$.

Finally, we quantify the prediction uncertainty at x_k using:

$$\sigma_k = 1 - \max_i \{\hat{p}_i^k\} = Prob(\text{incorrect label for } x_k)$$

This measure represents the probability of assigning an incorrect label to x_k, providing a direct assessment of the model's uncertainty in the multi-class setting.

4.2.2 Experimental Evaluation

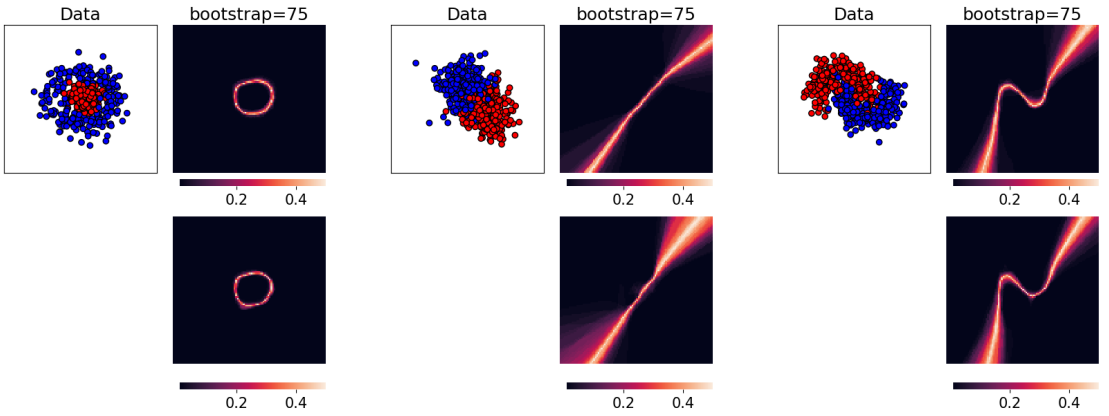
To demonstrate the effectiveness of the Beta and Dirichlet estimators in capturing variance uncertainty, we conduct several experiments on the toy datasets introduced earlier. These experiments will showcase how these estimators capture uncertainty in various settings, particularly highlighting their ability to reflect the variance of the predicted labels.

Binary classification

To demonstrate the effectiveness of the Beta estimator in capturing variance uncertainty for binary classification, we conduct experiments on three toy datasets: Moons, Gaussian Blobs, and a Circular dataset with the size of 500 points. These experiments showcase how the Beta estimator captures uncertainty in various settings, particularly highlighting its ability to reflect the variance of the predicted labels while being more tolerant to probability score variance far from the decision boundary in this 0.5 threshold setting.

Across all three datasets, the Beta estimator effectively captures the variance uncertainty in the model's predictions. Notably, the estimator demonstrates a greater tolerance for variance in probability scores when they are far from the decision boundary. This is evident in the heatmaps, where the uncertainty scores are lower in regions where the probability scores are confidently close to 0 or 1, even if there is some variance among the bootstrap predictions. This behavior aligns with the Beta estimator's ability to directly model the variance of the predicted labels, rather than solely focusing on the variance of the probability scores. Also, by construction, the score are scaled to the range of $[0, 0.5]$, which makes it easier to be compared across datasets.

Furthermore, the bootstrapping approach used to generate multiple models for the Beta estimator proves to be effective in approximating the true variance uncertainty. Despite using only 75 bootstrap samples, the estimator successfully captures the overall patterns of uncertainty across the different datasets. This



(a) Circular (b) Gaussian Blobs (c) Moons

Figure 4.7: Beta Estimator for Variance Uncertainty in Binary Classification. This figure displays the Beta estimator's variance uncertainty estimates for three different binary classification datasets, all using a Neural Net classifier. The first row shows the true data distribution, while the second row presents the corresponding heatmaps of the Beta estimator's uncertainty scores, obtained using 75 bootstrap samples.

highlights the practicality and efficiency of the Beta estimator in quantifying variance uncertainty for binary classification tasks.

Beta Estimator with Large Training Set

To further investigate the behavior of the Beta estimator, we conduct an experiment with a significantly larger training set size of 10,000 data points on the Circular dataset. This experiment aims to analyze the impact of a dense dataset on the Beta estimator's uncertainty estimates.

In contrast to the previous experiment with a smaller training set (Figure 2.6), we observe that the dark ring, which previously appeared in regions where the probability scores stabilized around 0.5, disappears when using a large training set. This indicates that even with low variance in probability scores near the decision boundary, the model still exhibits label flips, contributing to higher uncertainty.

This observation highlights the Beta estimator's ability to capture uncertainty stemming from label flips, even when the probability scores appear stable. The estimator's sensitivity to label variations, rather than solely relying on probability score variance, ensures a more practical representation of the model's uncertainty.

Furthermore, the Beta estimator continues to demonstrate its "slim" characteristic, with lower uncertainty scores in regions where the probability scores are confidently close to 0 or 1. This behavior is particularly pronounced in the Circular dataset with a large training set, as the model achieves high confidence and stability in its predictions due to the dense data representation.

Multi-class classification: Dirichlet score on Titanic dataset

To evaluate the performance of the Dirichlet score on a real-world dataset, we conducted an experiment using the Titanic dataset, with the same setting in section 4.1.2. In this multi-class setting, we focused on predicting the "Pclass" feature, which represents the passenger class (1st, 2nd, or 3rd), based on other

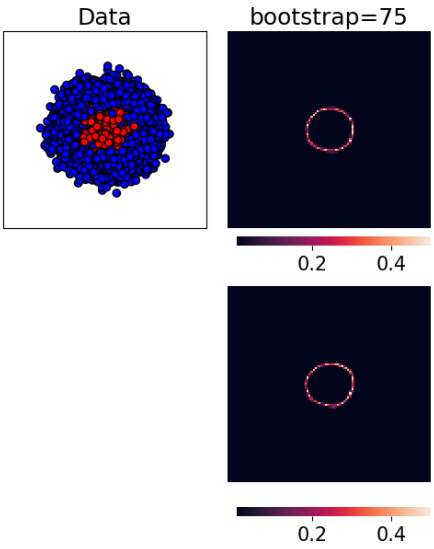


Figure 4.8: Beta Estimator with Large Training Set on Circular Dataset. This figure displays the Beta estimator's variance uncertainty estimates for the Circular dataset with 10,000 training points, using a Neural Net classifier and 75 bootstrap samples. The first row shows the true data distribution, while the second row presents the corresponding heatmap of the Beta estimator's uncertainty scores.

passenger characteristics. We masked out passengers younger than 16 years old from the training data and applied the Dirichlet score estimator to quantify the uncertainty associated with these masked data points.

From Figure 4.9, we can observe a distinction between the Dirichlet score distributions for the training data and the masked data points. The training data exhibits a high concentration of scores near 0, indicating high confidence in the predictions. In contrast, the masked data points show a more spread-out distribution with a significant portion of scores closer to 0.5, reflecting higher uncertainty in the predictions for these under-represented individuals.

This difference in distributions is more pronounced than what we observed with the Bregman Information score in Figure 4.6, suggesting that the Dirichlet score is more sensitive to the uncertainty associated with data points that are not well-represented in the training data. However, the difference is still not substantial, which could be attributed to the difficulty in precisely identifying the decision boundary in this high-dimensional setting. Without prior knowledge of the decision boundary's location, it is challenging to definitively assess the sensitivity of the Dirichlet score in capturing the uncertainty associated with specific data points.

Despite this limitation, the observation that the Dirichlet score exhibits greater differentiation between the training data and the masked data points compared to the Bregman Information score provides evidence for its increased sensitivity in capturing uncertainty. This finding supports the potential of the Dirichlet score as a valuable tool for quantifying uncertainty in real-world scenarios with potential biases or under-representation of certain demographics.

This observation suggests that the Dirichlet score can effectively capture the uncertainty associated with data points that are not well-represented in the training data, even in a multi-class setting. By providing a probability-based measure of uncertainty, the Dirichlet score offers a more intuitive and interpretable

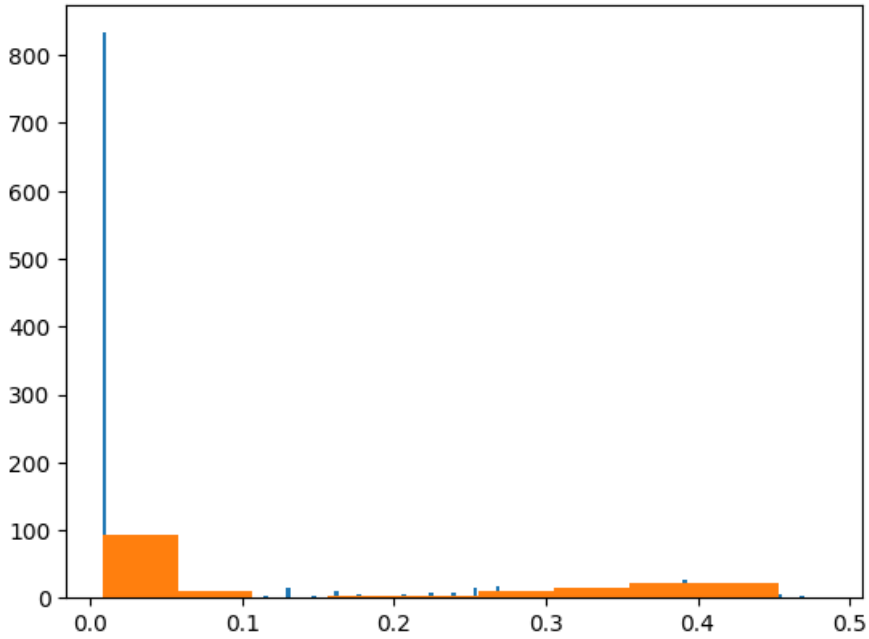


Figure 4.9: Dirichlet Scores on Titanic Dataset. This figure shows the distribution of Dirichlet scores for the training data (blue) and the masked data points (orange). The x-axis represents the Dirichlet score, indicating the estimated probability of the classifier making an incorrect prediction. The y-axis represents the number of data points with that score.

way to assess the reliability of predictions for individual data points, particularly in real-world scenarios with potential biases or under-representation of certain demographics.

5 Bias Uncertainty

In this chapter, we shift our focus to bias uncertainty, a critical component of the general bias-variance decomposition that arises from the model's inability to accurately capture the true underlying relationship in the data. As discussed in previous chapters, bias uncertainty reflects the systematic error of the model, leading to consistent deviations from the true predictions. We will explore practical approaches to assess and analyze bias uncertainty, drawing upon the theoretical foundations laid out earlier.

Our starting point is the bias term in Theorem 10, which is represented by the Bregman divergence between the model's prediction and the true distribution. However, in real-world scenarios, the true distribution remains unknown, and we often only have access to the feature vectors of new data points without label information, on which a trained model will be applied. To address this challenge, we leverage the model we have already trained and utilize the feature information of new data to perform out-of-distribution (OOD) detection.

The rationale behind this approach is that if a new, in-distribution data point is identified as an OOD sample, it suggests that the model may not have accurately captured the underlying true distribution in that region. This indicates a potential bias, arising from the model's lack of exposure to sufficient data in that area, and highlights the risk of systematic errors in the model's predictions. It's important to emphasize that we're primarily interested in ID data points located in sparse regions of the data distribution, rather than truly OOD data points, but the OOD detection can still help with it.

We will explore two distinct methods for assessing bias uncertainty through the lens of OOD detection: GradNorm[18] and TracIn[17]. GradNorm utilizes information extracted from the gradient space to identify OOD samples. TracIn, on the other hand, employs a follow-up training approach, where the model is keep trained with the inclusion of the potential OOD sample. By analyzing the impact of this new sample on the model's performance, TracIn can effectively gauge the bias uncertainty associated with the new data point.

By exploring these two complementary methods, we aim to provide a more comprehensive understanding of bias uncertainty and its implications for predictive modeling. These techniques offer valuable tools for identifying potential biases in the model and developing more robust and reliable prediction systems.

5.1 GradNorm

To assess bias uncertainty, we leverage out-of-distribution (OOD) detection methods. The underlying principle is that if a new data point is identified as an OOD sample, it suggests the model may not have accurately captured the true underlying data distribution, indicating potential bias. One effective OOD detection method is GradNorm, which utilizes information extracted from the gradient space.

5.1.1 Methodology

GradNorm leverages the observation that the magnitude of gradients is typically higher for in-distribution (ID) data compared to OOD data. This difference in gradient magnitudes can be exploited to effectively

distinguish between the two. We start with KL divergence.

Formally, the KL divergence is defined as:

$$D_{KL}(p||q) = \sum_i p_i \log \frac{p_i}{q_i} = -\sum_i p_i \log\ q_i + \sum_i p_i \log\ p_i = H(p,q) - H(p),$$

where $p = (p_1, \ldots, p_C)$ is the reference distribution (uniform distribution in this case), and $q = (q_1, \ldots, q_C)$ is the model-predicted distribution.

The idea here is that we can compare the model's output with a no-information vector $u = (\frac{1}{C}, \ldots, \frac{1}{C}) \in \mathbb{R}^C$, normally a uniform probability vector, which can be interpret as the information at that point. This can be further expressed as:

$$D_{KL}(u||\text{softmax}(f(x)) = -\frac{1}{C}\sum_{c=1}^{C} \log \frac{e^{f_c(x)/T}}{\sum_{j=1}^{C} e^{f_j(x)/T}} - H(u),$$

where the first term represents the cross-entropy loss between the softmax output and a uniform vector u, and the second term $H(u)$ is a constant.

The cross-entropy loss measures the difference between two probability distributions. In GradNorm, it quantifies the difference between the predicted probability distribution and the uniform distribution.

The cross-entropy loss is formally defined as:

$$\mathscr{L}_{CE}(f(x), y) = -\log \frac{e^{f_y(x)/T}}{\sum_{e=1}^{C} e^{f_c(x)/T}}.$$

We then backpropagate this KL divergence through the network to calculate the gradients with respect to the model parameters. The gradient of the KL divergence with respect to a model parameter w is:

$$\frac{\partial D_{KL}(u||\text{softmax}(f(x))}{\partial w} = \frac{1}{C}\sum_{i=1}^{C} \frac{\partial \mathscr{L}_{CE}(f(x), i)}{\partial w}.$$

This gradient is equivalent to averaging the derivative of the categorical cross-entropy loss for all labels.

Finally, we compute the vector norm of these gradients to obtain the OOD-score:

$$S(x) = \| \frac{\partial D_{KL}(u||\text{softmax}(f(x))}{\partial w} \|_p.$$

A larger gradient norm suggests that the input is more likely to be in-distribution, while a smaller norm indicates a higher likelihood of being OOD [18]. This because that for OOD inputs, the model is uncertain about the correct class. Ideally, the softmax output should have similar probabilities for all classes, closer to a uniform distribution. This leads to a smaller KL divergence from the uniform distribution, resulting in smaller gradients during backpropagation. By analyzing the distribution of OOD scores for known in-distribution data, a threshold can be determined to classify new samples as either ID or OOD.

To assess bias uncertainty, we leverage out-of-distribution (OOD) detection methods. The underlying principle is that if a new data point is identified as an OOD sample, it suggests the model may not have

accurately captured the true underlying data distribution, indicating potential bias. However, a crucial caveat exists: we are aim to find the OOD of the underlying true distribution. Since we already suspect a potential bias between the model and the true distribution, what the model identifies as in-distribution (ID) might still be OOD with respect to the true data distribution.

From a new perspective, a larger gradient norm implies a greater degree of information change within the model upon encountering this new data point. This suggests a higher risk that the model's information at this point is insufficient, potentially leading to a greater bias. Therefore, in our setting, a larger gradient norm can be interpreted as an indicator of bias uncertainty.

5.1.2 Experimental Evaluation

To evaluate the effectiveness of GradNorm in assessing bias uncertainty, we conduct experiments on the toy datasets and real-world datasets introduced earlier. These experiments aim to demonstrate how GradNorm captures the bias uncertainty associated with new data points, particularly highlighting its ability to identify out-of-distribution samples.

GradNorm on Classic Toy Datasets

We begin by evaluating the performance of GradNorm on the toy datasets: Moons and Gaussian Blobs (Figure 5.1). For each dataset, we train a neural network model and analyze the GradNorm scores for different data points.

Comparing the two GradNorm heatmaps in each subplot, we observe that using only the last layer's weights for gradient calculation yields better results. This is likely because the weights of the shallow layers do not change significantly after a few rounds of training, while the deeper layers, especially the last layer, are more sensitive to changes in the input data and thus better reflect the model's response to new data points.

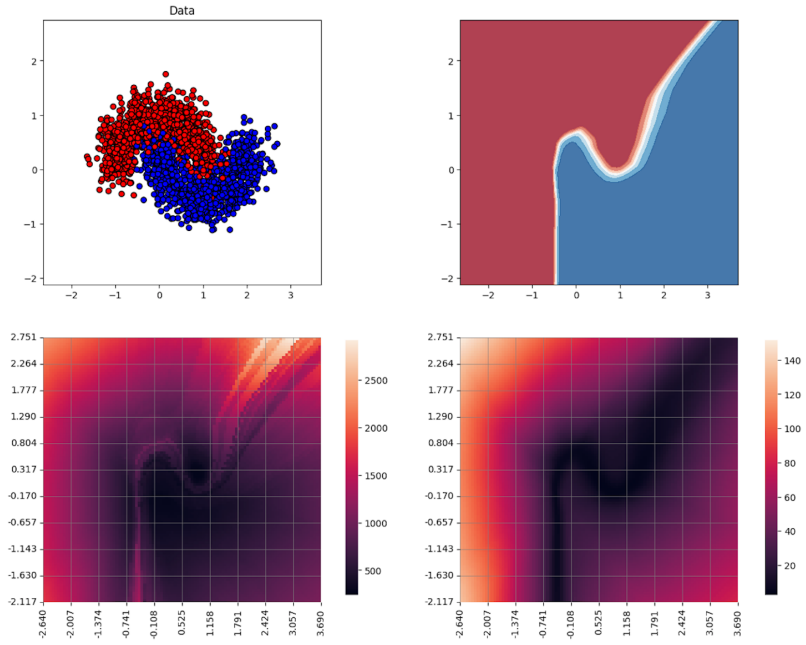
Focusing on the GradNorm heatmaps generated using the last layer's weights, we see a clear pattern. The area near the training data exhibits low GradNorm scores (darker regions), indicating that these regions are considered in-distribution by the model. As we move further away from the training data, the GradNorm scores increase (brighter regions), suggesting a higher likelihood of those regions being out-of-distribution. However, the decision boundary consistently shows low GradNorm scores, regardless of its distance from the training data. This is because adding a data point near the decision boundary does not significantly alter the model's predictions or confidence, as the model is already certain that the probability score is almost 0.5 for both classes there.

These observations demonstrate that GradNorm effectively captures the bias uncertainty associated with different regions in the data space. The method successfully identifies out-of-distribution areas while failed to recognize the uncertainty near the decision boundary. Still, this provides valuable insights into the model's behavior and its potential biases, aiding in the assessment of model reliability.

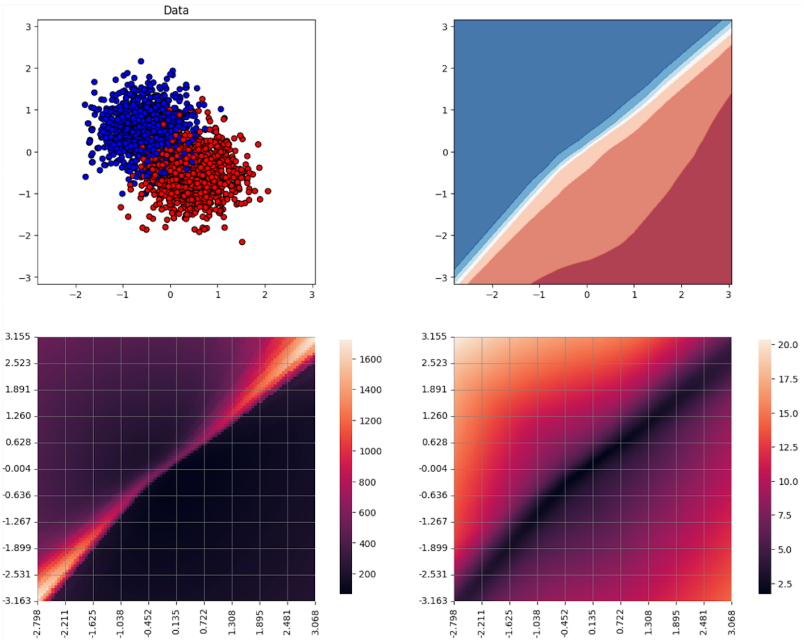
GradNorm on Toy Datasets with Large Class Separation

To further analyze the behavior of GradNorm, we investigate its performance on the Moons dataset with varying levels of class separation. This experiment aims to understand how GradNorm captures bias uncertainty when the distinction between classes becomes more or less pronounced.

As the class separation increases (Figure 5.2), we observe that the GradNorm score heatmaps, particularly those generated using the last layer's weights, become more unstable. This instability manifests as a

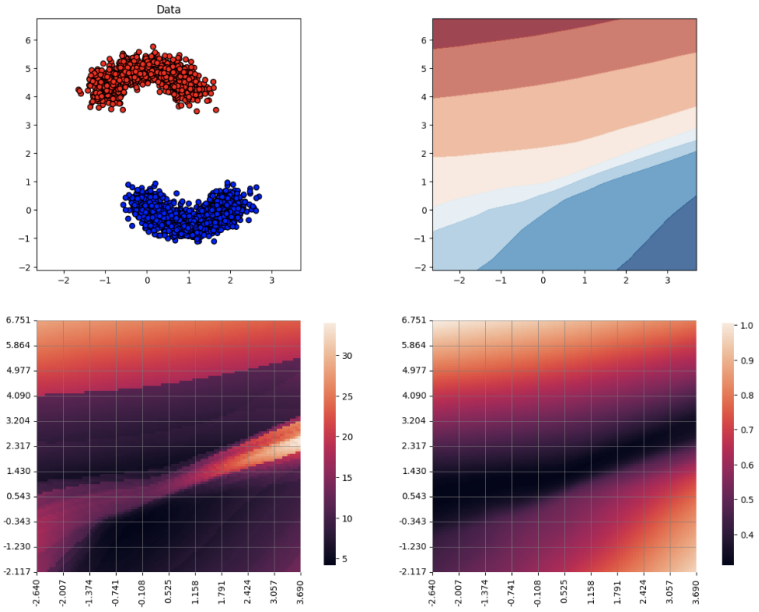


(a) Moons

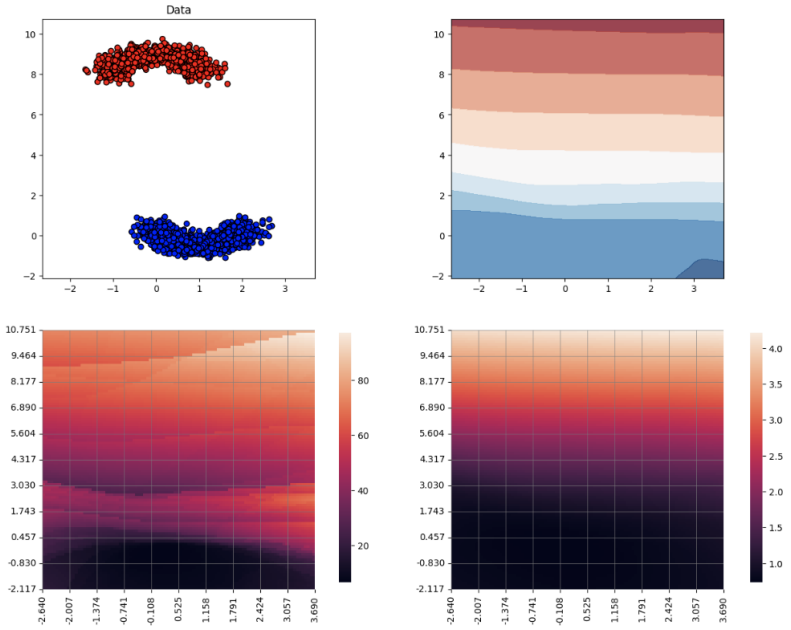


(b) Gaussian Blobs

Figure 5.1: GradNorm Visualization for Moons and Gaussian Blobs Datasets. This figure presents two subplots, for the Moons dataset and the Gaussian Blobs dataset. Each subplot displays the training data as a scatter plot in the top left, and the probability score landscape map in the top right. The bottom row of each subplot shows GradNorm score heatmaps, with the bottom left using gradients of all weights in the neural network, and the bottom right using gradients of only the last layer's weights. This allows for a direct comparison of the impact of using different sets of weights for calculating the GradNorm scores. Note that the high GradNorm represents high bias uncertainty.



(a) Class separation = 6



(b) Class separation = 8

Figure 5.2: GradNorm Visualization for Moons Dataset with Large Class Separations. This figure presents two subplots, each corresponding to a different class separation level in the Moons dataset. The layout of each subplot is identical to Figure 5.1, displaying the training data, probability score landscape map, and GradNorm score heatmaps using all weights and only the last layer's weights.

less clear distinction between in-distribution and out-of-distribution regions, with more fluctuations and variations in the GradNorm scores across the data space.

This observation can be attributed to the increased instability of the decision boundary as the class separation increases. With a larger separation between classes, the model's decision boundary becomes more sensitive to variations in the training data, leading to greater fluctuations in its shape and position. This instability in the decision boundary is reflected in the GradNorm scores, which become less consistent and more sensitive to the specific data points used for training.

Similar to the observations on the Moons dataset, we find that as the class separation increases in the RingBlobs dataset, the GradNorm score heatmaps(Figure 5.3), particularly those generated using the last layer's weights, become more unstable. This instability is again reflected in the less clear distinction between in-distribution and out-of-distribution regions and the increased fluctuations in the GradNorm scores.

Furthermore, the GradNorm heatmaps exhibit a strong resemblance to the probability score maps, indicating a high dependence of GradNorm on the model itself. This dependence raises concerns about the reliability of GradNorm for bias uncertainty estimation, as we aim to assess the bias between the model and the true distribution, rather than simply reflecting the model's own predictions.

Despite these limitations, GradNorm still offers valuable insights into the model's behavior and potential biases. The heatmaps generally show increasing uncertainty as we move further away from the training data, even with varying class separations. This information can be used to identify regions where the model is more prone to bias and guide further model improvement or data collection efforts. However, the observed dependence on the model's predictions highlights the need for further refinement and improvement of GradNorm or the exploration of alternative methods for a more robust and independent assessment of bias uncertainty.

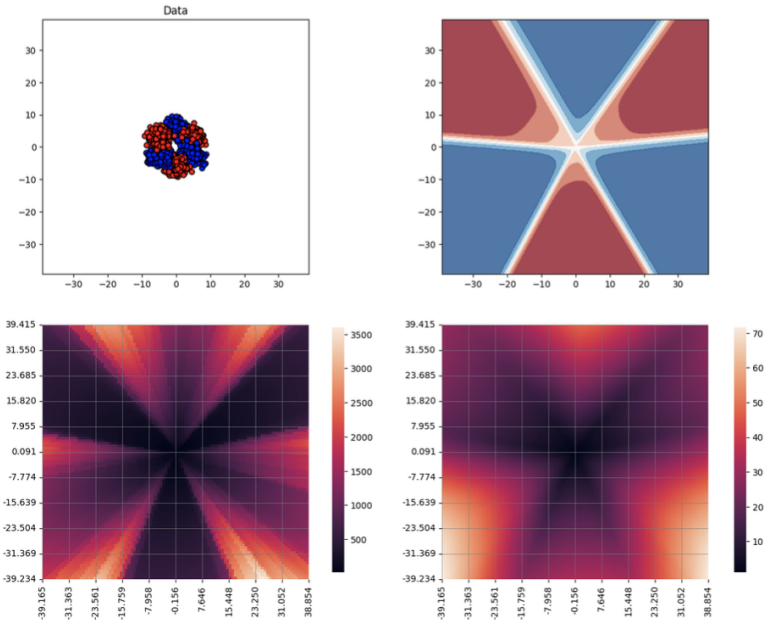
5.2 TracIn

Here we explore an alternative approach, TracIn, a method that leverages follow-up training to quantify bias uncertainty.

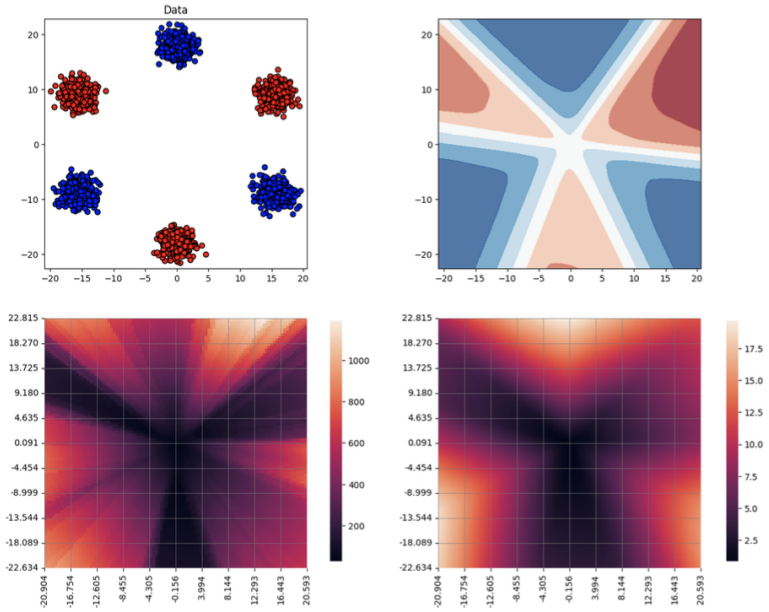
TracIn addresses the fundamental question: "How much information do we need to gain to correct a mistake made by the model?" Since the true label for a new data point is unknown, TracIn takes a comprehensive approach by averaging over all possible labels. This involves iterating over each potential label, assuming it to be the true label, and then follow-up training the model with this new information.

The key idea is that the amount of information required to "correct" the model's prediction reflects the degree of bias between the model and the true distribution. If a significant amount of information is needed to align the model's prediction with a particular label, it suggests a higher likelihood of bias associated with that label. Conversely, if the model readily adapts to a label with minimal information gain, it indicates a lower likelihood of bias.

TracIn offers a unique perspective on bias uncertainty by quantifying the information gain required to correct potential prediction errors. This approach provides a complementary view to GradNorm, further enriching our understanding of bias uncertainty and its implications for model reliability.



(a) Class separation = 5



(b) Class separation = 15

Figure 5.3: GradNorm Visualization for RingBlobs Dataset with Varying Class Separations. This figure presents two subplots, each corresponding to a different class separation level in the RingBlobs dataset. The layout of each subplot is identical to Figure 5.1, displaying the training data, probability score landscape map, and GradNorm score heatmaps using all weights and only the last layer's weights.

5.2.1 Methodology

To delve into the theoretical foundation of TracIn, we first introduce the concept of mutual information, a fundamental measure in information theory that quantifies the amount of information shared between two random variables.

Mutual information (MI) measures the mutual dependence between two random variables [25]. More specifically, it quantifies the 'amount of information' obtained about one random variable by observing the other random variable.

In essence, mutual information captures how much knowing one variable reduces uncertainty about the other. A higher mutual information indicates a stronger relationship between the variables, implying that observing one provides more information about the other.

Mathematically, the mutual information between two discrete random variables X and Y is defined as:

$$I(X;Y) = H(X) - H(X \mid Y)$$

where $H(X)$ is the is the individual entropy of X, and $H(X \mid Y)$ the conditional entropy of X given Y respectively.

TracIn leverages mutual information to quantify the information gain achieved by follow-up training the model with the inclusion of a new data point and its assumed label. This information gain reflects the reduction in uncertainty about the model's prediction after incorporating the new information. By analyzing this information gain across all possible labels, TracIn provides a comprehensive assessment of the bias uncertainty associated with the new data point.

In our specific case, we consider X as the probability distribution of predicted labels ($\hat{l}$) and Y as the probability distribution of true labels (l). The mutual information between these two distributions can be expressed as:

$$\begin{aligned}
I(\hat{l},l) &= H(\hat{l}) - H(\hat{l} \mid l) \\
&= -\sum_{i=1}^{C} p_{\hat{L}}(i) \log p_{\hat{L}}(i) - \left(-\sum_{j=1}^{C} p_L(j) H(\hat{l} \mid l = j) \right) \\
&= -\sum_{i=1}^{C} p_{\hat{L}}(i) \log p_{\hat{L}}(i) - \left(-\sum_{j=1}^{C} p_L(j) \sum_{i=1}^{C} p_{\hat{L}|L}(i \mid l = j) \log p_{\hat{L}|L}(i \mid l = j) \right) \\
&= -\left(\sum_{j=1}^{C} p_L(j) \left(\sum_{i=1}^{C} p_{\hat{L}}(i) \log p_{\hat{L}}(i) - \sum_{i=1}^{C} p_{\hat{L}|L}(i \mid l = j) \log p_{\hat{L}|L}(i \mid l = j) \right) \right)
\end{aligned}$$

This formulation captures the reduction in uncertainty about the predicted labels ($\hat{l}$) given the knowledge of the true labels (l). By analyzing this mutual information, TracIn provides a measure of the information gain achieved by follow-up training the model with the assumed true label, ultimately reflecting the bias uncertainty associated with the new data point.

Since we don't know the true label for the new data point, we can estimate the information gain by averaging over all possible labels, effectively assuming a uniform distribution for the true label: $p_L(j) = \frac{1}{C}$, where C is the number of classes. This simplifies the calculation of the mutual information.

The remaining challenge lies in estimating the conditional probability $p_{\hat{L}|L}(i \mid l = j)$, which represents the probability of predicting label i given that the true label is j. This conditional probability can be obtained by follow-up training the model with the new data point and its assumed label j, and then observing the model's prediction. By repeating this process for each possible label j, we can estimate the conditional probabilities and ultimately compute the average information gain, providing a comprehensive measure of bias uncertainty. Following, we would like to explore how can we set the follow-up training procedure to achieve a good performance and make it practical.

5.2.2 TracIn on Toy generators

To evaluate the effectiveness of TracIn in quantifying bias uncertainty, we conduct experiments on the toy datasets introduced earlier. These experiments aim to demonstrate how TracIn captures the information gain necessary to correct potential model errors, reflecting the degree of bias between the model and the true distribution.

As previously discussed, TracIn involves follow-up training the model with the inclusion of a new data point and its assumed label to estimate the information gain. This follow-up training process, however, requires careful consideration of several factors that can significantly influence the estimated bias uncertainty.

Specifically, we need to address the following questions regarding the follow-up training procedure:

- Batch Size: When adding the new data point to the training set for follow-up training , how many data points from the original training set should be included in each batch? This determines the balance between the influence of the new data point and the original training data on the model's updated parameters.
- Number of Iterations: How many iterations or epochs of follow-up training should be performed? This determines the extent to which the model is allowed to adapt to the new data point and its assumed label.
- Learning Rate: What learning rate should be used during the follow-up training process? This controls the step size taken by the optimizer in updating the model's parameters based on the new information.
- Weighting the New Data Point: Should we assign a higher weight to the new data point compared to the data points from the original training set? This can influence the degree to which the model prioritizes fitting the new data point versus retaining its knowledge from the original training data.

By carefully considering and experimenting with these factors, we can optimize the follow-up training process to ensure a reliable and informative estimation of bias uncertainty using TracIn. The following section presents our experimental setup and analysis, exploring the impact of these factors on TracIn's performance.

TracIn with Single Iteration Retraining on Toy Generators

We begin our exploration of TracIn by considering the simplest retraining scenario: using only the new data point and training for a single iteration. This setup allows us to analyze the immediate impact of the new data point on the model's predictions and its relation to bias uncertainty.

The TracIn heatmaps in this experiment exhibit a striking resemblance to the GradNorm heatmaps observed in previous experiments. This similarity is not surprising, as performing a single iteration of

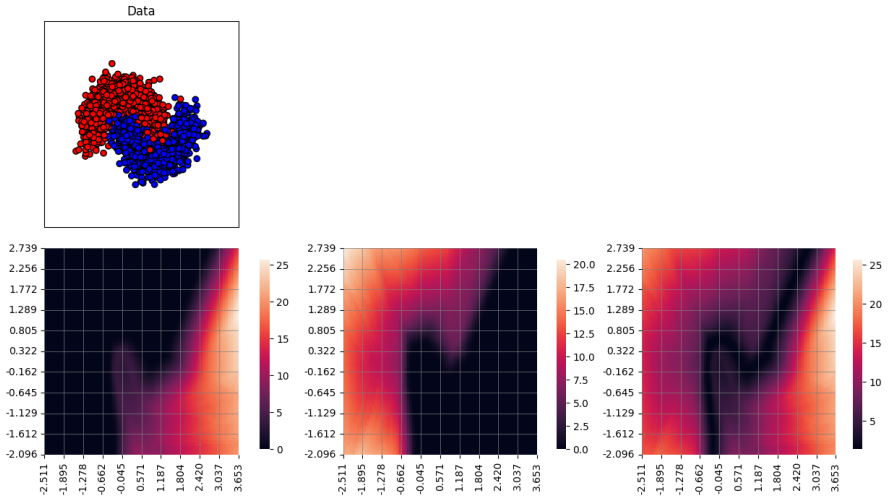


Figure 5.4: TracIn Visualization with Single Iteration Retraining. This figure displays the TracIn score heatmaps for the Gaussian Blobs dataset, where the model is retrained for a single iteration using only the new data point, i.e. any point in the space. The top subplot shows the original data distribution. The bottom row presents three heatmaps, from left to right: TracIn score when the new data point is assigned label 0, TracIn score when the new data point is assigned label 1, and the sum of the TracIn scores for both label assignments.

training with a new data point is conceptually similar to analyzing the gradients induced by that data point. Both approaches capture the immediate impact of the new data on the model's parameters and predictions.

We observe the following patterns in the TracIn heatmaps. Regions near the training data exhibit lower TracIn scores, indicating less information gain is required to incorporate those points, suggesting they are more likely to be in-distribution. As we move further away from the training data, the TracIn scores increase, reflecting a higher information gain needed to accommodate those points, suggesting they are potentially out-of-distribution. However, the decision boundary consistently shows lower TracIn scores, regardless of its distance from the training data. This indicates that incorporating data points near the decision boundary does not need that much information, as it has no preference for each class at the beginning.

TracIn with Multiple Iterations on Toy Generators

In this part, we extend the analysis of TracIn on toy generators by retraining the model for multiple iterations after adding back the masked data points. This allows us to investigate how the TracIn scores evolve over time as the model continues to learn and adapt to the changing training data distribution.

Figure 5.5 shows the TracIn scores for a toy dataset with two Gaussian blobs after one and three retraining iterations. In the one-iteration setting, the TracIn scores primarily highlight the decision boundary between the two classes. However, after three iterations, the TracIn scores capture not only the decision boundary but also the shape of the training data distribution. This indicates that with more training iterations, TracIn can better reflect the model's uncertainty in regions with varying data density.

This observation can be attributed to the need for multiple iterations to allow the model to converge to

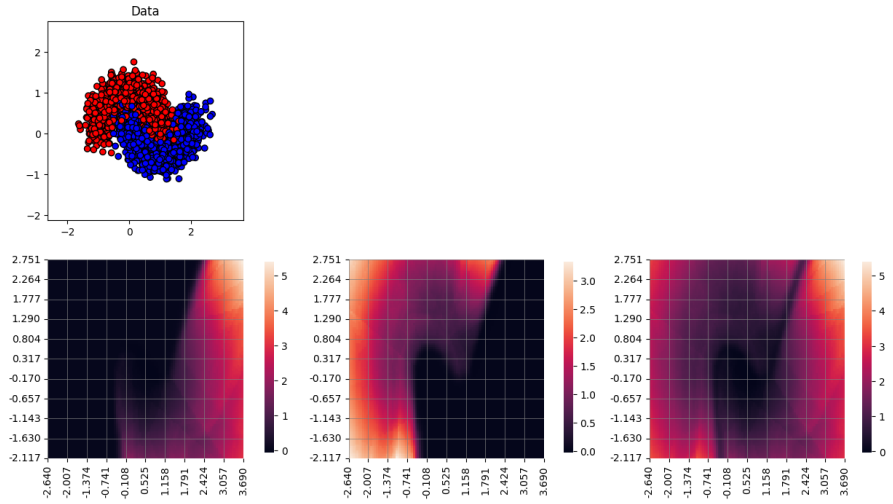


Figure 5.5: TracIn Visualization with Partial Training Set and Multiple Iterations.

This figure displays the TracIn score heatmaps for the Gaussian Blobs dataset, where the model is retrained for three iterations using the new data point and 25% of the original training set, randomly selected. The layout is similar to Figure 5.4, with the top subplot showing the original data distribution and the bottom row presenting three heatmaps for different label assignments of the new data point.

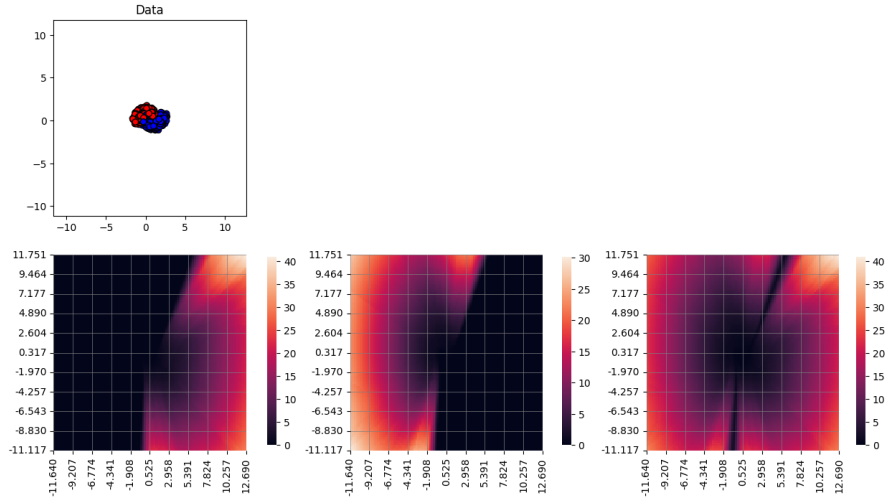


Figure 5.6: TracIn Visualization with Partial Training Set and Multiple Iterations. A zoomed out version of Figure 5.5.

the mini-batch training data. With a single iteration, the model may not have fully adapted to the newly added data points, leading to a less accurate representation of the uncertainty landscape. However, with multiple iterations, the model has more opportunities to learn and refine its understanding of the data distribution, resulting in a more comprehensive uncertainty estimation.

By comparing the results of the single and multiple iterations experiments, we gain a deeper understanding of the dynamics of TracIn scores during the training process and assess its effectiveness in capturing the evolving bias uncertainty as the model learns from the data. This analysis highlights the importance of considering multiple retraining iterations when using TracIn to quantify bias uncertainty, especially in scenarios with limited data or complex data distributions.

Figure 5.6 provides a zoomed-out view of the TracIn scores for the same toy dataset. This visualization reveals an interesting trend: the further away a point is from the training data, the larger its TracIn score. This observation aligns with the intuition that points far from the training data are more likely to be out-of-distribution and thus exhibit higher uncertainty.

This finding further supports the effectiveness of TracIn in capturing bias uncertainty. By assigning higher scores to points further from the training data, TracIn provides a valuable tool for identifying regions of the input space where the model's predictions are more likely to be unreliable due to limited training data or systematic errors.

From Figure 5.7, we can observe several interesting trends:

- **Convergence of TracIn distribution:** As the number of training iterations increases, the distribution of TracIn scores over the training data converges. This is evident in the rightmost histograms, which become more stable and concentrated with more iterations.
- **Stability of heatmap and contour lines:** Correspondingly, the heatmap and contour lines also stabilize as the TracIn distribution converges. This suggests that once the model has learned the underlying data distribution sufficiently, the TracIn scores become more consistent and less sensitive to further training.
- **Early convergence:** In this particular case, the TracIn scores and visualizations appear to stabilize relatively early in the training process (around iteration 5). But this may by highly use-case specific.

These observations provide further evidence for the effectiveness of TracIn in capturing bias uncertainty. The stability of the TracIn scores and visualizations after convergence suggests that they provide a robust and reliable measure of uncertainty, even with variations in the training process. Moreover, the early convergence in this example highlights the potential for efficient uncertainty estimation using TracIn.

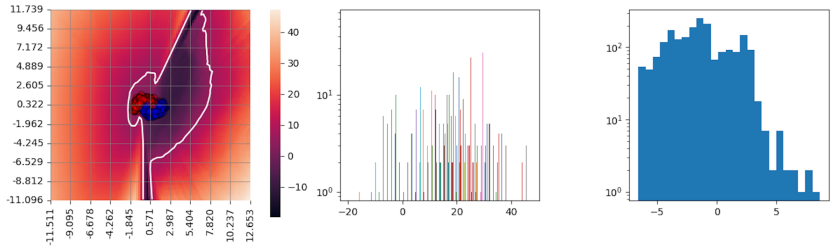
TracIn with Different Learning Rates on Toy Generators

In this experiment, we investigate the impact of the learning rate on TracIn scores during retraining. The learning rate is a crucial hyperparameter in the training process, as it controls the step size taken in the direction of the gradient. We hypothesize that the learning rate can significantly affect the TracIn scores and the resulting uncertainty estimates.

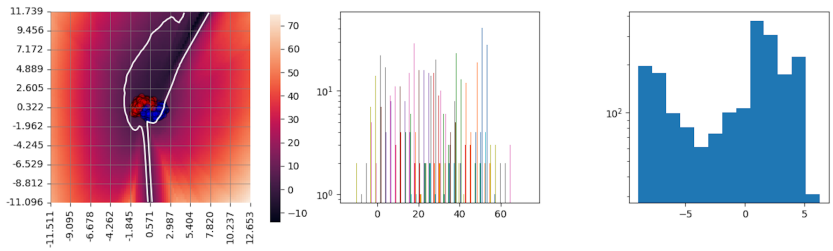
To explore this relationship, we utilize the Ringblobs dataset and vary the learning rate from 0.1 to 0.0001. We employ the same model architecture and retraining procedure as in the previous experiment, with three retraining iterations.

Figure 5.8 illustrates the TracIn scores for the Ringblobs dataset under different learning rates. Each row represents a specific learning rate and comprises three subplots: the leftmost subplot displays the TracIn heatmap overlaid with the training data scatter plot and a contour line demarcating the mean TracIn score of the top 20 data points in the training set. The middle subplot shows the histogram of TracIn scores over the grid points, while the rightmost subplot presents the histogram of TracIn scores for the training data.

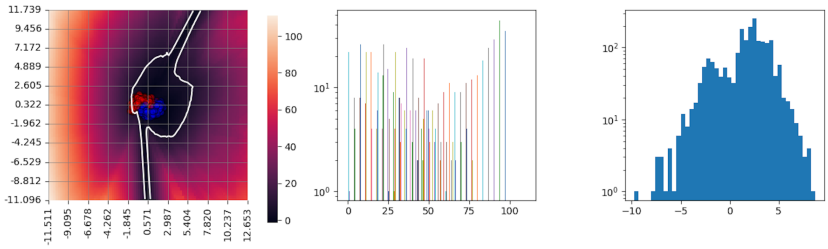
From Figure 5.8, we can observe the following trends:



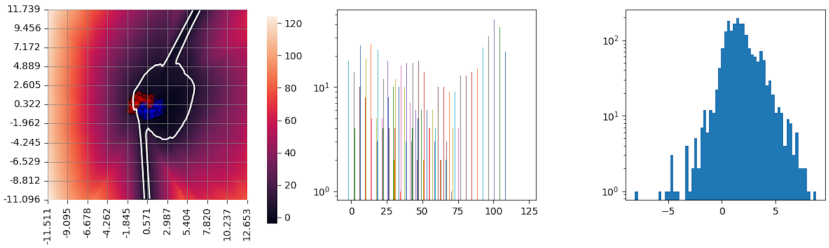
(a) Iteration = 1



(b) Iteration = 3

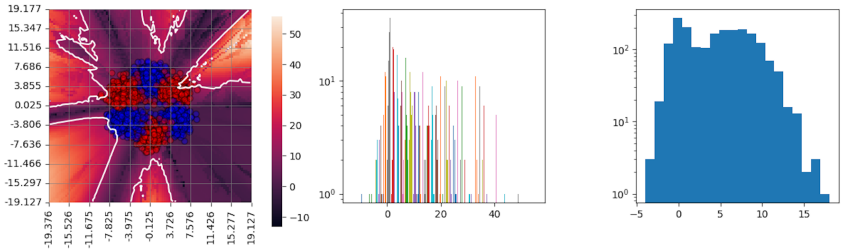


(c) Iteration = 5

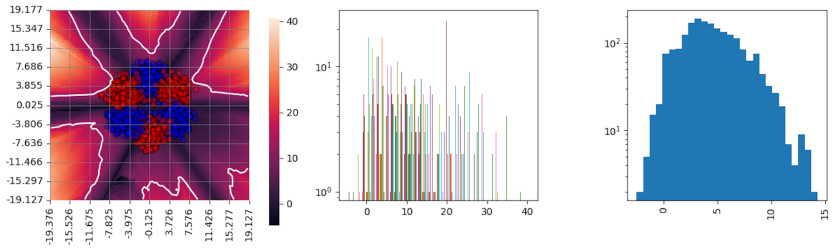


(d) Iteration = 7

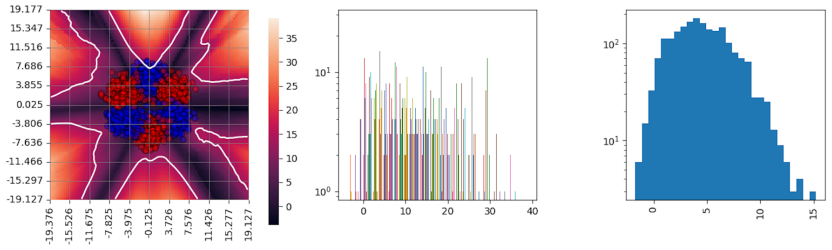
Figure 5.7: TracIn on Moons dataset with different iterations. This figure illustrates the evolution of TracIn scores on the Moons dataset across different training iterations (1, 3, 5, and 7), with each row representing a specific iteration. Each row comprises three subplots: the leftmost subplot displays the TracIn heatmap overlaid with the training data scatter plot and a contour line demarcating the mean TracIn score of the top 20 data points in the training set. The middle subplot shows the histogram of TracIn scores over the grid points, while the rightmost subplot presents the histogram of TracIn scores for the training data.



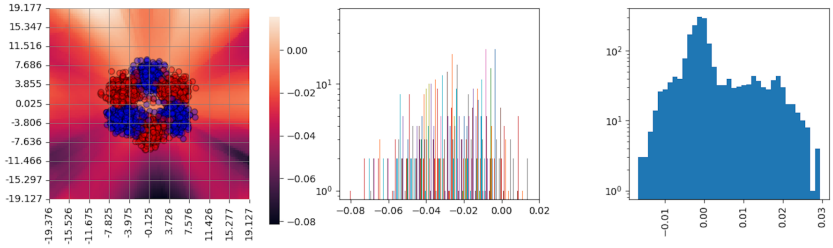
(a) Learning rate = 0.1



(b) Learning rate = 0.03



(c) Learning rate = 0.01



(d) Learning rate = 0.0001

Figure 5.8: TracIn on Ringblobs dataset with different learning rates.

- **Jagged shape with high learning rate:** When the learning rate is too high (e.g., 0.1), the TracIn heatmap and contour lines exhibit a jagged shape, indicating that the TracIn scores are not continuous. This suggests that a high learning rate can lead to instability in the training process and affect the reliability of the uncertainty estimates.
- **Small TracIn scores with low learning rate:** Conversely, when the learning rate is too low (e.g., 0.0001), the TracIn scores are very small. This implies that the model requires a large number of iterations to converge during retraining, which can be computationally expensive.
- **Optimal learning rate:** In this case, a learning rate of 0.01 appears to be optimal among the tested

values. It effectively captures the training data distribution and provides a smooth and continuous uncertainty estimate.

These findings highlight the importance of carefully selecting the learning rate when using TracIn for uncertainty quantification. An inappropriate learning rate can lead to unreliable uncertainty estimates or excessive computational cost. By tuning the learning rate appropriately, we can ensure that TracIn provides accurate and efficient uncertainty quantification.

TracIn with Different Batch Sizes on Toy Generators

In this experiment, we delve into the influence of batch size on TracIn scores during retraining. The batch size determines the number of training examples used in each iteration of the optimization algorithm. It plays a crucial role in the training dynamics and can affect the generalization performance and uncertainty estimates of the model.

To investigate this relationship, we utilize the Circular dataset and vary the batch size from 1% to 50% of the training set size, which is 2000 in this case. We maintain the same model architecture and retraining procedure as in the previous experiments, with three retraining iterations.

Figure 5.9 illustrates the TracIn scores for the Circular dataset with different batch sizes. Each row represents a specific batch size and again comprises three subplots: the leftmost subplot displays the TracIn heatmap overlaid with the training data scatter plot and a contour line demarcating the mean TracIn score of the top 20 data points in the training set. The middle subplot shows the histogram of TracIn scores over the grid points, while the rightmost subplot presents the histogram of TracIn scores for the training data.

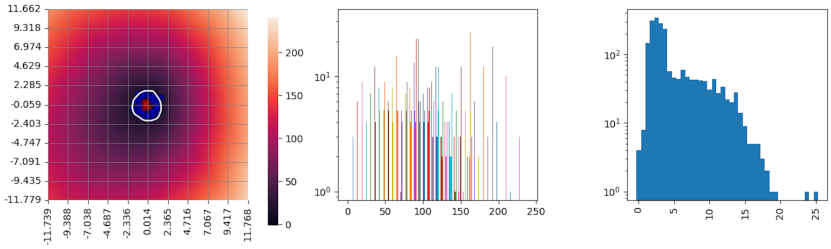
From this figure, we can discern the following trends:

- **Incomplete enclosure with small batch size:** When the batch size is too small (e.g., 1% of the training set size), the contour line fails to enclose the entire training set. This indicates that some regions within the training data still exhibit relatively high TracIn scores, suggesting that the model may not have fully captured the data distribution due to insufficient sampling.
- **Negative TracIn scores with large batch size:** Conversely, when the batch size is too large (e.g., 50% of the training set size), we observe negative TracIn scores. This phenomenon arises because we evaluate the TracIn score for a single data point at a time. With a large batch size, the influence of this single data point becomes diluted, and the model effectively overfits to the training set, leading to negative mutual information and negative TracIn scores.

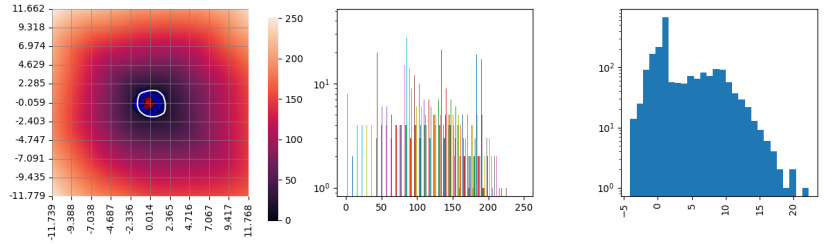
These findings underscore the importance of selecting an appropriate batch size when employing TracIn for uncertainty quantification. A batch size that is too small can hinder the model's ability to learn the data distribution comprehensively, while a batch size that is too large can lead to overfitting and unreliable uncertainty estimates. By tuning the batch size carefully, we can ensure that TracIn provides accurate and meaningful uncertainty quantification.

Insensitivity of TracIn

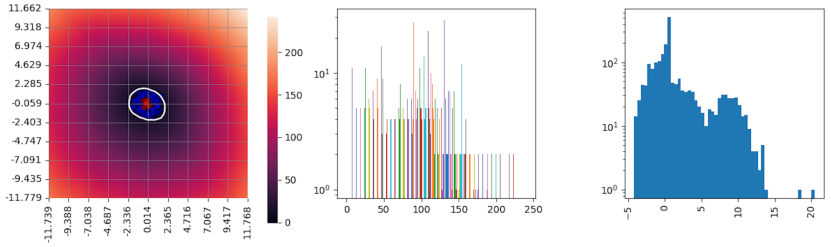
While our experiments have demonstrated the effectiveness of TracIn in quantifying bias uncertainty in various scenarios, it is important to also investigate its potential limitations. In this section, we delve into the insensitivity of TracIn to certain types of changes in the data distribution. We explore how TracIn scores may not exhibit significant variations even when encountering out-of-distribution data, potentially hindering its ability to reliably detect and quantify bias.



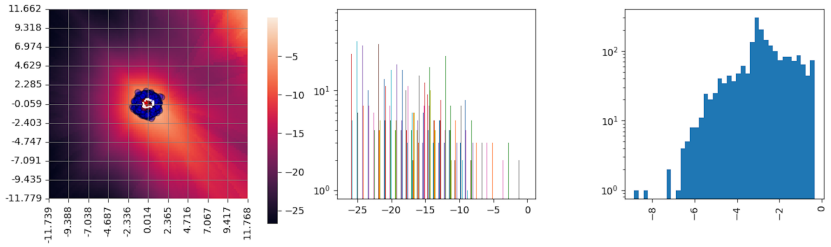
(a) Batch size = 1% training set size



(b) Batch size = 5% training set size



(c) Batch size = 10% training set size



(d) Batch size = 50% training set size

Figure 5.9: TracIn on Circular dataset with different batch sizes.

To illustrate this insensitivity, consider the scenario depicted in Figure 5.10, where we construct a synthetic circular dataset. The dataset comprises two concentric circles, each representing a different class. We then introduce a small square ”dead zone” in the center of the dataset, effectively removing a portion of the data from both classes.

As evident from the heatmap in Figure 5.11, TracIn does not detect the dead zone, as the scores remain relatively uniform across the entire region, including the area where data has been removed. This observation suggests that TracIn may not be sensitive enough to detect subtle or localized difference, imaging if there are a small density of blue points distributed in that area and not be sampled in the training set, in the

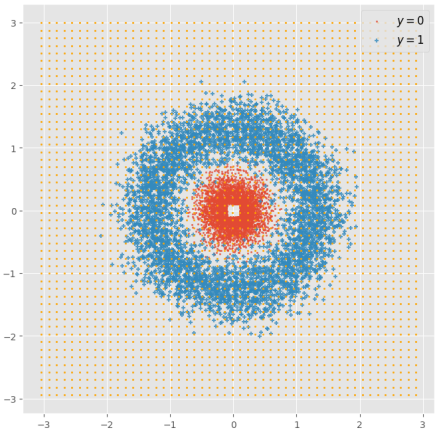


Figure 5.10: Circular dataset with a ”dead zone”.

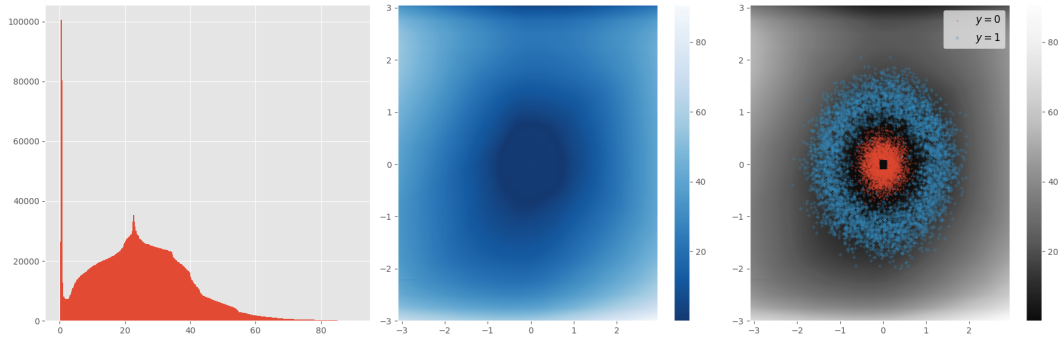


Figure 5.11: TracIn score of Circular dataset with a ”dead zone”. The left subplot shows the histogram of TracIn scores over the grid points, the middle subplot displays the TracIn heatmap, and the right subplot shows the heatmap overlaid with the training data scatter plot.

data distribution particularly when these changes do not significantly alter the overall class boundaries.

To further investigate this insensitivity, we added a significant weight (250) to the point in the center of the dead zone during the keep training process. Figure 5.12 shows the log loss over the grid points of the classifier when retraining on this point for 75 iterations. As you can see, it takes a considerable number of iterations (75 in this case) for the classifier to significantly ”notice” this point and converge. This observation suggests that TracIn may struggle to capture the influence of data points that are significantly under-represented or lie in regions where the model has low confidence.

Figure 5.13 depicts the loss curves for this experiment. The curves represent the loss for the center point in the dead zone, the mean loss for the points in the mini-batch and the mean loss for all points in the training set. We observe that the loss for the individual point diverges from the mean losses after around 60 iterations. This big number indicates that the model is not easy to fit this particular point, even when given a significant weight. This observation further supports the notion that TracIn may not be sensitive enough to capture the influence of such under-represented or unusual data points.

This insensitivity can be problematic in real-world scenarios where subtle sparse areas may exist. If TracIn fails to detect these changes, it may provide misleading uncertainty estimates and hinder the

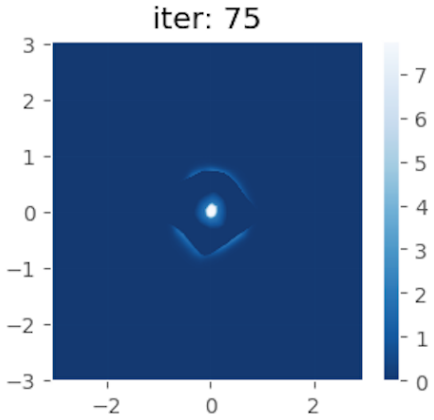


Figure 5.12: Heatmap of the loss space of adding the center point.

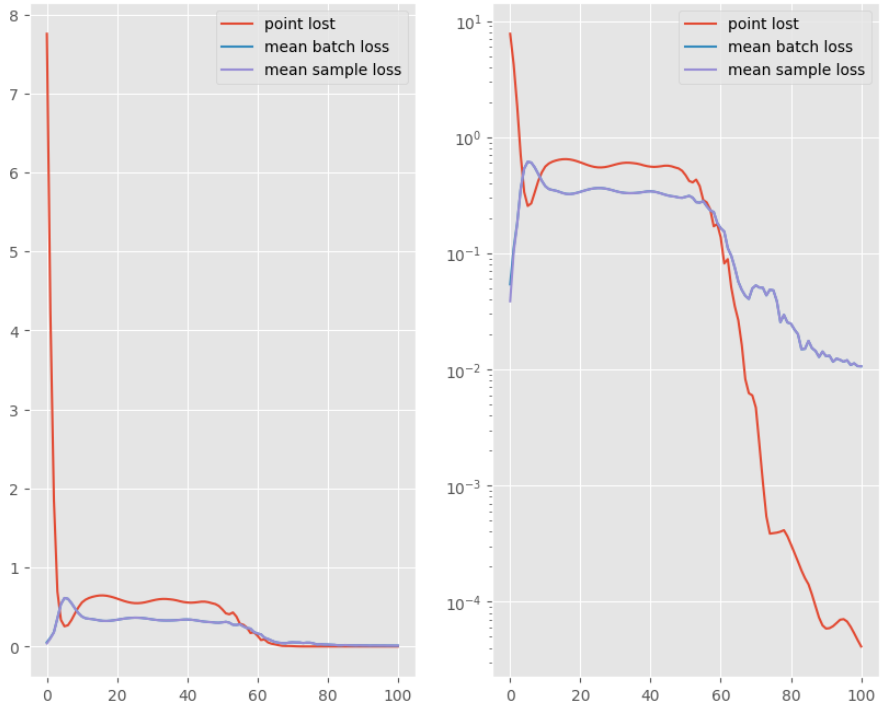


Figure 5.13: The loss of the center point and the average loss versus number of training iterations.

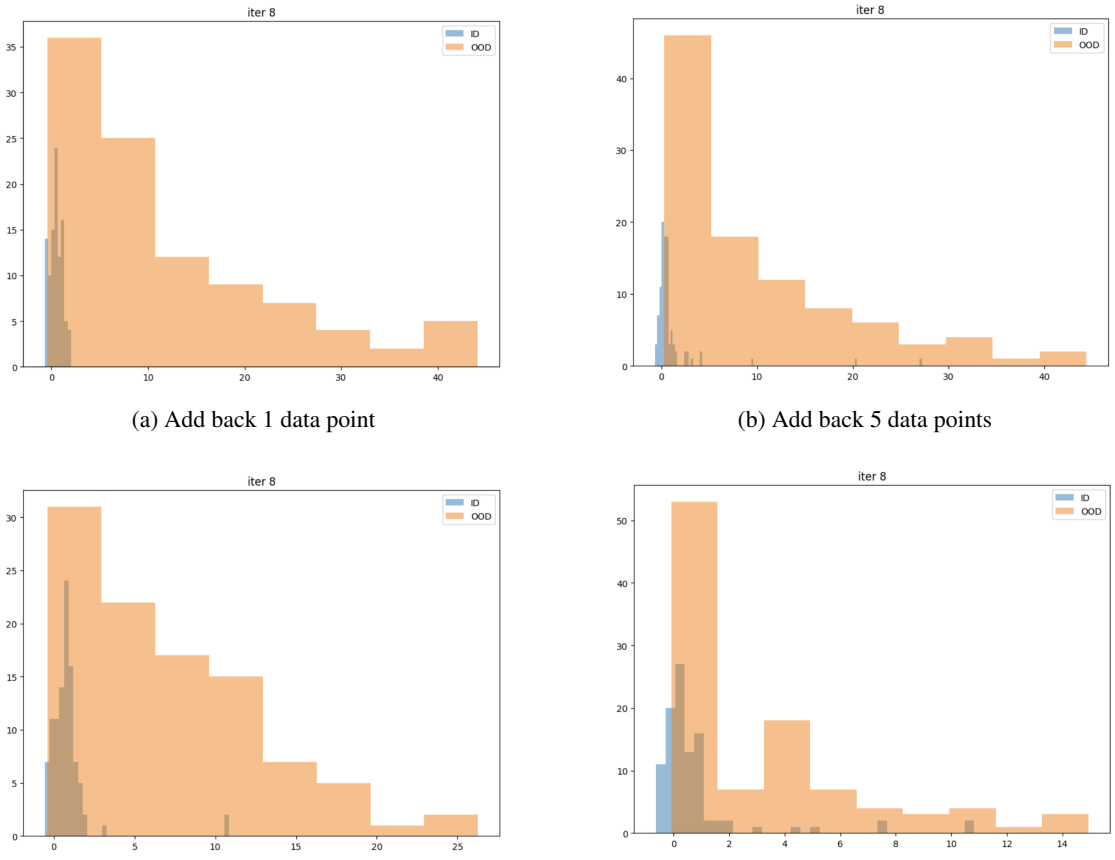
identification and mitigation of bias in deep learning models. Further research is needed to develop a more nuanced understanding of the factors that influence TracIn's sensitivity and to develop more robust variants that can effectively capture bias uncertainty across a wider range of datasets and tasks.

5.2.3 TracIn on Titanic Dataset

In this section, we evaluate the effectiveness of TracIn for quantifying bias uncertainty on the Titanic dataset with the same setting in section 4.1.2. The target variable, "Pclass," represents the ticket class, with 1 indicating first class, 2 indicating second class, and 3 indicating third class.

To evaluate the performance of TracIn in quantifying bias uncertainty, we design an experiment where we mask out passengers younger than 16 years old as the out-of-distribution data (134 in total). This creates a scenario where the model has limited exposure to this specific passenger demographic during training. We then progressively add back these masked data points to the training set, using four different settings:

- Add back 1 data point.
- Add back 5 data points.
- Add back 10 data points.
- Add back 100 data points.



(a) Add back 1 data point

(b) Add back 5 data points

(c) Add back 10 data points

(d) Add back 100 data points

Figure 5.14: TracIn on Titanic Dataset.

Figure 5.15 shows the distributions of TracIn scores for in-distribution (ID) and out-of-distribution (OOD)

data points under the four different settings. As we progressively add back the masked data points (passengers younger than 16 years old) to the training set, we observe the following trends:

- **Clear separation for small additions:** When adding back 1, 5, or 10 data points, there is a clear distinction between the TracIn score distributions for ID and OOD data. This indicates that TracIn can effectively identify the OOD data points, even when they are under-represented in the training set.
- **Less distinction for large additions:** When adding back 100 data points, the separation between the ID and OOD distributions becomes less pronounced. This suggests that as the model gains more exposure to the previously masked demographic, its ability to distinguish between ID and OOD data diminishes.
- **Decreasing TracIn scores:** Overall, we observe a decrease in the TracIn scores for the OOD data points as more of them are added back to the training set. This supports the theory that increasing the representation of a specific demographic in the training data can reduce the model's bias towards that demographic.

While there is some variability in the results due to the randomness of the training procedures, the overall trend is consistent with our expectations. These findings demonstrate the effectiveness of TracIn in quantifying bias uncertainty and its potential for identifying and mitigating bias in machine learning models.

To further assess the robustness of TracIn to unseen data, we conducted an additional experiment where we randomly masked out 30% of the data as OOD data. This simulates a scenario where the model encounters data points that are significantly different from the training distribution.

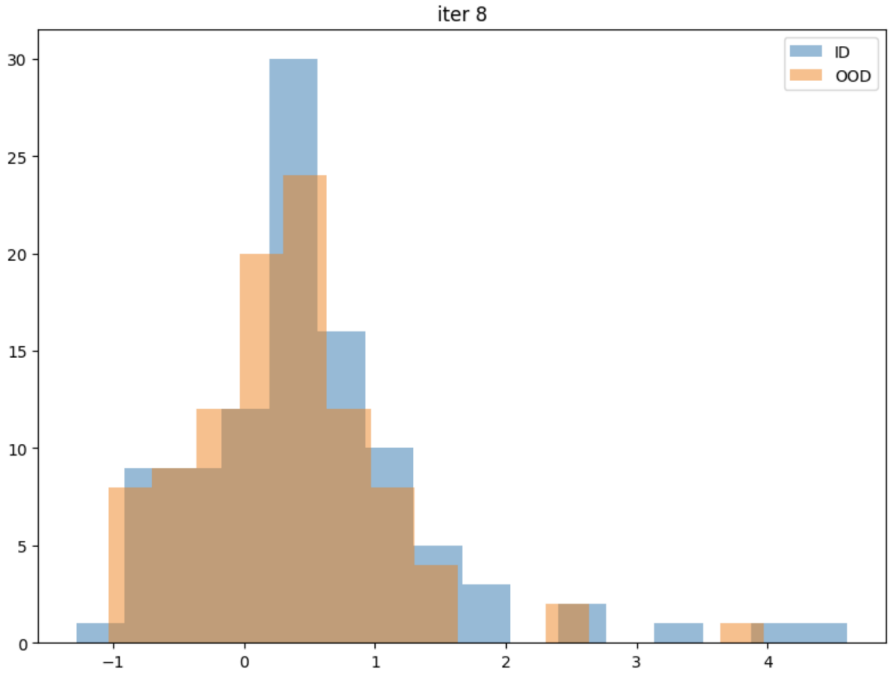


Figure 5.15: TracIn on Titanic Mask Out Random Passengers.

Figure 5.15 shows the distribution of TracIn scores for ID and OOD data in this setting. We observe no significant difference between the two distributions, indicating that TracIn is not overly sensitive to

unseen data. This finding suggests that TracIn can effectively quantify bias uncertainty even when the model encounters data points that deviate from the training distribution.

This robustness to unseen data is a crucial property for uncertainty quantification methods, as it ensures that the estimated uncertainty remains reliable even in the presence of novel or unexpected inputs. This robustness enhances the practical applicability of TracIn for real-world scenarios where models are likely to encounter data that differs from the training data.

By analyzing the TracIn scores for these masked data points as they are gradually added back to the training set, we can observe how the model's bias uncertainty changes with increasing exposure to this under-represented demographic.

6 Discussions and Limitations

This chapter provides a comprehensive discussion of the findings presented in this thesis, highlighting their implications for uncertainty quantification and bias mitigation in deep learning. We also address the limitations of our work and outline potential avenues for future research.

6.1 Discussion of Findings

Throughout this thesis, we have explored the concept of point prediction uncertainty and its decomposition into variance and bias components. We have investigated various methods for quantifying these components, including Bregman Information and Beta/Dirichlet scores for variance uncertainty, and GradNorm and TracIn for bias uncertainty. Our experimental evaluations on both synthetic and real-world datasets have provided valuable insights into the strengths and limitations of these methods, as well as the factors that influence their performance and the dynamics of uncertainty estimation during training.

6.1.1 Bregman Information versus Beta and Dirichlet score for Variance Uncertainty

We have delved into the quantification of variance uncertainty, exploring two distinct approaches: Bregman Information score and Beta (Dirichlet) score. Each approach presents unique advantages and limitations.

Bregman Information Score

The Bregman Information score exhibits a strong connection with variance-bias decomposition and effectively captures uncertainty caused by variance in the model's predictions. However, this score solely considers the probability scores without incorporating the actual labels. This characteristic can lead to insensitivity in regions near the decision boundary, where uncertainty should be high due to the model's difficulty in assigning definitive classifications. Furthermore, the Bregman Information score suffers from scaling issues, being unbounded and not representing a probability. This can hinder interpretability, as uncertainty is often more intuitively understood as the probability of the classifier making an incorrect prediction.

Beta and Dirichlet Score

To address these limitations, we have explored the Beta score for binary classification and the Dirichlet score for multi-class settings. These scores effectively resolve the scaling and interpretability issues associated with the Bregman Information score, providing a probability-based measure of uncertainty and performing well near the decision boundary. However, these scores rely solely on labels, discarding the classifier's output probability scores. This can result in a loss of valuable information, particularly in regions far from the decision boundary where the probability scores can provide nuanced insights into the model's confidence.

These findings highlight the trade-offs between different approaches for quantifying variance uncertainty. While the Bregman Information score effectively captures the relationship between variance and uncertainty, it suffers from limitations in sensitivity and interpretability. Conversely, the Beta/Dirichlet scores offer improved interpretability and performance near the decision boundary but may discard valuable information encoded in the probability scores.

6.1.2 GradNorm and TracIn for Bias Uncertainty

In this thesis, we extensively investigated two methods for quantifying bias uncertainty: GradNorm and TracIn. Both methods offer unique perspectives and capabilities for understanding and mitigating bias in deep learning models.

GradNorm

GradNorm leverages information from the gradient space, specifically focusing on the gradients of the loss with respect to the weights in the network, to identify and quantify bias. By analyzing these gradients, GradNorm can pinpoint areas within the network that are particularly sensitive to changes in the input data. This sensitivity can be indicative of bias, as it suggests that the model may be overly reliant on certain features or patterns in the data.

Our experiments with GradNorm have demonstrated its effectiveness in detecting out-of-distribution (OOD) data, particularly data that is OOD with respect to the classifier. However, it is important to acknowledge that there may be a gap between the OOD data detected by GradNorm and the true OOD data with respect to the underlying ground truth distribution. This gap arises because GradNorm focuses on the model's internal representation of the data, which may not perfectly align with the true data distribution.

Despite this limitation, GradNorm provides a valuable tool for identifying potential biases in datasets and improving the robustness of deep learning models. By highlighting inputs that deviate significantly from the training distribution, GradNorm can guide interventions to mitigate these biases and enhance the model's generalization capabilities.

TracIn

TracIn offers a more granular perspective on bias uncertainty by focusing on the influence of individual training examples on the model's predictions. It quantifies this influence by tracing the changes in the loss function during training, effectively measuring how much each training example contributes to the final prediction for a given input. This granular approach allows TracIn to identify specific training examples that have a disproportionate impact on the model's predictions, potentially revealing hidden biases or under-representation of certain data points.

Our experiments with TracIn demonstrated its ability to quantify bias uncertainty. By analyzing the TracIn scores for different data points, we were able to identify specific training examples that exerted a strong influence on the model's predictions. This information can be valuable for understanding the model's behavior and identifying potential biases or areas where the model may be overly reliant on specific training examples.

For instance, in our experiments with the Titanic dataset (Figure 5.14), we observed that TracIn scores were higher for passengers younger than 16 years old, a demographic that was under-represented in the training data. This observation suggests that the model's predictions for these passengers were more sensitive to the specific training examples that included individuals from this age group. This finding highlights TracIn's potential for identifying and quantifying bias related to under-represented groups or unusual data points.

Moreover, our experiments with synthetic datasets (Figures 5.8 and 5.9) revealed that TracIn scores can be influenced by various factors, including the model's architecture, the training process, and the data distribution. Specifically, we found that using a high learning rate can lead to TracIn scores exhibiting a jagged shape, indicating instability in the training process. Conversely, a very low learning rate can result in small TracIn scores, requiring a large number of iterations for convergence. We also observed that small batch sizes can hinder the model's ability to fully capture the data distribution, while large batch sizes can lead to overfitting and negative TracIn scores.

These findings underscore the importance of carefully considering the choice of hyperparameters and the characteristics of the data distribution when interpreting TracIn scores and using them to guide bias mitigation strategies.

Overall, TracIn provides a valuable tool for understanding and quantifying bias uncertainty in deep learning models. Its ability to identify influential training examples can aid in the development of more robust and fair machine learning systems. However, it is crucial to be mindful of its limitations, particularly its sensitivity to hyperparameters and potential insensitivity to subtle changes in the data distribution, as discussed in Section 5.2.2.

6.2 Practical Considerations

While the previous sections have focused on the theoretical and experimental aspects of uncertainty quantification, it is equally important to consider the practical implications of these methods. In this section, we discuss the practical considerations for applying the four methods explored in this thesis: Bregman Information, Beta/Dirichlet Score, GradNorm, and TracIn. We highlight their strengths and weaknesses in practice, offering guidance on their appropriate use and interpretation.

6.2.1 Bregman Information

Bregman Information, while theoretically grounded, presents challenges in practical application due to its scaling issues. The range of Bregman Information scores can vary significantly across different datasets(Figure 4.3 and Figure 4.4) and even within a single dataset(Figure 4.1), particularly in high-dimensional feature spaces. This makes it difficult to establish a universal threshold for identifying high uncertainty points. Moreover, without prior knowledge of the decision boundary's location, it can be challenging to determine what constitutes a truly high Bregman Information score.

Despite these limitations, Bregman Information can still be useful in practice for identifying regions of relatively high variance uncertainty within a dataset. By comparing the scores of different data points, practitioners can gain insights into the relative uncertainty associated with different areas of the feature space.

One notable advantage of Bregman Information is its model agnosticism. It only requires feeding the learner with random inputs and evaluating the resulting predictions, making it applicable to a wide range of models without requiring any modifications or specific assumptions about their architecture or training process.

6.2.2 Beta/Dirichlet Score

Unlike Bregman Information, the Beta/Dirichlet Score effectively addresses the scaling issue, making it suitable for both within-dataset and cross-dataset comparisons(Figure 4.7). This property enhances

its practical applicability, as practitioners can use it to compare uncertainty across different models or datasets without worrying about scaling discrepancies.

Like Bregman Information, the Beta/Dirichlet Score is also model agnostic, making it widely applicable. However, it is important to note that this score primarily focuses on the decision boundary, potentially neglecting uncertainty in other regions of the feature space. Therefore, it may not be the ideal choice for applications requiring high sensitivity to uncertainty across the entire data space.

6.2.3 GradNorm

In practice, GradNorm is often overshadowed by TracIn due to its similar limitations and higher model dependence. GradNorm requires access to the model's gradients and is therefore not model agnostic. This limits its applicability compared to TracIn, which can be applied to any model that produces probability scores.

6.2.4 TracIn

TracIn offers a more nuanced approach to quantifying bias uncertainty but requires careful tuning of hyperparameters, such as the learning rate and batch size. However, practitioners can leverage information from the training set to guide this tuning process. For instance, observing the convergence behavior on the training set can provide insights into the appropriate number of iterations for retraining.

Unlike variance uncertainty, where identifying high uncertainty regions can be challenging, low bias uncertainty areas are often easier to discern. Most of the training data, especially the dense regions, are less likely to exhibit high bias uncertainty(Figure 5.14). This observation can help practitioners set thresholds for detecting high uncertainty inputs, aided by visualizations such as contour plots.

While TracIn may not accurately capture bias uncertainty near the decision boundary, methods for quantifying variance uncertainty, such as Bregman Information or Beta/Dirichlet scores, can complement TracIn in these regions, providing a more comprehensive assessment of uncertainty.

Lastly, while TracIn is not strictly model agnostic in the sense that it requires follow-up training the model, it can be applied to any model that supports this functionality. This includes a wider range of models than, making TracIn a versatile tool for quantifying bias uncertainty.

6.3 Limitations

While our research has yielded valuable insights into uncertainty quantification and bias mitigation in deep learning, it is essential to acknowledge the limitations of our work. These limitations provide opportunities for future research and refinement of the methods we have explored.

- **Challenges in merging Bregman Information and Beta/Dirichlet scores for variance uncertainty:** Our exploration of variance uncertainty revealed a fundamental challenge in merging the strengths of the Bregman Information score and the Beta/Dirichlet score. While the Bregman score aligns well with variance-bias decomposition, it suffers from scaling and boundary insensitivity issues. Conversely, the Beta/Dirichlet score addresses these issues but discards valuable information from the probability scores. Our inability to effectively combine these approaches highlights a key area for future research in developing methods that can balance information retention with desirable properties like boundedness and sensitivity near the decision boundary.

- **Scaling issues with TracIn and GradNorm:** While both TracIn and GradNorm provide valuable insights into bias uncertainty, their scores can be difficult to use in practical case directly due to scaling issues. The scores are not bounded or normalized, making it challenging to compare them across different datasets or models. Although comparing the scores of new data points with those of points in the training set can provide some context, a more robust and standardized scaling mechanism would enhance the interpretability and comparability of these methods.
- **Sensitivity of TracIn to hyperparameters:** The performance of TracIn can be sensitive to the choice of hyperparameters, such as the learning rate and batch size. These hyperparameters influence the model's training dynamics and can affect the TracIn scores significantly. For instance, a high learning rate can lead to overfitting, which in turn can make TracIn overly sensitive, even for in-distribution data. This sensitivity necessitates careful tuning of hyperparameters, which can be time-consuming and may not generalize well across different datasets or tasks. Developing more robust and automated methods for hyperparameter tuning or more robust variants of TracIn that are less sensitive to these choices is crucial for ensuring its reliability and broader applicability.
- **Insensitivity of TracIn to subtle sparse area:** As demonstrated in the section 5.2.2, TracIn can exhibit insensitivity to certain types of changes in the data distribution. Specifically, it may not reliably detect subtle or localized changes, such as the "dead zone" in the circular dataset example. This limitation can hinder its ability to identify and quantify bias in scenarios where the data distribution changes are not substantial enough to significantly alter the overall class boundaries or model behavior.

7 Conclusion

In this thesis, we embarked on a detailed exploration of point prediction uncertainty in deep learning models, with a focus on decomposing this uncertainty into its variance and bias components. We sought to understand the sources of these components and develop methods to quantify them effectively.

We introduced a theoretical foundation using the general bias-variance decomposition framework. This framework allowed us to analyze and quantify uncertainty in a broader class of predictive tasks, utilizing proper scoring rules and Bregman divergences.

To quantify variance uncertainty, we investigated two approaches: Bregman Information and Beta/Dirichlet scores. Bregman Information, derived from the theoretical framework, effectively captured the variance component but had limitations in sensitivity and interpretability, particularly near the decision boundary. The Beta/Dirichlet scores offered better interpretability and performance near the decision boundary but discarded probability score information.

Our exploration of bias uncertainty led us to leverage out-of-distribution (OOD) detection methods, GradNorm and TracIn. GradNorm, using information from the gradient space, identified OOD samples, highlighting potential biases in the model's representation of the true data distribution.

TracIn employed a follow-up training approach, quantifying the information gain required to correct potential prediction errors. This approach offered a unique perspective on bias uncertainty by assessing the information needed to align the model's predictions with the true distribution.

Through experiments on synthetic and real-world datasets, we demonstrated TracIn's effectiveness in quantifying bias uncertainty and identifying potential sources of bias. We analyzed the impact of various factors on TracIn scores, including learning rate, batch size, and training iterations. The results highlighted the importance of careful hyperparameter tuning and the need for further research to address TracIn's limitations, such as its insensitivity to subtle or localized changes in the data distribution.

From a practical perspective, the techniques explored in this thesis offer valuable tools for identifying and mitigating bias in deep learning models, leading to more robust and reliable prediction systems. However, the limitations we identified suggest that further research is needed to refine these techniques and enhance their practical applicability. Future work could focus on addressing the scaling issues, improving sensitivity and interpretability, and developing more robust and automated methods for hyperparameter tuning.

Bibliography

[1] Katherine Haynes et al. “Creating and evaluating uncertainty estimates with neural networks for environmental-science applications”. In: *Artificial Intelligence for the Earth Systems* 2.2 (2023), p. 220061.

[2] Moloud Abdar et al. “A review of uncertainty quantification in deep learning: Techniques, applications and challenges”. In: *Information fusion* 76 (2021), pp. 243–297.

[3] Xavier Bouthillier et al. “Accounting for Variance in Machine Learning Benchmarks”. In: *Proceedings of Machine Learning and Systems*. Ed. by A. Smola, A. Dimakis, and I. Stoica. Vol. 3. 2021, pp. 747–769. URL: https://proceedings.mlsys.org/paper_files/paper/2021/file/0184b0cd3cfb185989f858a1d9f5c1eb-Paper.pdf.

[4] Yarin Gal et al. “Uncertainty in deep learning”. In: (2016).

[5] David JC MacKay. “A practical Bayesian framework for backpropagation networks”. In: *Neural computation* 4.3 (1992), pp. 448–472.

[6] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[7] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[8] Andreas Damianou and Neil D Lawrence. “Deep gaussian processes”. In: *Artificial intelligence and statistics*. PMLR. 2013, pp. 207–215.

[9] Tongfei Chen et al. “Confidence scoring using whitebox meta-models with linear classifier probes”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 1467–1475.

[10] Sebastian G Gruber and Florian Buettner. “Uncertainty estimates of predictions via a general bias-variance decomposition”. In: *arXiv preprint arXiv:2210.12256* (2022).

[11] Yoshua Bengio and Yves Grandvalet. “No unbiased estimator of the variance of k-fold cross-validation”. In: *Advances in Neural Information Processing Systems* 16 (2003).

[12] Andrey Malinin and Mark Gales. “Predictive uncertainty estimation via prior networks”. In: *Advances in neural information processing systems* 31 (2018).

[13] Hanjing Wang and Qiang Ji. “Epistemic Uncertainty Quantification For Pre-Trained Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 11052–11061.

[14] Weitang Liu et al. “Energy-based out-of-distribution detection”. In: *Advances in neural information processing systems* 33 (2020), pp. 21464–21475.

[15] Adrian Schwaiger et al. “Is uncertainty quantification in deep learning sufficient for out-of-distribution detection?” In: *Aisafety@ ijcai* 54 (2020).

[16] Taeseong Yoon and Heeyoung Kim. “Uncertainty Estimation by Density Aware Evidential Deep Learning”. In: *arXiv preprint arXiv:2409.08754* (2024).

[17] Garima Pruthi et al. “Estimating training data influence by tracing gradient descent”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19920–19930.

[18] Rui Huang, Andrew Geng, and Yixuan Li. "On the importance of gradients for detecting distributional shifts in the wild". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 677–689.

[19] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[20] Pedro Domingos. "A unified bias-variance decomposition for zero-one and squared loss". In: *AAAI/IAAI* 2000 (2000), pp. 564–569.

[21] Dieter Jungnickel. *Optimierungsmethoden: Eine Einführung*. Springer-Verlag, 2014.

[22] Eric Schechter. *Handbook of Analysis and its Foundations*. Academic Press, 1996.

[23] Tilmann Gneiting and Adrian E Raftery. "Strictly proper scoring rules, prediction, and estimation". In: *Journal of the American statistical Association* 102.477 (2007), pp. 359–378.

[24] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton: Princeton University Press, 1970. ISBN: 9781400873173. DOI: doi:10.1515/9781400873173. URL: https://doi.org/10.1515/9781400873173.

[25] J Kreer. "A question of terminology". In: *IRE Transactions on Information Theory* 3.3 (1957), pp. 208–208.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.