UNIVERSITAT DE BARCELONA

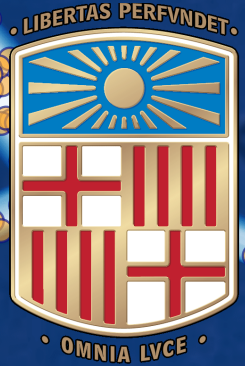
Aleatoric Uncertainty Modelling in Regression Problems using Deep Learning

Axel Brando Guillaumes

DOCTORAL DISSERTATION

ALEATORIC UNCERTAINTY MODELLING IN REGRESSION PROBLEMS USING DEEP LEARNING

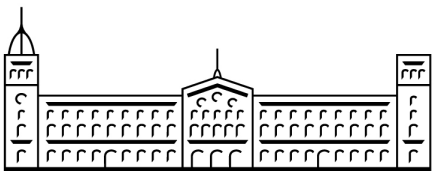
AXEL BRANDO



UNIVERSITAT DE BARCELONA

Ph.D. thesis

Aleatoric Uncertainty Modelling in Regression Problems using Deep Learning



Axel Brando

Department of Mathematics and Computer Science
Universitat de Barcelona (UB)

This dissertation is submitted for the degree of
Doctor of Philosophy

UB advisor: Jordi Vitrià
BBVA D&A advisor: Jose A. Rodríguez-Serrano

Barcelona, April 2022

Abstract

Uncertainty is always around us. Every decision we take has an associated likelihood of success and decisions made by autonomous systems are not an exception. Despite the great advances in the field of artificial intelligence, the inability of these systems to identify a higher risk scenario a priori could prevent their inclusion as part of solutions to many real problems. This is why it is essential that these systems learn how to model and deal with uncertainty. Starting from a probabilistic approach, this thesis proposes to formalize the different types of uncertainty and, in particular, focus its research on one type of uncertainty, the aleatoric uncertainty, since it is detected as the main uncertainty for the financial real-world problem that motivates this doctorate. Based on such research, the thesis proposes new models to improve the state of the art in modeling such uncertainty as well as introduces a new real problem that appears when there is a fixed predictive system that does not model uncertainty and we want to model uncertainty a posteriori. without changing the original model. This problem will be denoted as the modeling of the uncertainty of a black box system and will motivate the proposal of new models specialized in maintaining the predictive advantages, such as Quantile Regression (QR), but for the black box problem. Subsequently, the QR research will motivate the proposal of new models to solve a QR literature problem known as the crossing quantile phenomena, which appears when different quantiles are predicted simultaneously and they do not preserve their correct order with respect to their quantile value. Finally, all of the above research will be summarized in visualization and evaluation methods for the predicted uncertainty to produce uncertainty-tailored methods.

Summary

Nowadays, we live in an intrinsically uncertain world from our perspective. We do not know what will happen in the future but, to infer it, we build the so-called models. These models are abstractions of the world we live which allow us to conceive how the world works and that are, essentially, validated from our previous experience and discarded if their predictions prove to be incorrect in the future. This common scientific process of inference has several non-deterministic steps.

First of all, our measuring instruments could be inaccurate. That is, the information we use a priori to know what will happen may already contain some irreducible error.

Besides, our past experience in building the model could be biased (and, therefore, we would incorrectly infer the future, as the models would be based on unrepresentative data).

On the other hand, our model itself may be an oversimplification of the reality (which would lead us to unrealistic generalizations).

Furthermore, the overall task of inferring the future may be downright non-deterministic. This often happens when the information we have a priori to infer the future is incomplete or partial for the task to be performed (i.e. it depends on factors we cannot observe at the time of prediction) and we are, consequently, obliged to consider that what we want to predict is not a deterministic value.

One way to model all of these uncertainties is through a probabilistic approach that mathematically formalizes these sources of uncertainty in order to create specific methods that capture them.

Accordingly, the general aim of this thesis is to define a probabilistic approach that contributes to artificial intelligence-based systems (specifically, deep learning) becoming robust and reliable systems capable of being applied to high-risk problems, where having generic good performance is not enough but also to ensure that critical errors with high costs are avoided.

In particular, the thesis shows the current divergence in the literature - when it comes to dividing and naming the different types of uncertainty - by proposing a procedure to follow. In addition, based on a real problem case arising from the industrial nature of the current thesis, the importance of investigating the last type of uncertainty is emphasized, which arises from the lack of a priori information in order to infer deterministically the future, the so-called aleatoric uncertainty.

The current thesis delves into different literature models in order to capture aleatoric uncertainty using deep learning and analyzes their limitations. In addition, it proposes new state-of-the-art approaches that allow to solve the limitations exposed during the thesis.

As a result of applying the aleatoric uncertainty modelling in real-world problems, the uncertainty modelling of a black box systems problem arises. Generically, a Black box system is a pre-existing predictive system which originally do not model uncertainty and where no requirements or assumptions are made about its internals. Therefore, the goal is to build a new system that wrappers the black box and models the uncertainty of this original system. In this scenario, not all previously introduced aleatoric uncertainty modelling approaches can be considered and this implies that flexible methods such as Quantile Regression ones need to be modified in order to be applied in this context.

Subsequently, the Quantile Regression study brings the need to solve one critical literature problem in the QR literature, the so-called crossing quantile, which motivates the proposal of new additional models

to solve it.

Finally, all of the above research will be summarized in visualization and evaluation methods for the predicted uncertainty to produce uncertainty-tailored methods.

Extracte

Estem rodejats d'incertesa. Cada decisió que prenem té una probabilitat de sortir com un espera i, en funció d'aquesta, molts cops condicionem les nostres decisions. De la mateixa manera, els sistemes autònoms han de saber interpretar aquests escenaris incerts. Tot i això, actualment, malgrat els grans avenços en el camp de la intel·ligència artificial, ens trobem en un moment on la incapacitat d'aquests sistemes per poder identificar a priori un escenari de major risc impedeix la seva inclusió com a part de solucions que podrien revolucionar la societat tal i com la coneixem. El repte és significatiu i, per això, és essencial que aquests sistemes aprenguin a modelar i gestionar totes les fonts de la incertesa. Partint d'un enfocament probabilístic, aquesta tesi proposa formalitzar els diferents tipus d'incerteses i, en particular, centra la seva recerca en un tipus anomenada com incertesa aleatòrica, ja que va ser detectada com la principal incertesa decisiva a tractar en el problema financer original que va motivar el present doctorat industrial. A partir d'aquesta investigació, la tesi proposa nous models per millorar l'estat de l'art en la modelització de la incertesa aleatòrica, així com introdueix un nou problema, a partir d'una necessitat real industrial, que apareix quan hi ha un sistema predictiu en producció que no modela la incertesa i es vol modelar la incertesa a posteriori de forma independent. Aquest problema es denotarà com la modelització de la incertesa d'un sistema de caixa negra i motivarà la proposta de nous models especialitzats en mantenir els avantatges predictius, com ara la Regressió Quantílica (RQ), adaptant-los al problema de la caixa negra. Posteriorment, la investigació en RQ motivarà la proposta de nous models per resoldre un problema fonamental de la literatura en RQ conegut com el fenomen

del creuament de quantils, que apareix quan, a l'hora de predir simultàniament diferents quantils, l'ordre entre quantils no es conserva. Finalment, tota la investigació anterior es resumirà en mètodes de visualització i avaluació de la incertesa reportada per tal de produir mètodes que mitjançant aquesta informació extra prenguin decisions més robustes.

Resum

Vivim en un món intrínsecament incert, des de la nostra perspectiva. Desconeixem què passarà en el futur però, per a inferir-ho, construïm els anomenats models. Aquests models són abstraccions del món on vivim que ens permeten concebre el funcionament d'aquest i que, essencialment, es validen a partir de la nostra experiència prèvia i es descarten si les seves prediccions es demostren incorrectes en el futur. Aquest procés habitual d'inferència en la ciència té diversos passos que no són deterministes.

Primer de tot, els nostres instruments de mesura podrien ser imprecisos. És a dir, que la informació que usem a priori per a saber què passarà ja pot contenir un cert error irreductible.

A part, la nostra experiència passada per a construir el model podria estar esbiaixada (i, per tant, inferiríem incorrectament el futur ja que els models estarien basats en dades no representatives).

Per altra banda, el nostre model en si pot ser una simplificació massa exagerada de la realitat (el qual ens conduiria a generalitzacions no realistes).

Inclús, la mateixa tasca d'inferir el futur pot ser pròpiament no determinista. Això passa sovint quan la informació que es té a priori per inferir el futur és incompleta o parcial per a la tasca a realitzar (i.e. depèn de factors que no podem observar en el moment de predir) i, per tant, estem obligats a considerar que el que volem predir no és un valor determinista.

Una forma de modelar totes aquestes incerteses és mitjançant un enfoc probabilístic que formalitzi matemàticament aquestes fonts de la incertesa per tal de crear mètodes específics que les capturin.

L'objectiu general d'aquesta tesis es usar un enfoc probabilístic que contribueixi en que els sistemes basats en intel·ligència artificial (en concret, en aprenentatge profund) es converteixin en sistemes robustos i fiables capaços de ser aplicats en problemàtiques d'alt risc, on predir generalment sense errors no és suficient sinó que cal evitar els errors crítics amb alts costos. En particular, la tesis mostra l'actual divergència que hi ha a la literatura - a l'hora de dividir i anomenar als diferents tipus d'incertesa -, proposant procediment a seguir. A més, a partir d'un cas real provinent de la part industrial de la tesis, es remarca la importància d'investigar el darrer tipus d'incertesa provinent de la falta d'informació a priori per a poder inferir determinísticament el futur, la anomenada incertesa aleatòrica. Sobre aquesta incertesa, la tesis aprofundeix sobre els diferents tipus de models actuals per tal de capturar-la mitjançant xarxes neuronals i analitza les limitacions que tenen. A més, proposa noves aproximacions que permeten solucionar algunes de les dificultats exposades durant la tesis.

Fruit de la modelització de la incertesa aleatòria en problemàtiques reals sorgeix la modelització de la incertesa de sistemes de caixa negre. Els sistemes de caixa negre són sistemes predictius ja existents els quals no modelen la incertesa i on no es realitza cap assumpció sobre el seu funcionament intern. Per tant, l'objectiu serà construir un nou sistema que l'envolti i permeti modelar la incertesa d'aquest sistema original. En aquest escenari, no totes les aproximacions prèviament introduïdes de modelatge de la incertesa aleatòrica poden ser aplicades i això implica que mètodes flexibles com la regressió quantílica hagin de ser modificats per tal de poder ser aplicats en aquest context.

Posteriorment, l'anàlisi en detall de la regressió quantílica portarà a estudiar i proposar diferents solucions per una problemàtica crítica en la literatura de regressió quantílica, l'anomenat creuament de quantils.

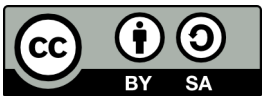
Finalment, tota la recerca realitzada anteriorment es sintetitzarà en mètodes de visualització i avaluació de la incertesa predita per fer ús d'aquesta informació.

Dedicació

Sempre un intenta mirar endavant. No plantejar-se objectius com desenllaços sinó com part d'un camí que ha de continuar. Però hi ha certs moments que és important fer retrospectiva i donar la rellevància que ha tingut el camí traçat; per aprendre, no oblidar i, sobretot, agrair als que ens envolten o han envoltat. Aquest agraïment obert té noms i cognoms i, de ben segur me'n deixaré d'essencials. Tot i això, tothom qui em coneix sap que no hi ha cap dubte que haig de començar amb qui m'ho ha donat tot: Gràcies per ajudar-me incondicionalment, Marisa, per la teva persistència i integritat que sempre has tingut, on també hi veig reflectit l'avi, sense cap dubte. Al Martin, per ensenyar-me a prioritzar les coses importants a la vida i a la família, en general, amb especial menció a la Sesi, per ser la millor padrina, i l'Oriol i l'Esteve per reafirmar-me que la família s'escull. En aquest trajecte, no puc oblidar a en Miquel Àngel, per la seva vocació i constància en donar-me suport durant molts anys. I és que aquests anys han estat molt intensos... Però res hagués estat igual sense tu, Joan, per tots els moments compartits i els que vindran, ni sense en Fèlix (per les bogeries), la Mònica (per sempre ser-hi), en Carles (per saber mantenir la amistat), la Marta (per ensenyar-me a fluir i créixer), la Esther (per tot el que estem construint), en Fran (per ser un bon amic), la Eli (perquè sé que sempre podrem reprendre les converses on les vam deixar), l'Aram (per ser el millor company), l'Edu (per les "rises"), la Fra ("Chissà", jo sí), la Carla (per la il·lusió que em desperta pensar en el futur), la Itziar (per la intensitat inexorable), la Lorena (per totes les converses pendents), la Natasha (per sempre connectar, malgrat la distància), l'Andreu (per la teva noblesa), "la" María (per

tots els moments compartits), l'Andrei (per aguantar-me en moments baixos), en Guillermo i en Damià (pels moments d'apofènia absoluta), l'Alejandro (per saber que puc comptar amb tu) i en Noel (per les llargues converses i mai oblidar-nos dels moments importants), també a l'Hèctor, Toni i Jonathan (per tots els projectes junts passats i futurs) i a l'Aleix i en Toni (pels que no arrenquen). També, voldria destacar els/les companys/es del departament, en especial, en Carlos, la Mariona i la Paula (per (no) haver donat us a aquelles RPi), en Pere i en Pablo (per haver-nos conegut recentment) i, també el suport imprescindible de la Ino. No me n'oblido pas dels (ex-?)companys de feina, entre tots/es: destacar l'Alberto, en Jordi, en Joan, en Luís, en José Luís i la Irene, així com en Jesús. Ni tampoc dels actuals, que també voldria destacar per haver-me ajudat en tot en aquest últim tram i per tots els que vindran: a en Fran, la Isabel i en Jaume. Tot això, em porta a mencionar els meus tutors, en Jordi i en Jose, qui junts m'han ensenyat a "surfejar" grans reptes "en paral·lel" i que espero que puguem continuar caminant.

A tothom que m'ha ajudat en aquest tram del viatge, gràcies per formar-hi part. Continuem.



Contents

Chapter 1

INTRODUCTION

1.1 Motivation and main objectives

Autonomous forecasting systems are increasingly being used to fulfil critical tasks, where the cost of an erroneous decision is significantly greater than the benefit of obtaining a generally good performance. What happens when the system makes a significant error in cases like these? The common approach is to associate such an error to the the "lack of estimation capabilities" of that system, which leads to an attempt to improve its performance without changing the prediction task, e.g. without adding the possibility to detect ambiguous cases. However, what if this imprecision derives from unknown exogenous variables affecting the variables to be predicted? In this work, we argue that uncertainty modelling must be viewed as an essential part of any critical forecasting system in order to detect such cases.

Figure 1.1: Illustrative critical tasks where autonomous systems can be used but need to model the associated uncertainty in each forecasting process.

Rather than focusing on the model's lack of complexity, which avoids forecasting scenarios that are inherently misleading, the current work enhances the prediction task by providing extra information that helps detect doubtful scenarios, i.e. where there are several possible valid answers given the same input variable values. Such scenarios are present in financial problems - the motivation behind this work - or situations where a variable that influences the response variable value is, for instance, a human decision based on spurious criteria.

Given this context, the current work will consider Deep Learning (DL) models as appropriate universal function estimators to address the following two issues:

1. How uncertainty modelling can be performed using such DL models.
2. How the high flexibility of such DL models can be used to enrich the uncertainty modelling process of even non DL-based predictive systems.

1.2 Contribution and thesis outline

The current work was mainly carried out as part of an industrial PhD, which consists of a Research, Development and Innovation (R&D&I) project aimed at enhancing knowledge transfer collaboration between a university or research centre (in our case, the University of Barcelona) and the industrial sector (in our case, BBVA Data & Analytics, the BBVA bank's data science spin-off, which now forms part of the bank and goes by the name of BBVA AI Factory).

The industrial context for this thesis prompted us to focus on regression problems where the tackled financial data sets are assumed to have a large size (millions of data points), the input variables are high dimensional and the output variable is typically a continuous one-dimensional variable to be predicted. Specifically, one of the main industrial problems tackled in this thesis is forecasting customers' upcoming expenses and incomes given the historical financial data from their account by identifying such uncertain scenarios and providing task-tailored methods based on the reported level of confidence.

The constraint of working with large scale data sets has led us to develop all of the theory explained later in this thesis using one of the state-of-the-art

models for these kinds of problems: the deep learning or the neural networks models. That being said, all of the uncertainty-estimation proposals presented in this work are intended to be agnostic regarding the kind of model used, and any parametric model that can be optimized via the derivative with respect to their parameters can therefore be easily applied with the solutions proposed in Chapter 3 and Chapter 4.

Aside from its industrial focus, the current research directly outlined several contributions in conferences, journals, event talks, workshops, presentations, white papers, newspaper and blog articles as well as impacting several of the company's internal products. Details of some of the public results and presentations are provided below, in chronological order:

In Chapter 2, we explore the reasons for studying uncertainty modelling, introduce relevant deep learning concepts required for the following sections, present the state-of-the-art regarding uncertainty types proposed by several competitors, discuss two of the main kinds of problems we will be tackling and explain why we have focused on aleatoric uncertainty. Then, in Chapter 3, we focus on how to model aleatoric uncertainty using deep learning (from a parametric and non-parametric distribution point of view). Following this, in Chapter 4, we introduce the black-box uncertainty modelling problem as an improvement to post-hoc uncertainty modelling. In Chapter 5, we focus on a certain limitation that appears in Quantile Regression models that predict several quantiles simultaneously - the crossing quantile phenomenon. Then, in Chapter 6, we highlight the importance of visualizing and correctly evaluating the forecast uncertainty, and finally, we recap all of the results presented in this doctoral thesis in Chapter 7.

TABLE 1.1 Timeline

2018/03	Maths for Industry 4.0, how to develop applied research. Highlight the importance of creating applied research that serves as a link between the academic and the industrial efforts. (Brando et al., 2018d).
2018/05	Presentation at first BCN.ai event. Presentation regarding the state of the art of uncertainty modelling using Deep Learning techniques for regression and classification problems. (Brando et al., 2018b).
2018/09	Paper at the 16th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) A comparison was made of several types of uncertainty modelling in financial problems, showing that aleatoric uncertainty prevails over other models. (Brando et al., 2018a).
2018/09	Presentation at Machine Learning Summer Schools (MLSS). Poster presentation of research done regarding aleatoric uncertainty modelling for expenses and income forecasting. (De Pablo et al., 2018).
2018/10	White paper between Google Cloud and BBVA. White paper presenting collaboration between Google Cloud and BBVA. The developed prediction engine uses the uncertainty modelling devised previously. (Maestre et al., 2018; Mejia et al., 2018).
2018/12	Blog article on the BBVA website. Blog article highlighting why it is important to model uncertainty in income and expenses forecasting. (Brando et al., 2018e).
2019/07	Paper at the ninth Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA). Extension of black-box uncertainty modelling for regression models to classification systems. (Mena et al., 2019).
2019/08	Workshop article for the anomaly detection workshop at the Knowledge Discovery Data Mining Conference (KDD). Presentation of the internal use case to detect highly uncertain forecasting scenarios for the expenses and incomes prediction problem. (Brando et al., 2019a).
2019/10	Presentation at the BCN.ai event. Presentation of UMAL as an aleatoric conditional model to approximate heterogeneous distributions. (Brando et al., 2018c).

Table 1.2 Timeline

2019/12	Paper at the thirty-third Conference on Neural Information Processing Systems (NeurIPS) UMAL is proposed as a new method to estimate aleatoric uncertainty in a flexible manner. (Brando et al., 2019b).
2019/12	"Las redes neuronales tienen derecho a no poner la mano en el fuego". Newspaper article in the Retina Section of El País. (Hidalgo, 2018).
2020/06	Journal article for IEEE Access. Proposal of the black-box uncertainty problem applied to regression scenarios. (Brando et al., 2020).
2020/08	Workshop article for the financial workshop at the Knowledge Discovery Data Mining Conference (KDD). Proposal of a new iterative model which is also able to propagate the uncertainty modelling it performs. This is applied to tackle a different internal problem related to customers' generic balance forecasting. (Muelas et al., 2020).
2021/08	Poster at the Eastern European Machine Learning (EEML) summer school. Poster presentation of UMAL as a solution for heterogeneous modelling. (Brando et al., 2022a).
2021/11	Paper at the twenty-fifth International Conference on Artificial Intelligence and Statistics (AISTATS). Tackling the crossing quantile phenomenon by means of constraining the derivative of the predicted function with respect to the quantile variable. (Brando et al., 2022a).
2022/04	(Under revision) Journal article for the Journal of Machine Learning Research (JMLR). CheNet is proposed as a solution for the Explicit Modelling of the Black-box Uncertainty problem. (Brando et al., 2022b).

Chapter 2

THE RESEARCH CONTEXT AND GOALS

Nowadays, forecasting systems are commonly used to automatically tackle problems that previously required human intervention. This clearly involves ethical aspects (Hevelke and Nida-Rümelin, 2015; Lin, 2016; Maxmen, 2018), especially when wrong decisions imply significant costs. The range of fields where such systems are applied includes, for instance, medicine (Deo, 2015; Rajkomar et al., 2019), industrial processes (Diez-Olivan et al., 2019), self-driving cars (Michelmore et al., 2018; Stilgoe, 2018; Tunga et al., 2021), AI protein folding (Jumper et al., 2021; Noé et al., 2020) and autonomous financial decisions (Culkin and Das, 2017; De Spiegeleer et al., 2018). From an engineering modelling viewpoint, forecasting systems are models conceived to be the closest as possible to the real process to be approximated. However, this approximation is not usually perfect. Thus, there is a need to analyse the reliability of these systems in order to identify troublesome situations and, in all likelihood, act in a special way. For instance, requiring expert human supervision for such cases.

Uncertainty is often simply understood in terms of variability with respect to the prediction. In the present work, we wish to emphasize the importance of developing a formal methodology to identify sources of uncertainty. This will prove crucial in designing solutions based on uncertainty-source performance and, consequently, producing consistent risk-based decision-making.

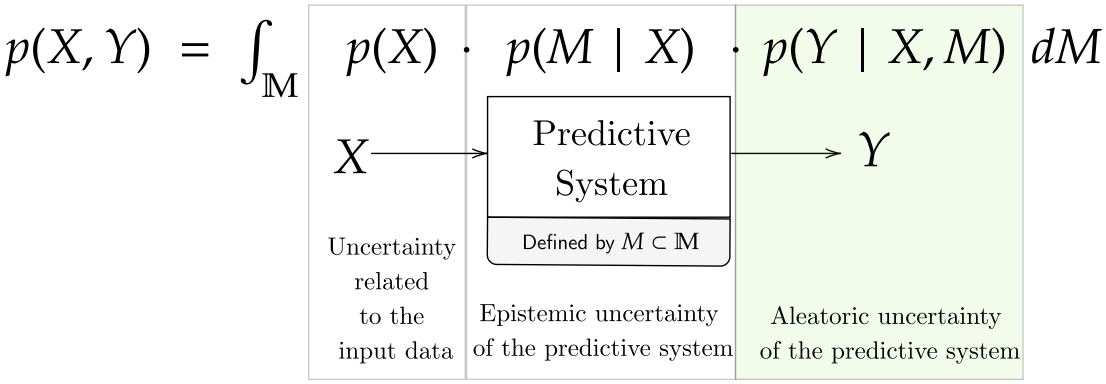


Figure 2.1: Probabilistic sources of uncertainty originated before or while the predictive system is being used. The novel aspects of this thesis correspond to the aleatoric uncertainty of the response variable Y given X, highlighted in green.

In supervised learning, the initial scenario involves having a data set,

$$\mathcal{D} = (X, Y) = \{(x_i, y_i)\}_{i=1}^{N}, x_i \in \mathbb{X}^I, y_i \in \mathbb{Y}^O, \tag{2.1}$$

where $\mathbb{X}$ and $\mathbb{Y}$ are spaces of dimensions I and O, respectively. $\mathcal{D}$ contains two different types of data: x_i, which correspond to the data points used as input variables for the predictive system, and y_i, which are assumed to depend on x_i. Although the aim of the predictive system will be to imitate this dependence, in most of the problems, this "simple" goal implies capturing different types of uncertainties, as Figure 2.1 shows.

As a rule, structural reliability or risk analysis problems use probability theory to quantify uncertainty. Specifically, Bayesian inference is used to disentangle the different sources of uncertainty, which include intrinsic variability in the measurement, but also ignorance in the learning process itself.

Following the notation introduced above, we will assume that X and Y are random variables, where the joint distribution of all the presented variables is $p(X, Y)$. Given that we assumed that Y depends on X, expressing it as conditional probability we obtain $p(X, Y) = p(Y \mid X)p(X)$, where $p(X)$ corresponds to a marginal distribution that represents the likelihood of the input data itself[1].

[1]Unless explicitly stated, the new evaluated value, x^*, or the value to be predicted, y^*, are implicitly considered as an abuse of notation. Specifically, when $p(Y \mid X, M)$, it means the

To calculate the latter, an additional model could be considered, such as a density estimation model with respect to X. Although this corresponds to a source of uncertainty that can ultimately affect the system prediction, the goal of this work is to analyse the uncertainty of the predictive system and not that produced by a new type of input data. Thus, although we are aware that this uncertainty is also crucial, this kind of uncertainty, which includes the irreducible noise in X or the presence of outliers in the input data, will not be considered in the framework presented here.

On the other hand, the relationship between X and Y is modelled using the predictive system. Importantly, a mismatch between the best model, usually defined by some model properties, and the real predictive relationship between X and Y could produce an extra source of uncertainty. Given that these properties define the model, we need to consider all the possible combinations of properties in the properties-space, $\mathbb{M}$, that defines the predictive system, each one denoted as M. For instance, parametric models are normally defined by a collection of hyper-parameters and parameters or weights. In such cases, the conditional distribution can be expressed as

$$p(Y \mid X) = \int_{\mathbb{M}} p(Y, M \mid X) dM = \int_{\mathbb{M}} p(Y \mid X, M) \cdot p(M \mid X) dM \tag{2.2}$$

where the integral is present due to the number of combinations usually being uncountable (e.g. if one of the model properties is a real number).

Considering Eq. (2.2), our aim is to estimate the properties, M, that maximize the overall conditional probability. Following (Der Kiureghian and Ditlevsen, 2009; Kendall and Gal, 2017), we will denote:

- $p(M \mid X)$ as the **epistemic** source of uncertainty. The term epistemic comes from the Greek "episteme", which corresponds to the concept of knowledge. This uncertainty is reducible, given that the more input data points we have about the relationship between X and Y, the more the

variable to be predicted is y^*. When X is given - e.g. $p(M \mid X)$ and $p(Y \mid X, M)$ - it means the whole data set $\mathcal{D}$ is given except for the only input-dependent $p(x^* \mid X)$, included in $p(X)$.

model will behave like the original predictive process, if the family of models defined by $\mathbb{M}$ is rich enough.

- $p(Y \mid X, M)$ as the **aleatoric** source of uncertainty. Similarly to the previous case, aleatoric comes from the Greek term "alea", which means dice. This kind of uncertainty is considered inherently different, because it is assumed to be irreducible due to probabilistic variability. Consequently, we need to model the conditional probability distribution of Y given X for a certain M, and this will not decrease even providing new data.

Epistemic and aleatoric uncertainty are commonly defined according to how assumptions are selected, i.e. answering the questions: can the model be rich enough to perfectly imitate the approximated real process? Or, is there an inevitable lack of information or intrinsic randomness regarding the response variable that forces us to model it as a random variable?

Epistemic uncertainty has traditionally been the main type of uncertainty to be tackled in the literature due to the common hypothesis that "the system is not rich enough to approximate the real process". As a consequence, a huge range of alternatives, mostly based on Bayesian methods, have been proposed to improve the modelling of this kind of uncertainty, as we will detail in Section 2.3. However, in the different analysis we presented publicly as part of this thesis and within the company (Brando et al., 2018a,e, 2019a), we have seen that there are problems in which the main source of uncertainty comes from the intrinsic randomness of the response variable conditioned to the input value. For instance, one of the problems analysed was the monthly-aggregated incomes/expenses forecast of bank users described in Section 2.5 and presented in Brando et al. (2018a); Ciprian et al. (2016). Assuming the source of the main uncertainty to be epistemic would imply that the reason for incorrectly predicting the next aggregated movement originates from the use of a model that is not rich enough. However, we know - and the assumption therefore makes sense - that the input information available to the model is only partially informed to make this complex forecast, i.e. spurious behaviours and decisions are made by the bank's client that are not encoded considering only the last n-months of movements. This means that we find an

intrinsic variability of $p(Y \mid X, M)$ here that cannot be reduced and, thus, must be modelled.

In short, we have seen that the impact of modelling aleatoric or both (aleatoric and epistemic) uncertainties is clearly critical for the kind of problems mentioned here. This is due to there not being enough available input data to predict the response variable and obtain a unique family of models, or those that maximize $p(M \mid X)$, which individually perform an appropriate point-wise prediction.

2.1 Uncertainty nomenclature disagreement

Understanding the concept of uncertainty is not strictly a data science or machine learning dilemma. In fact, there is a science known as *uncertainty quantification*, which characterizes it as being used by experts in many fields, including physics, finance, medicine, psychology and metrology; even the ancient Greece philosophers pondered over it. In this characterization, a common question is to define what constitute the uncertainty sources in a forecasting process.

Taking one of the widely accepted classifications of uncertainty as a starting point, the initial aim of this section, avoiding the previously introduced uncertainty types,is to present part of the disagreement that exists in the literature regarding the possible taxonomies of uncertainty. In particular, what we wish to highlight in this section is that there is no perfect classification, but rather certain common ideas that help us understand the sources of uncertainty. As in the previous section, our uncertainty sources will be directly linked to a mathematical term (a probability), leaving aside the discussion on terminology as an important topic but not one crucial to the overall proposals presented in this research.

In a forecasting scenario, uncertainty is considered to be the consequence of a lack of knowledge regarding the possible future. This absence exists because we do not have enough information to infer the future in a deterministic way, and a probabilistic approach can therefore be considered: in such a prediction process, we will use a certain input information (set of *data points* with their *attributes*) to infer the future. Consequently, if we do not have enough information, there are two different scenarios that define the two most common uncertainty terms:

- **Epistemic uncertainty**: This appears when the new input point used to forecast the future is unknown, meaning it is a new point completely differing from previous ones. Therefore, this kind of uncertainty can be resolved when sufficient new sets of points are provided - because the new anomalous point will stop being so - and, consequently, this uncertainty is **reducible**.
- **Aleatoric uncertainty**: This emerges when the attributes that define the input point are not sufficient to perfectly know what to forecast, i.e. there exist several valid predictions given the same input attributes. Thus, considering more new data points will not solve the lack of attributes to reduce such uncertainty and, therefore, this uncertainty is **irreducible**.

In fact, the irreducible property is controversial in fields such as physics, since it can be argued that there are always hidden variables that allow such uncertainty to be reduced. Ultimately, the aforementioned terminology may make more sense when we are faced with a problem in which it is not possible to add new covariables (or attributes) to the input information. This assumption allows us to divide uncertainty sources into those which can be reduced and those which, crucially, can only be detected or modelled as ambiguity.

Given the above, aleatoric uncertainty can be related to an unavoidable occlusion, an inherent stochasticity or, equivalently, a scenario that is essentially partially observable. In the present research, we focus on modelling conditional aleatoric uncertainty - i.e. given a certain input, the possible variability that exists in the response variable to be predicted. However, if we define aleatoric uncertainty as simply the "irreducible uncertainty type" then this does not need to be conditional, i.e. an unavoidable precision error of measurement in the gathering of input values constitutes aleatoric uncertainty under these terms.

Similarly, the epistemic uncertainty related to our lack of knowledge can be conditioned to the predictive model or the data themselves as two different variables: on the one hand, when a certain model is used for the forecasting process, this implies assuming some restrictions, which avoids obtaining the real prediction process. As a consequence of this lack of knowledge, several diverse predictive systems together as an ensemble can provide a more confident forecast than a

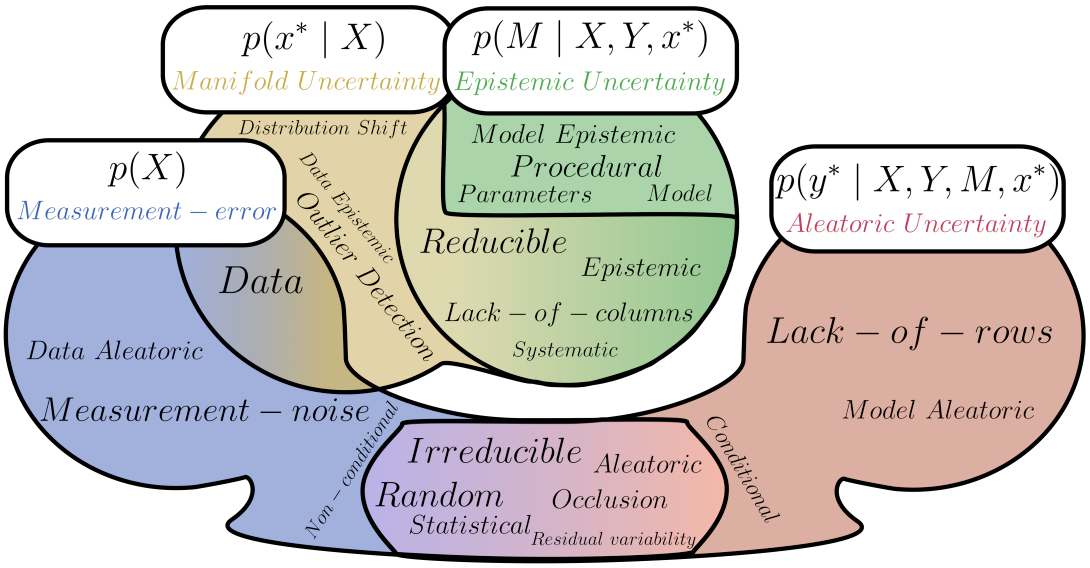


Figure 2.2: Summary of common uncertainty sources terms linked to their simplified mathematical formulation. Each colour represents one uncertainty type. Shared zones corresponds to terms that are commonly used for different uncertainty types. Each white box has our proposed notation (matching Figure 2.1).

single one, which will be more prone to overfitting. We therefore need to consider that a certain prediction made using a certain model inherently assumes the model's limitations, and that this produces an epistemic kind of uncertainty, which also includes the procedural variability arising from the training procedure. On the other hand, if we focus only on the input data information in a supervised learning task, when the previous data set to train our model differs from the new data to be tested, this constitutes another source of lack of knowledge, i.e. we could evaluate the predictive system in an outlier with respect to our previous knowledge, which produces an unreliable outcome.

The particular features listed above - and especially the fact that different meanings of uncertainty use the same terminology - exacerbates the debate on the appropriate use or not of the terms "epistemic" and "aleatoric" for this disentanglement. With the aim of clarifying these meanings, Figure 2.2 provides a short compilation of equivalent ways of referring to each of these uncertainty types and, importantly, a link to the mathematical notation previous introduced

in Figure 2.1. These terms corresponds to the ones that usually precede the word "uncertainty". In particular, the common areas between the different types of uncertainty shown in the Figure 2.2 correspond to those terms that are used both to refer to one or another type of uncertainty, which leads to the previously highlighted confusion and debates. For example, the "aleatoric" uncertainty term can be used simultaneously for uncertainty our measurement-error and for our aleatoric uncertainty but the former is a non-conditional probability and the latter is a conditional one.

2.2 Relevant Deep Learning Concepts

This section is not intended as an introduction to Deep Learning, as in Bishop (1994a); Chollet (2017); Goodfellow et al. (2016b); LeCun et al. (2015); Murphy (2012). Rather, the aim is to highlight relevant concepts present in the deep learning or neural networks[1] literature that should be taken into consideration for this work.

Although deep learning can be used in generic function estimation, in this work we focus on proposing solutions to supervised learning problems. Following the notation in Eq. (2.1), given a data set $\mathcal{D} = (X, Y) = \{(x_i, y_i)\}_{i=1}^{N}, x_i \in \mathbb{X}^I, y_i \in \mathbb{Y}^O$, this will be divided into three different randomly selected splits denoted as

1. The training set: The subset of data that will be used to optimize the neural network model.

2. The validation set: The subset of data that will be used to decide when to stop optimizing the neural network.

3. The test set: The subset of data that will be used to verify the quality of the performance or accuracy of the neural network.

In this context, a linear model is a transformation $\hat{y}_i = W \cdot \boldsymbol{x}_i$ with $W \in \mathbb{R}^{I \times O}$, $\boldsymbol{x}_i \in \mathbb{X}^I$ and $\hat{y}_i \in \mathbb{Y}^O$, where the goal is to find the weight W, which minimizes

[1] In this text, the term "deep learning" or "neural network" will be used indifferently.

a certain distance function (known as the loss function) between the predicted output and the real response variable value with respect to the training split set. The neural network is still a parametric function

$$\begin{aligned} \phi\colon\ & \mathbb{X}^I \to \mathbb{Y}^O \\ & \boldsymbol{x}_i \mapsto \hat{y}_i, \end{aligned} \tag{2.3}$$

but, instead of a single matrix multiplication, it considers several internal **non-linear transformations** from the input, $\boldsymbol{x}_i$, to produce the output variable, $\hat{y}_i$. Each of these transformations is known as a "layer", which combines its input layer values with its weights to produce its internal layer output. Generically, the neural network combines a mixture of weights and these input values to minimize a certain distance function (known as the loss function) between the predicted output and the real response variable value with respect to the training data set. As we will describe later, this loss function is usually the Mean Square Error (MSE), i.e. $\mathcal{L}(X, y) = \sum_{i=1}^{N} (y_i - \phi(\boldsymbol{x}_i))^2$. Typically, the optimization process is performed using a kind of gradient descent method, calculating the derivative of the loss function with respect to each weight; this will be analysed in greater depth in the following sections.

2.2.1 Defining the architecture

How the input values are combined with the weights in a neural network is defined by its architecture: each neural network is defined by different stacked transformation processes, with each transformation being known as a layer. For instance, a so-called "dense layer" is a combination of all of the layer inputs, h_{i-1}, to provide the outputs of the layer h_i. This is equivalent to the linear model but, in addition, a non-linear function σ, known as the activation function (Goodfellow et al., 2016b), is applied in order to estimate non-linearities, i.e. $h_i = \sigma(W \cdot h_{i-1})$. Although the sigmoid function (Han and Moraga, 1995), hyperbolic tangent (Anastassiou, 2011) or ReLU (Fukushima, 1969; Glorot et al., 2011a; Hahnloser et al., 2000) functions are the most common activation functions used, there are plenty of alternatives to choose from. Similarly, specific

layers, understood as transformation processes, are designed for a certain type of data: for instance, images are commonly stored as a matrix of pixel values, where their patterns are combinations of those present in different parts of the image, and spatial proximity therefore matters. In these cases, layers such as convolutional layers (Fukushima, 1988; LeCun et al., 1989), which perform a space invariant operation - known as convolution - that is applied over the input features, providing a translation equivariant response - known as the feature maps - and learning their corresponding mask (or kernel), are suitable for these types of problems. Alternatively, when a temporal relationship exists, other layers such as recurrent layers e.g. the GRU (Cho et al., 2014) or the LSTM layer (Hochreiter and Schmidhuber, 1997) allow the network to capture useful relationships more easily because they impose temporal-constraints compared to a standard dense layer.

All of these design decisions regarding the neural network architecture (such as the number and type of layers or the number of hidden output dimensions for each layer) are normally taken prior to the training process (Goodfellow et al., 2016a), which is why they are considered hyper-parameters. Given that neural networks are considered universal function approximators (Hornik et al., 1989; Zhou, 2020), the common goal is to tackle the desired problem with a rich enough architecture that it can learn a combination of weights to obtain the expected performance in the test set.

2.2.2 Defining the loss function

Following Goodfellow et al. (2016a); Smithing (1999), the function representing the cost or error between the model - mainly defined by its parameters or weights, w - and the evaluated data is called the "loss function". Specifically, the learning process takes place when we minimize the expected value of this evaluated function for all of the evaluated points. From a statistical viewpoint, this optimization process using the loss function can be understood as a maximization process of the posterior probability of w, $p(w \mid X, Y)$, which relates the data and a desired distribution with respect to the model parameters. Applying the Bayes' rule,

$$p(w \mid X, Y) = \frac{p(Y \mid X, w) \cdot p(w)}{p(Y \mid X)} \tag{2.4}$$

we can see that to obtain the posterior probability it would be necessary to calculate the likelihood of w, $p(Y \mid X, w)$, and the prior probability of w, $p(w)$, and in some way tackle the so-called evidence, $p(Y \mid X)$.

The frequentist viewpoint performs different assumptions. Firstly, it considers the weights of the neural network as deterministic variables. Therefore, the previous prior distribution is not considered. Secondly, the evidence is also not considered, given that it is a constant with respect to the weight optimization. This process is known as Maximum Likelihood Estimation (MLE) and corresponds to

$$\boldsymbol{w}^{MLE} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\ p(Y \mid X, \boldsymbol{w}). \tag{2.5}$$

Additionally, if only the first assumption is relaxed - i.e. the weights of the neural network are considered as random variables - this is known as Maximum A Posteriori (MAP) and corresponds to

$$\boldsymbol{w}^{MAP} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\ p(Y \mid X, \boldsymbol{w}) \cdot p(\boldsymbol{w}, \tag{2.6}$$

where the additional prior term can be considered as a regularization term. In fact, if the assumed prior distribution corresponds to a standard normal distribution, this is also known as l_2-regularizer (Moore and DeNero, 2011; Wu et al., 2017) or l_1-regularizer in the case of a standard Laplace distribution, as in Schmidt et al. (2007).

In the MLE or MAP approach, $p(Y \mid X, w)$ is typically approximated considering the logarithm of a normal distribution with a constant scale parameter (as shown in the introduction to Chapter 3). That is to say, the prediction of the neural network is used as the location parameter, and the neural network consequently learns the conditional mean,

$$\mathcal{L}\left(\boldsymbol{w}; \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \phi(\boldsymbol{x}_i)\right)^2. \tag{2.7}$$

This method is commonly denoted as Mean Squared Error (MSE), Least Squares Deviation (LSD) or Mean Squared Deviation (MSD) (Pishro-Nik, 2020; Vault, 2020). Similarly, we must consider that the logarithm of a Laplace distribution with constant scale parameter approximates the conditional median and corresponds to the - also well-known - Mean Absolute Error (MAE) or Least Absolute Deviation (LAD); that is

$$\mathcal{L}\left(\boldsymbol{w}; \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \phi(\boldsymbol{x}_i)|, \tag{2.8}$$

which provides results that are more robust to outliers and more interpretable than the commonly used MSE (Chai and Draxler, 2014; Willmott and Matsuura, 2005) (as shown in detail in Chapter 3). However, for the continuous variable y, the conditional mean or median represents only a very limited statistic. As Bishop (2006) stated and we demonstrate in this thesis, in many scenarios it is considered beneficial to obtain a much more complete description of the response variable's probability distribution. This gives rise to one of our main reasons for proposing the more complex frequentist solutions described in Chapter 3.

2.2.3 Minimizing the loss function for the neural weights

Once our neural network architecture and loss function are decided, we need to find a way to minimize the error function to modify the weights in order to obtain an expected result. In order to do this, we need to calculate the derivatives of the loss function with respect to the weights in the neural network. According to (Bishop, 1994b), one way of solving this problem is to use the standard *back-propagation* procedure, provided we obtain suitable expressions for the derivatives of the error function with respect to the activations of the neural network's output units. The requirement here is that the loss function must be differentiable with respect to the outputs of the neural network present in this function. The

derivatives act as *errors* that can be back-propagated through the network to find the derivatives with respect to the network weights. Much has been written about this process of optimization, including Bishop (2006); Goodfellow et al. (2016a); Nielsen (2015). As (Bishop, 1994b) noted, standard optimization algorithms such as conjugate gradients or quasi-Newton methods can then be used to find a minimum of the loss function, $\mathcal{L}$. Alternatively, if an optimization algorithm such as stochastic gradient descent is used, the weight updates can be applied after the separate presentation of each pattern. In recent years, many new gradient descent optimization algorithms have been developed, such as the Nesterov accelerated gradient, Adagrad, Adadelta, RMSprop and Adam (Ruder, 2016).

Nowadays, this differentiation process is implemented in most relevant deep learning libraries, which allows native code to be differentiated automatically. As with the Autograd Library (Dougal Maclaurin and Johnson, 2016) - or its new version, JAX (Bradbury et al., 2018) - most libraries use reverse-mode differentiation (also called *reverse accumulation*[1]). By using these libraries, we can therefore efficiently take gradients of scalar-valued functions with respect to array-valued arguments, thus simplifying the gradient-based optimization problem and allowing us to focus on other aspects.

2.3 Deep epistemic uncertainty modelling

2.3.1 Introduction

Once we have established a minimum deep learning notation - and we know that these neural networks are defined by their architecture, weights and other hyper-parameters - then we can return to the uncertainty modelling discussion (introduced in Chapter 2).

Regarding epistemic uncertainty modelling, following the concept proposed in Section 2.1, different methods can be found in the literature that tackle this igno-

[1]The process is explained in page 7 of the Ilya Sutskever's PhD thesis.

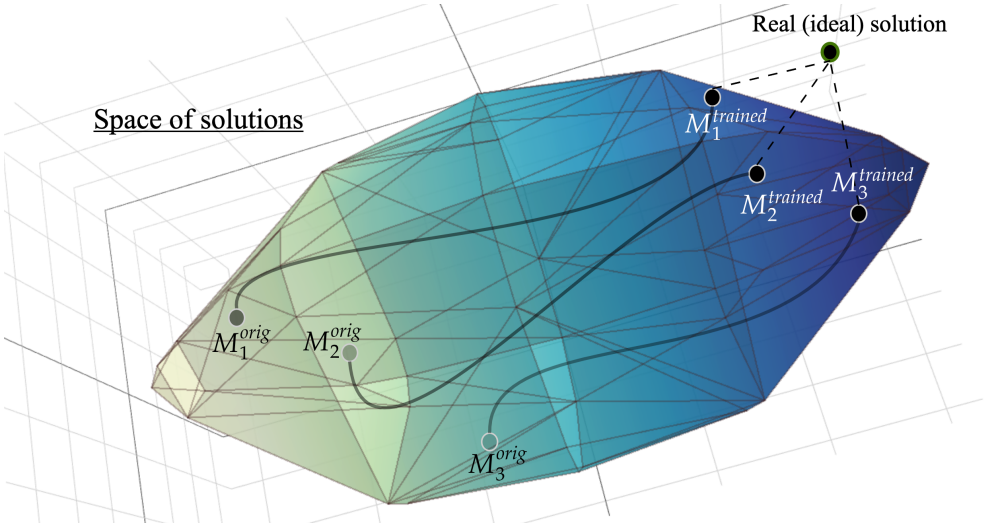


Figure 2.3: Synthetic representation of an enclosed solutions space given a certain family of models $\mathbb{M}$ and how different initial models M_1, M_2, M_3 end up obtaining similar errors compared to the real solution despite being different. Epistemic uncertainty aims to exploit this variability.

rance regarding the correct model to choose[1] (as represented in Figure 2.3). For instance, the weights of the neural network can be considered as random variables and Variational Inference (VI) (as shown in in (Blei et al., 2017; Blundell et al., 2015)) can be applied to learn their parametric distributions, or the Dropout technique used as a Bayesian inference training process (Gal and Ghahramani, 2016), or even an ensemble of neural networks considered and properly combined (Lakshminarayanan et al., 2017). In the following sections, we will describe how these approaches are implemented to highlight the difference between the main proposals in this work.

2.3.2 Bayesian neural networks using variational inference

In this subsection, we will describe how to create a Bayesian neural network using VI as the method to optimize their weights. Considering $\boldsymbol{w}$ the parameters of a standard neural network ϕ, since we do not know which family of distribution our posterior - $p(\boldsymbol{w} \mid X, Y)$ - belongs to, VI chooses one that is plausible and tractable

[1] This phenomenon is related to the concept of "indentifiability" in statistics.

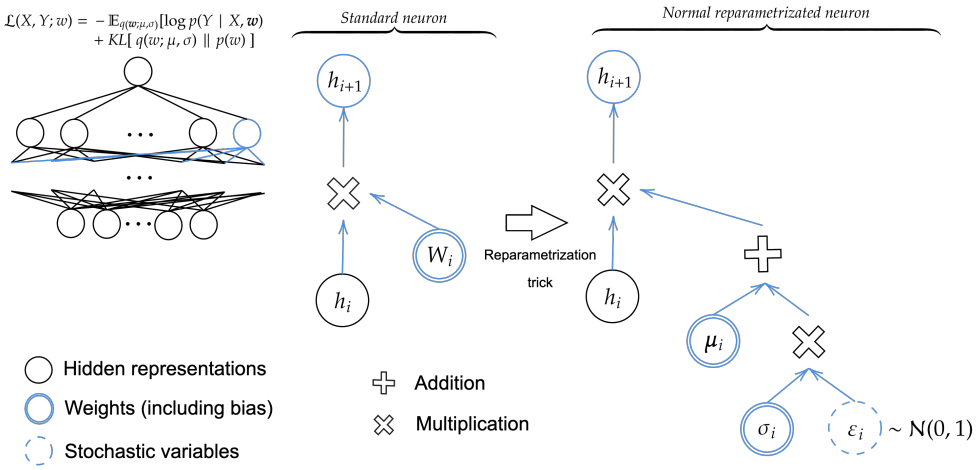


Figure 2.4: Graphic representation of how to modify the neurons of a standard neural network to make it a Bayesian neural network using the reparametrization trick.

and calls it q. This distribution q has some parameters θ. The variational view finds the parameters θ of a distribution on the weights $q(\boldsymbol{w}; \theta)$, which minimize the Kullback-Leibler (KL) divergence (Blei et al., 2017) with the true Bayesian posterior on the weights,

$$\theta^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathrm{KL}[q(\boldsymbol{w}; \theta) \parallel p(\boldsymbol{w} \mid X, Y)] = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \underbrace{\mathbb{E}_{q(\boldsymbol{w};\theta)}[\log q(\boldsymbol{w}; \theta) - \log p(\boldsymbol{w}, Y \mid X)]}_{-\mathrm{ELBO}(\theta)} + \log p(Y \mid X). \quad (2.9)$$

Due to the evidence $\log p(Y \mid X)$ being fixed with respect to θ, then minimizing the KL divergence is equivalent to maximizing the underbraced term, known as the Evidence Lower BOund (ELBO). Unlike the evidence, the ELBO is a tractable lower bound that can be optimized over the variational parameters if we rewrite it as

$$\mathbb{E}_{q(\boldsymbol{w};\theta)}[\log q(\boldsymbol{w};\theta) - \log p(\boldsymbol{w}, Y \mid X)] = \underbrace{-\mathbb{E}_{q(\boldsymbol{w};\theta)}[\log p(Y \mid X, \boldsymbol{w})]}_{\substack{\text{likelihood part} \\ \text{(data dependent)}}} + \underbrace{\mathrm{KL}[q(\boldsymbol{w};\theta) \parallel p(\boldsymbol{w})]}_{\substack{\text{complexity part} \\ \text{(prior dependent)}}}. \quad (2.10)$$

Without entering into detail about which particular kind of parametric distribution to take as the likelihood, we can consider that the parameters of this distribution are obtained from the neural network ϕ, i.e. $p(Y \mid X, \boldsymbol{w}) = pdf\,(y_i; \phi(\boldsymbol{x}_i))$, thus Eq. (2.10) can be expressed as

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \int_{\boldsymbol{w}} \sum_{i=1}^{N} q(\boldsymbol{w};\theta) \log pdf\,(y_i; \phi(\boldsymbol{x}_i))\, d\boldsymbol{w} + \int_{\boldsymbol{w}} q(\boldsymbol{w};\theta) \log \frac{q(\boldsymbol{w};\theta)}{p(\boldsymbol{w})} d\boldsymbol{w}. \quad (2.11)$$

To approximate the first integral, we can use the Monte Carlo estimate of the expectation by independently and randomly generating n_s samples $w_1, \ldots, w_{n_s}$ of weights from the distribution $q(\boldsymbol{w};\theta)$. In this manner, the best parameters, θ^*, are those that satisfy

$$\theta^* \approx \underset{\theta}{\operatorname{argmin}} \frac{1}{n_s} \sum_{s=1}^{n_s} \sum_{i=1}^{N} [\log pdf\,(y_i; \phi_s(\boldsymbol{x}_i))] + \mathbb{E}_{q(\boldsymbol{w};\theta)}[\log q(\boldsymbol{w};\theta) - \log p(\boldsymbol{w})], \quad (2.12)$$

where ϕ_s represents the neural networks, with their corresponding weights $\boldsymbol{w}_s$ for each sample s.

At this point, we need a way to equivalently estimate the expectation of the last summing part of Eq. (2.12). Usually, a Gaussian distribution for q is imposed (Blundell et al., 2015), for instance, so that $q(\boldsymbol{w}; \theta = (\mu, \sigma)) \sim \mathcal{N}(\mu, \sigma^2)$. If we suppose that $p(\boldsymbol{w}) \sim \mathcal{N}(\mu_0, \sigma_0^2)$, where μ_0, σ_0 are the prior of the Gaussian distri-

bution parameters, in an extended way, then we need to optimize the following loss function

$$(\mu^*, \sigma^*) \approx \underset{(\mu,\sigma)}{\operatorname{argmin}} \frac{1}{n_s} \sum_{s=1}^{n_s} \Bigg[\sum_{i=1}^{N} \left[\log pdf\left(y_i; \phi_s(\boldsymbol{x}_i)\right) \right] - \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}(\mu - \boldsymbol{w}_s)^2 + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(\sigma_0^2) + \frac{1}{2\sigma_0^2}(\mu_0 - \boldsymbol{w}_s)^2 \Bigg]. \quad (2.13)$$

Then, we apply the *reparametrization trick* (Kingma and Welling, 2014) of $\boldsymbol{w}$: since $\boldsymbol{w} \sim \mathcal{N}(\mu, \sigma^2)$ is a Gaussian random variable, we can reparameterize it as a deterministic function with respect to the distribution parameters like $\boldsymbol{w} = \mu + \varepsilon \cdot \sigma$, where $\varepsilon \sim \mathcal{N}(0, 1)$ does not depend on the distribution parameters, i.e. we are expressing the Gaussian distribution in terms of a standard Gaussian distribution, as shown in Figure 2.4. This is done in order to be able to compute the derivative of our loss function regarding the parameters (μ, σ). Therefore, the parameters of the model ϕ to be optimized are (μ, σ), and we perform the sampling process to generate the weights $\boldsymbol{w}$ during the forward step of the neural network's back-propagation training algorithm. In other words, we optimize the (μ, σ)-parameters of the Normal distribution, which will be used to generate the weights of our neural network. Hence, the sampling process of that distribution generates different possible neural networks ϕ_s to be used for prediction.

Following Section 2.2.2, $\log pdf\left(y_i; \phi_s(\boldsymbol{x}_i)\right)$ can be substituted by its MLE or MAP version to avoid explicitly approximating the aleatoric uncertainty and focus on the epistemic uncertainty, as we did in Brando et al. (2018a). If it is desirable to combine both uncertainties in a single model, it can be done using this step. However, given that this section focuses on epistemic uncertainty, we can consider an MLE with respect to the location parameters, similar to Eq. (2.7).

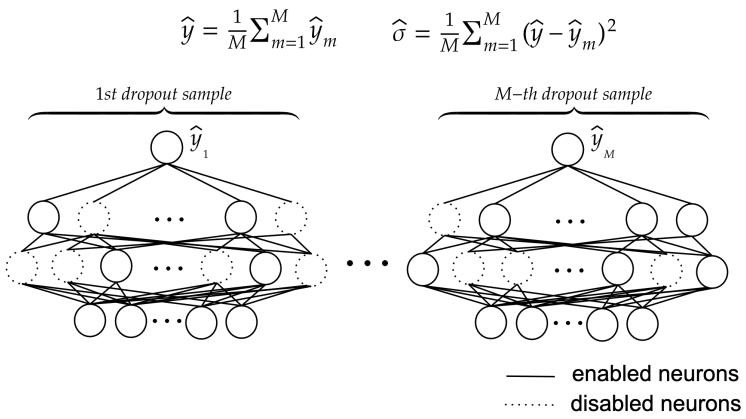


Figure 2.5: Graphic representation of how to obtain an ensemble of models to calculate their predicted mean and variance using a single model trained with Dropout.

2.3.3 Monte Carlo dropout

In this subsection we will discuss the Monte Carlo (MC)-dropout technique (proposed in Gal and Ghahramani (2016)). Originally, the standard Dropout technique (Srivastava et al., 2014) was introduced to avoid overfitting in deep learning models as a stochastic regularization method. This method consists in randomly removing certain connections between neurons in training time - as shown in any of the neural networks in Figure 2.5 - depending on the "dropout ratio" value, which indicates the probability of removing each connection or not. Following the training step, all of the connections are considered and the ratio value is used to adjust the learnt weights, as described in (Baldi and Sadowski, 2013).

Therefore, this previously described dropout procedure is deterministic at test time. However, several variations of the original dropout have been proposed. One of them is MC-dropout, which simply extends the dropout removing process in training and test time (Gal and Ghahramani, 2016). Consequently, the neural network prediction is no longer deterministic, because it depends on which randomly choose neuron connections are maintained, allowing each different prediction to be interpreted as a sample of a Bernoulli variational distribution and, therefore, to provide a Bayesian interpretation of the predicted values, as we can see in Figure 2.5, where the expected value and its empirical variance is

considered.

2.3.4 Ensemble of neural networks

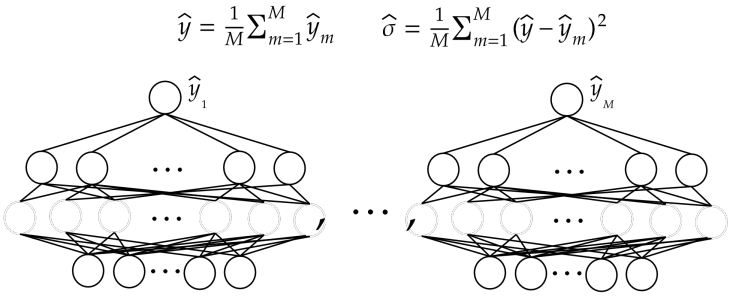


Figure 2.6: Graphic representation of an ensemble of neural networks and how to calculate its predicted mean and variance using M different models.

In this subsection, we will highlight how a set of several different neural networks trained over the same problem can be used to capture epistemic uncertainty, following Lakshminarayanan et al. (2016). As introduced in Section 2.1, our definition of epistemic uncertainty corresponds to the bias produced by the use of a certain model. In these terms, a fairly simple but effective way of estimating this uncertainty is by considering a set of different models (e.g. an ensemble of several neural networks with different hyper-parameters, such as different architecture, optimized in a different way, etc.).

In line with the above, Lakshminarayanan et al. (2016) proposed training several "different" models and considering each of its predictions as a sample from a probabilistic distribution as MC-Dropout in the previous Section 2.3.3. A graphical illustration of the procedure is shown in Figure 2.6, where the concept of having a sufficiently "different" set of models is crucial to be able to rely on their variability.

2.3.5 Additional epistemic alternatives

Generally speaking, there are plenty of ways to capture epistemic uncertainty (Postels et al.; Tagasovska and Lopez-Paz, 2019; van Amersfoort et al., 2020). For

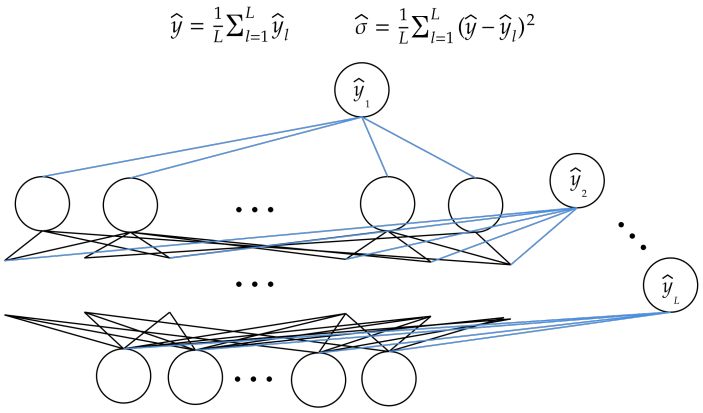


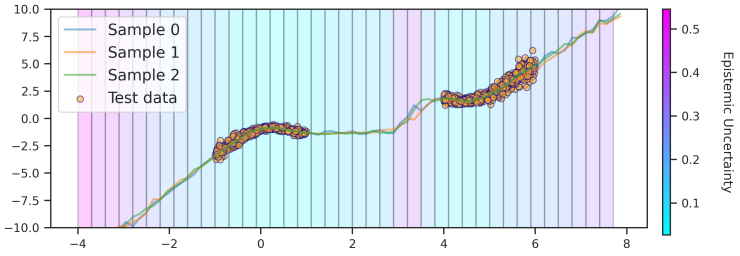
Figure 2.7: Simplified version of the alternative way to model epistemic uncertainty based on the architecture of the neural network. Each y_i can be considered a different output of the model.

instance, if it is important to minimize the computational cost of training several models as in Section 2.3.4, the Depth Uncertainty Network (DUN), presented in Antoran et al. (2020), can be considered. In this case, epistemic uncertainty is captured by making diverse predictions in a single neural network, as shown in Figure 2.7, where each depth level is used to produce a different prediction, yielding a set of predictions in a single forward pass and avoiding the Mean Field assumption, as discussed in depth in Foong et al. (2020).

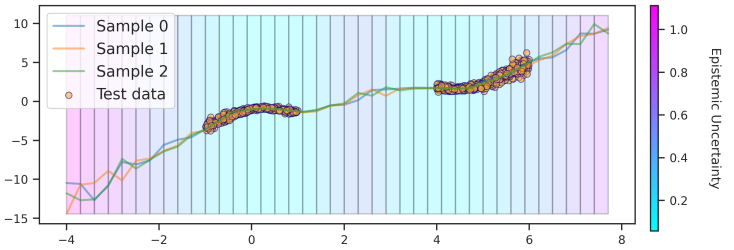
In general terms, there are two key points to pursue in epistemic models: "expressivity", to ensure each sample of the approximated probabilistic distribution $p(M \mid X, Y, x^*)$ corresponds to an accurate solution, and "diversity" in the set of predictions, so as to avoid all the samples overfitting into a single forecast when several correct solutions exist with respect to the model being used.

2.3.6 Synthetic comparison of epistemic uncertainty

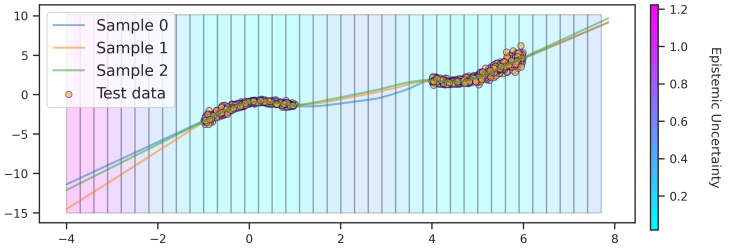
The goal of this subsection is to compare the epistemic performance of the three main baseline methods presented previously, i.e. the Variational Inference Bayesian neural network (described in Section 2.3.2), the Bayesian Dropout (described in Section 2.3.3) and the Deep Ensembles (described in Section 2.3.4), and ultimately to show how epistemic uncertainty can be visualized to avoid



(a) Bayesian Neural Network.



(b) Bayesian Dropout method.



(c) Deep Ensemble method.

Figure 2.8: Comparison of several epistemic methods. The colour indicates the degree of confidence. Three samples are shown for each case.

misleading interpretations.

Visualizing uncertainty poses a crucial problem (that will be discussed in detail in Chapter 6). Moreover, as we saw in Section 2.1, the very concept of uncertainty is not clear most of the time, and should be divided according to its source. Specifically, the definition of epistemic uncertainty used in this work, briefly represented as $p(M \mid X)$ or $p(M \mid X, Y, x^*)$ (following Figure 2.1 and Figure 2.2, respectively), considers combining a set of models to provide an ensemble of predictions. A mismatch between the predictions of the ensemble will indicate that there is greater epistemic uncertainty. Importantly, this disagreement does not capture conditional variability (aleatoric uncertainty), due to each

of the models in the ensemble being optimized using the same loss (i.e. typically, all of them are approximating the same conditional statistic but changing its initial conditions for each case, for instance, by using a different data split to train, using a diverse weight-initialization for each model, etc.). Consequently, visualizing epistemic uncertainty as conditional variability (e.g. as confidence intervals around the main prediction) could lead to a misunderstanding regarding just what is capturing this kind of uncertainty. Given the above, in this subsection we propose visualizing epistemic uncertainty as a gradient colour in the background where the predictions are made. In this case, highly uncertain areas will be indicated with a pinkish colour and more confidence in a bluish colour, as shown in Figure 2.8.

Analysing Figure 2.8, we observe that the margins of the data set are in pink due to there being no data points and, therefore, the predicted samples will tend to mismatch more often, providing a higher epistemic uncertainty. Similarly, in the central part, where there are no data points, the different samples tend to mismatch, which tends to yield a more uncertain prediction than where there are points. That being said, however, we observe that in this latter case, the tendency of neural networks to seek as simple a solution as possible results in non-ideal epistemic variability in this middle zone.

Since epistemic uncertainty is not the main goal of this work, the results of this synthetic experiment are illustrative and do not constitute a formal comparison between these methods. Mainly, the goal of showing these plots in Figure 2.8 is to demonstrate that the disagreement between the models in an ensemble may be useful in capturing a certain kind of uncertainty and, importantly, that it is crucial to visualize it in a proper manner to avoid misinterpretations regarding which information is providing us with this kind of uncertainty.

2.4 Real estate price per night forecasting

Since we will justify the original need in the next section, the current work will focus on aleatoric uncertainty. In order to quantitatively validate the proposed models developed during this thesis, we have considered real-world tasks where

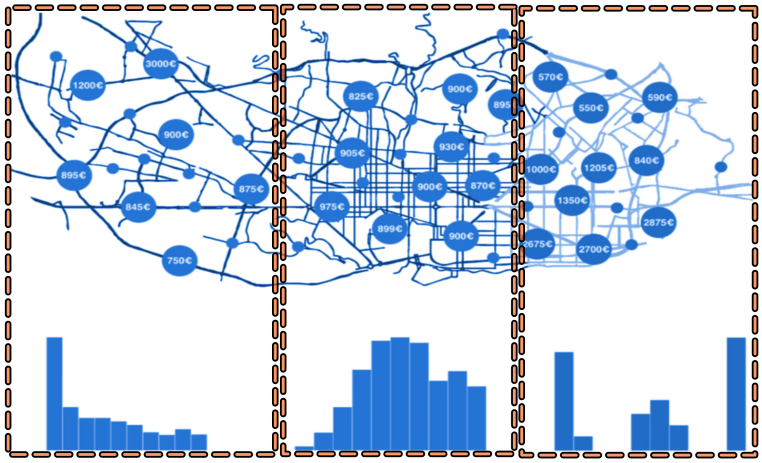


Figure 2.9: Three synthetic distribution of flat prices are shown on a city map. Slightly modified screenshots of the video (Brando and Llop, 2019) presented at NeurIPS 2020 for the article (Brando et al., 2019b).

the behaviour of the variable to be predicted has a heterogeneous distribution. For the sake of reproducibility[1], we have considered the use of an open data set of AirBnB apartment prices. Specifically, we predict prices for the cities of Barcelona and Vancouver by using public information downloaded from Cox (2019), selecting the last time each apartment appeared within the months available from April 2018 to March 2019. Therefore, the price prediction is based on informative features such as neighbourhood, number of beds and other characteristics associated with the apartments.

The regression problem will be defined as predicting the real price per night of each apartment in their respective currency, using the following information: the One Hot Encoding of the categorical attributes (present in the corresponding Inside Airbnb "listings.csv" files) including the district number, the postcode, information about the room and property type, as well as the number of bathrooms, the desired accommodates together with its latitude and longitude normalized according to the minimums and maximums of the corresponding city.

Given 36367 and 11497 apartments in BCN and YVC, respectively, we have considered 80% as a training set, 10% as a validation set and the remaining 10%

[1]The corresponding source code for obtaining and processing the data set can be found in Brando et al. (2019c)

as a test set. Regarding the trained models, all of them share the same neural network architecture for their ϕ, composed of 6 dense layers with ReLU activation in all but the last layer and their output dimensions of 120, 120, 60, 60, 10 and P, respectively, where P will be the number of required outputs depending on the task they are optimizing (as will be shown in Chapter 3).

2.5 Time-series prediction of financial expenses

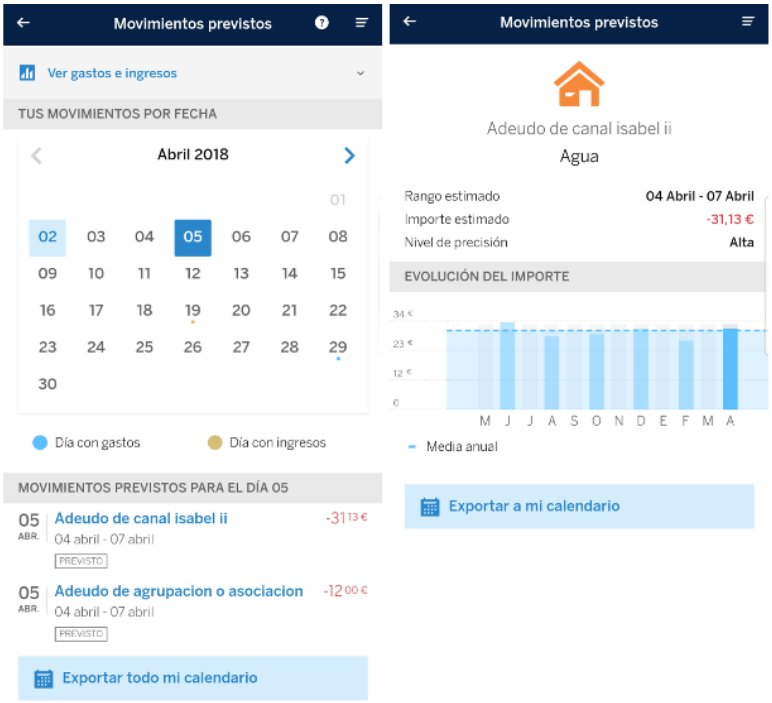


Figure 2.10: Screenshots of BBVA's mobile app showing expected incomes and expenses. Global calendar view (left) and expanded view of one of the forecasts (right).

The industrial nature of the current work requires that the results be applied to several internal and external problems pertaining to the company. One of the main problems - publicly addressed in Brando et al. (2018a, 2019b); Mejia et al. (2018) - is that of forecasting upcoming expenses and incomes in personal financial records, which will be seen in the mobile app shown in Figure 2.10.

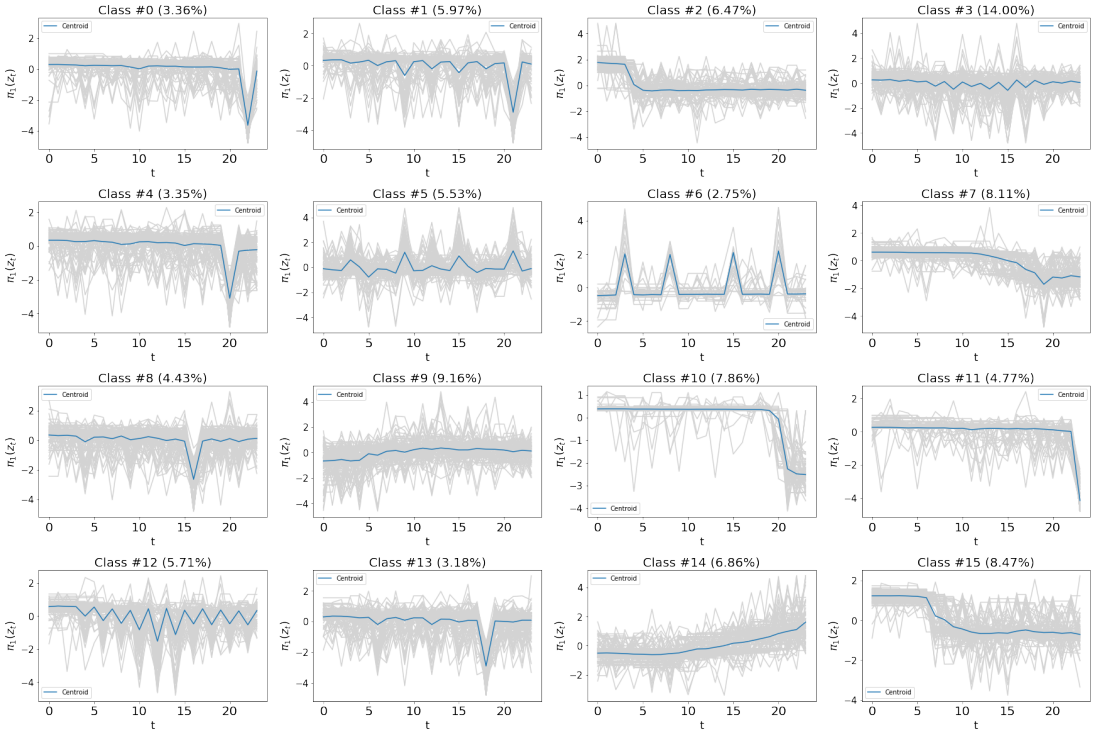


Figure 2.11: Clustering of the normalised 24 points time-series by using the π_1 transformation. The grey lines are 100 samples and the blue line is the centroid for each cluster.

The data were anonymized, e.g. customer IDs were removed from the series, and we do not use individual amounts, but rather aggregated monthly amounts. In addition, all of the experiments were carried out on the industrial servers where this research was performed. Technically, the input data consist of 24 consecutive monthly series of monetary expenses for certain customers in different selected expense categories. The ultimate goal was to forecast the aggregated value for the next month (as explained in detail in Brando et al. (2018a)).

To illustrate the nature and variability of the series, Figure 2.11 shows the values for the raw series grouped by clusters resulting from a k-means algorithm, where $k = 16$. Prior to clustering, the series were centred according to their mean and normalized by their standard deviation, with the result that the emergent clusters indicate scale-invariant behaviours such as periodicity.

The variability of certain clusters in Figure 2.11 reveals the challenge that

this problem can pose due to an expense or income resulting from erratic human actions, spurious events or depending on factors not captured by the past values of the series. Therefore, the main goal was to capture present uncertainty in order to detect such untrustworthy series and, for instance, therefore only communicate forecasts for those cases for which we are confident.

Table 2.1: Errors (MAE) found in each method at points of the error versus retention curve corresponding to a Retain=R. Each row corresponds to the combination of a predictor and an uncertainty score (described in the main text).

Predictor + uncertainty	R=25%	R=41%	R=50%	R=75%	R=99.5%	R=100%
mean + *var*	36.93	54.09	67.54	152.84	417.93	491.68
Zeros + *var*	36.95	54.21	67.75	154.16	429.29	504.45
Last + *var*	50.25	75.19	94.53	203.27	474.12	606.58
GAM + *var*	16.37	23.41	29.66	68.83	191.73	7539.91
GAM + *SE*	13.19	20.97	27.30	62.62	191.86	7539.91
RF* + prec	N/A	76.08	N/A	N/A	224.95	232.11
RF* + *var*	11.69	18.68	25.21	66.90	177.15	232.11
Dense + *var*	12.3 ± .24	18.1 ± .26	23.5 ± .26	56.3 ± .28	**146.** ± **.48**	**193.** ± **.49**
DenseD + *var*	14.9 ± .19	20.8 ± .13	26.4 ± .07	61.1 ± .30	160. ± .80	208. ± 1.0
DenseD + D	16.1 ± 1.3	23.2 ± .91	29.5 ± .76	64.9 ± .62	162. ± 1.4	208. ± 1.0
DenseB + *var*	16.1 ± 1.6	23.5 ± 1.9	30.6 ± 2.2	79.8 ± 4.4	198. ± 5.2	248. ± 5.6
DenseB + B	19.9 ± 1.1	28.5 ± 1.2	36.0 ± 1.7	89.4 ± 4.6	199. ± 4.7	248. ± 5.6
DenseHo + *var*	12.3 ± .23	18.1 ± .22	23.5 ± .22	56.3 ± .30	**145.** ± **.89**	**193.** ± **.84**
DenseHet + *var*	10.7 ± .08	16.4 ± .10	21.7 ± .11	55.0 ± .30	150. ± .97	199. ± .84
DenseHet + b_{het}	8.24 ± 1.3	12.9 ± .90	18.1 ± .59	45.3 ± .67	153. ± 1.1	199. ± .84
LSTM + *var*	11.6 ± .13	17.6 ± .12	23.1 ± .13	55.7 ± .43	146. ± .81	205. ± 1.2
LSTMHo + *var*	11.9 ± .37	18.0 ± .59	23.5 ± .76	56.6 ± 2.0	150. ± 9.1	210. ± 14.
LSTMHet + *var*	10.5 ± .03	16.2 ± .07	21.6 ± .10	53.9 ± .17	147. ± 1.3	218. ± 1.6
LSTMHet + b_{het}	**5.04** ± **.23**	**10.7** ± **.41**	**15.2** ± **.24**	**41.1** ± **.54**	149. ± 2.9	218. ± 1.6

Consequently, in Brando et al. (2018a) we considered a recurrent neural network model (such as the one presented in Section 2.2) and modelled the epistemic and aleatoric uncertainty to obtain the results presented in Table 2.1. Each of the values in this table correspond to the MAE between the real values and a certain prediction of the retained points that have a confidence value below a certain point, i.e. the $K = 25\%$ column will be the MAE value of 25% of the points with lower predicted uncertainty value. Therefore, the lower the value, the better.

As can be seen in Table 2.1, the aleatoric approaches identify the untrustworthy time-series better by more effectively filtering highly confident predictions,

in all cases obtaining the best results below 75%, which are the main case of interest. Specifically, if we compare the purely aleatoric case (DenseHet+ b_{het}) - highlighted in green - with two epistemic solutions (the DenseD + D, which corresponds to the Monte Carlo Dropout solution presented in Section **??** and the DenseB + B, which corresponds to the Bayesian neural network described in Section 2.3.2) - both highlighted in red - using the same hidden architecture for all of the cases, then we can see that **aleatoric uncertainty plays an important role**. Hence the reason for analysing aleatoric uncertainty in the next Chapter 3 and the rest of the work done in the thesis.

Chapter 3

ALEATORIC UNCERTAINTY MODELLING USING NEURAL NETWORKS

In Chapter 2, we reviewed the different sources of uncertainty and reasons for modelling aleatoric uncertainty. In this chapter, we propose novel solutions for aleatoric uncertainty modelling and tackle new problems deriving from this kind of uncertainty based on the real-world limitations detected when applying these models. We have already presented several approaches for tackling epistemic uncertainty in Section 2.3, which is the uncertainty related to modelling the family of possible models that better fits the data we have, i.e. the $p(M \mid X)$ term in Eq. (2.2).

In contrast, the current section presents several alternatives for modelling aleatoric uncertainty, which is critical in scenarios where the response variable cannot be predicted in a deterministic way (as discussed in Section 1.2). This includes scenarios where we do not have enough input information to uniquely report a prediction, but also when the response variable inherently has a noise that cannot be reduced (such as the forecasting of monthly aggregated expenses and incomes of clients in the bank operations presented in Section 2.5, or the real state price prediction problem presented in Section 2.4).

Following the uncertainty disentanglement presented in Eq. (2.2), aleatoric un-

certainty corresponds to modelling conditional probability $p(Y \mid X, M)$. Specifically, the selected model here is a neural network, which means that, following Section 2.2, it is a parametric model denoted by M, i.e. its corresponding hyperparameters and parameters. Considering aleatoric uncertainty separately implies fixing the family of models defined by M to a single model that will be used to maximize this conditional likelihood $p(Y \mid X, M)$ (i.e. we avoid the integral of Equation (2.2) to focus on only the aleatoric uncertainty part).

Although aleatoric uncertainty may seem like a rare type of uncertainty, it is quite the opposite. In fact, as we will see below, this type of uncertainty is implicitly assumed in most approaches aimed at solving a real regression problem; it simply appears when we are solving a regression problem rather than as a functional approximation, but solving it from a probabilistic perspective.

3.1 Preliminaries and notation

Now we have established the reasons for studying aleatoric uncertainty, we will start to study approaches to modelling it. Before that, however, we should highlight a simplification in the notation.

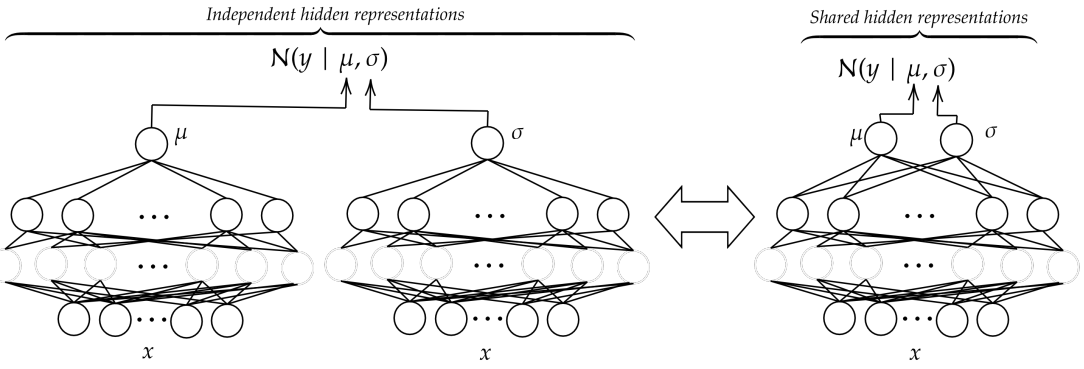


Figure 3.1: Graphic representation of the difference between sharing the hidden layers of the neural network or not.

The assumption of shared hidden representations The following proposed models will predict parameters of parametric distributions or quantiles of

the response conditional distribution $p(Y \mid X, M)$. As we can see from Figure 3.1, when a single model predicts different parameters or quantiles, there is a decision to be made regarding whether to share some hidden layers or not, with the ultimate aim of forecasting them, which implies assuming that a shared internal representation benefits the approximation of the desired probabilistic function. Based on our previous internal and public works (Brando et al., 2018a), we conclude that this decision is not critical in most of our scenarios, due to neural networks being capable of learning several parameters simultaneously if enough parameters are provided. Therefore, without loss of generality, we will simplify the notation by not indicating whether the hidden representation is shared, unless it needs to be highlighted, i.e. $\mu_w(x)$ and $\sigma_v(x)$ can be the outputs of the first or the second model presented in Figure 3.1. Furthermore, in terms of optimization, the same optimizer over w and v can be used to optimize the different models - with different w and v weights - connected for the loss function or considering the single model - where w and v are mixed - with its corresponding different outputs.

3.2 Learning the parameters of a conditional parametric distribution

In this section, we use neural networks as good function approximators to learn the parameters of conditional parametric density functions via the Maximum Likelihood Estimation (MLE), which was described in Section 2.2.

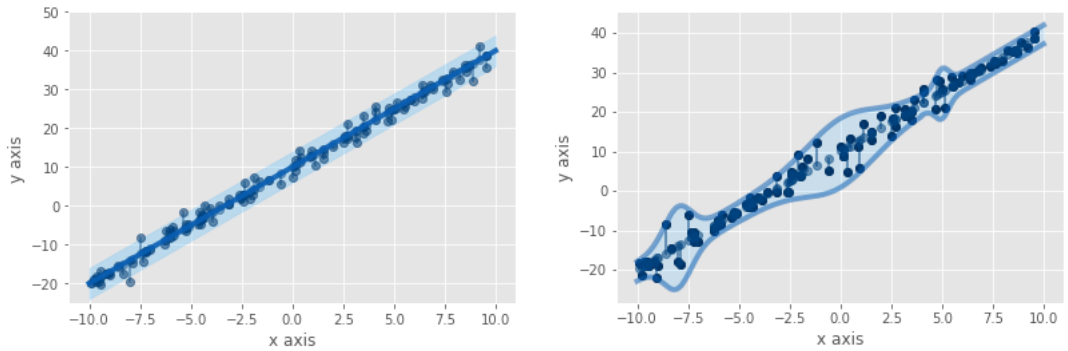
3.2.1 Unimodal distribution

As stated in Section 2.2, when we are optimizing the Mean Squared Error (MSE), from a probabilistic viewpoint, we are implicitly optimizing the location parameter of a Normal distribution with a fixed scale parameter via the Maximum Likelihood Estimator (MLE); that is

$$\mathcal{L}_{MSE}\left(y, \mu(x); \boldsymbol{w}\right) = \left(y - \mu(x)\right)^2 \quad \underset{MLE}{\Longleftarrow} \quad \mathcal{N}(y \mid \mu(x), \sigma(x)) = \frac{1}{\sqrt{2\pi\sigma(x)^2}} e^{-\frac{(y-\mu(x))^2}{2\sigma(x)^2}}. \tag{3.1}$$

Similarly to (Bishop, 1994b; Kendall and Gal, 2017), we are defining a likelihood function over the output of a neural network with a normal distribution, $\mathcal{N}(\mu(\boldsymbol{x}), \sigma^2)$, where μ is the neural network function and σ is the scale parameter.

When dealing with location–scale family distributions, the scale parameter can be considered a constant. In that case, we will denote we are in a **homoscedastic** scenario. In contrast, when the scale parameter is a function of x, such as the location parameter, this will be denoted as a **heteroscedastic** scenario. The difference between the two can be seen in Figure 3.2, where the same data set is approximated using a homoscedastic or heteroscedastic normal distribution. As we can see, the homoscedastic case finds a constant value of σ, while the heteroscedastic one, $\sigma(x)$, has a varied value depending on x.



(a) Homoscedastic approximation. (b) Heteroscedastic estimation.

Figure 3.2: Comparison of an input-dependent and a non-input-dependent uncertainty estimation.

Importantly, the scale parameter can only take positive real values, i.e. $\sigma \in \mathbb{R}_+$. Consequently, we should force its output to always be positive, and this can be done by applying a positive function to the scale parameter neural network output. For instance, we can apply a softplus function (Zheng et al., 2015) with a certain shift, specifically $\sigma(\boldsymbol{x}) = 10^{-3} + \texttt{softplus}(NN(\tau, \boldsymbol{x}) + 10^{-5})$, where

$NN(\tau, \boldsymbol{x})$ is the corresponding output of the neural network. The reason why this could be a good decision over other positive functions will be discussed in greater depth in Section 3.2.2.

In the event of learning a heterogeneous distribution, Figure 3.3 shows how a neural network learning the location and scale parameter of a normal distribution approximates all the parts of the synthetic data set proposed in Brando et al. (2019b).

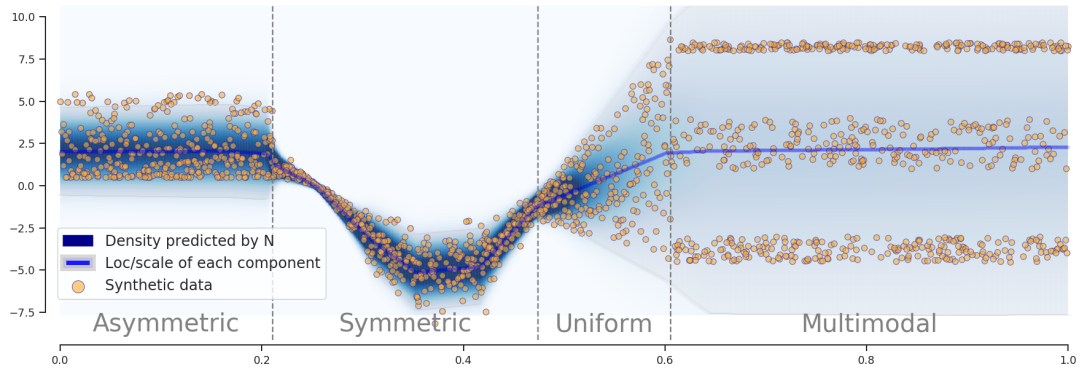


Figure 3.3: Regression problem with heterogeneous output distributions modelled with a normal distribution.

Generically, the idea of unimodal conditional modelling is to pose neural network learning as a probabilistic function learning using a conditional parametric distribution. Thus, there is no restriction on using another distribution function if it is more convenient for our problem, as in Brando (2017); Brando et al. (2018a) and we will discuss below.

Reminder 1 (The potential role of Laplace vs normal distributions) *Historically, the normal distribution has had a strong presence in the literature. However, despite the existence of other distributions which are potentially valuable due to their properties, these do not receive the same attention as the normal distribution. A special case in this regard, perhaps affected by being apparently similar to the normal distribution, is that of the Laplace (or double exponential) distribution (Geraci, 2017). This distribution dervied from the heterogeneous normal sub-population and was proposed at a similar historical point as the normal one (the former by Pierre-Simon Laplace in* 1774*, and the latter, depending on the source (Wilson, 1923), in* 1778*). As shown in Section 2.2.2, the MLE of the normal distribution corresponds to the mean (with the MSE). Similarly, it is well-known that the MLE over the Laplace distribution corresponds to the median (with the MAE).*

Over the last decade, theoretical developments related to MAE regression have led to a renewed interest in the Laplace distribution (Geraci, 2017) and its asymmetric extensions as pseudo-likelihood for Quantile Regression, as we will see in Section 3.3. At the same time, computational advances based on interior point algorithms have made MAE estimation a serious competitor of MSE methods. Moreover, as stated in Section 2.2, another reason for the "comeback" of the double exponential is its robustness, which makes normal distribution undesirable in many research areas, especially where there is a presence of outliers. For instance, it had a very notable impact on the bank's expenses and incomes financial forecasting problem described in Section 2.5 and addressed in Brando et al. (2018a).

As a consequence of the above, following Reminder 5, there are several scenarios where the Laplace distribution may be a better choice than the normal one. As stated in Section 2.2, the MLE of a Laplace distribution over its location parameter is the Mean Absolute Error (MAE); that is

$$\mathcal{L}_{MAE}\left(y, \mu(x); \boldsymbol{w}\right) = |y - \mu(x)| \quad \underset{MLE}{\Longleftarrow} \quad L\mathcal{P}(y \mid \mu(x), b(x)) = \frac{1}{2b(x)} e^{-\frac{|y-\mu(x)|}{b(x)}}, \tag{3.2}$$

where $L\mathcal{P}$ corresponds to the Laplace distribution, which, as we have said, has similar properties to the normal distribution and two (location and scale) parameters. Specifically, however, the Laplace distribution avoids the square difference and square scale denominator of the normal distribution, which results in empirically unstable behaviour in the initial points of the neural network training time or when the absolute error is sizeable (Brando et al., 2018a).

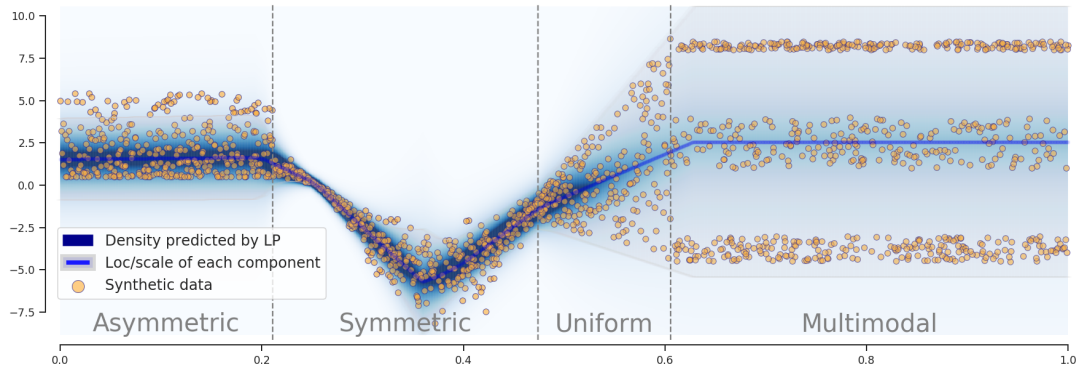


Figure 3.4: Regression problem with heterogeneous output distributions modelled with a Laplace distribution.

Figure 3.4 shows how a neural network learning the location and scale parameter of a Laplace distribution approximates all the parts of the synthetic data set sampled from a heterogeneous distribution that was proposed in Brando et al. (2019b). In contrast with Figure 3.3, here the Laplace distribution is less affected by the long tail of the asymmetric part.

3.2.2 Mixture of distributions

Reminder 2 (Multivariate normal distribution) *Given* $X \sim \mathcal{N}(\mu, \Sigma)$ *where* $\mu = [\mu_1, \ldots, \mu_n]^T$ *is the mean and* Σ *is the covariance matrix (a positive definite matrix of dimensions* $k \times k$*), then its probability density function is*

$$pdf_{\mathbf{x}}(x_1, \ldots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)} \tag{3.3}$$

Similar to the previous unimodal cases, the standard approach to find the mean and variance matrix parameters is the ***maximum likelihood method****, which requires maximization of the log-likelihood function:*

$$\ln \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{k} \ln f(x_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= -\frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) - \frac{k}{2}\ln(2\pi)$$

where here $\boldsymbol{x}$ *is a vector of real numbers.*

In the same way that we previously showed how the parameters of a conditional unimodal distribution can be approximated using a neural network, following (Bishop, 1994b), we can replace this unimodal distribution in the conditional density of the complete target vector with a mixture model (McLachlan G. J, 1988), which has the flexibility to approximate a richer distribution function. The probability density of the target data is then represented as a linear combination of kernel functions in the form

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i(\boldsymbol{x}) pdf_i(\boldsymbol{y}|\boldsymbol{x}) \tag{3.4}$$

where m is the number of components in the mixture and $\alpha_i(\boldsymbol{x})$ are called *mixing coefficients*. Specifically, if we consider the kernel functions as Gaussian distributions, we obtain

$$pdf_i(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{(2\pi)^{c/2}\sigma_i(\boldsymbol{x})^c} \exp\left\{-\frac{\| \boldsymbol{y} - \boldsymbol{\mu}_i(\boldsymbol{x}) \|^2}{2\sigma_i(\boldsymbol{x})^2}\right\}, \tag{3.5}$$

where $\boldsymbol{\mu}_i$ represents the centre of the i^{th} kernel. In this case, we are assuming that the components of the output vector are statically independent within each component of the distribution, and the distribution can be described by a common variance $\sigma_i(\boldsymbol{x})$. As (Bishop, 1994b) stated, to be more generic, the assumption of independence can be relaxed by introducing a full covariance matrix for each Gaussian kernel. However, according to (McLachlan G. J, 1988) and (Bishop, 1994b), a Gaussian mixture model with this simplified kernel can approximate any given density function to arbitrary accuracy, provided the mixing coefficients and the Gaussian parameters (means and variances) are correctly chosen. Note that this assumption simplifies the calculation of the inverse of the covariance matrix $\boldsymbol{\Sigma}_i$ since we will have a diagonal matrix with the same variance σ_i across all dimensions,

$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \sigma_i & 0 & \cdots & 0 \\ 0 & \sigma_i & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_i \end{bmatrix}, \tag{3.6}$$

which simplifies the $\mid \boldsymbol{\Sigma}_i \mid^{-1}$ computation of Eq. (3.3) to the σ_i^{-c} of Eq. (3.5).

As we can see in Figure 3.5, given an input vector $\boldsymbol{x}$, the Mixture Density Network (MDN) model provides a general formalism for modelling an arbitrary conditional density function $p(Y \mid X)$. This union between the traditional neural network and the mixture model part is achieved by using the log-likelihood of the linear combination of kernel functions as a loss function of the neural network. According to Bishop (1994b), by choosing a mixture model with a sufficient number of kernel functions, and a neural network with a sufficient number of hidden units, the MDN can theoretically approximate any conditional density function, $p(Y \mid X)$, as closely as desired.

Regarding the number of model outputs, compared to the single distribution modelling shown in Section 3.2.1, building this MDN increases the number of output parameters from c to $(c + 2) \times m$ parameters, where c remains the dimension of the output and m is the number of components of the mixture we are considering in the MDN.

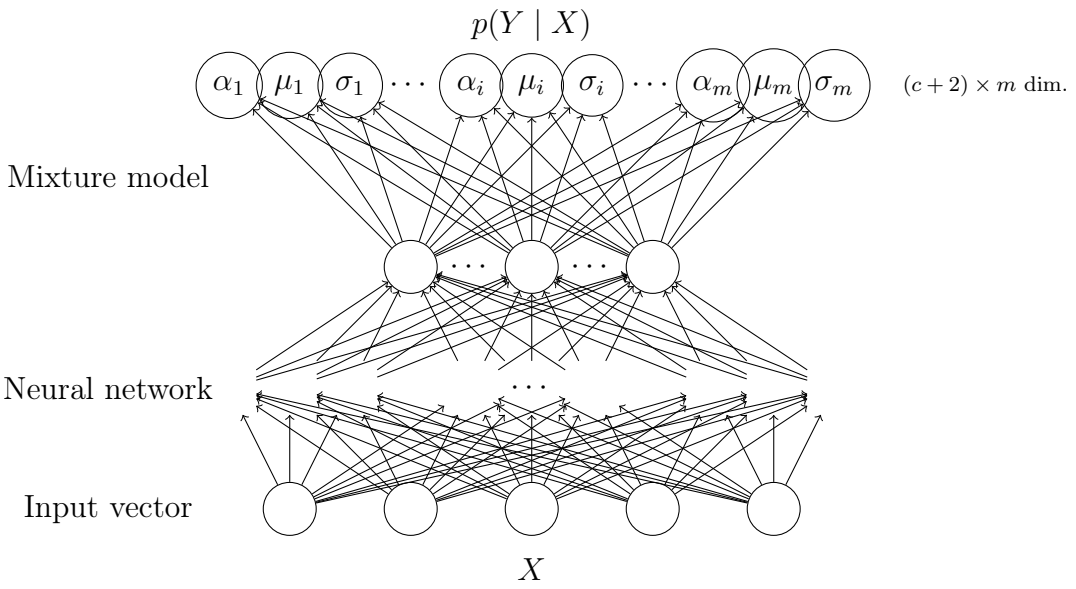


Figure 3.5: Representation of the Mixture Density Network (MDN) model. The output of the feed-forward neural network determines the parameters in a mixture density distribution. This image was extracted from Brando (2017).

Similarly to the unimodal case and following Bishop (1994b), there are some restrictions that each parameter should satisfy:

1. To ensure that the sum of the mixture components represents a probability, the mixing coefficients $\{\alpha_i\}_{i=1}^m$ should satisfy that $\sum_{i=1}^{m} \alpha_i = 1$. This restriction can be satisfied by computing a Softmax of the corresponding neural networks outputs as

$$\alpha_i = \frac{e^{(z_i^\alpha)}}{\sum_{j=1}^{m} e^{(z_j^\alpha)}}, \tag{3.7}$$

   which forces α_i to lie in the range $(0, 1)$ and sum to unity.

2. Regarding the scale parameter, σ_i, which is required to always be positive (similarly to Eq. 3.1), the classical approach (Bishop, 1994b) is to compute the exponential of the corresponding network output, z_i^σ, as

$$\sigma_i = e^{(z_i^\sigma)} \tag{3.8}$$

which, in a Bayesian framework, would correspond to the choice of an uninformative Bayesian prior, assuming that the corresponding network outputs z_i^σ had uniform probability distribution (Bishop, 1994b; Jacobs et al., 1991; Nowlan and Hinton, 1992). Theoretically, the use of this representation avoids pathological configurations in which one or more of the variances goes to zero. However, as we detected in Brando (2017); Brando et al. (2018a) and happens similarly in other deep learning models (Detlefsen et al., 2019; Nazabal et al., 2020), the use of an exponential function as an activation function for the scale parameters introduces a new issue due to its high growth in positive values. Nevertheless, the exponential decrease in negative values is desired, and an "ELU plus 1 function" can therefore be considered,

$$g(x) = ELU(\alpha, x) + 1 = \begin{cases} \alpha(e^x - 1) + 1 & \text{for } x < 0 \\ x + 1 & \text{for } x \geq 0 \end{cases}. \tag{3.9}$$

This function is the piece-wise function equivalent to the Softplus function,

$$g(x) = log(1 + e^x). \tag{3.10}$$

Both functions preserve the desired behaviour of the exponential function in the negative values, while it has a linear growth in the positive values to avoid exploding issues.

3. Finally, regarding the location parameters, $\{\mu_i\}_{i=1}^k$, the notion of an uninformative prior would be represented directly by the network outputs, i.e.

$$\mu_{i,k} = z_{i,k}^\mu \tag{3.11}$$

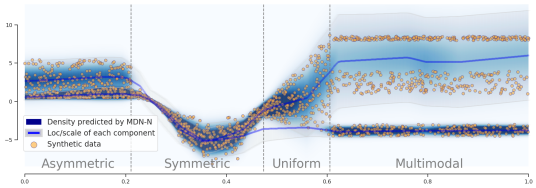
To sum up, as explained in Reminder 2, it is possible to construct a likelihood

function using the conditional density of the complete target vector. Thus, in order to optimize the MDN, the maximisation of the log-likelihood function or, equivalently, minimisation of the following negative logarithm of the likelihood will be considered

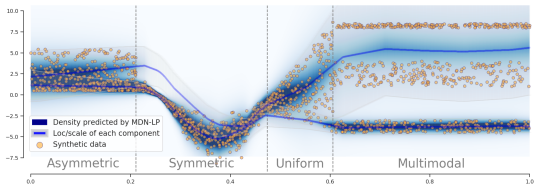
$$\log \mathcal{L}(\boldsymbol{y}, \{\phi_i\}_{i=1}^{m}; \boldsymbol{w}) = -\log\left(p(Y \mid X)\right) = -\log\left(\sum_{i=0}^{m} \alpha_i(\boldsymbol{x})\phi_i(\boldsymbol{x})\right)$$
$$= -\log\left(\sum_{i=0}^{m} \exp\left(\log \phi_i(\boldsymbol{x}) - \log(\alpha_i(\boldsymbol{x}))\right)\right) \quad (3.12)$$

where $\phi_i(\boldsymbol{x})$ is the same as in the Eq. (3.5). As explained in Bishop (1994b), the term $\sum p(\boldsymbol{x})$ has been dropped as it is independent from the parameters of the mixture model, and therefore independent from the network weights. Thus, the aim of MDNs is to model the complete conditional probability density of the output variables. From this density function, any desired statistic involving the output variables can, in principle, be computed.

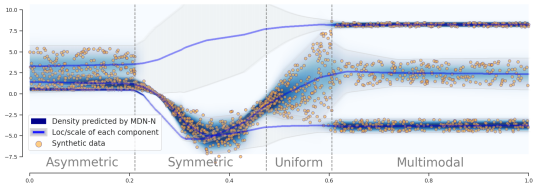
However, a wrong choice of hyper-parameter regarding the number of mixture components can directly skew the conditional density estimation of an MDN. To demonstrate this, we can analyse Figure 3.6, where different MDNs executed over the synthetic heterogeneous distribution clearly return a different performance depending on the number of components initially chosen. Importantly, we can see that the kernel function, $\phi_i(\boldsymbol{x})$, can be freely selected as in the unimodal case (shown in Eq. (3.2)). Extending this, we considered an MDN with normal and Laplace distributions, as shown in Figure 3.6. Although a mixture with a sufficient number of components of this type would theoretically be enough to approximate any kind of conditional distribution, the reality is that, empirically, when we increase this number, the behaviour of the MDN prediction, and its optimization process, become more unstable Brando (2017); Brando et al. (2019b). This was one of the main reasons for developing the method in the following section.



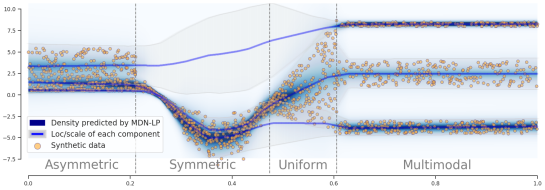
(a) MDN of 2 Normal distributions.



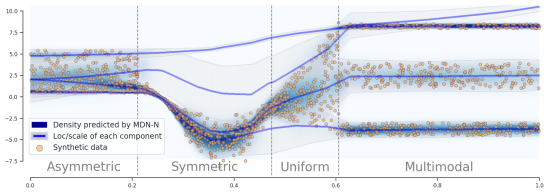
(b) MDN of 2 Laplace distributions.



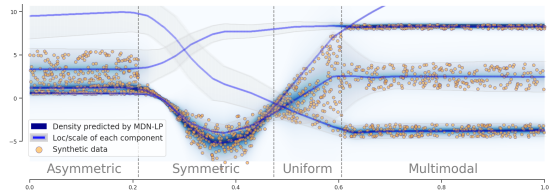
(c) MDN of 3 Normal distributions.



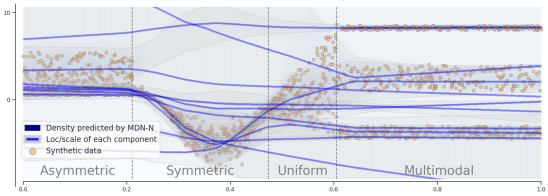
(d) MDN of 3 Laplace distributions.



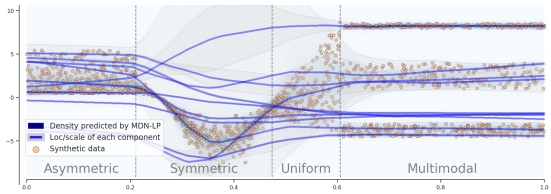
(e) MDN of 4 Normal distributions.



(f) MDN of 4 Laplace distributions.



(g) MDN of 10 Normal distributions.



(h) MDN of 10 Laplace distributions.

Figure 3.6: Heterogeneous conditional density estimation using several MDNs.

3.2.3 Uncountable mixture of distributions and the UMAL

The limitations regarding the finite mixture model highlighted in the previous Section 3.2.2 lead us to consider the following model, presented in Brando et al. (2019b).

The first objective is to avoid setting the number of components of the mixture m as a hyper-parameter that needs to be pre-decided. Therefore, one solution is to consider a mixture distribution marginalizing over a variable ϵ, i.e.

$$p\left(y \mid x, \boldsymbol{w}\right) = \int p(\epsilon) \cdot pdf\left(y \mid \theta(x)\right) \, d\epsilon. \tag{3.13}$$

where θ represents the conditional parameters of the *pdf* distribution. This is not new in the literature; for instance, in Titsias and Ruiz (2019), the hierarchical proposal using a mixing parameter to model the distribution implicitly is similar to the previous model in Eq. (3.13). Comparing this equation with the generic Eq. (3.4) of an MDN, the current mixture model has an uncountable set of components that are combined to produce the uncountable mixture[1] distribution.

Following classical approaches, let us consider the components as normal distributions, such that,

$$p\left(y \mid x; \boldsymbol{w}\right) = \int p(\epsilon) \cdot \mathcal{N}\left(y \mid \mu(x), \sigma(x)\right) \; d\epsilon. \tag{3.14}$$

We can now make two considerations, as we did in the model presented in Brando et al. (2019b). On the one hand, we assume a uniform distribution for each component $p(\epsilon)$ of the mixture model. Therefore, the weight $p(\epsilon)$ is the same for all the normal distributions, maintaining the restriction to integrate to 1. On the other hand, in order to make the integral tractable at training time, following the strategy proposed in implicit learning models using neural networks (Dabney et al., 2018a; Tagasovska and Lopez-Paz, 2018), we consider that the random variable $\epsilon \sim \mathcal{U}(0, 1)$ and apply the Monte Carlo (MC) integration (Rasmussen, 1996), selecting N_ϵ random values of ϵ in each iteration, so that we discretize the integral. This results in the following expression:

$$p\left(y \mid x; \boldsymbol{w}\right) \approx \frac{1}{N_\epsilon} \sum_{t=1}^{N_\epsilon} \mathcal{N}(y \mid \mu_{\epsilon_t}(x), \sigma_{\epsilon_t}(x)). \tag{3.15}$$

Therefore, the Uncountable Mixture of Normal distributions (UMN) model is optimized by minimizing the following negative log-likelihood function with respect to $\boldsymbol{w}$,

[1]The concept of "uncountable mixture" refers to the marginalisation formula that defines a compound probability distribution (Fred et al., 2017) and avoids the confusion with the literature Infinite Gaussian Mixture model (Rasmussen et al., 1999).

$$-\log p\left(Y \mid X, \boldsymbol{w}\right) \approx -\sum_{i=1}^{n} \log \left(\sum_{t=1}^{N_\epsilon} \exp\left[\log \mathcal{N}(y_i \mid \mu_{\epsilon_t}(x_i), \sigma_{\epsilon_t}(x_i))\right] \right) - \log(N_\epsilon), \tag{3.16}$$

where, as is commonly considered in mixture models (Khan et al., 2010), we have a "logarithm of the sum of exponentials". This form allows application of the log-sum-exp trick (Nielsen and Sun, 2016) during optimization to prevent overflow or underflow when computing the logarithm of the sum of the exponentials. The result can be seen in Figure 3.7.

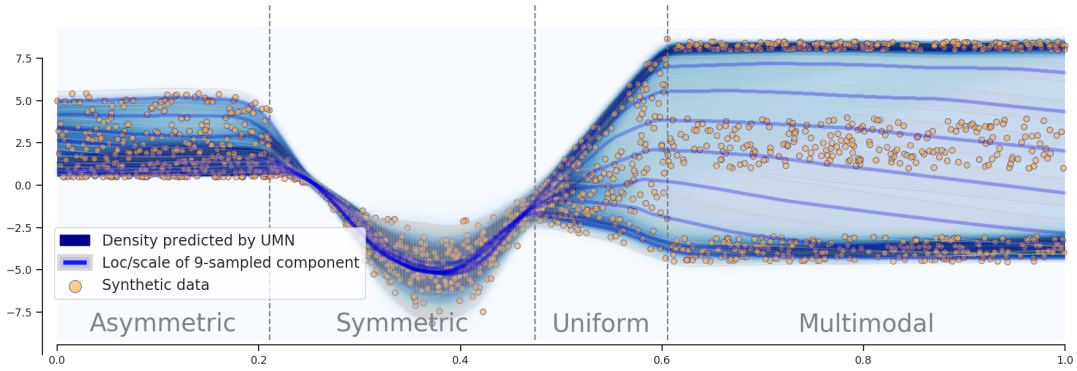


Figure 3.7: Regression problem with heterogeneous output distributions modelled with an Uncountable Mixture of Normals.

Uncountable Mixture of Asymmetric Laplacian distributions (UMAL) The second improvement with respect to the MDN is to see whether any distribution exists whose shape may be beneficial or which has a characteristic that can be combined with the marginalization process to enhance the conditional distribution approximation. Similarly to in previous unimodal distributions (described in Section 3.2.1) or in the mixture distribution case (defined in Section 3.2.2), where we searched for an appropriate distribution to use in our model (that is, a Laplace distribution can be better than a normal one in certain scenarios, as shown in Reminder 5), here we can consider different types of distributions in this uncountable mixture. Specifically, one interesting property could be to give a meaning to the marginalized variable, ϵ. This would mean, for instance, trying to

find a distribution where a parameter exists that has a special meaning, belongs to the real values (or even better, is within a real interval) and can therefore be marginalized in the required manner. The distribution proposed in (Brando et al., 2019b) for tackling these requirements was the Asymmetric Laplace distribution, as we will see below.

Given a location parameter $\mu \in \mathbb{R}$, a scale parameter $b \in \mathbb{R}^+$ and an asymmetry parameter $\tau \in [0, 1]$, the Asymmetric Laplace distribution is defined as:

$$\mathcal{ALD}\left(y \mid \mu, b, \tau\right) = \frac{\tau(1-\tau)}{b(x)} \exp\left\{-\left(y - \mu(x)\right) \cdot \left(\tau - \mathbb{1}[y < \mu(x)]\right) / b(x)\right\}. \tag{3.17}$$

here $\mathbb{1}[p]$ is the indicator function that verifies the condition p.

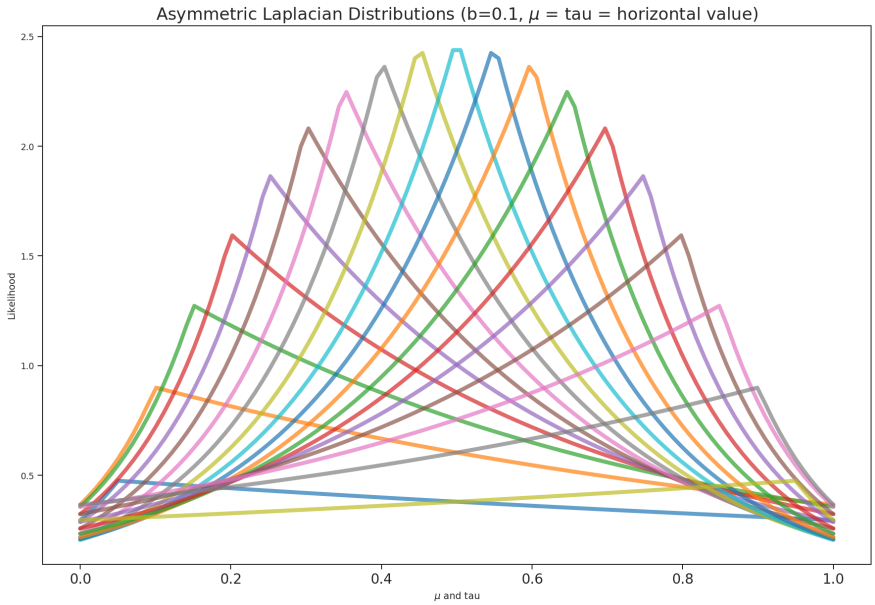


Figure 3.8: Visualization of several distributional shapes of the Asymmetric Laplace distribution depending on the $\tau = \mu$ values matching the horizontal axis.

Next, we combine all $\mathcal{ALD}$s, as can be seen partially in Figure 3.8, to infer a response variable distribution as an uncountable mixture.

Let $\boldsymbol{w}$ be the weights of the deep learning model to estimate, ϕ: $\mathbb{R}^{F+1} \rightarrow$

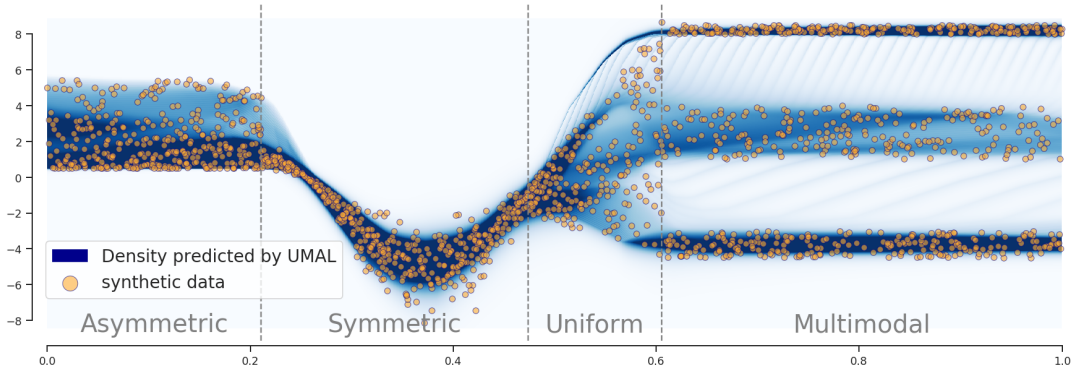


Figure 3.9: Regression problem with heterogeneous output distributions modelled with an UMAL, extracted from Brando et al. (2019b).

$\mathbb{R} \times (0, +\infty)$, which predicts the (μ_τ, b_τ) parameters of the different $\mathcal{A}$LDs conditioned to a τ value. We can then consider the following compound model marginalizing over τ:

$$p\left(y \mid x; \boldsymbol{w}\right) = \int \alpha_\tau(x) \cdot \mathcal{A}LD\left(y \mid \mu_\tau(x), b_\tau(x), \tau\right) \, d\tau. \tag{3.18}$$

Now, we can make the two prior considerations regarding the uniform distribution for each component α_τ and discretize the integral, applying MC integration and selecting N_τ random values of τ in each iteration. The expression to be calculated is:

$$p\left(y \mid x; \boldsymbol{w}\right) \approx \frac{1}{N_\tau} \sum_{t=1}^{N_\tau} \mathcal{A}LD(y \mid \mu_{\tau_t}(x), b_{\tau_t}(x), \tau_t). \tag{3.19}$$

Therefore, the Uncountable Mixture of Asymmetric Laplacians (UMAL) model is optimized by minimizing the following negative log-likelihood function with respect to $\boldsymbol{w}$,

$$-\log p\left(Y \mid X, \boldsymbol{w}\right) \approx -\sum_{i=1}^{n} \log \left(\sum_{t=1}^{N_\tau} \exp\left[\log \mathcal{A}LD(y_i \mid \mu_{\tau_t}(x_i), b_{\tau_t}(x_i), \tau_t)\right] \right) - \log(N_\tau), \tag{3.20}$$

where we can use the log-sum-exp computation (as we did in Section 3.2.2) during optimization to prevent overflow or underflow when computing the logarithm of the sum of the exponentials. This obtains a behaviour such as the one presented in Figure 3.9, where multiple modes and skewed distributions are estimated without specifying the number of components or the desired distributional shape, as shown in Figure 3.10 extracted from (Brando et al., 2019b).

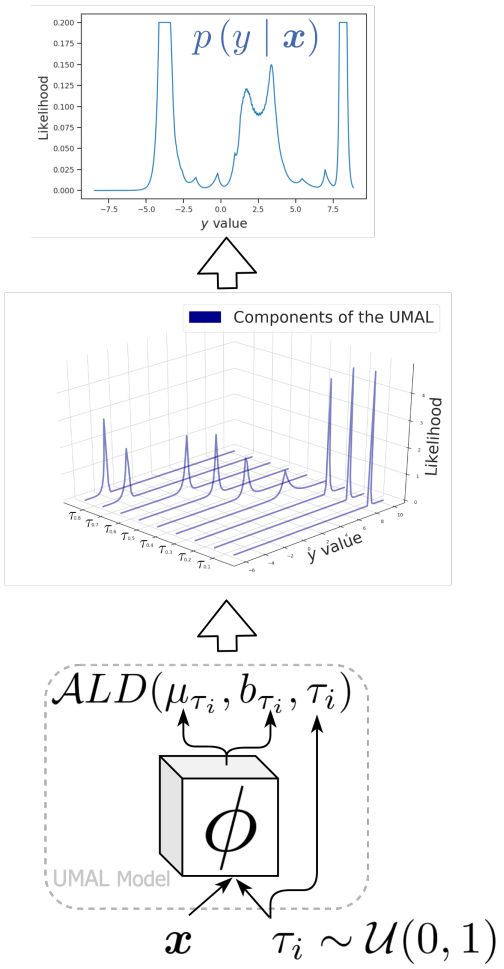


Figure 3.10: On the bottom, we see a representation of the proposed regression model that captures all the components τ_i of the mixture of $\mathcal{A}$LDs simultaneously. In the middle, we observe a visualization of some $\mathcal{A}$LD components predicting the distribution of the upper plot, which corresponds to the *Multimodal* part of Figure 3.9.

Pseudo-code of the UMAL

UMAL can be viewed as a framework for upgrading any point-wise estimation regression model in deep learning to an output distribution shape forecaster, as show in Algorithm 2. This implementation can be performed using any automatic differentiation library such as TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2017) or JAX (Bradbury et al., 2018). Additionally, it also performs the Monte Carlo step within the procedure by means of the N_τ dimension, which results in more efficient computation in training time.

Therefore, in order to obtain the conditioned mixture distribution, we should perform Algorithm 3.

Prerequisites 1 Definitions and functions used for the following Algorithms

▷ $\boldsymbol{x}$ has batch size and number of features as shape, $[bs, F]$.
▷ RESHAPE($tensor, shape$): returns $tensor$ with shape $shape$.
▷ REPEAT($tensor, n$): repeats last dimension of $tensor$ n times.
▷ CONCAT(T_1, T_2): concat T_1 and T_2 by using their last dimension.
▷ LEN(T_1): number of elements in T_1.

Algorithm 2 How to build UMAL model using any deep learning architecture for regression

procedure BUILD_UMAL_GRAPH(input vectors $\boldsymbol{x}$, deep architecture ϕ, MC sampling N_τ)
  $\boldsymbol{x} \leftarrow$ RESHAPE(REPEAT($\boldsymbol{x}, N_\tau$), $[bs \cdot N_\tau, F]$) ▷ Adapt $\boldsymbol{x}$ so it has an associated τ.
  $\boldsymbol{\tau} \leftarrow \mathcal{U}(0, 1)$ ▷ $\boldsymbol{\tau}$ must have $[bs \cdot N_\tau, 1]$ shape.
  $\boldsymbol{i} \leftarrow$ CONCAT($\boldsymbol{x}, \boldsymbol{\tau}$) ▷ The $\boldsymbol{i}$ has $[bs \cdot N_\tau, F + 1]$ shape.
  $(\boldsymbol{\mu}, \boldsymbol{b}) \leftarrow \phi_\tau(\boldsymbol{i})$ ▷ Applying any deep learning function ϕ.
  $\mathcal{L} \leftarrow$ Eq. (3.20) ▷ Applying the UMAL Loss function by using the $(\boldsymbol{\mu}, \boldsymbol{b}, \boldsymbol{\tau})$ triplet.
  return $\mathcal{L}$

Algorithm 3 How to generate the final conditioned distribution using the UMAL model

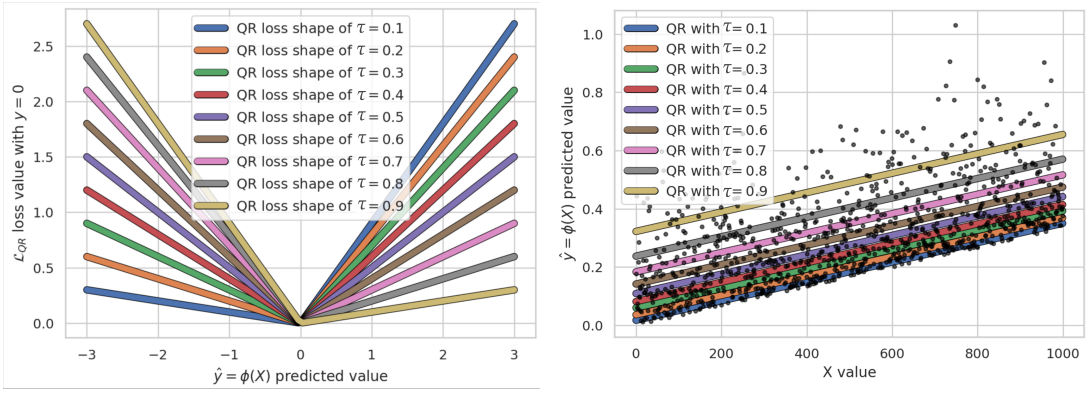
procedure PREDICT(input vectors $\boldsymbol{x}$, response vectors $\boldsymbol{y}$, deep architecture ϕ, selected τs sel_τ)
  $\boldsymbol{\tau} \leftarrow$ RESHAPE(REPEAT(sel_τ, bs), $[bs \cdot \text{LEN}(sel_\tau), 1]$) ▷ Adapting $\boldsymbol{\tau}$ shape.
  $\boldsymbol{x} \leftarrow$ RESHAPE(REPEAT($\boldsymbol{x}, sel_\tau$), $[bs \cdot \text{LEN}(sel_\tau), F]$) ▷ Adjusting $\boldsymbol{x}$ shape.
  $\boldsymbol{i} \leftarrow$ CONCAT($\boldsymbol{x}, \boldsymbol{\tau}$) ▷ The $\boldsymbol{i}$ has $[bs \cdot N_\tau, F + 1]$ shape.
  $(\boldsymbol{\mu}, \boldsymbol{b}) \leftarrow \phi_\tau(\boldsymbol{i})$ ▷ Apply the trained deep learning function ϕ.
  $p(\boldsymbol{y} \mid \boldsymbol{x}) \leftarrow \frac{1}{N_\tau} \sum_{t=1}^{N_\tau} \mathcal{ALD}(\boldsymbol{y} \mid \mu_{\tau_t}, b_{\tau_t}, \tau_t)$ ▷ Mixture model of sel_τ for each y.
  return $p(\boldsymbol{y} \mid \boldsymbol{x})$

3.3 Distribution-free estimation with quantile regression

Reminder 3 (Quantile Regression) *In statistics and econometrics, an extension to classic regression has been proposed: Quantile Regression (QR). Given $\tau \in (0,1)$, the τ-th QR loss function over a linear model is*

$$\mathcal{L}_\tau\left(w; \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - w \cdot \boldsymbol{x}_i\right) \cdot \left(\tau - \mathbb{1}[y_i < w \cdot \boldsymbol{x}_i]\right), \tag{3.21}$$

where $\mathbb{1}[p]$ is the indicator function that verifies the condition p. A visual representation of the quantile loss for different values can be seen in Figure 3.11.



(a) QR loss depending on τ.

(b) Corresponding conditional quantile τ.

Figure 3.11: Visualization of QR loss shape and the corresponding conditional approximated quantile using a linear function $f(x) = w \cdot x$.

Following (Koenker and Hallock, 2001) and Reminder 3, Quantile Regression may be viewed as a natural extension of the mean approximation that was previously presented as MSE in Eq. (2.7). Specifically, when $\tau = 0.5$ it corresponds to the MAE case presented in Eq. (2.8). In contrast to the MSE or MAE approaches, taking the combined estimation of different conditional quantiles together will provide a complete view of the distribution shape of the response variable as a distributional-free approximation i.e. without imposing strong assumptions such as symmetry or unimodality.

This chapter will take a more in-depth look at the methods used to model aleatoric uncertainty in this thesis by using this discrete approximation of the conditional distribution, $p(Y \mid X, M)$, through the conditional quantiles.

3.3.1 Fixed quantile regression

If we extend the quantile regression loss function (shown in Eq. 3.21) to a non-linear function learning, we can create quantile regression forests (Meinshausen and Ridgeway, 2006), gradient boosted quantile regression models (Zhang et al., 2018) or even neural network models (Dabney et al., 2018b) that can be used to approximate a discrete set of quantiles. In all of these cases, the set of quantile values to approximate, denoted as $\{\tau_t\}_{t=1}^{N_\tau}$, is fixed a priori. Thus, the number of quantiles to approximate in such a QR scenario constitutes a hyperparameter of the model definition.

In the following sections, we will continue using neural networks as a good function estimator to build several models that can serve to meet different goals.

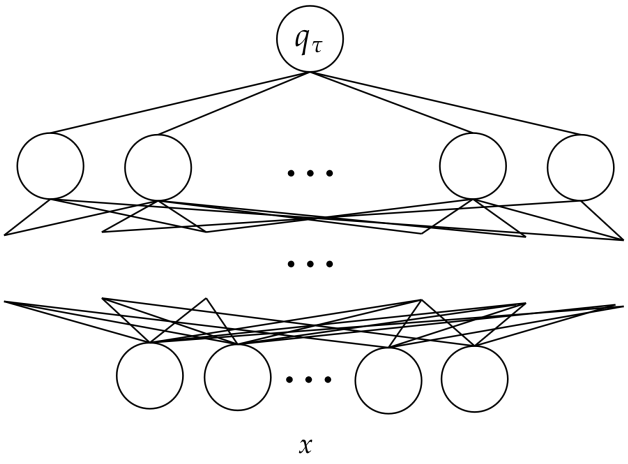


Figure 3.12: Graphic representation of the Single Quantile Network.

3.3.1.1 Single quantile estimation with a neural network

As we have seen, the mean and the median can both be useful statistics when our goal is to find a central value. However, in fields such as finance, structural

engineering or earth sciences, the existence of extreme deviations from the median is an important matter of study due to the extreme consequences that can result from this[1]. Given this context, estimating a point-wise threshold can be one way of tackling such problems. The objective, therefore, will not be to predict a central expected value, but rather to search for a distribution point where the accumulation of probability mass corresponds to the desired value.

The generic formulation of quantile regression with deep learning can be expressed as follows:

Definition 1 (Conditional quantile regression) *Given $\tau \in (0, 1)$, the τ-th quantile regression loss function with a NN, ϕ, would be defined as*

$$\mathcal{L}_\tau \left(\boldsymbol{w}; \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \right) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \phi(\boldsymbol{x}_i) \right) \cdot \left(\tau - \mathbb{1}[y_i < \phi(\boldsymbol{x}_i)] \right), \tag{3.22}$$

where $\mathbb{1}[p]$ is still the indicator function that verifies the condition p.

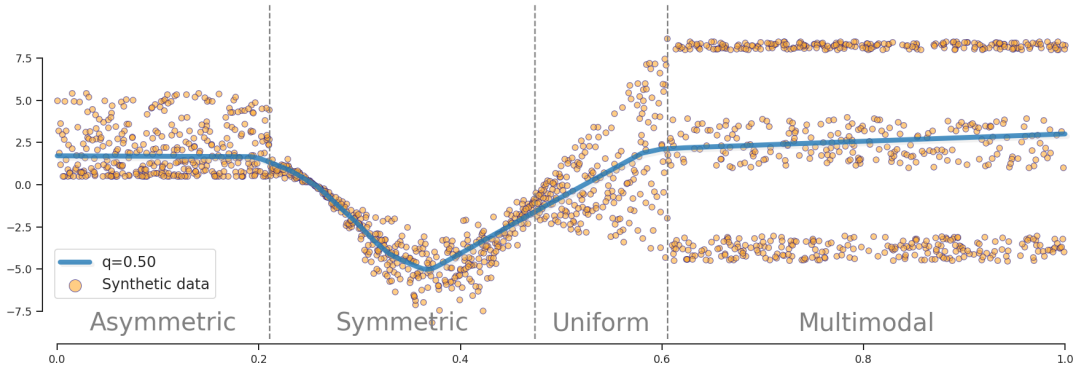


Figure 3.13: Regression problem with heterogeneous output distributions modelling the quantile 0.5, i.e. the median.

In Figure 3.13, we show the result of a non-linear approximation of the synthetic data set over the median, $\tau = 0.5$, using quantile regression.

[1]The analysis of these rare events is a branch of statistics known as extreme value theory.

3.3.1.2 Multiple quantiles with a neural network

Along the same lines, if we are able to predict not only one quantile but several simultaneously, then we can build confidence intervals over the distribution to be predicted. These intervals will seek to contain the desired probability mass. Importantly, the definition of these intervals may or may not be centred on the median. For example, in certain cases we might be interested in predicting a centred interval that includes 60% of the prediction points (to ensure an expected medium value is included or to analyse the width of that confidence interval). However, if the problem is more similar to the one described in the previous section - where it is of greater importance to detect extreme values - it may be of interest to detect the highest 10% of the values to be predicted, which might involve performing an analysis of the distribution tail, for instance.

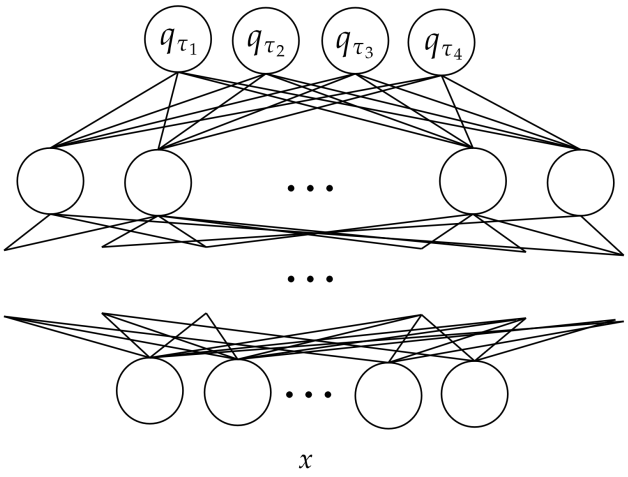


Figure 3.14: Graphic representation of the Quantile Network that approximates $N = 4$ fixed quantiles.

When several quantiles are to be approximated, this will be done in the following manner: firstly, by considering a multi-output neural network model such as the one presented in Figure 3.14. Secondly, for each of the outputs, by optimizing the quantile regression loss function with its desired quantile to be predicted. This approach will be similar to the simultaneous estimation of several parameters using the single model presented in Section 3.2, although here each output will correspond to a different pre-decided quantile. The formal definition is described

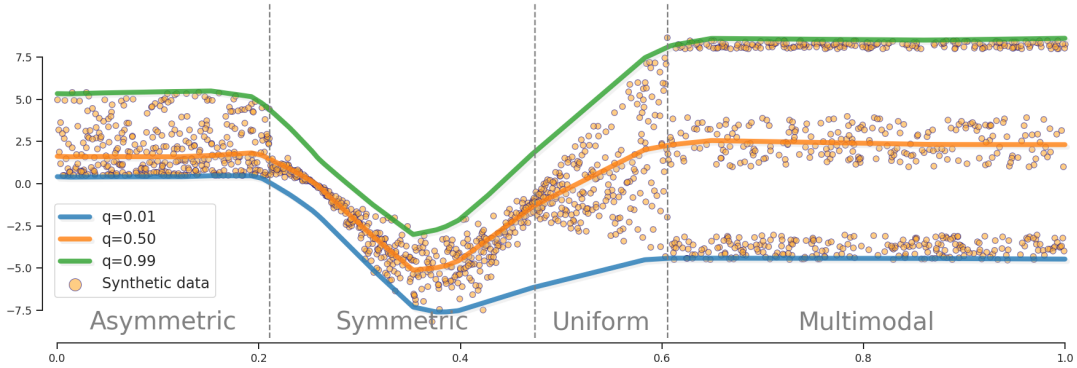


Figure 3.15: Regression problem with heterogeneous output distributions modelling the quantile $0.01, 0.5$ and 0.99.

below.

Given a fixed set of quantiles $\{\tau_i\}_{i=1}^{N_\tau} \in (0,1)$, the simultaneous approximation of this set using an NN with N_τ-outputs, $\{\phi_i\}_{i=1}^{N_\tau}$, will be done using the following loss function,

$$\mathcal{L}_\tau(x, y; \boldsymbol{w}, \{\tau_i\}_{i=1}^{N_\tau}) = \sum_{i=1}^{N_\tau} (y - \phi_i(x)) \cdot (\tau_i - \mathbb{1}[y < \phi_i(x)]) \tag{3.23}$$

where $\mathbb{1}[p]$ is the indicator function that verifies the condition p.

Equivalently to the single quantile approximation, the non-linear approximation of the three quantiles $\{\tau_i\}_{i=1}^{3} = \{0.1, 0.5, 0.99\}$ is shown in Figure 3.15.

To sum up, we see that by using this method we can define distribution-free intervals of confidence using DL functional approximation capabilities, which will be extremely useful in situations where corner values are problematic (for instance, in several internal financial problems such as the one presented in Section 2.5).

3.3.2 Implicit quantile regression

The fixed quantile approximation, previously presented in Section 3.3.1, can be extended to a non-fixed quantile approximation. In other words, a quantile function approximation can be built, where the desired quantile to predict is an input value, τ, of that function, as shown in Figure 3.16. Following Brando et al.

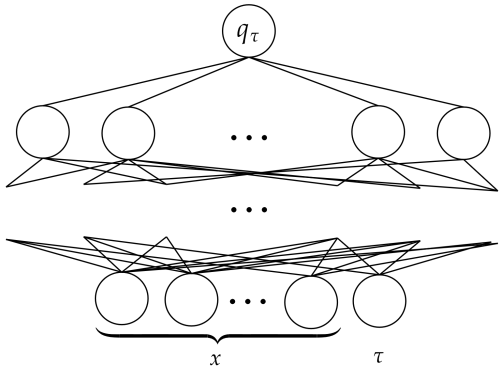


Figure 3.16: Graphic representation of the Implicit Quantile Network.

(2019b); Dabney et al. (2018a); Tagasovska and Lopez-Paz (2019), if the models are trained stochastically (such as neural networks using stochastic gradient descent), they can be defined to learn the quantile function implicitly, i.e. where the quantile to predict is an input parameter, τ. As Dabney et al. (2018a) stated, this approach allows any conditional distribution to be approximated, given sufficient model capacity. In such cases, we can extend Eq. (3.21) to learn the full quantile distribution using the predicted quantiles, as follows:

Definition 2 (Conditional quantile function) *A function* $\Phi_w \colon [0,1] \times \mathbb{R}^D \to \mathbb{R}$ *with parameters* w *approximates the quantile function when it minimizes the conditional quantile regression loss function defined as*

$$\mathcal{L}(X,Y) = \mathbb{E}\left[\int_0^1 \Big(Y - \Phi_w(\tau, X)\Big) \cdot \Big(\tau - \mathbb{1}[Y < \Phi_w(\tau, X)]\Big)\, d\tau\right], \tag{3.24}$$

where $\mathbb{1}[c]$ *denotes the indicator function that verifies the condition* c*.*

Due to the integral, Eq. (3.24) is difficult to compute because the analytical expression of Φ_w is not generally known. Following the notation in Brando et al. (2019b), we can apply a Monte-Carlo strategy to provide a feasible loss function. This is based on considering a uniform random variable $\tau \sim \mathcal{U}(0,1)$ and for each evaluation of the loss function in Eq. (3.24), a τ-sample set, $\{\tau_t\}_{t=1}^{N_\tau}$, of N_τ points

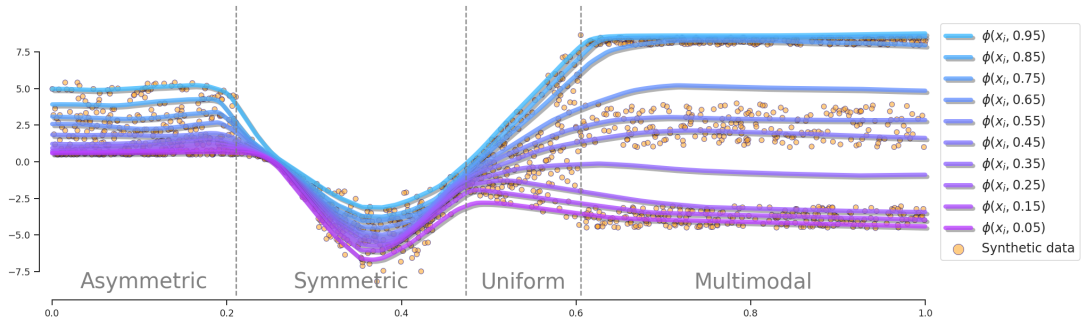


Figure 3.17: Regression problem with heterogeneous output distributions modelling the entire distribution of quantiles implicitly.

is generated in each training iteration, such that

$$\mathcal{L}(X, Y) \approx \mathbb{E}\left[\frac{1}{N_\tau}\sum_{t=1}^{N_\tau}\Big(Y - \Phi_w(\tau_t, X)\Big) \cdot \Big(\tau_t - \mathbb{1}[Y < \Phi_w(\tau_t, X)]\Big)\right]. \tag{3.25}$$

Therefore, the learnt quantile function corresponds to a full discretization of the conditional probability $p\,(Y \mid X, M)$, which we are interested in predicting based on Eq. 2.2. As we have said, Φ_w must be a model that can be trained using Monte-Carlo, and neural network models are therefore appropriate in this case.

3.3.3 The connection with asymmetric Laplace

Until now, the quantile regression approaches described above were designed to obtain point-wise or discrete forecasts corresponding to any desired set of quantiles. As we have seen, this has the advantage of performing a free-distributional estimation of the conditional distribution. However, similarly to the MSE in Eq. (2.7) (or the MAE in Eq.(2.8)) which is related to the normal distribution (or the Laplace distribution) by means of the MLE over the location parameter, we can build a distribution where its MLE corresponds to the QR loss function presented in Eq.(3.21)? The Asymmetric Laplace distribution (shown in Eq. 3.28) can be formulated to have a direct connection to the QR loss function.

Typically, the Asymmetric Laplace distribution has the following probability density function expression (Jammalamadaka and Kozubowski, 2004; Kozubowski

and Podgórski, 2000),

$$pdf(x \mid m, \lambda, \kappa) = \left(\frac{\lambda}{\kappa + 1/\kappa} \right) e^{-(x-m)\lambda s \kappa^s}, \tag{3.26}$$

where $m \in \mathbb{R}$ corresponds to the location parameter, $\lambda \in \mathbb{R}_+$ the scale parameter and $\kappa \in \mathbb{R}_+$ the asymmetry parameter. Considering Eq. 3.26 and performing the following change of variable,

$$\tau = \frac{\kappa^2}{\kappa^2 + 1} \qquad b = \frac{\kappa}{(\kappa^2 + 1)\lambda}, \tag{3.27}$$

then we can arrive at the proposed Asymmetric Laplace distribution ($\mathcal{ALD}$) formulation used for the Bayesian quantile regression approach stated in Yu and Moyeed (2001) and presented in Eq. 3.28,

$$\mathcal{ALD}\left(y \mid \mu, b, \tau\right) = \frac{\tau(1-\tau)}{b} e^{\frac{-(y-\mu)\cdot(\tau - \mathbb{1}[y<\mu])}{b}}, \tag{3.28}$$

where $\mu \in \mathbb{R}$ corresponds to the location parameter, $b \in \mathbb{R}_+$ the scale parameter and $\tau \in \mathbb{R}_+$ the asymmetry parameter. Consequently, the MLE regarding the location parameter of the Eq. 3.28 produces the original QR loss function as follows,

$$\begin{aligned} \mathcal{L}_\tau(x, y; \boldsymbol{w}) &= (y - \mu_\tau(x)) \cdot (\tau - \mathbb{1}[y < \mu_\tau(x)]) \\ &\underset{MLE}{\Longleftarrow} \quad \mathcal{ALD}\left(y \mid \mu, b, \tau\right) = \frac{\tau(1-\tau)}{b(x)} e^{\frac{-(y-\mu(x))\cdot(\tau - \mathbb{1}[y<\mu(x)])}{b(x)}}. \end{aligned} \tag{3.29}$$

Importantly, similarly to the Normal (or Laplace) distribution compared to the MSE (or MAE), we can state that the Asymmetric Laplace distribution is a non-point-wise approach of the quantile regression. Therefore, this allows us to approximate not only a certain quantile but a corresponding variability around this. This result led us to develop the following sections.

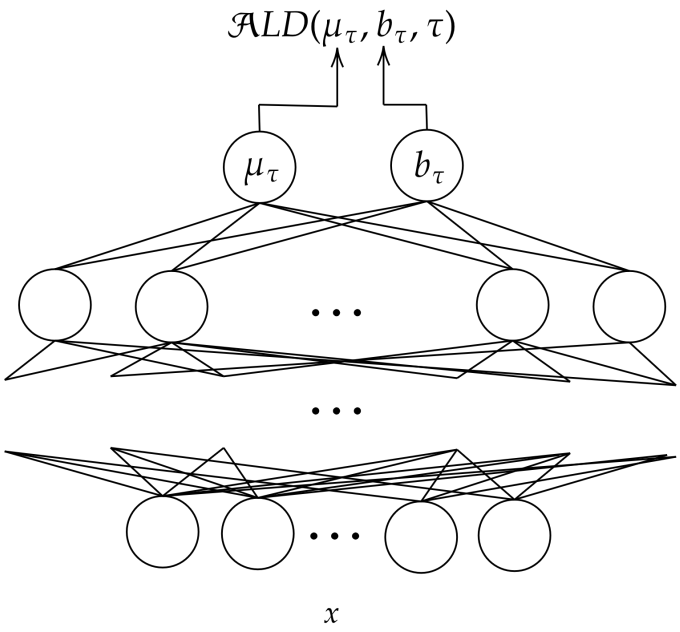


Figure 3.18: Graphic representation of the Fixed ALD Network.

3.3.3.1 Fixed asymmetric Laplace distributions

Similarly to Section 3.2.1, where a unimodal distribution was approximated, or Section 3.3.1, where a fixed quantile τ is approximated, the relationship detected between quantile regression and the Asymmetric Laplace distribution can motivate the proposal of a model which approximates the conditional parameters of this latter probabilistic function using a neural network, as shown in Figure 3.18, minimizing the log-likelihood as the next loss function,

$$-\log p\left(Y \mid X, \boldsymbol{w}\right) = -\sum_{i=1}^{n} \log \mathcal{ALD}(y_i \mid \mu_\tau(x_i), b_\tau(x_i), \tau). \tag{3.30}$$

The result of estimating the conditional location and scale of an Asymmetric Laplace distribution with $\tau = 0.5$ can be seen in Figure 3.19. As we can see, the variance is higher when the predicted median quantile could correspond to a higher variability of values.

Since we know that each asymmetry parameter, τ, corresponds to a different

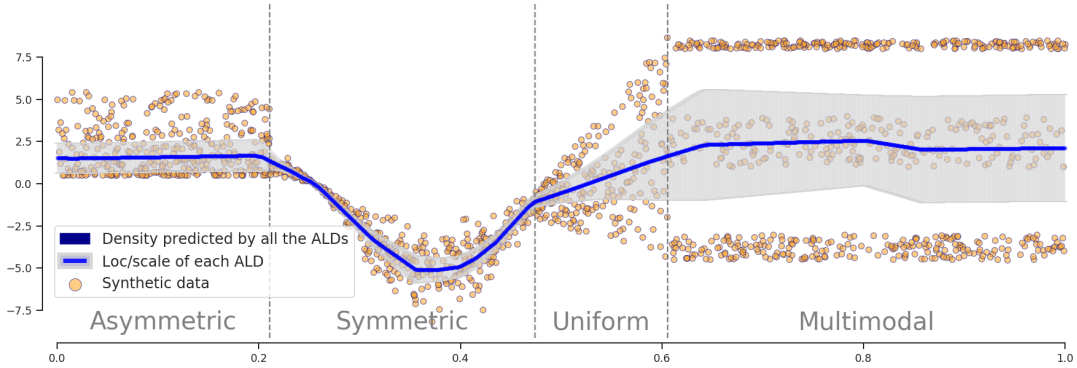


Figure 3.19: Regression problem with heterogeneous output distributions modelling the Asymmetric Laplace distribution with $\tau = 0.5$.

quantile value, another step forward would be to approximate several Asymmetric Laplace distributions with different pre-selected asymmetry values. This is similar to Section 3.3.1.2, where several quantiles were predicted simultaneously. However, in the former case, the loss function is the sum of the log-likelihoods,

$$-\log p\left(Y \mid X, \boldsymbol{w}\right) \approx -\sum_{i=1}^{n} \frac{1}{N_\tau} \left(\sum_{t=1}^{N_\tau} \log \mathcal{A}LD(y_i \mid \mu_{\tau_t}(x_i), b_{\tau_t}(x_i), \tau_t) \right), \quad (3.31)$$

where N_τ is the number of the approximated Asymmetric Laplace distributions with asymmetry values $\{\tau_t\}_{t=1}^{N_\tau}$. Importantly, although at the level of neural network architecture it may be the same, this model is different than considering a mixture model (as described in Section 3.2.2). When an MDN is considered, then each Asymmetric Laplace distribution is a component of the mixture, with a particular asymmetry value for each one, τ_t. Therefore, they are combined to approximate the conditional likelihood $p\left(Y \mid X, \boldsymbol{w}\right)$. In other words, the loss function considers the logarithm of the likelihoods and implements the log-sum-exp trick as follows,

$$-\log p\left(Y \mid X, \boldsymbol{w}\right) \approx -\sum_{i=1}^{n} \log \left(\sum_{t=1}^{N_\epsilon} \exp\left[\log \mathcal{ALD}(y_i \mid \mu_{\tau_t}(x_i), b_{\tau_t}(x_i), \tau_t)\right] \right) - \log(N_\tau). \tag{3.32}$$

Based on the "uncountable" term introduced in Section 3.2.3 for the UMAL model (Brando et al., 2019b), the authors in Klotz et al. (2021) proposed to refer to this model as the Countable Mixtures of Asymmetric Laplacians (CMAL).

In Figure 3.20, the result of approximating several Asymmetric Laplace distributions can be shown in the synthetic heterogeneous problem.

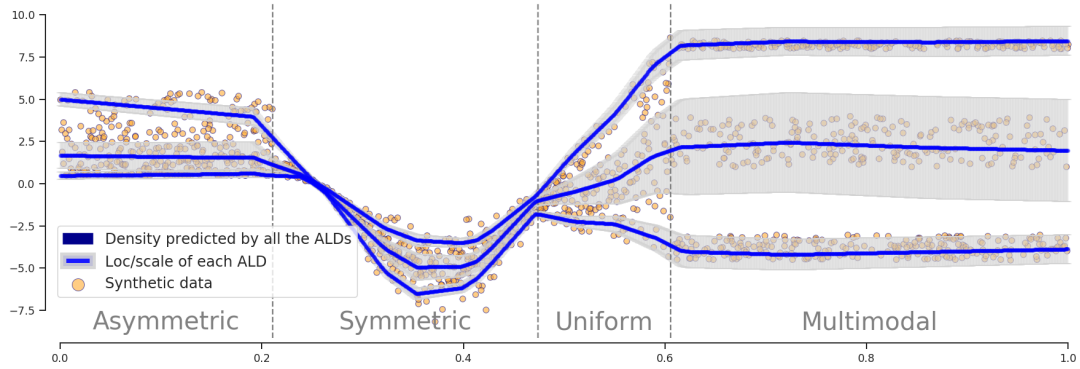


Figure 3.20: Regression problem with heterogeneous output distributions modelling the Asymmetric Laplace distributions with $\tau = \{0.01, 0.5, 0.9\}$.

3.3.3.2 Independent asymmetric Laplace distributions

Following Section 3.3.2, if we consider an IQR model where the entire range of quantiles is implicitly and independently approximated, then the mode of an $\mathcal{ALD}$ can be directly inferred because it corresponds to the location parameter. Thus, in inference time there is a perfect solution using a combination of $\mathcal{ALD}$ that estimates the real distribution but in a "Dirac delta manner".

Using this idea of learning the conditional quantiles implicitly, an alternative approach would be to minimize the entire negative logarithm of $\mathcal{ALD}$s as a sum of distributions where each one "independently" captures the variability for each quantile. Therefore, the model will predict the location and scale parameters

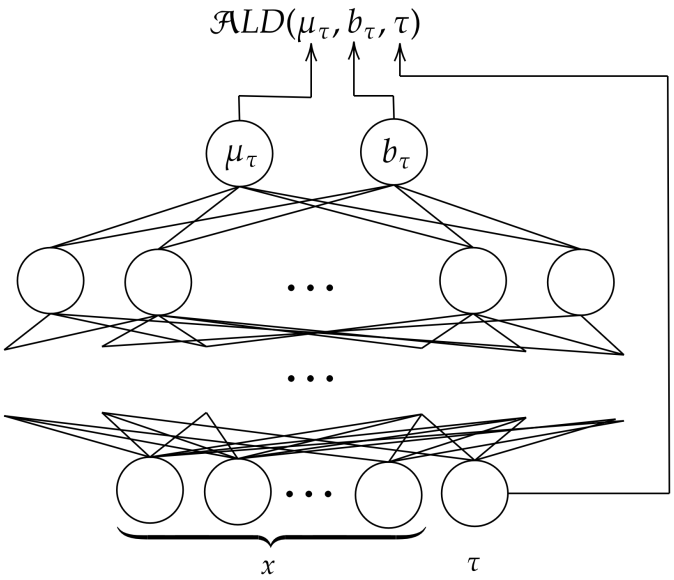


Figure 3.21: Graphic representation of a generic implicit mixture of the Asymmetric Laplace distributions model.

given a τ input value, as shown in Figure 3.21.

This solution is, in fact, an upper bound of the UMAL model presented in Section 3.2.3. Considering N_τ the number of MC-samples of the random variable $\tau \sim \mathcal{U}(0,1)$ and applying Jensen's Inequality to the negative logarithm function of Eq. (3.20) gives us an expression that corresponds to considering all $\mathcal{ALD}$s as independent elements,

$$-\log p\left(Y \mid X, \boldsymbol{w}\right) \leq -\sum_{i=1}^{n}\left(\sum_{t=1}^{N_\tau} \log \mathcal{ALD}(y_i \mid \mu_{\tau_t}(x_i), b_{\tau_t}(x_i), \tau_t)\right) - \log(N_\tau). \tag{3.33}$$

We will refer to this upper bound solution as the *Independent* $\mathcal{ALD}$ model. Figure 3.22 shows the result of approximating the synthetic heterogeneous distribution.

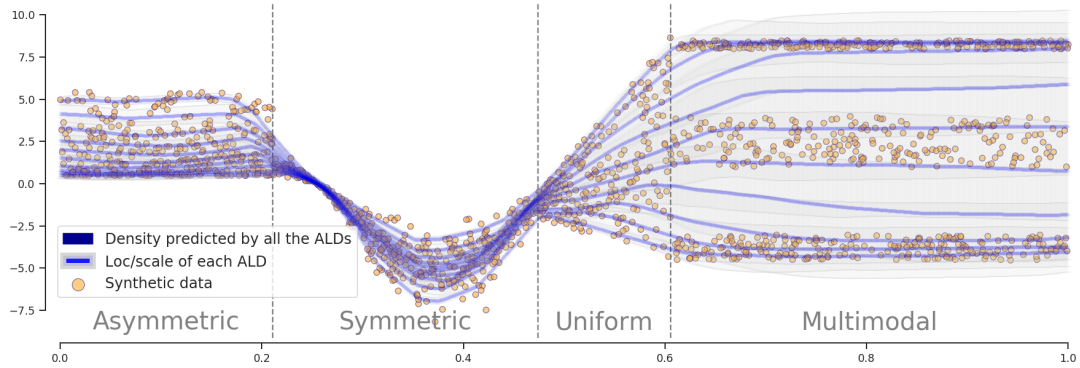


Figure 3.22: Regression problem with heterogeneous output distributions independently modelling all of the Asymmetric Laplace distributions with respect to τ.

3.3.3.3 The UMAL as a dependent quantile model

In Eq. 3.18, the UMAL model was presented as a mixture of Asymmetric Laplace distributions. After highlighting the link between this distribution and the quantile regression loss function, we can re-visit UMAL as a "dependent" combination - in contrast to the "independent" combination presented in Section 3.3.3.2 - of $\mathcal{A}LD$s, in the knowledge that a solution exists where each location parameter will correspond to the conditional quantile value.

In this case, the UMAL architecture will be the same as that presented in Figure 3.21, but obtaining the results shown in Figure 3.23 (or in Figure 3.9).

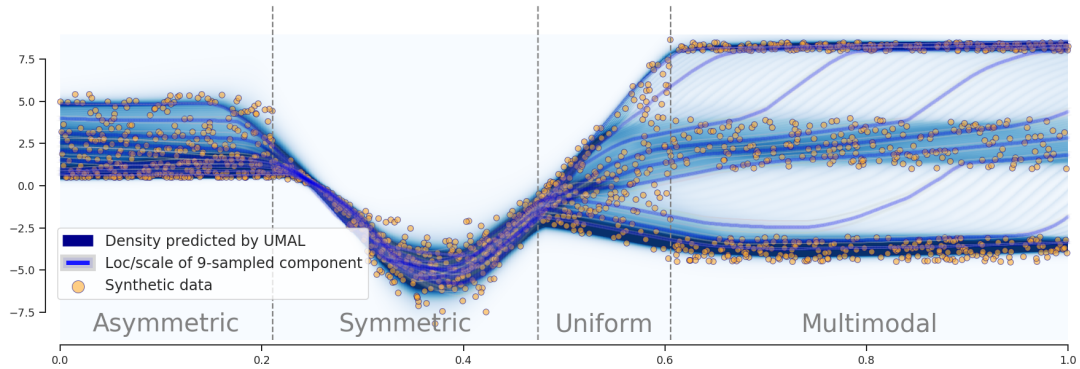


Figure 3.23: Regression problem with heterogeneous output distributions modelled using an Uncountable Mixture of Asymmetric Laplacians.

3.4 Results and comparison

3.4.1 Data sets and experiment settings

In this section, we show the performance of the proposed model, as shown in Brando et al. (2019b). All experiments were implemented in TensorFlow (Abadi et al., 2015) and Keras (Chollet et al., 2019), running in a workstation with Titan X (Pascal) GPU and GeForce RTX 2080 GPU. Regarding parameters, we used a common learning rate of 10^{-3}. In addition, to restrict the value of the scale parameter, b, to strictly positive values, the respective output had a softplus function (Zheng et al., 2015) as activation. We will refer to the number of parameters to be estimated as P. On the other hand, the Monte Carlo sampling number, N_τ, for Independent QR, $\mathcal{A}LD$ and UMAL models will always be fixed to 100 at training time. Furthermore, all public experiments were trained using an early stopping training policy with 200 epochs of patience for all compared methods.

Synthetic regression The synthetic heterogeneous data set presented throughout Chapter 3 corresponds to the following data set. Given $(X, Y) = \{(x_i, y_i)\}_{i=1}^{3800}$ points where $x_i \in [0, 1]$ and $y_i \in \mathbb{R}$, the data are obtained from 4 different fixed synthetic distributions depending on the X range of values. Specifically, if $x_i < 0.21$, then the corresponding y_i came from a distribution

$$\text{Beta}(\alpha = 0.5, \beta = 1). \tag{3.34}$$

Next, if $0.21 < x_i < 0.47$, then their y_i values are obtained from the distribution

$$\mathcal{N}(\mu = 3 \cdot \cos x_i - 2, \sigma = \mid 3 \cdot \cos x_i - 2 \mid) \tag{3.35}$$

which depends on the x_i value. Then, when $0.47 < x_i < 0.61$, their respective y_i values are obtained from an increasing uniform distribution and, finally, all values above 0.61 are obtained from three different uniform distributions: $\mathcal{U}(8, 0.5)$, $\mathcal{U}(1, 3)$ and $\mathcal{U}(-4.5, 1.5)$. A total of 50% of the uniformly generated data were

considered as test data, 40% for training and 10% for validation.

For all of the compared models, we will use the same neural network architecture for ϕ. This consists of 4 dense layers with 120, 60, 10 and P output dimensions, respectively, and all but the last layer with ReLu activation. Regarding training time, all models took less than 3 minutes to converge.

Room price forecasting (RPF) This problem and data set were described in Section 2.4. Specifically, the goal is to estimate the price per night of several flats using its properties. These data are collected from a publicly available data base known as the Inside Airbnb platform Cox (2019), where we selected Barcelona (BCN) and Vancouver (YVC) as the cities to carry out the comparison of the models in a real situation.

Financial estimation Similarly, this problem was introduced in Section 2.5. Specifically, the aim here is to anticipate personal expenses and income for each specific financial category in the upcoming month by only considering the last 24 months of aggregated historical values for that customer as a short-time series problem. This private data set contains monthly aggregated expense and income operations for each customer in a certain category as time series of 24 months. 1.8 million time series from a selected year will be used as the training set, 200 thousand as the validation set and 1 million from the following year will be the test set.

Regarding the neural network architecture for all compared models, after an internal previous refinement task to select the best architecture, we used a recurrent model that contains 2 concatenated Long Short-Term Memory (LSTM) layers (Hochreiter and Schmidhuber, 1997) of 128 output neurons each, and then two dense layers of 128 and P outputs, respectively. It is important to note that because all compared solutions used for this article are agnostic with respect to the architecture, the only decision we need to take is how to insert the extra τ information into the neural network function in the QR, $\mathcal{A}LD$ and UMAL models. In these cases, for simplicity, we add the information τ repeatedly as one more attribute of each point of the input time series.

Table 3.1: Comparison of the Log-Likelihood of the test set over different alternatives to model the distribution of the different proposed data sets. The scale for each data set is indicated in parentheses.

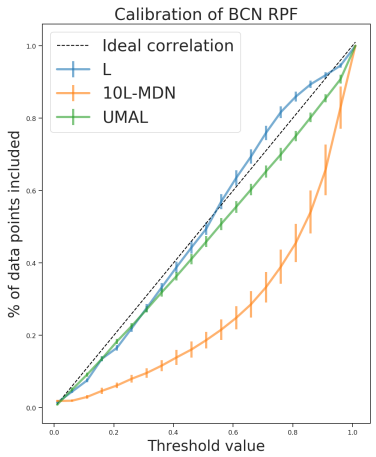
Log-Likelihood comparison				
Model	Synthetic (10^2)	BCN RPF (10^3)	YVC RPF (10^2)	Financial (10^6)
Normal distribution	-39.88 ± 13.4	-38.44 ± 6.55	-70.79 ± 3.26	-8.56
Laplace distribution	-41.30 ± 0.78	-19.84 ± 0.93	-82.87 ± 8.01	-7.88
Independent QR	-119.0 ± 7.68	-32.98 ± 1.63	-113.54 ± 10.4	-8.26
2 comp. Normal MDN	-43.14 ± 6.12	-28.59 ± 3.38	-74.11 ± 3.26	-6.37
3 comp. Normal MDN	-51.79 ± 21.0	-31.66 ± 4.85	-74.22 ± 2.37	-7.25
4 comp. Normal MDN	-111.6 ± 43.27	-28.60 ± 7.22	-76.85 ± 5.95	-6.75
10 comp. Normal MDN	-184.3 ± 35.5	-27.72 ± 2.81	-77.26 ± 6.12	-10.40
2 comp. Laplace MDN	-42.83 ± 1.54	-19.76 ± 0.18	-65.52 ± 0.40	-10.83
3 comp. Laplace MDN	-64.13 ± 36.70	-19.57 ± 0.30	-78.80 ± 3.79	-5.84
4 comp. Laplace MDN	-52.53 ± 8.79	-19.89 ± 0.44	-66.58 ± 1.10	-5.72
10 comp. Laplace MDN	-155.9 ± 32.9	-21.45 ± 0.83	-82.51 ± 9.66	-6.28
Independent $\mathcal{A}LD$	-39.03 ± 0.45	-19.03 ± 0.81	-64.16 ± 0.19	-5.66
UMAL model	$\mathbf{-28.14 \pm 0.44}$	$\mathbf{-18.04 \pm 0.72}$	$\mathbf{-62.68 \pm 0.21}$	$\mathbf{-5.49}$

3.4.2 Experimental results

Log-Likelihood comparison We compared the log-likelihood adaptation of all models presented in Table 3.1 for the three type of problems introduced. For all public data sets, we give their corresponding mean and standard deviation over the 10 runs of each model we performed. Due to computational resources, the private data set is the result after one execution per model. Furthermore, we take into account different numbers of components for the different MDN models. We observe that the best solutions for MDN are far from the UMAL cases. Thus, we conclude that the UMAL models achieve the best performance in all of these heterogeneous problems.

Calibrated estimated likelihoods We performed an additional empirical study to determine whether the learned likelihood is useful (i.e. if UMAL yields calibrated outputs). We would highlight here that our system predicts an output distribution $p(y|x, \boldsymbol{w})$ (not a confidence value). Specifically, we have computed the % of actual test data that falls into different thresholds of predicted probability. Ideally, given a certain threshold $\theta \in [0, 1]$, the amount of data points with a predicted probability above or equal to $1 - \theta$ should be similar to θ. We

Figure 3.24: Plot with the performance of three different models in terms of calibration. The mean and standard deviation for all folds of the mean absolute error between the predicted calibration and the perfect ideal calibration is represented in the table.



Likelihood calibration comparison		
Model	BCN RPF	YVC RPF
Normal distribution	$.12 \pm .04$	$\mathbf{.04 \pm .01}$
Laplace distribution	$\mathbf{.03 \pm .00}$	$.06 \pm .01$
Independent QR	$.10 \pm .02$	$.12 \pm .02$
2 comp. Normal MDN	$\mathbf{.05 \pm .02}$	$.12 \pm .05$
3 comp. Normal MDN	$.07 \pm .02$	$.14 \pm .04$
4 comp. Normal MDN	$.10 \pm .03$	$.17 \pm .06$
10 comp. Normal MDN	$.19 \pm .04$	$.19 \pm .06$
2 comp. Laplace MDN	$.05 \pm .01$	$.09 \pm .01$
3 comp. Laplace MDN	$.08 \pm .02$	$.11 \pm .02$
4 comp. Laplace MDN	$.13 \pm .05$	$.12 \pm .03$
10 comp. Laplace MDN	$.24 \pm .03$	$.18 \pm .05$
Independent $\mathcal{ALD}$	$.06 \pm .01$	$\mathbf{.02 \pm .01}$
UMAL model	$\mathbf{.04 \pm .01}$	$.07 \pm .01$

have plotted these measures for different methods (in green, our model) when considering the BCN RPF dataset on the left side of Figure 3.24. Furthermore, on the right side of the same Figure, we report the mean absolute error between the empirical measures and the ideal ones for both rental-price data sets. As we can see, the conditional distribution predicted by UMAL has low error values. Therefore, we can state that UMAL produces proper and calibrated conditional distributions especially suitable for heterogeneous problems[1].

Predicted distribution shape analysis From right to left in Figure 3.25, we show a 50 perplexity with Wasserstein distance t-SNE (Maaten and Hinton, 2008) projection from 500 linearly spaced discretization of the normalized predicted distribution to 2 dimensions for each room of the test set in Barcelona. Each colour of the palette corresponds to a certain DBSCAN (Ester et al., 1996) cluster obtained with $\epsilon = 5.8$ and 40 minimum samples as DBSCAN parameters. We show a Hex-bin plot over the map of Barcelona, where the colours correspond to the mode cluster of all the rooms inside the hexagonal limits. A similar study

[1]In the Appendix section of Brando et al. (2019b), we have evaluated calibration quality and negative log-likelihood on the UCI data sets with the same architectures as Hernández-Lobato and Adams (2015a); Lakshminarayanan et al. (2017).

would be useful to extract patterns inside the city, and consequently adapt specific actions to them.

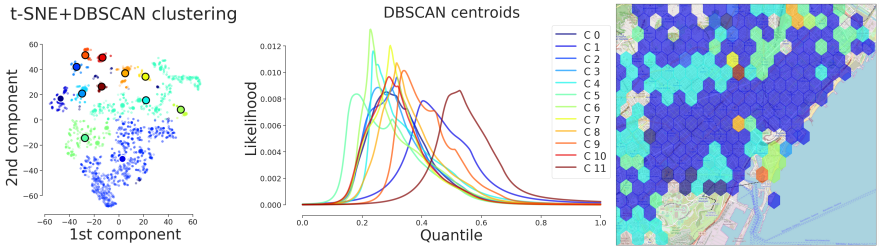


Figure 3.25: DBSCAN clustering of the t-SNE projection to 2 dimensions of normalized Barcelona predicted distributions. Hexbin plot of most common clusters for each hexagon over the map.

3.5 Conclusions

This chapter has introduced how to model aleatoric uncertainty (using the definition proposed in Section 2.1) by means of conditional parametric and distributional-free approaches.

In the parametric proposed models, it starts from the homoscedastic conditional model, and proposing several heteroscedastic models that includes single parametric distributions, a finite number of components to define a mixture and, ultimately, an infinite number of components defining an uncountable mixture. Particularly, the Uncountable Mixture of Asymmetric Laplacians (UMAL) model was proposed, a framework that uses deep learning to estimate output distribution without strong restrictions (Figure 3.9). As shown in Figure 3.10, UMAL is a model that implicitly learns infinite $\mathcal{A}LD$ distributions, which are combined to form the final mixture distribution. Thus, in contrast with mixture density networks, UMAL does not need to increase its internal neural network output, which tends to produce unstable behaviours when it is required as it is discussed in Section 3.2.3. Furthermore, the Monte Carlo sampling of UMAL could be considered as a batch size that can be updated even during training time.

On the other hand, the distributional-free proposed models starts from the single quantile regression models, which allows to perform conditional threshold

estimation. Increasing to simultaneously estimation of several quantiles, which allows to create distributional-free confidence intervals. As it was shown in X, the probabilistic training of neural networks allows to extend these models to implicit function estimations, which allows to estimate quantile functions. After that, the connection between $\mathcal{A}LD$ and quantile regression was highlighted to connect the previously proposed UMAL model from a quantile regression viewpoint to these distributional-free models.

Finally, we have presented a benchmark comparison in terms of log-likelihood adaptation in the test set of three different types of problems. The first was a synthetic experiment with distinct controlled heterogeneous distributions that contains multimodality and skewed behaviours. Next, we used public data to create a complex problem for predicting the room price per night in two different cities as two independent problems. Finally, we compared all the presented models in a financial forecasting problem anticipating the next monetary aggregated monthly expense or income of a certain customer given their historical data. We showed that the UMAL model outperforms the capacity to approximate the output distribution with respect to the other baselines as well as yielding calibrated outputs.

In introducing UMAL we emphasise the importance of taking the concept of aleatoric uncertainty to a whole richer level, where we are not restricted to only studying variability or evaluating confidence intervals to make certain actions but can carry out shape analysis in order to develop task-tailored methods. Additionally, we saw the connection between UMAL and the distributional-free models approximated by means of quantile regression.

Chapter 4

BLACK-BOX WRAPPER FOR UNCERTAINTY MODELLING

Until now, we have been designing methods that exclusively estimate uncertainty. However, can these uncertainty modelling solutions be somehow applied to model the uncertainty of an already existing non-uncertainty predictive system as a post-hoc improvement to uncertainty modelling? In such a case, the fewer assumptions we require regarding the already existing system the better, and it would therefore be useful to assume as generic an approach as possible. Consequently, we started by considering this already existing system as a plain mathematical function where, given an input, it produces an output and there are no other strong assumptions or requirements, regardless of how this output is provided. In such cases, we call this already existing predictive system a black box.

Specifically, a black box constitutes a deterministic (point-wise) relationship between the response Y and the covariate X variables presented in Eq. 2.1. This relationship usually corresponds to an estimation of a conditional summary statistic of $p(Y \mid X, M)$ (e.g. a percentile, a moment or any interesting conditional value), which can then be used as a point-wise predictor. Here, we are going to combine this point-wise estimation with the rest of the uncertainty modelling for "recovering" the original conditional probability, $p(Y \mid X, M)$. Explicitly, the aim is to determine a new conditional density model, $q(Y \mid X, M)$ that uses black-box predictions to improve the prediction or, even, satisfies that the values of its cho-

sen conditional summary statistic for $q(Y \mid X, M)$ correspond to the black-box values.

There are two main reasons for considering the predictive system as a black box:

- Forecasting systems applied in real-world solutions often require much more effort than mere code development. For instance, they typically require that dependencies are fulfilled, documentation written and unit tests developed, but this must also align with businesses interests. The costs of this are included in the *technical debt* debate, a concept introduced in 1992 to tackle long-term costs incurred when moving quickly in software engineering (Cunningham, 1992; Sculley et al., 2015). Considering the system as a black box for uncertainty modelling enables us **to give a longer life to the current implementation**, thereby avoiding hard implementation changes while focusing on the uncertainty modelling part and its subsequent new applications.

- Modelling the uncertainty of a point-wise predictive system, considered as a black box, forces us to disentangle the estimation of aleatoric uncertainty from the point-wise prediction so that uncertainty modelling techniques based on deep learning can be mixed with **the original predictive system, which does not need to depend on deep learning**.

  Thus, the black-box uncertainty modelling approach introduces a way of using deep learning estimation while maintaining the original prediction, which has potential beneficial aspects for industrial applications. For instance, critical real-world scenarios require that an appropriate regulation is satisfied (e.g., medical (Ustun and Rudin, 2016) or financial models (Rudin, 2019) may have to accomplish interpretability constraints, which nowadays can prevent techniques such as deep learning models from being used. Consequently, scenarios where it is necessary to have a "certain kind of" point-wise predictive system that does not model uncertainty, but, at the same time, where the cost of an erroneous prediction is high, can be tackled by providing a hybrid solution that incorporates uncertainty modelling information without changing the non-deep learning original system prediction.

Thus, black-box uncertainty modelling aims to provide what we denote as an "uncertainty wrapper", which avoids replacing the point-wise predictive system by preserving all the internals of that system (i.e. it can be a parametric model, a hand-crafted rule-based system or even a human decision). Therefore, the M in Eq. 2.2 is mostly formed by a presumed unknown piece of information, which leads us to focus on the aleatoric $p(Y \mid X, M)$ part where M is given.

On the whole, considering the predictive system as a black box allows us to give the original software a longer lifespan and to model aleatoric uncertainty while a certain condition between the point-wise predictive system and the uncertainty modelling wrapper is desired or enforced, depending on the problem statement. When this condition is enforced, it will henceforth be referred as the "open" case, in contrast with the "covert" one. A formal description of the two problem statements presented here may be summarized with the following definitions:

Definition 3 (Covert Black box) *Let X be an independent random variable in $\mathbb{R}^D$ and let Y be a response random variable to be regressed. A model representing a function $K\colon \mathbb{R}^D \to \mathbb{R}$ is said to be a covert black box - or simply a black box - if it approximates an unknown conditional statistic (i.e. a percentile or a moment) of $p(Y \mid X, M)$ and no other strong hypothesis regarding the internals of that model - or the summary statistic that is approximating - are made.*

For instance, a rule-based system that approximates the conditional mean of $p(Y \mid X, M)$ can be a covert black-box model if we consider it as a simple function, $K(\boldsymbol{x})$, avoiding extra considerations such as it approximating the conditional mean or avoiding any internal information regarding the model for the next uncertainty modelling process. In this case, the goal will be as follows:

Definition 4 (Uncertainty Modelling of a Black box) *Let X and Y be respectively covariate and response variables and let $K(\boldsymbol{x})$ be a covert black box (in the sense of Def. 3) over $p(Y \mid X, M)$. The Uncertainty Modelling of a Black box (UMB) consists in approximating the conditional density $p(Y \mid X, M)$ with a new model $q(Y \mid X, M)$ using the learnt $K(\boldsymbol{x})$ as additional information.*

Importantly, Def. 4 does not link the new conditional density model, $q(Y \mid X, M)$, with the previously learnt statistic, $K(\boldsymbol{x})$, more than as extra information to predict the former. Therefore, the uncertainty modelling of a covert black box considers that we do not have access to the statistic that is approximating this black box. However, having this extra information could be something independent that still preserves $K(\boldsymbol{x})$ being a black box, i.e. it preserves our non-knowledge requirements regarding the internals of $K(\boldsymbol{x})$. In fact, if we are able to infer or know that the statistic is approximating the black box, then we can use this information in favour of a better approximation or even to ensure that the new predicted distribution, $q(Y \mid X, M)$, satisfies that its chosen conditional statistic is still the black box value. This additional objective requirement introduces the following definition:

Definition 5 (Honest Black box) *Let X be an independent random variable in $\mathbb{R}^D$ and let Y be a response random variable to be regressed. Assume a fixed certain desired conditional summary statistic (i.e. a percentile or a moment) of $p(Y \mid X, M)$. A model representing a function $K\colon \mathbb{R}^D \to \mathbb{R}$ is said to be an honest black box if it approximates this summary statistic of $p(Y \mid X, M)$ and no other strong hypothesis regarding the internals of that model are made.*

Importantly, by using an honest black box, since we know the desired conditional summary statistic is approximating of $p(Y \mid X, M)$, we can build a goal of predicting a new conditional density model that preserves the learnt statistic by K as follows:

Definition 6 (Uncertainty Modelling of an Honest Black Box) *Let X and Y be respectively covariate and response variables and let K be an Honest black box (in the sense of Def. 5) that approximates a desired statistic of the unknown conditional distribution $p(Y \mid X, M)$. The Uncertainty Modelling of an Honest Black Box (UMHB) consists in determining a new conditional density model, $q(Y \mid X, M)$, which approximates $p(Y \mid X, M)$, such that the* desired *summary statistic of $q(Y \mid X, M)$ corresponds to the black box, $K(\boldsymbol{x})$.*

Note that a model that tackles the UMHB problem is able to disentangle a

point-wise *desired* summary statistic estimation - and associate it with $K(\boldsymbol{x})$ - from the aleatoric $q(Y \mid X, M)$ distribution approximation.

Furthermore, in the UMHB context, the mismatch between the real conditional statistic and the black box, $K(\boldsymbol{x})$, produces a new source of aleatoric uncertainty, which differs from the one derived from the data. However, the way it is estimated still entails using $p(Y \mid X, M)$. Importantly, as we will see in the Experimental Section 4.2.4.3, a poorly estimated $K(\boldsymbol{x})$ will impact the modelling of $p(Y \mid X, M)$, given that we always impose the constraint to be satisfied with respect to $q(Y \mid X, M)$.

Regarding epistemic uncertainty modelling, the UMB and UMHB problems, by definition, cannot access the K part of M in Eq. 2.2. Thus, given that $K(\boldsymbol{x})$ can be considered fixed throughout the process (because it corresponds to the black box), the main uncertainty we will tackle will be the aleatoric one.

In this chapter, firstly in Section 4.1, we will provide a literature compilation and clarify which methods cannot be considered for the uncertainty modelling of a black box and, then, in Section 4.2 we will tackle a generic UMB scenario. Following that, a restricted UMHB case will be analysed in Section 4.3. For both cases, we will have an experimental section with its respective results.

4.1 Related work

Uncertainty estimation of a black-box predictive system in regression problems was initially tackled in Brando et al. (2020). In contrast, in Brando et al. (2022b) we propose imposing a constraint between the predicted quantiles and the original black-box system, consequently adding the "honest" term introduced previously to the aleatoric uncertainty model. This imposition ensures that the black box is always the desired statistic with respect to the new predicted distribution.

As stated previously, the UMHB problem involves considering models that not only approximate the $p(Y \mid X, M)$ with a $q(Y \mid X, M)$, but also constrain $q(Y \mid X, M)$ in such a way that the desired conditional summary statistic of that corresponds to $K(\boldsymbol{x})$. At the same time, this $K(\boldsymbol{x})$ will correspond to a previously fixed function that estimates the desired conditional summary statistic

of $p(Y \mid X, M)$. Consequently, not all models that approximate $p(Y \mid X, M)$, such as the common QR models (Koenker and Hallock, 2001) or a mixture model (Bishop, 1994b), can be considered to tackle the present problem. We need models that in some way disentangle the conditional summary statistic approximation from the rest of the $p(Y \mid X)$ modelling.

The following sections are divided among those models that can be applied to the MBU problem (shown in Section 4.2) and those that only can be applied to the UMHB one (shown in Section 4.3).

4.2 Uncertainty modelling of a black box

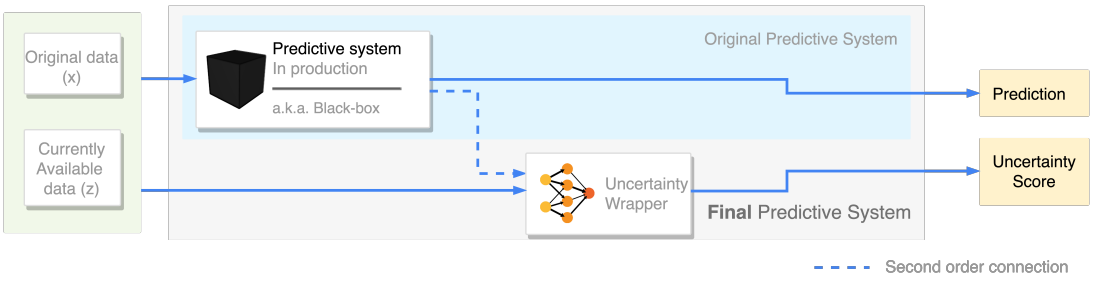


Figure 4.1: Graphic representation - obtained from (Brando et al., 2020) - of how to upgrade any black-box predictive system with an Uncertainty Score.

The covert statement introduced in the UMB provides an obscure scenario compatible with the following viewpoint: The black box can be seen as the implementation of a predictive model presenting a function of the form $\hat{y} = K(\mathbf{x})$. Here, $\mathbf{x}$ represents a set of inputs, and $\hat{y}$ represents a predicted quantity, which we will denote *prediction*. The fundamental assumption is that we do not know the functional form of K, and all we can do is invoke $K(\mathbf{x})$ and observe the result $\hat{y}$. The goal in this generic Modelling of Black-box Uncertainty is to use the point-wise information of the black-box predictor as input information for the uncertainty modelling part.

As highlighted in Brando et al. (2020), for instance, experienced data scientists probably identify with this situation when using a model method implemented in an off-the-shelf library (Meng et al., 2016; Pedregosa et al., 2011), API, or cloud

service – in many implementations, the function K is called `predict()`.

To further frame our problem, let us discuss in greater detail some considerations related to the inputs, output and assumptions regarding K (following Figure 4.1).

Inputs to K The inputs $\mathbf{x}$ represent the attributes that would be passed to K. We consider two situations. On the one hand, when we have access to the actual values of $\mathbf{x}$ at the time we invoke K, referred to as *preserved input*. On the other hand, we consider a more general situation, noted as *distorted input* $\mathbf{z}$, where one may not be able to observe the inputs $\mathbf{x}$ directly and instead have access to some features of them, or to some public contextual variables, and can still see the result of $B(\mathbf{x})$ to build the *Uncertainty Wrapper*, ψ, as shown in Figure 4.1.

- The case of *preserved input* happens when we have access to a data set with the same or equivalent data points as the original data set that was used to train K.
- The case of *distorted input* happens when we have access to the prediction of the black-box system $K(\mathbf{x})$ but not to its input $\mathbf{x}$. For example, consider a public API to forecast electricity consumption in some geographic area. While we can call the API, and obtain its *prediction*, in this case we do not have access to the input variables. However, as shown in later experiments, we see that we can still use the available variables $\mathbf{z}$ (e.g. previous consumption values) to provide a reasonable input context to forecast uncertainty.

Finally, we should consider two different model configurations. First of all, when the input of the *Uncertainty Wrapper* is $\mathbf{z}$, we refer to the model as a *First Order Decision model* $\psi(\mathbf{z})$. Otherwise, when the input of the model is $\mathbf{z}$ and the black-box prediction $K(\mathbf{x})$ are both considered input attributes of our uncertainty estimator, we denote this model *Second Order Decision model* $\psi(\mathbf{z}, K(\mathbf{x}))$.

In the experiments Section 4.2.4.3, we will consider *preserved* and *distorted* input problems and also compare *First* and *Second Order Decision models*, as shown in Figure 4.1.

Output of K The output, $\hat{y}$, represents the predicted quantity. In this thesis, we consider regression problems and so $\hat{y}$ is a scalar. Without loss of generality, the methods proposed in this thesis apply to vector-valued outputs. Moreover, extensions to binary classification problems would be possible if we considered $\hat{y}$ a classification score.

The black-box K We assume we do not know the functional form of K, which is the common situation when using an off-the-shelf piece of software. However, we note that our proposal also applies to situations where K may be known but changes in K are not allowed or would be expensive. Therefore, our definition of black box in fact means that K is *immutable* (either because it is unknown, or because it is costly or impossible to replace for any of the given reasons).

4.2.1 Probabilistic distribution fitting

As we saw in Section 3.2, certain conditional distributions have the property that one of its parameters corresponds to its mode. In scenarios where the black box is trying to estimate this value, these distributions can be useful, since we can replace the mode parameter value for the black-box value, as we will see later.

Specifically, when considering neural network models, until now we have seen how to model the target variable distribution as a Normal (Bishop, 2016) or Laplace distribution (Brando et al., 2018a) and solve a Maximum Likelihood estimation problem in order to find the optimal network parameters. Both distributions are a sub-case of the parametric family of symmetric distributions known as Generalized Normal Distribution (GND). Hence, we can cover the previous approaches by assuming $\hat{y} \sim \mathcal{GN}(B(\mathbf{x}), \psi(\mathbf{z}), \beta)$, where $\mathcal{GN}$ is a GND with the black-box output, $B(\mathbf{x})$, is a location parameter and the neural network wrapper, $\psi(\mathbf{z})$, is a scale parameter. In this case, the distribution function is:

$$\mathcal{GN}(y \mid B(\mathbf{x}), \psi(\mathbf{z}), \beta) = \frac{\beta}{2\psi(\mathbf{z})\Gamma(1/\beta)} \exp\left(-\frac{|y - B(\mathbf{x})|}{\psi(\mathbf{z})}\right)^{\beta} \tag{4.1}$$

where Γ denotes the Gamma function. Note that β is another parameter to be optimized and, as highlighted previously, it represents the Laplace distribution

when $\beta = 1$ and the Normal distribution when $\beta = 2$.

Importantly, the scale parameter of the distribution, $\psi(\mathbf{z})$, is a function that depends on $\mathbf{z}$, the *preserved* or *distorted* information available at the input. Therefore, our goal can be stated as that of finding a functional form of $\psi(\mathbf{z})$ and the parameter β which maximizes the corresponding log-likelihood:

$$\mathcal{L}\left(y, B(\mathbf{x}), \mathbf{z}\right) = \log\Gamma(1/\beta) - \log\beta + \log\psi(\mathbf{z}) + \left(\frac{|y - B(\mathbf{x})|}{\psi(\mathbf{z})}\right)^{\beta} \tag{4.2}$$

As mentioned earlier, the reason to consider $\psi(\mathbf{z})$ as a deep neural network is clear: we are trying to approximate a function that might be complex and non-linear, which is why a high-capacity model is advised. We assume that estimating the possible variability of an output given the input is as challenging as predicting the expected value. This setting is model-agnostic with respect to the *Uncertainty Wrapper* neural network architecture. Given the neural network output, $NN(\mathbf{z})$, which by default can take positive and negative values, the only requirement is that $\psi(\mathbf{z})$ needs to be strictly positive, and we therefore apply a `Softplus` function to the output of the neural network, i.e. $\psi(\mathbf{z}) = \log\left[1 + \exp\left(NN(\mathbf{z})\right)\right]$, similarly to that used in Section 3.2.1.

In our particular case, however, a crucial detail is that $K(\mathbf{x})$ is fixed, and thus, the loss function will not be optimized with respect to it, so the values of $|y - K(\mathbf{x})|$ are always the same.

Finally, on the one hand, we can interpret the loss function of Eq. (4.2) as the sum of a regularization term and a reconstruction term. If $\psi(\mathbf{z})$ is smaller than the scale of these errors, the reconstruction term $\left(\frac{|y-K(\mathbf{x})|}{\psi(\mathbf{z})}\right)^{\beta}$ penalizes the loss value. Otherwise, if $\psi(\mathbf{z})$ is too large, then the regularization term $\log\psi(\mathbf{z})$ becomes dominant. So we have an equilibrium that yields $\psi(\mathbf{z})$ as the predicted scale of the errors depending on the input.

On the other hand, the job of β is less evident for some fixed set of errors $|y - K(\mathbf{x})|$ and *Uncertainty Scores*, $\psi(\mathbf{z})$. The optimal β is such that the shape of the distribution fits better, i.e. high values of β correspond to plateau-like distributions, low values of β to point-shaped ones.

4.2.2 Distribution estimation of the residuals errors

Until now, the proposed approaches to model black-box uncertainty assume a certain conditional distribution of the response variable. However, an alternative approach would be to directly estimate the residual error between the black box and the value to be predicted. Implicitly, we can do this because y is an unknown function of $\mathbf{x}$. Assuming the residuals follow a Generalized Normal Distribution of unknown variance, the loss function for this case is the negative logarithm of the likelihood, out of constants and dependence from the variance:

$$\mathcal{L}\left(y, K(\mathbf{x}), \mathbf{z}\right) = \log\Gamma(1/\beta) - \log\beta + |y - K(\mathbf{x}) - \psi(\mathbf{z})|^{\beta} \tag{4.3}$$

4.2.3 Quantile regression of residuals

All of the above being said, the previously proposed approach assumes a certain conditional distribution for the residual errors. Following the distributional-free idea of Section 3.3, we can estimate the quantiles of the residuals. As introduced in Reminder 3, a well-known approach to compute confidence intervals and deal with prediction with uncertainty is quantile regression, in which the target estimate is a certain quantile of the distribution of real values, rather than the mean (Hao et al., 2007; Koenker and Hallock, 2001). The length of the centred 90% confidence interval can be used as a proxy of the uncertainty of the estimation. To do this, we define two functions $\phi^{+}(\mathbf{z})$, $\phi^{-}(\mathbf{z})$ as the $(\frac{19}{20} =)95\%$ and $(\frac{1}{20} =)5\%$ percentiles of the distribution of the residual error, $y - K(\mathbf{x})$, for each function and use a surrogate hinge loss to them Pereira et al. (2014) in the following way:

$$\begin{cases} \mathcal{L}^{+}\left(y, B(\mathbf{x}), \mathbf{z}\right) = \max\left[19(y - K(\mathbf{x}) - \phi^{+}(\mathbf{z})), \phi^{+}(\mathbf{z}) - y - K(\mathbf{x})\right] \\ \\ \mathcal{L}^{-}\left(y, B(\mathbf{x}), \mathbf{z}\right) = \max\left[y - K(\mathbf{x}) - \phi^{-}(\mathbf{z}), 19(\phi^{-}(\mathbf{z}) - y - K(\mathbf{x}))\right]. \end{cases} \tag{4.4}$$

From these, as we did in the shape parameter in Section 3.2.1, we define the *Uncertainty Score* as the Softplus of the quantile difference,

$$\psi(\mathbf{z}) = \log\left[1 + \exp(\phi^{+}(\mathbf{z}) - \phi^{-}(\mathbf{z}))\right]. \tag{4.5}$$

4.2.4 Results and comparison

4.2.4.1 Baselines under evaluation

In order to evaluate our proposals, we compare the previous proposals with two methods that allow an uncertainty proxy to be obtained given a dataset of triplets $(\mathbf{z}_n, \hat{y}_n, y_n)$. Additionally, we define a simple prediction baseline as a sanity check.

Nearest Neighbour Distance The distance to the n-th nearest neighbour in the input space Thompson (1956), $\mathbf{z}$, can be seen as a proxy of "normality", which is sometimes related to reliability. As a simple baseline, we used the distance to the $5th$ neighbour to sort predictions by reliability. The distance to other neighbours can also be considered. We have abbreviated this method to *NN*.

Nearest Neighbour Regression As well as using distance, we can also use the targets of the nearest neighbour to obtain a direct estimation of uncertainty (Altman, 1992). Specifically, if we call $Y_{neigh} = y_{\pi(1)}, \ldots, y_{\pi(K)}$, the targets of the K first neighbours, we use $\text{std}(Y_{neigh})$ as an estimate of prediction uncertainty. We denote this method as *NN-Reg*.

Gaussian Processes Following the product-of-GP-experts model proposed in Deisenroth and Ng (2015), we built an ensemble of N Gaussian processes, $\{GP_i\}_{i=1}^{N}$, where each one is trained with a different part of the training set to predict the difference between the black-box prediction and the real value, i.e. $|y - K(\mathbf{x})|$. Thereafter, as each Gaussian process predicts a mean, $\mu_i(\mathbf{z})$, and a variance, $\sigma_i(\mathbf{z})$, one way of defining the uncertainty score could be:

$$\psi(\mathbf{z}) = \frac{1}{N}\sum_{i=1}^{N} \mu_i^2(\mathbf{z}) + \sigma_i^2(\mathbf{z}) \tag{4.6}$$

In this way, we avoid the scalability problems typically found in Gaussian Processes. We refer to this baseline as *GP*.

Standard deviation of z When forecasting the next value of a time series, the standard deviation of the previous time series points can be used as the simplest means of estimating the uncertainty of the next value. While we would expect this to yield a poor uncertainty proxy, we added it as good experimental practice to make sure that our proposal yields much better results than simple cases. From now on, we refer to this as *std*.

4.2.4.2 Data sets and experimental settings

All the datasets used are of the form $[\mathbf{z}, y, \hat{y}_i]$, where $[\mathbf{z}, y]$ are the real data, $\mathbf{z}$ the available inputs and y the target variable(s), and $\hat{y}_i$ are each of the black-box estimates for y.

Forecasting Bank Customers' Impending Financial Expenses and Incomes Following Brando et al. (2018a), our problem is to forecast upcoming monthly expenses and incomes in a certain aggregated financial category for each bank client. Each time series contains 24 points and the goal is to predict the next aggregated month. To build the dataset, we used 2 million randomly-selected time series for a single-year training set and 1 million more for the test corresponding to the following year.

Estimating Electrical Power Demand In this problem, we have to forecast the mean electrical power demand in two hours, given the means for each two-hour period over the previous 72 hours. Thus, we have time series of 36 points and the problem is forecasting the next one. The data were prepared from the sets made publicly available by Red Eléctrica of Spain, which can be found at REE (2020). The public series are at intervals of 10 minutes, so we averaged every 12 points to obtain the mean value for 2 hours. In this experiment, data were captured for the period comprising 1-1-2014 to 18-10-2018, so we have 250,000 points, split evenly between train and test.

Predicting a Biological Response Challenge We also considered a real-world dataset from a public Kaggle challenge (Ingelheim, 2012). The data comprised $1,776$ numerical descriptors representing the size, shape or elemental constitution of each molecule. The aim here was to predict whether the molecule was seen to elicit a biological response. Although the challenge is a binary classification problem, for the purposes of this article, we regard it as a *regression* task, where it is necessary to predict a real-valued score (0 or 1). The interest of this dataset lies in the fact that one of the participants published the code for her solution (Olivetti, 2012), which we used as a black box. The dataset had $3,751$ points: 500 were used to train the black box, $2,000$ to train the confidence estimator and $1,251$ were used as a test set.

Black boxes used for evaluation

The aim of using several black boxes was to simulate different situations encountered in real-world problems, where an interpretation is needed or the function $B(\mathbf{x})$ cannot be changed for practical reasons. For each dataset, we took *existing real systems* as black boxes, and in order to extend the study, in some cases complemented them with additional simulated black boxes, as follows:

For the **Financial dataset**, we used the following black boxes:

- *Mean*: The average forecast of the historical input values: $\hat{y} = \bar{x}$.
- *Last*: The last observed value in the time series with 24 points: $\hat{y} = x_{24}$, which is known as the "naive method" in the forecasting literature (Hyndman and Athanasopoulos, 2014).
- *In Production*: The in-production system that produces forecasts for the banking app. This system is highly optimized for production purposes, difficult to replace, and uses a number of different models and software components. It outputs a point forecast but not a prediction interval. Therefore, it complies with many of the black-box assumptions described in this work.

For the **Electrical Power Demand dataset**, we considered:

- *Company*: Red Eléctrica's own forecasting consumption, computed by the company itself. These forecasts are available as part of the dataset, but we ignored how the predictive system is designed. Thus, we could be in a distorted input case, following the initial part of the current Section 4.2.
- *RT*: A regression tree model with depth 4, available from the `sklearn` library.

For the **Biological Response dataset**, we used a participant's solution to the Kaggle challenge, as published in Olivetti (2012), hereafter referred to as *Kaggle*.

At this point, it is important to highlight the different kinds of black boxes used. Firstly, the initial two black boxes (*Mean* and *Last*) proposed for the Financial dataset are a clearly *preserved input* case, since both the forecasting method and the uncertainty estimation method work with the expense time series data. However, the third case (*In Production*) is a *distorted input* scenario, since the in-production forecaster uses more attributes than just the previously predicted series, which our uncertainty wrapper has no access to. Similarly, the Electrical Power Demand estimation done by the company probably uses additional information that was not available to us (we assume these to be weather conditions, dummies for special dates, etc.), meaning that *Company* is also a *distorted input* scenario.

Finally, we considered *First* and *Second Order Decision* models for all of the alternatives.

Deep learning specifications

Different architectures were combined with different loss functions. In the two time series datasets, we used recurrent and dense networks (called *LSTM* and *dense*, respectively), but only the latter in the classification/regression dataset.

In the **Financial** and the **Electrical Power Demand datasets**, the dense network has two hidden dense layers with 50 and 20 units, respectively, while the recurrent network has a first hidden layer of 50 LSTM units and a second of 20 dense neurons.

In the **Biological Response dataset**, the dense network has two hidden layers with $3,000$ and $1,000$ units, respectively, due to the high number of input attributes.

The *Second Order* dense *decision models* have the same structure as the *First Order* ones, but adding the black-box output as an extra input, whereas the *Second Order* recurrent *decision models* have an extra dense layer to extract features from the black-box output to 10 units, which feeds the second hidden layer as well as the output of the LSTM layer.

All the deep learning models were implemented using the automatic differentiation library TensorFlow Abadi et al. (2016) and, specifically, the Keras wrapper Chollet et al. (2019). Their corresponding parameters were optimized using a grid search of different parameter combinations for each model. Furthermore, they were trained in two phases using 500 epochs with early stopping and 10% of the training set as a validation set. As explained in Section 4.2.4.2, in the first phase, β is trainable so the model learns the shape of the distribution, whereas in the second phase, β is constant, so we have more numerical stability to learn ψ.

Note that when we perform quantile regression, we train two models, one for the interval's upper-bound function and another for the lower-bound one.

On the whole, we considered three different loss functions:

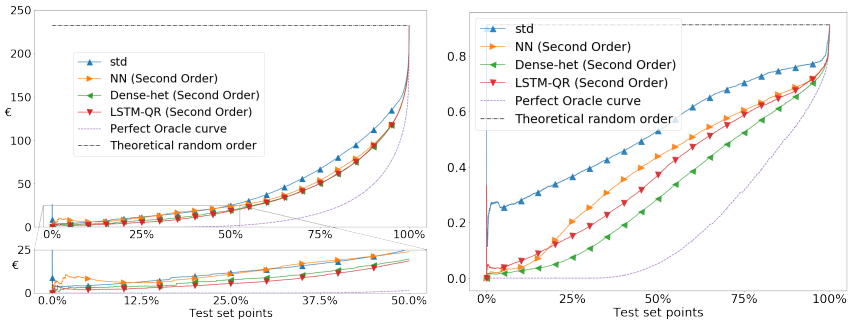
- *het*: Heteroscedastic aleatoric estimation loss, Eq. (4.2).
- *res*: Deterministic bias estimation loss, Eq. (4.3).
- *QR*: 90% centered coverage Quantile Regression, Eq. (4.4).

Hyperparameter policy of β

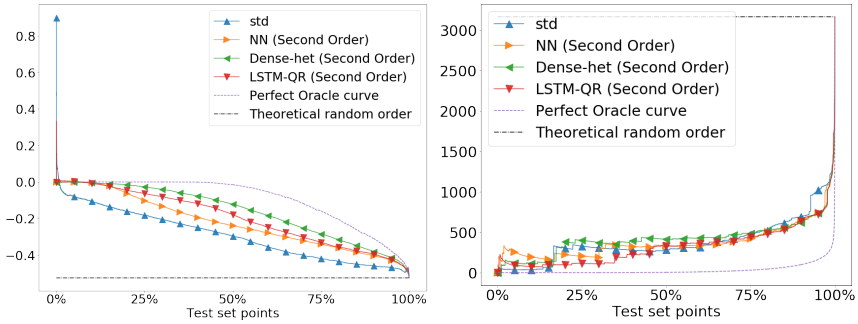
In both frameworks (heteroscedastic and residual), we first optimized the network with all the trainable parameters, and then froze β when it had converged (i.e. low variation of that parameter is regarded as convergence), and continued training the other weights (those used to compute $\psi(\mathbf{z})$). This provides more numerical stability, as minor changes in the value of β affect the loss value.

4.2.4.3 Experimental results

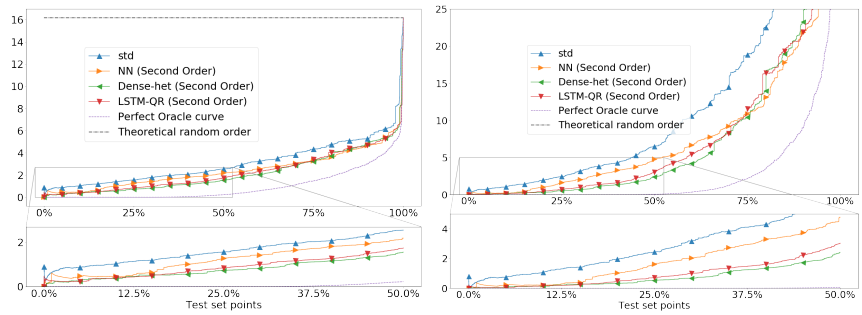
In this section, we evaluate quantitatively and qualitatively certain properties of the Uncertainty Wrapper, both for our in-house example, as well as in the real-world, public datasets.



(a) MAE measure, Eq. (4.7). (b) MAPE measure, Eq. (4.8).



(c) MPE measure, Eq. (4.9). (d) RMSE measure, Eq. (4.10).



(e) RMSPE measure, Eq. (4.11). (f) MSPE measure, Eq. (4.12).

Figure 4.2: Error-retain plot of the *In Production* black box for our Financial Forecasting problem using different scoring measures. Sub-figures (a), (e) and (f) have a zoomed shot of the initial 50% at the bottom.

Does the Uncertainty Wrapper predict the confidence? The first question we wish to solve is whether the uncertainty wrapper has the ability to filter those points where our black box makes larger errors. In other words, do the uncertainty scores rank the predictions by increasing error?

Note that traditional ways of evaluating regression (or forecasting) methods are real-valued error metrics, such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). However, these are computed based only on the predictions $B(\mathbf{z})$ and the true values, whereas here we are looking for a way to evaluate the quality of $\psi(\mathbf{z})$.

The Error versus Retention Curve (ERC) One way to compare the ordering quality of different uncertainty score functions, ψ_k, is by contrasting their different error-retain curves (Brando et al., 2018a). Figure 4.2 shows the curves for different methods applied to the financial dataset with respect to the "In Production" industrial black box. Each sub-figure corresponds to a different scoring measure indicated in the corresponding caption. These curve plots, on the y-axis, have the respective cumulative scoring measure, $\mathcal{D}$, corresponding to the subset of the predictions such that $\psi(\mathbf{z}) < \kappa$, where κ is a threshold. The x-axis shows the fraction of points under the threshold. Following (Fomby, 2008), if $\Psi_i = [\psi_k(\boldsymbol{z}_i) < \kappa]$, the different selected scoring methods are the Mean Absolute Error (MAE),

$$\mathrm{ERC}_{MAE}\left(y, \boldsymbol{z}, K(\mathbf{x}), \kappa\right) = \frac{\sum_{i=1}^{N} |y_i - K(\mathbf{x}_i)| \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}, \tag{4.7}$$

the Mean Absolute Percentage Error (MAPE),

$$\mathrm{ERC}_{MAPE}\left(y, \boldsymbol{z}, K(\mathbf{x}), \kappa)\right) = \frac{\sum_{i=1}^{N} \left|\frac{y_i - K(\mathbf{x}_i)}{y_i}\right| \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}, \tag{4.8}$$

the Mean Percentage Error (MPE),

$$\mathrm{ERC}_{MPE}\left(y, \boldsymbol{z}, K(\mathbf{x}), \kappa)\right) = \frac{\sum_{i=1}^{N} \frac{y_i - K(\mathbf{x}_i)}{y_i} \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}, \tag{4.9}$$

the Root Mean Squared Error (RMSE),

$$\mathrm{ERC}_{RMSE}\left(y, \boldsymbol{z}, K(\mathbf{x}), \kappa)\right) = \sqrt{\frac{\sum_{i=1}^{N} \left(y_i - K(\mathbf{x}_i)\right)^2 \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}}, \tag{4.10}$$

the Root Mean Square Percentage Error (RMSPE),

$$\mathrm{ERC}_{RMSPE}\left(y, \boldsymbol{z}, K(\mathbf{x}), \kappa)\right) = \sqrt{\frac{\sum_{i=1}^{N} \left(\frac{y_i - K(\mathbf{x}_i)}{y_i}\right)^2 \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}}, \tag{4.11}$$

and the Mean Square Percentage Error (MSPE),

$$\mathrm{ERC}_{MSPE}\left(y, \boldsymbol{z}, K(\mathbf{x}), \kappa)\right) = \frac{\sum_{i=1}^{N} \left(\frac{y_i - K(\mathbf{x}_i)}{y_i}\right)^2 \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}. \tag{4.12}$$

As we can see in Figure 4.2, for all methods we obtain a trade-off curve showing that the error decreases as we "accept" only forecasts with decreasing values of the uncertainty $\psi(\mathbf{z})$. In the experiment presented here, we have successfully added an uncertainty wrapper on top of an industrial black box, and we can now use it to filter the forecasts we are unsure about and not display them to the user.

We are in the scenario that metrics posses advantages in interpretability are preferable due to the predicted value is monetary. Following (Pontius et al., 2008), we consider better to focus on metrics that uses MAE instead of RMSE as they are fundamentally easier to understand than the latter. Consequently, although the standard deviation could seems better, in the initial part, if we

only show Figure 4.2.c case, we can observe that in all other cases the proposed methods based on heteroscedastic networks and quantile regression obtain the best performances.

Exhaustive Quantitative Evaluation Since it is impractical to visualize all error versus retain figures for all of the methods, we summarize all of them into a single metric, which we will denote as ordering score. This value will be computed for every combination of dataset, method, black box and order (First and Second order), and evaluated for all possible combinations.

The ordering score quantifies whether the ordering induced by an uncertainty function $\psi(\cdot)$ is close to the perfect or, otherwise, to a random ordering. Its computation details are explained below.

First, let us consider the error-reject curve of a perfect ordering. It is clear that the best possible ordering happens when it is the same as ordering by the real error, that is $\psi_o(\mathbf{z}_i) = |K(\mathbf{x}_i) - y_i|$. We denote this ideal situation as the "perfect oracle curve".

Similarly, the least informed way to sort by uncertainty scores is randomly. Clearly, this would yield a constant error-retain curve with a value corresponding to the MAE of the dataset (up to random fluctuations). We can define

$$\delta_k = \left[\left(\frac{1}{N} \sum_{i=1}^{N} |\psi_k(\mathbf{z}_i) - y_i| \right), \underset{\text{Repeat } N \text{ times}}{\ldots} \right] \tag{4.13}$$

as the vector with the value of the whole MAE of ψ_k.

At this point, where we have defined a lower and upper bound curve, we are able to define the **ordering score** of an uncertainty wrapper function, ψ_k, as

$$\mathcal{S}(\psi_k) = 100 \left(1 - \frac{A(\psi_k) - A(\psi_o)}{\delta_k - A(\psi_o)} \right), \tag{4.14}$$

where $A(v)$ is the area of the *error versus retain curve* of the function v. The ordering scores is one minus the ratio between the difference of area between the selected uncertainty wrapper and the oracle divided by the difference of the area of a random ordering criteria and the oracle. Therefore, the closer the ordering

Table 4.1: Ordering scores values for each of the methods explained in all Section 4.2 and for each of the data sets of Section 4.2.4.2.

Dataset	Financial forecasting						Electrical Power Demand				Biology	
Black-box	Mean value		Last value		In Production		Company Forecast		Regression Tree		Kaggle solution	
std	87.1		87.0		86.5		4.8		11.1		-	
NN	73.4	74.0	84.6	88.0	83.5	89.0	5.8 ± 0.0	5.8 ± 0.0	16 ± 0.0	20.5 ± 0.0	25.6 ± 0.0	25.7 ± 0.0
NN-Reg	82.4	84.4	87.9	88.5	88.2	88.6	11.2 ± 0.0	10.8 ± 0.0	15.1 ± 0.0	14.7 ± 0.0	35.6 ± 0.0	35.4 ± 0.0
GP	60.2	60.2	67.7	79.0	64.5	77.9	7.1 ± 0.3	7.5 ± 0.4	25.4 ± 0.3	82.9 ± 0.0	21.6 ± 0.9	21.8 ± 0.8
Dense-res	80.6	81.4	−4.2	5.1	−2.4	3.0	0.6 ± 3.5	1.9 ± 2.2	38.7 ± 3.0	50.4 ± 4.2	4.9 ± 0.0	4.3 ± 1.6
LSTM-res	79.7	78.5	0.9	4.5	0.0	0.0	1.8 ± 7.6	5.4 ± 5.0	42.9 ± 3.8	$\mathbf{85.2 \pm 1.6}$	-	-
Dense-QR	90.1	89.9	**91.3**	**91.1**	**90.7**	**90.8**	18.6 ± 1.4	16.9 ± 1.6	32.1 ± 2.4	18.2 ± 4.8	14.8 ± 4.8	14.1 ± 5.5
LSTM-QR	89.0	88.7	**90.8**	**91.4**	**90.3**	**91.0**	12.7 ± 0.6	17.1 ± 1.6	32.2 ± 2.9	14.4 ± 5.6	-	-
Dense-het	**92.9**	**91.0**	88.8	88.3	88.4	90.6	$\mathbf{20.4 \pm 0.9}$	$\mathbf{20.8 \pm 0.8}$	$\mathbf{50.4 \pm 5.8}$	83.9 ± 4.2	$\mathbf{51.6 \pm 2.6}$	$\mathbf{52.2 \pm 1.9}$
LSTM-het	**93.3**	**93.4**	89.8	91.2	87.8	86.8	$\mathbf{17.0 \pm 4.9}$	$\mathbf{19.3 \pm 3.2}$	$\mathbf{50.4 \pm 4.2}$	71.3 ± 7.7	-	-

Background color meaning: First Order Decision version of the model Second Order Decision version of the model

score is to 100, the closer it is to a perfect sorting. On the other hand, a value closer to zero (it may even be negative given stochasticity) will mean an almost random ordering.

In Table 4.1 we show the ordering scores values for all the combinations of datasets, black-boxes, methods and First/Second order choice explained in Sections 4.2.1, 4.2.3, 4.2.2, 4.2.4.1 and 4.2.4.2. For all public dataset, the mean and variance ordering scores values of 10 independent executions are reported.

Returning to the original question about detecting whether the uncertainty wrapper improves in our confidence, if we look at the Table 4.1, we see that all values are positive values, with the exception of some cases corresponding to the *LSTM-res*. Therefore, taking into account the definition of the ordering score, we can ensure that by using the wrapper constitutes an improvement in the performance. Additionally, we can observe that the deep Uncertainty Wrapper strategies proposed in this article are the ones that get the best results for every point compared to other baselines.

Furthermore, in the comparison presented in Table 4.1 we can observe that the ordering score exhibit relevant variations in scale depending on the complexity of the problem. This complexity is related with the presence of distorted or preserved inputs implying bad performance such as the residual uncertainty wrappers that obtained close to zero or even negative values.

Which is the best uncertainty wrapper?

The results of the Table 4.1 leads us to wonder if there is a method that stands out from the others systematically. While the 1st ranked method varies over the columns of the table, we can see that clearly using the Heteroscedastic methods as wrapper gives us the first or second best position for any problem and black-box. Thus, we can consider that the Heteroscedastic model is the most stable when it comes to getting good generic solutions.

Fine-grained analysis of orderings

Table 4.1 and Figure 4.2 give insights on the overall *quality* of the orderings induced by uncertainty wrappers.

We would also like to check monotonicity properties of the ordering, i.e. to what degree does the uncertainty score approximately sort by real-error? In other words, we want to match the intuition that "easy to predict" inputs should be assigned low uncertainty scores, while "potentially disastrous prediction" should be avoided.

To that end, we first group the samples by MAE, $|y_i - K(\mathbf{x})_i|$ into 10 bins. These can be interpreted as 10 different degrees of prediction difficulty (from lowest to highest error). When we vary the uncertainty threshold κ we could measure the % of points that fall into each of the 10 bins (y-axis), while the x-axis corresponds to the retain % (induced by κ). We would expect the low-error bins to capture most % at low retain rates, and conversely, that the high-error bins dominate at high values of the retain rate. For instance, this could be used to detect wrong predictions with a high Uncertainty Score and other unwanted scenarios.

In Figure 4.3, we can show that for all methods displayed, the bins with large error (the purplish ones) appear at the end. On the other hand, we can observe that the behaviour of both Heteroscedastic and quantile regression wrappers is quite smooth even for lower bins (the yellowish ones). Therefore, we can conclude that considering ordering score values is a proper manner to ordering regarding different types of errors.

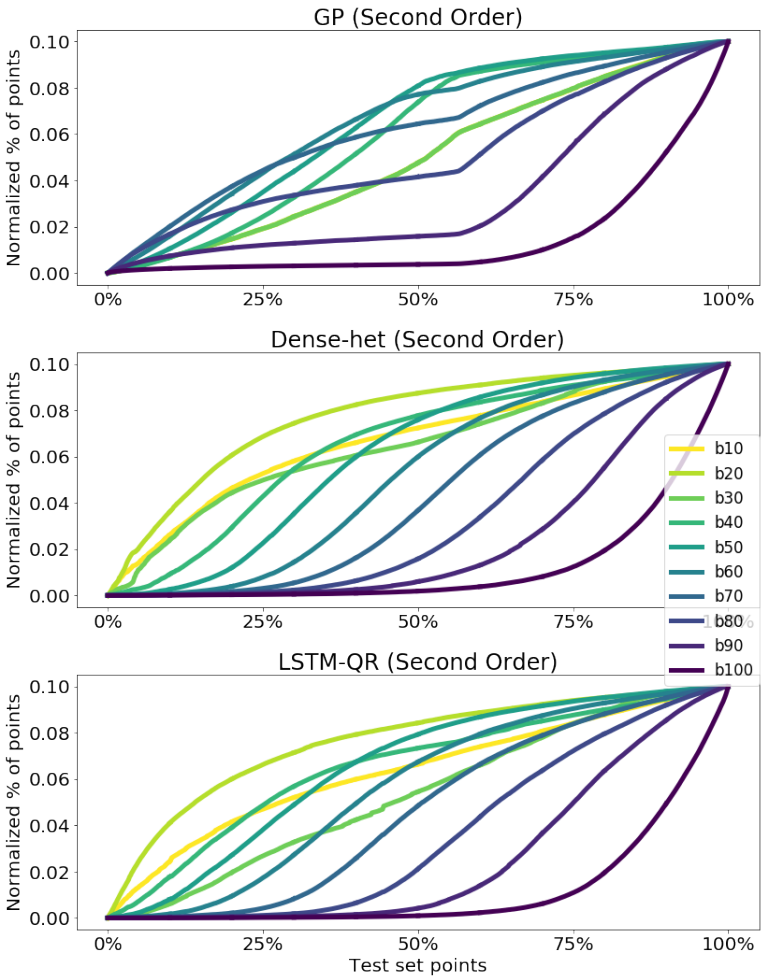


Figure 4.3: Percentage of points of each bin of real MAE error of the *In Production* black-box of our Financial Forecasting problem sorted by the uncertainty wrapper described in the title. Each color corresponds to a different real error bin indicated in the legend.

Table 4.2: β final value after the first part of the optimization for each model and problem.

Dataset	Financial forecasting						Electrical Power Demand				Biology	
Black-box	Mean value		Last value		In Production		Company Forecast		Regression Tree		Kaggle solution	
Dense-res	0.297	0.299	0.303	0.306	0.304	0.306	2.156 ± 0.000	2.156 ± 0.000	2.210 ± 0.002	2.165 ± 0.001	0.776 ± 0.014	0.777 ± 0.019
LSTM-res	0.298	0.297	0.301	0.304	0.304	0.306	2.155 ± 0.000	2.155 ± 0.000	2.210 ± 0.001	2.168 ± 0.001	-	-
Dense-het	0.629	0.675	0.673	0.673	0.677	0.393	1.170 ± 0.003	1.171 ± 0.004	2.655 ± 0.264	5.706 ± 0.654	0.689 ± 0.001	0.687 ± 0.001
LSTM-het	0.617	0.656	0.189	0.668	0.192	0.201	1.102 ± 0.009	1.190 ± 0.030	3.430 ± 0.414	10.17 ± 1.951	-	-

Background color meaning: First Order Decision version of the model Second Order Decision version of the model

Discretization in levels of confidence

Another important point to verify is the correlation between the ordering induced by each uncertainty wrappers and the 'true ordering' based on the real error (oracle). In this case, for each uncertainty wrapper we could define certain thresholds corresponding to the five required quantiles (i.e. values between the quantiles $0-20$, $20-40$, $40-60$, $60-80$ and $80-100$) and associate them to the five classes, respectively. Note that this is equivalent to defining five 'quality classes' (*Higher Error*, *High Error*, *Medium Error* , *Low error* and *Lower Error*) and computing a sort of confusion matrix between the true classes and the predicted classes.

In Figure 4.4, we observe the confusion matrix for our *In Production* problem where it is indicated in each box the normalized number of predictions that have coincided between the prediction of the oracle and the chosen uncertainty wrapper.

As we can see in Figure 4.4, it is easier for all the different presented models to detect the points with higher confidence than the others (given their more yellowish colour). However, the number of the percentage of points contained in such a box on the bottom left is different depending on the model: We can see that all baseline models, as well as the quantile regression model, have a significantly lower value than the *Heteroscedastic* case. This indicates us that, although the ordering score value in our *In Production* problem of the *Heteroscedastic* model may be slightly lower than the quantile regression model, the *Heteroscedastic* model detects in a better way the values that are more reliable. Accordingly, the *Heteroscedastic* model is the one that best orders its *High*, *Medium*, and *Lower*

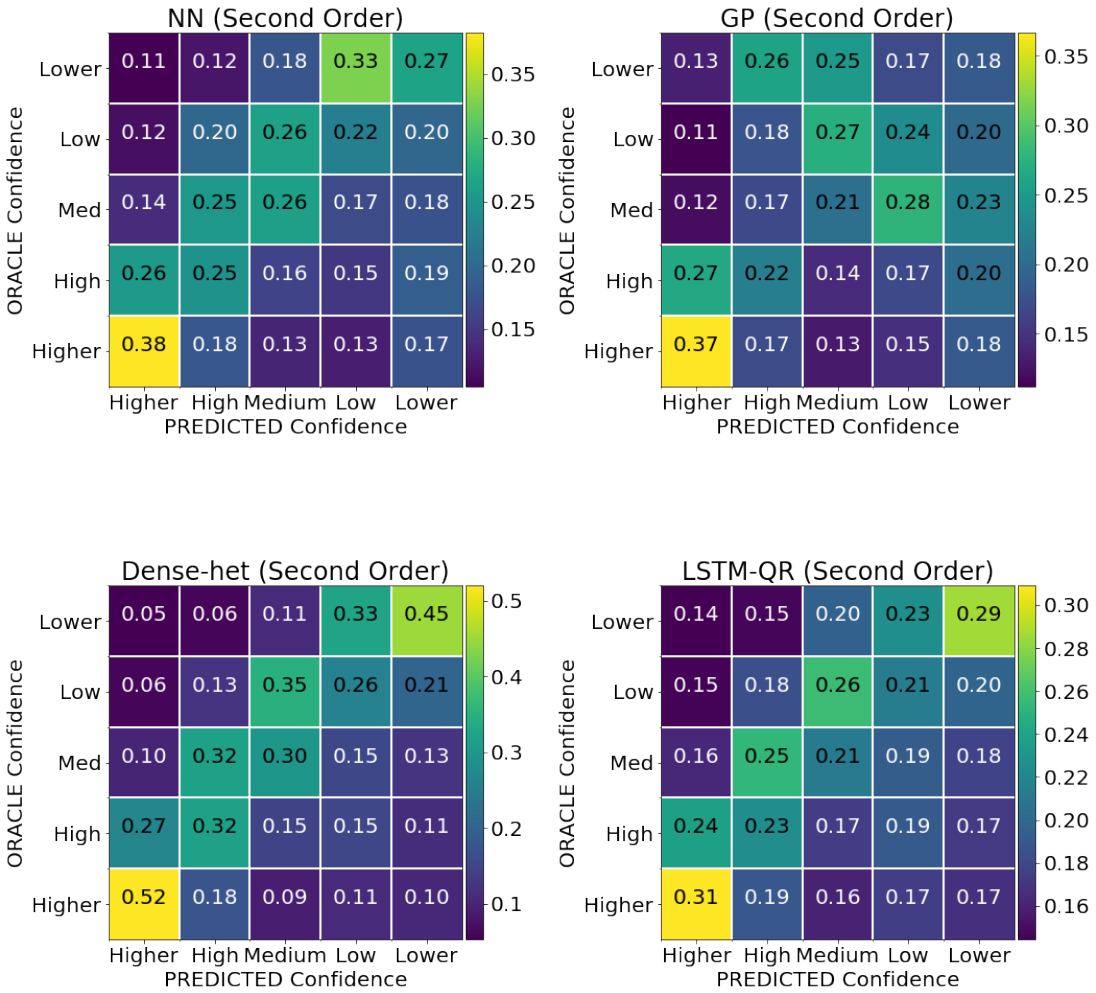


Figure 4.4: Normalized confusion matrix of the problem of classification into 5-levels of confidence for certain models. Each one has its own colour map scale.

values as well as those predictions that have a high probability of being erroneous in the upper right corner. In short, the model with the most central tendency is the *Heteroscedastic* one.

Checking the convergence of β

Before finishing, it is important to analyse the convergence of the extra hyperparameter of the heteroscedastic and estimation of residuals models, β, which we optimize in a first learning phase and allow us to optimize the type of distribution to be fitted, as we explained in Sections 4.2.1 and 4.2.4.2. Table 4.2 shows the final convergence value of the β hyperparameter after the first training phase of all the methods in the different problems. As we can see, the convergence values of β for the problems where the experiment could be repeated converges to a stable value. There is only an exception with the case of the Regression Tree with the *LSTM-het* model where, given that the convergence value of β ended in a high value, the variance also ends up being high. To tackle this situations, a predefined value of β could be considered and directly optimize the other parameters of the deep learning wrapper.

Distribution of errors review

Finally, we want to visualize the impact on the probability of an erroneous prediction when the uncertainty score value increases by using the more stable black-box for our Financial forecasting problem: the *Second Order* version of the *Dense-het* model. From the business point of view, this information would be very useful to take the decision on which value of the uncertainty score we discard the predictions. We will now show the whole distribution of the errors sorted by the uncertainty score.

How to build the distribution plot The process to generate the Figure 4.5 is the following: First of all, we stratify into 30 bins the distribution of the absolute values of the error by using our *In Production* black-box for all the test-set, i.e. $|y_i - B(\mathbf{x}_i)|$. Then, we assign to each bin a certain color of a defined colormap

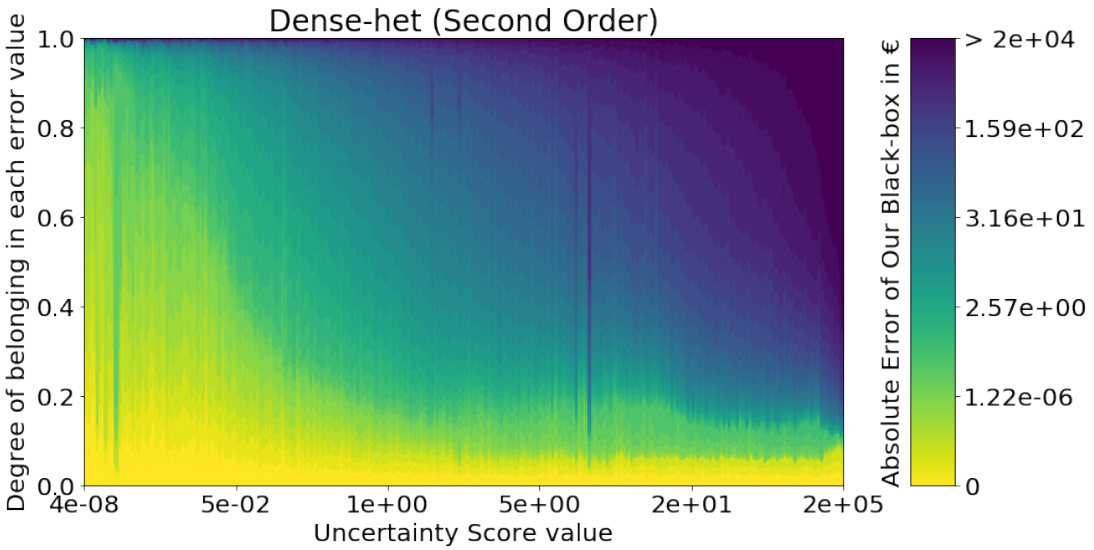


Figure 4.5: Density plot between the uncertainty score value and their corresponding Average Absolute Error produced by the Our *In Production* black-box when it is predicting in the Financial forecasting problem.

(as it can be seen in the right of the Figure 4.5). Afterwards, we order the absolute values of the errors of our black-box prediction by using their respective uncertainty score values. Thereafter, in order to reduce the dimensionality and make the behaviour smoother, we average in groups of 1024 points the previous sorted errors and their corresponding uncertainty score values. Finally, we draw vertically the degree of belonging in each of the error bins for each of the sorted groups.

Analysis of the distribution results Figure 4.5 exhibits the desired behaviour of a uncertainty score: the left part of the plot (low values of the uncertainty scores) concentrates the samples with lowest error (yellow-ish); the right part of the plot (high uncertainty values) contains a majority of samples with high and very high error (blue-ish).

These trade-off plots can also be used to inspect or debug the failure cases. For instance, we observe a yellow-ish band (bottom of the plot) with high uncertainty scores, which corresponds to samples with low error that the uncertainty model missed. Also, we observe some spikes, which may correspond to specific patterns

which do not follow the expected trend. Again, it would valuable to inspect these cases, but we recall the global error-retain trend of the plot remains as desired.

4.3 Intentional black-box uncertainty modelling

As we saw previously, there are a difference between the distributional fitting solutions presented until now and the residual error approximation: the former associates one of the parameters to be the black box, therefore, it has a meaning. The second one approximates a residual error, therefore, it models the distribution of the error which avoids directly know what is approximating the black box.

Although seemingly very straight-forward, conditional location-scale family distributions - like the heteroscedastic Normal or Laplace models presented in Section 3.2.1 - are directly applicable to the UMHB problem introduced at the initial part of this Chapter 4. This is because one of the conditional parameters can be considered as a certain statistic that it may be useful to predict, and the other gives us information about the confidence. The former can therefore be fixed as the black box and the latter learnt using the neural network model. In this section, we will extend the use of these models to tackle the UMHB problem and propose a new approach which transfers the distributional-free advantages of QR to the UMHB problem.

4.3.1 Heteroscedastic normal distribution

The deep heteroscedastic Normal model (N) As it was introduced in Section 3.2.1 and similarly to Bishop (1994b); Brando et al. (2019b); Kendall and Gal (2017); Lakshminarayanan et al. (2017); Tagasovska and Lopez-Paz (2019), by using two neural networks, $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$, it is possible to approximate the conditional normal distribution,

$$\mathcal{N}(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})) = \frac{1}{\sqrt{2\pi\sigma(\boldsymbol{x})^2}} e^{-\frac{(y-\mu(\boldsymbol{x}))^2}{2\sigma(\boldsymbol{x})^2}}, \tag{4.15}$$

such that they maximize the likelihood.

In the scenario of UMHB, $\mu(\boldsymbol{x}) = K(\boldsymbol{x})$ is the black-box function, which corresponds to the conditional mean statistic of the response variable, and we only need optimize the $\sigma(\boldsymbol{x})$ as a neural network model. Once optimized, the desired quantile τ can be obtained with $F(\tau, \boldsymbol{x}) = \mu(\boldsymbol{x}) + \sigma(\boldsymbol{x})\sqrt{2} \cdot \text{erf}^{-1}(2\tau - 1)$, $\tau \in (0, 1)$, where erf^{-1} is the inverse error function.

4.3.2 Heteroscedastic Laplace distribution

The deep heteroscedastic Laplace model (LP) Following the Reminder 5, as a more robust alternative to outlier values, a conditional Laplace distribution can be considered (Brando et al., 2018a),

$$LP\big(\mu(\boldsymbol{x}), b(\boldsymbol{x})\big) = \frac{1}{2b(\boldsymbol{x})} e^{-\frac{|y-\mu(\boldsymbol{x})|}{b(\boldsymbol{x})}}, \tag{4.16}$$

where the black-box function $\mu(\boldsymbol{x}) = K(\boldsymbol{x})$ corresponds to the conditional median of the response variable, and similarly to the previous case, we only need optimize $b(\boldsymbol{x})$ as a neural network. In contrast, here $F(\tau, \boldsymbol{x}) = \mu(\boldsymbol{x}) + \big(b \log(2\tau)\big) \cdot \mathbb{1}[\tau \leq \frac{1}{2}] - \big(b \log(2 - 2\tau)\big) \cdot \mathbb{1}[\tau > \frac{1}{2}]$, $\tau \in (0, 1)$.

4.3.3 The Chebyshev network

In this section we will tackle the uncertainty modelling of an Honest Black Box (denoted as the UMHB case). Until now, the only way to use the distribution-free approach based on quantile regression to tackle the covert MBU scenario was by approximating the residual error. However, this is not compatible with the UMHB statement, where the black box should be a pre-decided conditional statistic of the new conditional distribution model $q(Y \mid X, M)$. To transfer the distribution-free capabilities of quantile regression to the UMHB problem we propose the next model, which is based on the idea of modelling the partial derivative of quantile function with respect to the quantile value and linking the constant of integration, which can be expresses as a conditional statistic of the distribution, with the black-box predictor in the desired way.

Generically speaking, most of the techniques described in Chapters 2, 3 and

4 thus far either *(i)* estimate the conditional distribution, $p(Y \mid \boldsymbol{X}, M)$, – e.g the conditional normal or Laplace distributions (shown in Section 3.2.1) or quantile function models (based on the Definition 2) – or *(ii)* provide a point-wise predictive system to solve the regression problem – e.g. the models optimized using Eq. 2.7 or Eq. 2.8. None of them simultaneously tackle both estimations as two different outcomes. In critical real-world problems, however, the two pieces of information can have different but complementary meanings, that is, either *(i)* regarding the confidence of the prediction or *(ii)* regarding the output to report.

Let us imagine we have a point-wise predictive system, $K\colon \mathbb{R}^D \rightarrow \mathbb{R}$. This function could, for instance, be optimized using the mean squared error (shown in Eq. 2.7) and therefore approximate the conditional mean. Let us also suppose that this point-wise predictive system has some beneficial properties for the regression problem to be tackled, meaning that we are interested in preserving this conditional mean predictor. If we assume a considerable risk in the forecasting process, then we may be interested in modelling the prediction confidence of $p(Y \mid \boldsymbol{X}, M)$ apart from the conditional mean estimation $K(\boldsymbol{x})$. One solution to confidence modelling here could be to independently model $p(Y \mid \boldsymbol{X}, M)$, approximated as a known conditional distribution $q(Y \mid \boldsymbol{X}, M)$. However, the conditional mean of q will not necessarily correspond to $K(\boldsymbol{x})$, so we are obtaining a confidence that is not related to the original point-wise predictive system, $K(\boldsymbol{x})$. Therefore, instead we need to build a conditional distribution, $q(Y \mid \boldsymbol{X}, M)$, that approximates $p(Y \mid \boldsymbol{X}, M)$ such that its conditional mean corresponds to $K(\boldsymbol{x})$. This argument can be extended to any desired statistic, not only the mean, by considering the quantile function, as we will see hereafter.

Specifically, a quantile function, $\Phi\colon [0,1] \times \mathbb{R}^D \rightarrow \mathbb{R}$, which approximates $p(Y \mid \boldsymbol{X}, M)$, can be expressed according to any generic conditional statistic, $K\colon \mathbb{R}^D \rightarrow \mathbb{R}$, of that distribution. Broadly speaking, there exist a certain τ_0 or a set of τ_0s such that $K(\boldsymbol{x}) = \Phi(\tau_0, \boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^D$. For instance, the conditional median approximated in Eq. 2.8 corresponds to the quantile $\tau_0 = 0.5$, or the conditional mean estimated in Eq. 2.7 corresponds to the integral of all the quantiles, $K(\boldsymbol{x}) = \int_0^1 \Phi(t, \boldsymbol{x})dt$. Therefore, if we consider the partial derivative

of the quantile function with respect to the quantile value, i.e. $\partial\Phi(\tau, \boldsymbol{x})/\partial\tau =: \phi_w(\tau, \boldsymbol{x})$, we can conveniently build the constant of integration of ϕ_w with respect to τ as the certain statistic of $p\,(Y \mid \boldsymbol{X}, M)$ we had initially, named as $K(\boldsymbol{x})$, i.e. we can formulate $\Phi(\tau, \boldsymbol{x}) = K(\boldsymbol{x}) + K_w(\boldsymbol{x}) + \int_0^\tau \phi_w(t, \boldsymbol{x})\, dt$, where $K(\boldsymbol{x}) + K_w(\boldsymbol{x})$ is the constant of integration - that does not depend on τ - and, specifically, $K(\boldsymbol{x})$ is the desired conditional statistic of $p\,(Y \mid \boldsymbol{X}, M)$. Simultaneously, ϕ_w - that depends on τ and $\boldsymbol{x}$ - will help Φ to approximate $p\,(Y \mid \boldsymbol{X}, M)$ as a quantile function for all the quantiles τ.

Following this idea, we would like to estimate that derivative in that way obtaining the main definition that supports the proposed model of this section:

Definition 7 (Partial derivative of the conditional quantile function) *Let $\Phi_w \colon [0,1] \times \mathbb{R}^D \to \mathbb{R}$ be a quantile function, let $K \colon \mathbb{R}^D \to \mathbb{R}$ be a function, and let $\phi_w \colon [0,1] \times \mathbb{R}^D \to \mathbb{R}_+$ and $K_w \colon \mathbb{R}^D \to \mathbb{R}$ be two functions with parameters w. If we assume that*

$$\Phi_w(\tau, \boldsymbol{x}) = K(\boldsymbol{x}) + K_w(\boldsymbol{x}) + \int_0^\tau \phi_w(t, \boldsymbol{x})\, dt. \tag{4.17}$$

then the conditional quantile function can be approximated by minimizing the following loss function,

$$\begin{aligned}\mathcal{L}(\boldsymbol{X}, Y) = \mathbb{E}\Bigg[\int_0^1 \Bigg(Y - \Big(K(\boldsymbol{X}) + K_w(\boldsymbol{X}) + \int_0^\tau \phi_w(t, \boldsymbol{X})\, dt \Big) \Bigg) \\ \cdot \Bigg(\tau - \mathbb{1}\Big[Y < \Big(K(\boldsymbol{X}) + K_w(\boldsymbol{X}) + \int_0^\tau \phi_w(t, \boldsymbol{X})\, dt \Big) \Big] \Bigg)\, d\tau \Bigg],\end{aligned} \tag{4.18}$$

where $\mathbb{1}[c]$ denotes the indicator function that verifies the condition c.

Importantly, the assumption in Eq. 4.17 tells us that ϕ_w is a partial derivate of the quantile function and the term $K + K_w$ corresponds to the constant of integration. This term is divided as the sum of two terms (independently from any quantile); one depends on the parameters w in the optimization and the other, K, can be parametric or not, to allow it to be an honest black box, i.e. any

point-wise predictive system that approximates a desired conditional statistic of $p\left(Y \mid \boldsymbol{X}, M\right)$.

At first glance, it would seem that the previous formulation of the quantile function in Eq. 4.18 will only make the computation of Eq. 3.24 more difficult. However, the disentanglement in the integral will allow us to separately estimate a certain statistic of $p\left(Y \mid \boldsymbol{X}, M\right)$, using $K(\boldsymbol{x})$, which will be considered as a global bias used for the quantile function estimation of $p\left(Y \mid \boldsymbol{X}, M\right)$, mainly using the integral of $\phi_w(\tau, \boldsymbol{x})$. Thus, we are able to simultaneously but separately estimate (i) a point-wise prediction and (ii) its confidence.

Having this disentanglement property, particularly given the fact that we can express the constant of integration in terms of a summary statistic, will also be useful for estimating the confidence of any pre-trained point-wise predictive system that approximates a certain summary conditional statistic, $K(\boldsymbol{x})$.

Furthermore, given that we are estimating the partial derivative of the conditional quantile function, $\phi_w(\tau, \boldsymbol{x})$, we are able to impose restrictions on that derivative, and therefore propose a theoretical and empirical solution to the crossing quantile phenomena, which constitutes a limitation in most of the quantile regression models that predict several quantiles simultaneously. This phenomenon happens when a method cannot ensure that the quantile function will be monotonic with respect to the quantile value, i.e. given $\tau_1, \tau_2 \in [0, 1]$ such that $\tau_1 < \tau_2$, then $\Phi(\tau_1, \boldsymbol{x}) < \Phi(\tau_2, \boldsymbol{x})$ is not ensured, see, e.g. (Koenker and Hallock, 2001). However, as this is not the main aim of the present UMHB problem, these results will be presented in the next Chapter 5 based on the work Brando et al. (2022a).

In Section 4.3.3.1, the mechanism for estimating the derivative of a function using a neural network is briefly explained and extended to regress the partial derivative with respect to any desired inputs. Following that, in Section 4.3.3.2, the main contribution of this chapter and all its variants is presented, and then compared - theoretically and in terms of performance - with other baselines models.

4.3.3.1 Related works

Reminder 4 (Modelling the derivative of a function with a NN)
Aside from quantile function approximation, which requires a partial monotonic function, building generic monotonic functions poses an important problem in several areas (Archer and Wang, 1993; Daniels and Velikova, 2010; Gupta et al., 2016; Sill, 1998; Wang et al., 2020; You et al., 2017). We will start by describing how a monotonic function can be learnt by means of approximating the derivative of the final function.
Recently, a deep learning approach to building a monotonic function $H\colon \mathbb{R}^D \to \mathbb{R}$*, called the Unconstrained Monotonic Neural Network (UMNN), was proposed in Wehenkel and Louppe (2019). The UMNN estimates the derivative of that function as*

$$H(z) = H(0) + \int_0^z h(t)\, dt, \tag{4.19}$$

where the integral in Eq. (4.19) is approximated using the Clenshaw-Curtis quadrature (Clenshaw and Curtis, 1960).
Furthermore, this derivative can be approximated by means of a neural network $\hat{h}\colon \mathbb{R}^D \to \mathbb{R}_+$*, whose output is restricted to strictly positive values. Therefore, when* $h(z) :- \frac{\partial H}{\partial z}(z) > 0$*, the* $H(z)$ *behaves as a monotone function with respect to* z*, if* $\hat{h}(z) = h(z)$*.*
In the following section, we will see that a partial derivative is required to learn a quantile function such as the one presented in Eq. (3.24).

The main contribution of this Section 4.3 is the Chebyshev Network (CheNet), which builds a conditional quantile function $\Phi\colon [0,1] \times \mathbb{R}^D \to \mathbb{R}$ and constitutes a solution for the disentanglement between the confidence of the prediction, ϕ_w, and a point-wise predictive system, K (as shown in Definition 7), which, jointly with $K_w(\boldsymbol{x})$, will define the constant of integration.

Internally, CheNet uses a positive neural network $\phi_w\colon [0,1] \times \mathbb{R}^D \to \mathbb{R}$ with parameters w approximating the derivative of $\Phi(\tau; \boldsymbol{x})$ such that

$$\Phi(\tau; \boldsymbol{x}) = K(\boldsymbol{x}) + Q(\tau; \boldsymbol{x}), \qquad Q(\tau; \boldsymbol{x}) = K_w(\boldsymbol{x}) + \int_0^\tau \phi_w(t, \boldsymbol{x})\, dt, \tag{4.20}$$

where $K(\boldsymbol{x})$ will be in charge of modelling a certain conditional statistic of $p(Y \mid X)$ and $Q(\tau; \boldsymbol{x})$ will approximate the rest of the $p(Y \mid X)$ distribution as a new $q(Y \mid X)$, thus verifying that the desired conditional statistic corresponds to $K(\boldsymbol{x})$ using the $K_w(\boldsymbol{x})$ as an offset.

Furthermore, Eq. (4.20) is optimized by the CQR loss function in Eq. (3.25). As we will see below, CheNet takes advantage of considering the full range of possible quantiles, i.e. $[0, 1]$, in contrast with the UPMNN, which considers $[0, \tau]$. This is the main reason why CheNet is able to uniformly select a constant of integration function, $K(\boldsymbol{x})$, on $[0, 1]$, which corresponds to a certain desired conditional statistic of the distribution produced by the quantile function, $q(Y \mid X)$.

4.3.3.2 Model definition

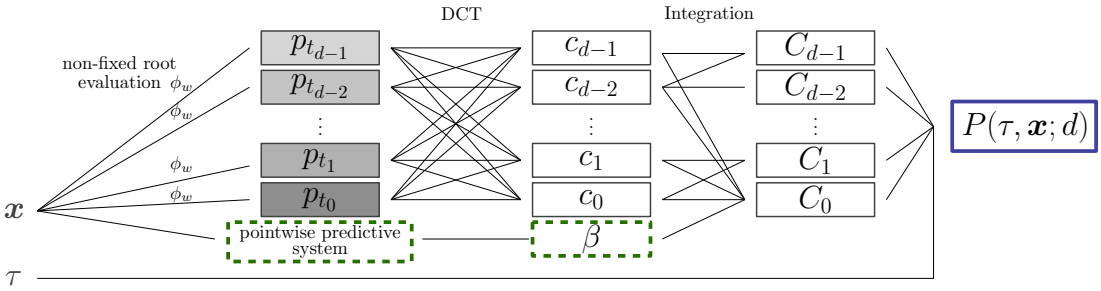


Figure 4.6: Graphic representation of CheNet. For any degree d, $\{p_{t_i}\}_{i=0}^{d-1}$ constitute the evaluation of the initial Chebyshev polynomial expansion, $\{c_k\}_{k=0}^{d-1}$ their coefficients, $\{C_k\}_{k=0}^{d-1}$ the coefficients of the integrated polynomial, K the constant of integration (or the black-box function) and P the conditional prediction of the quantile τ.

CheNet contains a neural network $\phi_w \colon [0, 1] \times \mathbb{R}^D \to \mathbb{R}_+$ that only produces positive outputs and models the derivative of the final function with respect to a given quantile τ. CheNet optimizes the neural network $\phi_w(\tau, \boldsymbol{x})$ by calculating the coefficients of a truncated Chebyshev polynomial expansion $p(\tau, \boldsymbol{x}; d)$ of degree d with respect to τ. Due to the truncation, we consider a finite mesh of quantile values, called *Chebyshev roots*, $\{t_k\}_{k=0}^{d-1} \subset [0, 1]$ which are defined by

$$t_k = \frac{1}{2} \cos\left(\frac{\pi(k + \frac{1}{2})}{d} \right) + \frac{1}{2}, \quad 0 \leq k < d. \tag{4.21}$$

Given that these Chebyshev roots only depend on the degree d and not on any quantile, CheNet exhibits global properties in the entire interval $[0, 1]$ regardless the desired quantile to predict.

Generically, the truncated Chebyshev expansion consists in expressing a function as a linear combination of Chebyshev polynomials. These are defined as mappings $T_k \colon [-1, 1] \to \mathbb{R}$ given by the recurrent formula

$$\begin{aligned} T_0(t) &:\!- 1, \\ T_1(t) &:\!- t, \\ T_{k+1}(t) &:\!- 2tT_k(t) - T_{k-1}(t), \qquad k \geq 1. \end{aligned} \tag{4.22}$$

In our case, τ is in $[0, 1]$ rather than in $[-1, 1]$ which slightly modifies the definition,

$$\phi_w(\tau, \boldsymbol{x}) \approx p(\tau, \boldsymbol{x}; d) :\!- \frac{1}{2}c_0(\boldsymbol{x}) + \sum_{k=1}^{d-1} c_k(\boldsymbol{x}) T_k(2\tau - 1). \tag{4.23}$$

The quantities $c_j(\boldsymbol{x})$ in Eq. (4.22) are independent of the quantiles[1]. These quantities are computed by

$$c_j(\boldsymbol{x}) :\!- \frac{2}{d} \sum_{k=0}^{d-1} \phi_w(t_k, \boldsymbol{x}) \cos\left(\frac{j\pi(k + \frac{1}{2})}{d}\right), \qquad 0 \leq j < d, \tag{4.24}$$

which is merely a product of the matrix-vector. It admits the computation algorithm known as the Discrete Cosinus Transform of type 2 (DCT-II), which reduces the matrix-vector multiplication from $\Theta(d^2)$ to a logarithmic complexity, $\Theta(d \log d)$.

The polynomials T_k in Eq. (4.23) do not need to be explicitly computed and, by construction of the coefficients $c_k(\boldsymbol{x})$ in Eq. (4.24), $p(t_k, \boldsymbol{x}; d)$ is "equals to" $\phi_w(t_k, \boldsymbol{x})$ for all Chebyshev roots, $\{t_k\}_{k=0}^{d-1}$, as previously defined in Eq. (4.21). This equality must, in practice, be understood in terms of machine precision of the numerical representation system, classically $\sim 10^{-16}$ in double-precision or $\sim 10^{-8}$ in single-precision arithmetic. That exactness is important to ensure monotonicity, see Brando et al. (2022a). This root evaluation step is denoted as

[1] In contrast to the UPMNN, as shown in Eq. (4.21).

p_{t_k} in Figure 4.6.

Once $p(t, \boldsymbol{x}; d)$ has been encoded by its $c_j(\boldsymbol{x})$ coefficients in Eq. (4.24), if we use these to compute $P(\tau, \boldsymbol{x}; d)$, which represents $\int_0^\tau p(t, \boldsymbol{x}; d)\, dt$, then P we will be an approximation of the integral of the neural network, ϕ_w. That is,

$$P(\tau, \boldsymbol{x}; d) \approx \Phi(\tau, \boldsymbol{x}) = K(\boldsymbol{x}) + Q(\tau; \boldsymbol{x}). \tag{4.25}$$

Additionally, given that the neural network $\phi_w(t, \boldsymbol{x})$ gives only positive values for all $t \in [0, 1]$, then $P(\tau, \boldsymbol{x}; d)$ would be an increasing function with respect to τ as long as d is large enough.

In the above procedure, the integral of ϕ_w in Eq. (4.20) is, in general, not straightforward to compute. However, the neural network $\phi_w(\cdot, \boldsymbol{x})$ is globally being represented by $p(\cdot, \boldsymbol{x}; d)$ on the quantile interval $[0, 1]$. Therefore, by using its Chebyshev coefficients $c_k(\boldsymbol{x})$, we can encode the integral of P to provide its Chebyshev coefficients $C_k(\boldsymbol{x})$. In fact, according to Clenshaw (1955), the integral of $p(\tau, \boldsymbol{x}; d)$ gives another Chebyshev expansion, say $P(\tau, \boldsymbol{x}; d)$, with coefficients $C_k(\boldsymbol{x})$, i.e.,

$$P(\tau, \boldsymbol{x}; d) = \frac{1}{2} C_0(\boldsymbol{x}) + \sum_{k=1}^{d-1} C_k(\boldsymbol{x}) T_k(2\tau - 1). \tag{4.26}$$

To deduce the expressions for $C_k(\boldsymbol{x})$ in Eq. (4.26), we need to recurrently integrate the polynomials $T_k(t)$, whose integral are

$$\begin{aligned}
&\int T_0(t)\, dt = T_1(t) + \text{constant}, \\
&\int T_1(t)\, dt = \frac{T_2(t) + T_0(t)}{4} + \text{constant}, \\
&\int T_k(t)\, dt = \frac{T_{k-1}(t)}{2(k-1)} - \frac{T_{k+1}(t)}{2(k+1)} + \text{constant}, \qquad k \geq 2.
\end{aligned} \tag{4.27}$$

By ordering the coefficients of the integral in Eq. (4.23), we deduce that

$$C_k(\boldsymbol{x}) :- \frac{c_{k-1}(\boldsymbol{x}) - c_{k+1}(\boldsymbol{x})}{4k}, \quad 0 < k < d-1, \qquad C_{d-1}(\boldsymbol{x}) \;\; :- \frac{c_{d-2}(\boldsymbol{x})}{4(d-1)}, \tag{4.28}$$

and $C_0(\boldsymbol{x})$ depends on the constant of integration $K(\boldsymbol{x})$ in Eq. (4.20) and the

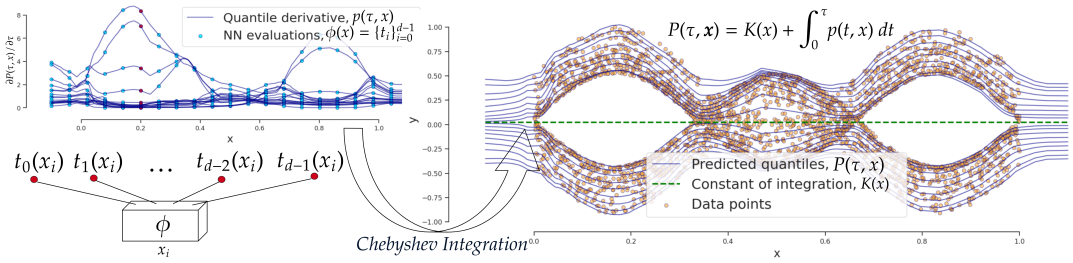


Figure 4.7: In general terms, CheNet uses a NN - ϕ at the bottom left - to generate d **positive values** –the red $\{t_j(\boldsymbol{x})\}_{j=0}^{d-1}$ points–, which will be used as roots for computing the coefficients of a Chebyshev polynomial, represented in the top left subfigure. This polynomial will approximate the partial derivative of the quantile function. To do that, we integrate it obtaining a new Chebyshev polynomial, the one in the right subfigure. Thus, for each $\boldsymbol{x}$ we have a Chebyshev polynomial modelling all the quantiles.

other coefficient values in Eq. (4.26). This freedom on the $C_0(\boldsymbol{x})$ value allows CheNet to impose different conditions to obtain $C_0(\boldsymbol{x})$ for the full interval $[0, 1]$.

As we will soon see in Section 4.3.3.4, this important property will give us the possibility to select which kind of point-wise predictive system, K, we want (following Def 7) and this can be exploited as a "constraint selection" for honest black boxes, which implies that CheNet constitutes a distributional-free solution for the UMHB problem (following Def 6).

On the whole, CheNet can be summarized as in Figures 4.7 and 4.6 - considering K as a point-wise predictive system - consisting of the following steps: for each $\boldsymbol{x}$ value,

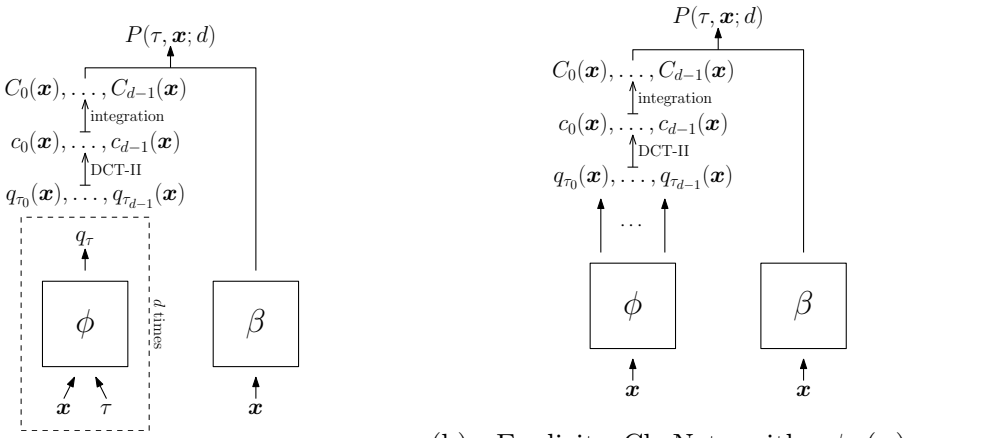
1. A set of values, $\{p_{t_k}(\boldsymbol{x})\}_{k=0}^{d-1}$ is obtained via the neural network ϕ_w at the corresponding roots, $\{t_k\}_{k=0}^{d-1}$.

2. These are transformed to d coefficients, $\{c_k(\boldsymbol{x})\}_{k=0}^{d-1}$ using a fast matrix-vector multiplication algorithm. This results in a truncated Chebyshev polynomial, $p(\tau, \boldsymbol{x}; d)$.

3. This polynomial is then integrated to obtain another Chebyshev polynomial, $P(\tau, \boldsymbol{x}; d)$, whose coefficients are $\{C_k(\boldsymbol{x})\}_{k=0}^{d-1}$ in Eq. (4.28).

Mostly, CheNet contains a Chebyshev polynomial for each $\boldsymbol{x}$ value that approximates all the conditional quantiles of $p(Y \mid X)$.

4.3.3.3 Implicit and explicit CheNet



(a) Implicit CheNet with $\phi_w(\tau, \boldsymbol{x}) = q_\tau$.

(b) Explicit CheNet with $\phi_w(\boldsymbol{x}) = \{q_{\tau_t}\}_{t=0}^{d-1}$.

Figure 4.8: Graphical comparison of the two versions of CheNet.

CheNet admits two different ways of managing the order of the Chebyshev expansion in its internal neural network ϕ_w. Both are illustrated in Figure 4.8. Implicit CheNet consists in evaluating ϕ_w as many times as the number of roots, and Explicit CheNet provides d outputs in evaluating ϕ_w.

Definition 8 (Implicit CheNet) *An implicit CheNet optimizes a neural network of the form* $\phi_w \colon [0,1] \times \mathbb{R}^D \to \mathbb{R}_+$ *constructing the Chebyshev polynomial of an arbitrary order* d *in Eq.* (4.26).

Definition 9 (Explicit CheNet) *Let* d *be a non-negative integer. An explicit CheNet optimizes a neural network of the form* $\phi_w \colon \mathbb{R}^D \to \mathbb{R}_+^d$ *constructing the Chebyshev polynomial of order* d *in Eq.* (4.26).

Both forms of CheNet can produce the same final Chebyshev polynomial $P(\tau, \boldsymbol{x}; d)$ at a given iteration and, importantly, such a polynomial can be evaluated at any desired τ.

The two forms are also suitable for different purposes. Put briefly, Implicit CheNet will allow us to choose the precision required in terms of having all the quantiles sorted after the training process. For instance, we will be able to address the crossing quantile phenomenon or, equivalently, ensure an always monotonic ϕ-function, as described in Brando et al. (2022a).

On the other hand, Explicit CheNet simplifies the neural network's learning process by only internally predicting a fixed number of roots regarding the derivative. This simplification helps to improve the overall performance in terms of quantile regression accuracy.

The main aim of this Chapter, which is to resolve the UMHB problem, can be tackled using both versions of CheNet. A more in-depth discussion will be presented in the following sections, which address selection of the appropriate constant of integration and the application of CheNet as a solution for the UMHB problem (see Definition 6).

4.3.3.4 Constant of integration selection for CheNet

In this section, we address how to define CheNet as a derivative of the quantile function (as shown in Eq. (4.18)), considering any desired statistic (such as a quantile or the conditional mean of $p(Y \mid X)$) using its constant[1] of integration. Following Definition 7, all that is needed to optimize the parameters w in the CheNet's Q (shown in Eq. (4.20)) are the corresponding K-evaluation values of the training set along with the input and response values, i.e. $\{(\boldsymbol{x}_i, K(\boldsymbol{x}_i)), y_i\}_{i=1}^{N}$. Therefore, K does not need to be a parametric function and the generic goal is reduced to approximate the conditioned response distribution to the input, $p(Y \mid X)$ by predicting all their conditional quantiles. This avoids asymmetry or unimodality assumptions with respect to $p(Y \mid X)$.

It is important to highlight that this constant of integration selection can be done thanks to the non-quantile dependence of the CheNet's roots (as it is shown in Eq. (4.21)), which allows us to use CheNet as a solution for the UMHB problem.

[1] The term "constant" refers to the quantile τ input and not to $\boldsymbol{x}$, as described in Eq. (4.20).

The formula used to calculate the constant of integration $C_0(\boldsymbol{x})$ (shown in Eq. (4.28)) will depend on which statistic we choose for $K(\boldsymbol{x})$, as shown in the following propositions.

Proposition 10 (CheNet-q_0) *Let (P, K) be a CheNet model and $q(Y \mid X)$ the distribution that P produces as quantiles. If K is a point-wise predictive system that is the **lowest quantile** ($\tau = 0$) of $q(Y \mid X)$, then the C_0 coefficient of the Chebyshev polynomial P verifies:*

$$C_0(\boldsymbol{x}) = 2K(\boldsymbol{x}) - 2\sum_{k=1}^{d-1} C_k(\boldsymbol{x})(-1)^k. \tag{4.29}$$

We will denote this as CheNet-q_0.

Proof The learning process of CheNet is subjected to the condition that the lowest quantile value, i.e. $\tau = 0$, must be the K. That is, $P(0, \boldsymbol{x}; d) = K(\boldsymbol{x})$. By taking the value $t = -1$ in Eq. (4.22), we derive that $T_k(-1) = (-1)^k$. Therefore, using Eq. (4.26) with $\tau = 0$, we obtain Eq. (4.29). ■

Proposition 11 (CheNet-q_1) *Let (P, K) be a CheNet model and $q(Y \mid X)$ the distribution that P produces as quantiles. If K is a point-wise predictive system that is the **highest quantile** ($\tau = 1$) of $q(Y \mid X)$, then the C_0 coefficient of the Chebyshev polynomial P verifies:*

$$C_0(\boldsymbol{x}) = 2K(\boldsymbol{x}) - 2\sum_{k=1}^{d-1} C_k(\boldsymbol{x}). \tag{4.30}$$

We will denote this as CheNet-q_1.

Proof The highest quantile corresponds to $\tau = 1$. Taking into account the definition of the Chebyshev polynomials, $T_k(1) = 1$. Therefore, imposing the condition $P(1, \boldsymbol{x}; d) = K(\boldsymbol{x})$, we deduce Eq. (4.30). ■

The CheNets of type q_0 and q_1 from the Propositions 10 and 11 can be applied to the prediction of extreme weather events, which entail the forecasting system predicting the maximum or the minimum values of $p(Y \mid X)$. In these cases, a pre-trained system that forecasts the maximum or minimum temperature value could be used as $K(\boldsymbol{x})$ in Eq. (4.29) or Eq. (4.30), respectively, to determine the overall quantile distribution of $p(Y \mid X)$, taking $K(\boldsymbol{x})$ as a reference point (i.e. preserving the constraint that the maximum or minimum of the predicted quantiles corresponds to $K(\boldsymbol{x})$).

Proposition 12 (CheNet-Median) *Let (P, K) be a CheNet model and $q(Y \mid X)$ the distribution that P produces as quantiles. If K is a point-wise predictive system that is the **median** ($\tau = 0.5$) of $q(Y \mid X)$, then the C_0 coefficient of the Chebyshev polynomial P verifies:*

$$C_0(\boldsymbol{x}) = 2K(\boldsymbol{x}) - 2 \sum_{\substack{k=1 \\ k \text{ even}}}^{d-1} (-1)^{k/2} C_k(\boldsymbol{x}). \tag{4.31}$$

We will denote this as CheNet-Med.

Proof The median corresponds to the quantile value of $\tau = 0.5$. Taking into account the definition of the Chebyshev polynomials, $T_k(0) = 0$ when k is odd and $T_k(0) = (-1)^{k/2}$ when k is even. Therefore, imposing the condition $P(0.5, \boldsymbol{x}; d) = K(\boldsymbol{x})$, we deduce Eq. (4.31). ■

Proposition 13 (CheNet-Mean) *Let (P, K) be a CheNet model and $q(Y \mid X)$ the distribution that P produces as quantiles. If K is a point-wise predictive system that is the **mean** of $q(Y \mid X)$, then the C_0 coefficient of the Chebyshev polynomial P verifies:*

$$C_0(\boldsymbol{x}) = 2K(\boldsymbol{x}) - 2 \sum_{\substack{k=1 \\ k \text{ odd}}}^{d-1} \frac{C_k(\boldsymbol{x})}{k^2 - 4}. \tag{4.32}$$

We will denote this as CheNet-Mean.

Proof Taking into account the definition of the (continous) mean, the condition we must impose is

$$\int_0^1 \tau P(\tau, \boldsymbol{x}; d)\, d\tau = K(\boldsymbol{x}). \tag{4.33}$$

Then, taking into account the linearity of the integral and after a change of coordinates ($t = 2\tau - 1$), the integral is reduced to compute the mean of the Chebyshev polynomials. That is,

$$\int_{-1}^1 tT_k(t)\, dt. \tag{4.34}$$

Taking into account the symmetries of the Chebyshev polynomials, we obtain the equality

$$\int_{-1}^1 tT_k(t)\, dt = (1 + (-1)^{k+1}) \int_0^1 tT_k(t)\, dt.$$

Then if k is even, Eq. (4.34) is zero. If k is now assumed to be an odd integer, then taking into account the recurrent definition of the Chebyshev polynomials, $2T_k(t) = T_{k+1}(t) + T_{k-1}(t)$ and then Eq. (4.27),

$$\int_{-1}^1 tT_k(t)\, dt = \frac{T_{k-2}(t)}{2(k-2)} - \frac{T_{k+2}(t)}{2(k+2)}\bigg|_0^1 = \frac{2}{k^2-4}.$$

From here, we can recover the expression in Eq. (4.32). ■

Additionally, $K(\boldsymbol{x})$ could be approximated by means of another neural network - depending on a new set of weights v - when we are optimizing w and optimize both at same time. In this latter case, the function learnt by this new $K_v(\boldsymbol{x})$ will approximate the choice of conditional summary statistic with respect to $p(Y \mid X)$, depending on the previously selected formula.

The pseudo-code implementation of CheNet

The pseudo-code for CheNet (including all of its variations) is shown in Algorithm 7 and can be implemented using any automatic differentiation library. We provide a TensorFlow (Abadi et al. (2016)) and Keras (Chollet et al. (2019))

implementation in a Github repository, which will be made public in the camera-ready version of this paper.

Algorithm 4 Evaluation of Eq. (4.23) or Eq. (4.26) at $\tau \in [0, 1]$

procedure EVAL_CHEB(τ, $c_0(\boldsymbol{x}), \ldots, c_{d-1}(\boldsymbol{x})$)
  $d_1(\boldsymbol{x}) \leftarrow d_2(\boldsymbol{x}) \leftarrow d_3(\boldsymbol{x}) \leftarrow 0$.
  $\sigma \leftarrow 2\tau - 1$.
  for $k = d-1, d-2, \ldots, 1$ **do**
    $d_3(\boldsymbol{x}) \leftarrow d_1(\boldsymbol{x})$.
    $d_1(\boldsymbol{x}) \leftarrow 2\sigma d_1(\boldsymbol{x}) - d_2(\boldsymbol{x}) + c_k(\boldsymbol{x})$.
    $d_2(\boldsymbol{x}) \leftarrow d_3(\boldsymbol{x})$.
  return $\sigma d_1(\boldsymbol{x}) - d_2(\boldsymbol{x}) + 0.5 c_0(\boldsymbol{x})$.

Algorithm 5 Definitions and functions used for the following algorithms

▷ $\boldsymbol{x}$ has batch size and number of features as shape, i.e. $[bs, D]$.
▷ RS(*tensor*, *shape*): reshape *tensor* to *shape*.
▷ RP($tensor, n$): repeats n times the last dimension of *tensor*.
▷ CC(T_1, T_2): concatenate T_1 and T_2 using their last dimension.

Algorithm 6 Obtaining all CheNet coefficients

procedure CHEB_CS($\boldsymbol{x}, d, \phi, K$)
  $\{t_k\}_{k=0}^{d-1} \leftarrow$ Apply Eq. 4.21
  $\{t'_k\}_{k=0}^{d-1} \leftarrow \texttt{RS}\big(\texttt{RP}\big(\{t_k\}_{k=0}^{d-1}, bs\big),\ [bs \cdot d, 1]\big)$
  $\boldsymbol{x}' \leftarrow \texttt{RS}(\texttt{RP}(\boldsymbol{x}, d),\ [bs \cdot d, D])$
  $\boldsymbol{i} \leftarrow \texttt{CC}\big(\{t'_k\}_{k=0}^{d-1}, \boldsymbol{x}'\big)$ ▷ $[bs \cdot d, D+1]$.
  $o \leftarrow \phi(\boldsymbol{i})$ ▷ Apply any NN function $\phi\colon [0,1] \times \mathbb{R}^D \to \mathbb{R}^+$ or $\phi\colon \mathbb{R}^D \to (\mathbb{R}^+)^d$.
  $\{c_k(\boldsymbol{x})\}_{k=0}^{d-1} \leftarrow$ DCT-II(o, d) ▷ Transformation.
  $\{C_k(\boldsymbol{x})\}_{k=1}^{d-1} \leftarrow$ Integration step wrt $\{c_k(\boldsymbol{x})\}_{k=0}^{d-1}$ ▷ Eq. (4.28)
  $C_0(\boldsymbol{x}) \leftarrow 2K(\boldsymbol{x}) - 2\sum_{k=1}^{d-1} C_k(\boldsymbol{x})(-1)^k$ ▷ Eq. (4.29)
  return $\{C_k(\boldsymbol{x})\}_{k=0}^{d-1}, \{c_k(\boldsymbol{x})\}_{k=0}^{d-1}$.

4.3.4 Results and comparison

The different data sets and architecture hyper-parameters used to obtain the following experimental results are described in greater detail in Brando et al.

Algorithm 7 How to build the CheNet model by using any deep learning architecture for regression

procedure BUILD_CHENET_GRAPH($\boldsymbol{x}, y, d, \phi, K, N_\tau$)
  ▷ N_τ is the number of non-roots to evaluate.
  $\{C_k(\boldsymbol{x})\}_{k=0}^{d-1}, \{c_k(\boldsymbol{x})\}_{k=0}^{d-1} \leftarrow$ `CHEB_CS`$(\boldsymbol{x}, d, \phi, K)$
  $\boldsymbol{\tau} \leftarrow \mathcal{U}(0,1)$ ▷ $\boldsymbol{\tau}$ must has $[bs \cdot N_\tau, 1]$ shape.
  $\boldsymbol{o}_P \leftarrow$ `EVAL_CHEB`$(\boldsymbol{\tau}, C_0(\boldsymbol{x}), \ldots, C_{d-1}(\boldsymbol{x}))$
  $\mathcal{L} \leftarrow (y - \boldsymbol{o}_P) \cdot (\boldsymbol{\tau} - \mathbb{1}[y < \boldsymbol{o}_P])$ ▷ Eq. (3.25) loss.
  return $\mathcal{L}$

Algorithm 8 How to obtain the CheNet's prediction for a set of desired quantiles

procedure PREDICT($\boldsymbol{x}, d, \phi, K, \boldsymbol{\tau}$)
  ▷ $\boldsymbol{\tau}$ - with $[bs \cdot N_\tau, 1]$ shape - is the desired quantiles to evaluate.
  $\{C_k(\boldsymbol{x})\}_{k=0}^{d-1}, \{c_k(\boldsymbol{x})\}_{k=0}^{d-1} \leftarrow$ `CHEB_CS`$(\boldsymbol{x}, d, \phi, K)$
  $\boldsymbol{o}_P \leftarrow$ `EVAL_CHEB`$(\boldsymbol{\tau}, C_0(\boldsymbol{x}), \ldots, C_{d-1}(\boldsymbol{x}))$
  return $\boldsymbol{o}_P$

(2022b).

Testing generic quantile calibration Table 4.3 shows a comparison of the quantile forecasting prediction for two given black-box systems - a Random Forest (RF) (Liaw et al., 2002) and an XGBoost (Chen and Guestrin, 2016) - in four data sets. The first four columns correspond to each part of the heterogeneous synthetic distribution proposed in Brando et al. (2019b) and shown in Figure 4.9, the fifth column is the full Year Prediction MSD UCI dataset (Dua and Graff, 2017), predicting the release year of a song from 90 audio features and, finally, the last two columns correspond to predicting the room price forecasting of Airbnb flats (RPF) in Barcelona and Vancouver, extracted from Brando et al. (2019b). The mean of the QR loss value (shown in Eq. 4.4) is evaluated for ten thousand randomly selected quantiles for ten executions of each model $\{m_k\}_{k=1}^{10}$ as

$$\mathcal{L}_{m_k}(X_{test}, Y_{test}) = \sum_{i=1}^{N_{test}} \sum_{j=1}^{N_\tau} \frac{\big(y_i - f_{m_k}(\tau_j, \boldsymbol{x}_i)\big) \cdot \big(\tau_j - \mathbb{1}[y_i < f_{m_k}(\tau_j, \boldsymbol{x}_i)]\big)}{N_{test} \cdot N_\tau}, \quad (4.35)$$

where the N_{test} is the number of points in the test set, the $N_\tau = 10,000$ is the number of Monte Carlo samplings and the f_{m_k} is any of the models considered in

Table 4.3: Mean and standard deviation of the QR loss value, mean ± std, of 10 executions for each Black box-*Uncertainty wrapper* using all of the test distributions in Figure 4.9 and three data sets (described in Brando et al. (2022b)). The ranges that overlap with the best range are highlighted in bold.

	Asymmetric	Symmetric	Uniform	Multimodal	Year-MSD	BCN-RPF	YVC-RPF
RF-N	42.37 ± 0.04	23.19 ± 1.00	66.44 ± 0.26	151.51 ± 0.24	57.50 ± .05	23.47 ± .14	**27.27 ± .39**
RF-LP	42.88 ± 0.04	**22.10 ± 0.03**	67.13 ± 0.09	153.06 ± 0.22	57.58 ± .02	**23.07 ± .17**	28.06 ± .12
RF-(I)CheNet	**41.52 ± 0.35**	23.19 ± 0.70	**65.98 ± 0.20**	148.39 ± 0.16	**48.28 ± .18**	**23.17 ± .07**	28.16 ± .14
RF-(E)CheNet	**41.23 ± 0.07**	**22.39 ± 0.46**	**66.23 ± 0.13**	**147.45 ± 0.31**	**48.25 ± .00**	**23.18 ± .10**	28.26 ± .17
XGBoost-N	42.42 ± 0.05	**23.35 ± 0.99**	66.38 ± 0.26	149.35 ± 0.40	51.17 ± .08	24.52 ± .26	27.79 ± .08
XGBoost-LP	42.90 ± 0.02	**23.02 ± 0.43**	67.13 ± 0.17	150.94 ± 0.12	51.24 ± .02	22.63 ± .11	27.86 ± .07
XGB.-(I)CheNet	**41.95 ± 0.40**	**23.69 ± 0.68**	**65.89 ± 0.17**	146.20 ± 0.30	**48.54 ± .08**	**22.00 ± .04**	**27.51 ± .13**
XGB.-(E)CheNet	**41.80 ± 0.33**	**23.68 ± 1.20**	66.57 ± 0.21	**143.62 ± 0.19**	**48.45 ± .00**	**22.01 ± .04**	**27.49 ± .07**
N	43.63 ± 2.89	**23.70 ± 6.85**	**67.45 ± 1.68**	148.78 ± 2.88	49.00 ± .24	27.28 ± 1.25	28.62 ± 1.61
LP	43.46 ± 0.15	**20.72 ± 0.47**	**68.06 ± 0.82**	149.99 ± 0.64	48.67 ± .28	23.51 ± .28	22.32 ± .06
(I)CheNet	**41.72 ± 0.24**	**22.94 ± 1.81**	**68.55 ± 6.61**	**145.93 ± 3.14**	**46.76 ± .25**	**20.67 ± .40**	**21.97 ± .12**
(E)CheNet	**42.02 ± 0.39**	**22.57 ± 0.69**	**66.52 ± 0.63**	**141.21 ± 0.51**	**46.72 ± .12**	**20.98 ± .71**	**21.90 ± .05**

Table 4.3. Importantly, the QR loss value not only reports about each system's performance but also how generically calibrated its predicted quantiles are, which does not mean that the number of crossing quantiles is necessarily less, see Brando et al. (2022a).

Furthermore, in Table 4.3 we observe that CheNet outperforms other methods in most cases due to it transferring the capacity to capture asymmetries and multimodalities of QR in $p(Y \mid X)$ to the UMHB problem, where our uncertainty modelling needs to be restricted in order to maintain the corresponding statistic associated with the black box.

What happens when the black box is inaccurate This restriction of conserving the black box can be seen qualitatively in the upper part of Figure 4.9, where it must always be the case, i.e. even if the performance worsens because the black box, $K(\boldsymbol{x})$, is not correctly fitted (as described in the initial part of the current Chapter 4). In this case, $K(\boldsymbol{x})$ is an inaccurate Random Forest predicting the mean. Importantly, the CheNet propagates the $K(\boldsymbol{x})$ noise to the predicted quantiles (in blue) because the constraint is always forced, therefore, the predicted quantiles must preserve the noisy conditional mean function (in green).

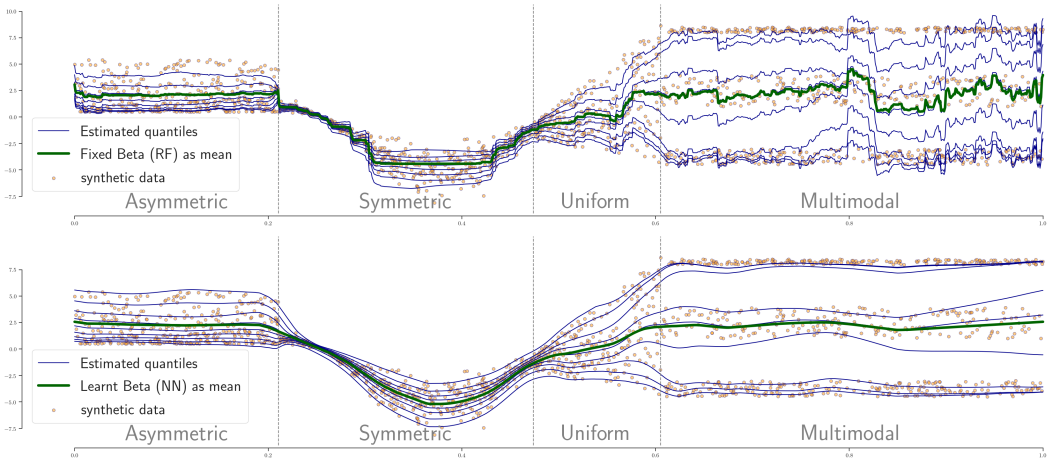


Figure 4.9: Heterogeneous synthetic distribution proposed by Brando et al. (2019b). In the upper part of the figure, the learnt quantiles, ϕ, are noisy because their mean is the black box defined as an inaccurate MSE Random Forest (RF), K, following Eq. (4.32). In the lower part, ϕ and K are learnt and asymmetries and multimodalities can be seen more clearly, while still respecting the constraint in Eq. (4.32).

Modelling heterogeneous distributions On the other hand, the ability of CheNet to model heterogeneous distributions using QR is better displayed in the lower part of Figure 4.9. In this case, the black box is a neural network that is learnt concurrently with the quantiles. Since the black box is better approximated, the quantiles are better learnt.

	$\boxed{XGBoost}$-N	$\boxed{XGBoost}$-LP	$\boxed{XGBoost}$-(I)CheNet	$\boxed{XGBoost}$-(E)CheNet
Concrete	46.55 ± 6.7(3.34 ± 0.4)	54.66 ± 5.5(4.38 ± 0.3)	$\mathbf{85.58 \pm 4.7}$($17.7 \pm 21.$)	$\mathbf{92.77 \pm 2.4}$(17.6 ± 2.2)
Power	79.44 ± 3.5(6.99 ± 0.4)	88.23 ± 3.7(9.19 ± 1.2)	$\mathbf{92.93 \pm 2.8}$(12.0 ± 1.6)	$\mathbf{95.36 \pm 2.1}$(13.1 ± 2.0)
Wine	$\mathbf{95.78 \pm 1.8}$(2.66 ± 0.2)	$\mathbf{96.78 \pm 1.8}$(2.77 ± 0.2)	$\mathbf{95.81 \pm 1.9}$(3.86 ± 0.6)	$\mathbf{96.25 \pm 2.7}$(3.60 ± 0.8)
Yacht	$\mathbf{93.22 \pm 5.1}$(2.42 ± 0.4)	$\mathbf{94.03 \pm 4.6}$(2.45 ± 0.4)	$\mathbf{99.19 \pm 1.7}$(8.16 ± 1.3)	$\mathbf{97.58 \pm 3.0}$(4.50 ± 1.3)
Naval	$\mathbf{99.52 \pm 1.3}$(0.05 ± 0.0)	$\mathbf{99.68 \pm 1.3}$(0.08 ± 0.0)	$\mathbf{96.95 \pm 4.2}$(0.11 ± 0.0)	$\mathbf{99.99 \pm 0.0}$(0.04 ± 0.0)
Energy	$\mathbf{95.45 \pm 4.4}$(2.16 ± 0.5)	$\mathbf{95.84 \pm 3.7}$(2.11 ± 0.3)	99.48 ± 0.6(6.17 ± 1.1)	$\mathbf{98.57 \pm 1.7}$(4.14 ± 0.8)
Boston	$51.86 \pm 10.$(3.25 ± 0.4)	63.73 ± 6.6(4.19 ± 0.3)	$\mathbf{89.12 \pm 3.7}$(10.4 ± 2.0)	$\mathbf{96.67 \pm 2.4}$($20.3 \pm 27.$)
Kin8nm	$\mathbf{94.77 \pm 1.0}$(0.67 ± 0.0)	98.33 ± 0.7(0.87 ± 0.0)	$\mathbf{95.51 \pm 1.8}$(1.39 ± 0.1)	$\mathbf{97.77 \pm 0.6}$(0.78 ± 0.1)

Table 4.4: Mean and standard deviation, mean $\pm$ std, of the 100·PICP (MPIW) value between the 0.975 and 0.025 quantile of the $\boxed{\text{black box}}$-*uncertainty wrapper* for the different test set folds presented in Hernández-Lobato and Adams (2015b).

Testing a standard interval of confidence calibration Table 4.3 shows the differences in performance between the baselines when using the RF or XGBoost

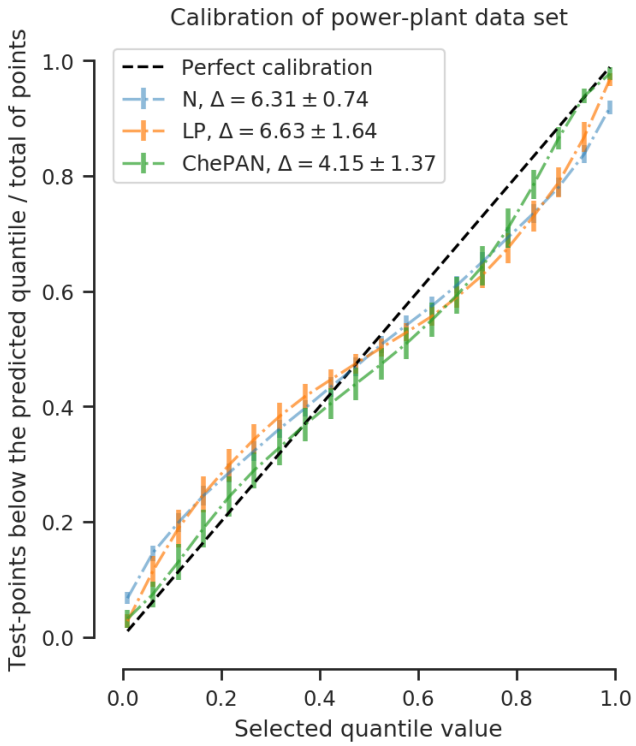


Figure 4.10: Calibration curve of the Normal, Laplace and CheNet models.

in terms of the QR loss function. However, we also want to show additional experiments that directly measure the calibration of the predicted quantiles and compare the predicted width of certain desired intervals. Following the UCI data sets used in Gal and Ghahramani (2016); Hernández-Lobato and Adams (2015b); Lakshminarayanan et al. (2017); Tagasovska and Lopez-Paz (2019), we performed two empirical studies to assess this point in a black-box scenario where the black box is an MSE-XGBoost. Following the proposed hidden layers architecture in Tagasovska and Lopez-Paz (2019), the Prediction Interval Coverage Probability (PICP) and the Mean Prediction Interval Width (MPIW) are reported in Table 4.4, considering the 0.025 and the 0.975 quantiles. We can state that generically, CheNet results are well calibrated compared to the other baselines.

Testing the generic calibration of the intervals of confidence For the sake of completeness, we also computed an additional metric, not only to verify

	$\boxed{XGBoost}$-N	$\boxed{XGBoost}$-LP	$\boxed{XGBoost}$-(I)CheNet	$\boxed{XGBoost}$-(E)CheNet
Concrete	14.40 ± 1.7	14.35 ± 1.5	$\mathbf{5.99 \pm 4.4}$	4.33 ± 1.8
Power	6.31 ± 0.7	$\mathbf{6.63 \pm 1.6}$	$\mathbf{4.15 \pm 1.4}$	2.89 ± 0.9
Wine	$\mathbf{4.40 \pm 1.2}$	$\mathbf{4.34 \pm 1.2}$	$\mathbf{4.58 \pm 1.6}$	5.23 ± 1.5
Yacht	$\mathbf{8.93 \pm 2.3}$	$\mathbf{6.98 \pm 2.4}$	14.20 ± 3.9	8.05 ± 3.0
Naval	$\mathbf{6.61 \pm 2.8}$	$\mathbf{7.21 \pm 2.1}$	$\mathbf{6.90 \pm 2.7}$	13.58 ± 3.5
Energy	$\mathbf{6.12 \pm 2.1}$	$\mathbf{5.05 \pm 2.0}$	$\mathbf{6.26 \pm 2.2}$	5.08 ± 1.9
Boston	15.55 ± 3.1	15.30 ± 2.1	$\mathbf{4.99 \pm 1.7}$	$\mathbf{6.98 \pm 4.1}$
Kin8nm	$\mathbf{3.55 \pm 0.6}$	4.79 ± 0.5	5.25 ± 0.8	4.40 ± 0.9
Protein	7.88 ± 1.9	7.30 ± 0.3	$\mathbf{5.39 \pm 0.8}$	5.49 ± 0.3

Table 4.5: Plot with performance in terms of calibration. The table contains the mean and standard deviation of all the folds using the mean absolute error between the empirical predicted calibration and the perfect ideal calibration of 980 equidistant quantiles using Eq. (4.36).

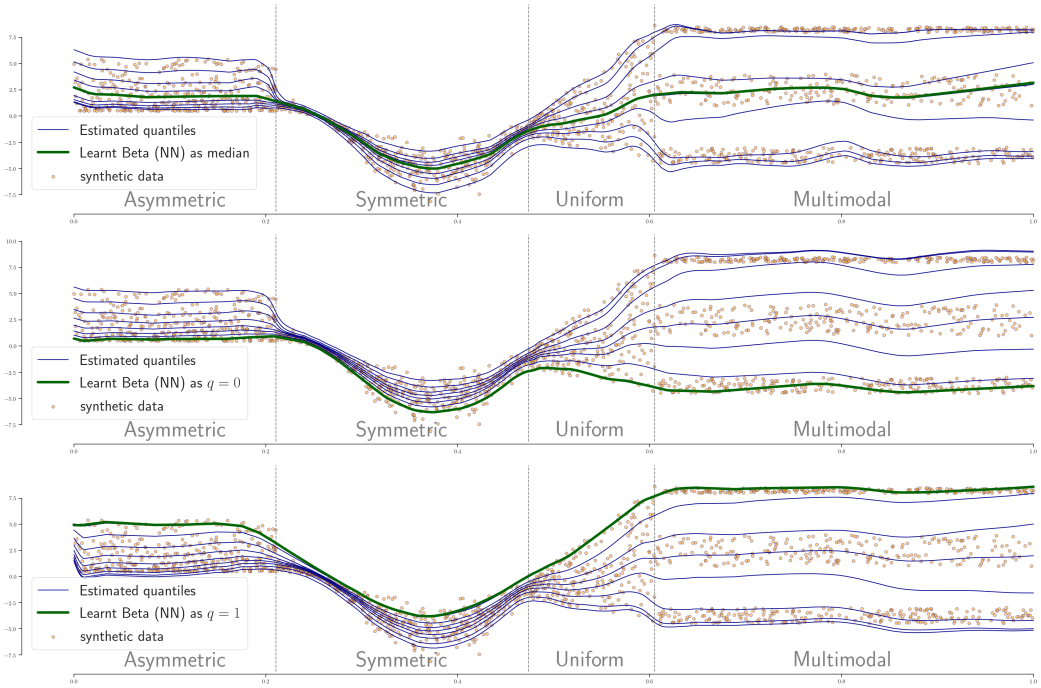


Figure 4.11: Heterogeneous synthetic distribution proposed by Brando et al. (2019b). In all cases, ϕ_w and $K(\boldsymbol{x})$ are learnt, while $K(\boldsymbol{x})$ corresponds to a different statistic in each case. In the upper part of the figure, $K(\boldsymbol{x})$ approximates the median, following Eq. (4.31). In the central figure, $K(\boldsymbol{x})$ regresses to the lower quantile 0, following Eq. (4.29). In the lower part, K corresponds to the higher quantile 1, following Eq. (4.30).

the calibration of the 0.025 and 0.975 quantiles, but also to obtain a measure of general calibration considering the entire quantile distribution (Figure 4.10 and

Table 4.5). Given N_τ-equidistant set of quantiles to evaluate, $\boldsymbol{\tau} = [10^{-2}, \ldots, 1 - 10^{-2}]$, the % of actual test data that falls into each predicted quantile can be compared to each real quantile value as

$$\mathcal{C}al(f; X_{test}, Y_{test}, \boldsymbol{\tau}) = \frac{1}{N_\tau} \sum_{j=1}^{N_\tau} |\tau_j - \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \mathbb{1}[y_i < f(\tau_j, \boldsymbol{x}_i)]|. \tag{4.36}$$

Comparing different summary statistics as $K(\boldsymbol{x})$ Finally, three extra figures showing the disentangled visualization of this calibration metric from each quantile can be found in Figure 4.11.

Summing up, all of the figures and tables related to calibration in this section show that CheNet generally displays a better performance in the black-box scenario than the other models.

4.4 Conclusions

The original motivation of the Uncertainty Modelling of a Black box (UMB) came from a real-world need where interpretability requirements forced us to implement an specific pointwise predictive model as a solution for the currently-in-use predictive system Brando et al. (2018a). In that case, this predictive system works generally well and, most importantly, satisfies the different imposed requirements (mostly related to interpretability needs) to be applied as a solution for our original sensitive problem. However, the particular risky environment where it is applied required us - once the model is in production - to model, additionally, the uncertainty related to its forecast but the replacement of this system is not advisable due to all the dependencies. Thus, we are in a scenario where we want to maintain the currently forecasting system without assuming any strong assumption about its internals (i.e. it will be considered as a black box) and, simultaneously, we would like to model the uncertainty regarding $p(y|x)$.

In the presented context, distributional-free solutions such as the QR models presented in previous Section 3.3 are not directly applicable as a solution for

all the presented black box uncertainty modelling problems. Yet, QR solutions can be considered if they are modelling the residual errors as it was proposed in Section 4.2 as a covert black box solution. Furthermore, alternative solutions, such as the location-scale family (in particular, the heteroscedastic normal and Laplace distribution which were previously introduced in Section 3.2.1) have an especial definition that allow us to disentangle the estimation of a certain point-wise prediction and its uncertainty estimation. Here, the special relation that can exist between the uncertainty estimation and the prediction introduces the concept of an honest black box, which motivates the second goal of this chapter: the Uncertainty Modelling of an Honest Black box (UMHB).

As the main proposed solution for the UMHB problem, in this chapter we presented the Chebyshev Network (CheNet), brings two important novelties in contrast to the previously introduced:

Firstly, it allows us to pre-decide the type of summary statistic constraint that will be defined using the $\beta(x)$ (as it is shown in Figures 4.6 and 4.9. Crucially, the CheNet Chebyshev coefficients are τ-independent (as shown in Eq. (4.21)). Therefore, the freedom of $C_0(x)$ allows us to define different types of constrains (see Section 4.3.3.4).

Secondly, it ables the possiblity to learn a distributional-free $q(y \mid x)$ that approximates $p(y \mid x)$ because it approximates implicitly all the quantiles distribution as a quantile function (as it is described in Algorithm 8 and also shown in Figure 4.9).

Introducing CheNet, we propose a generic deep learning wrapper to use any deep learning regression architecture to estimate the partial derivative of a conditional quantile function which can be used to enriching any previously defined point-wise predictive system with its aleatoric uncertainty estimation.

Empirical results over several real world data sets and synthetic ones were done to verify that the imposed restrictions between the point-wise predictive system and the distribution approximation would not adversely affect the performance of the model compared with other quantile regression baselines and likelihood-based models. As we saw in Section 4.2.4.3 and 4.3.4, the proposed models maintain the performance in terms of likelihood estimation and yields calibrated outputs with

respect the baselines as well as is able to tackle the UMB and UMHB problems with their corresponding models.

To sum up, this work is a further step in the research to improve the reliability and robustness of current production systems and state-of-the-art models. Future work could be continued in lines such as capturing other types of uncertainty as well as extending this method for classification problems.

Chapter 5

THE CROSSING QUANTILE PHENOMENON

The modelization of quantile functions (shown in Eq. (3.24)) constitutes an important goal, especially when the distribution to be predicted, $p(Y \mid X)$, is complex or when we are interested to impose the minimum assumptions about the shape of the distribution. Typically, an exponential power distribution (e.g. the Normal or Laplace distributions) is considered which implies to assume unimodality and symmetry and lose critical information regarding the shape of the distribution.

By building a quantile function we are able to approximate, in a discrete manner by means of quantiles, the distribution $p(Y \mid X)$. Typically quantiles are used to build confidence intervals. However, when these confidence intervals are so tight or when we want to recover the distribution shape from the discretization, we could face with a critical problem as follows.

Since the different predicted quantile values are estimated individually, they may not be ordered according to their quantile value, τ. For instance, the predicted quantile 0.1 may not be a value greater than the quantile 0.05. As a consequence, this leads to an invalid distribution of the response variable, which is known as the *crossing quantile* phenomenon (Koenker et al., 2017).

Different solutions have been proposed to overcome this phenomenon. Most of them are based on adding a penalisation term to regularise the optimization process and "encourage" the model to reduce the number of crossing quantiles

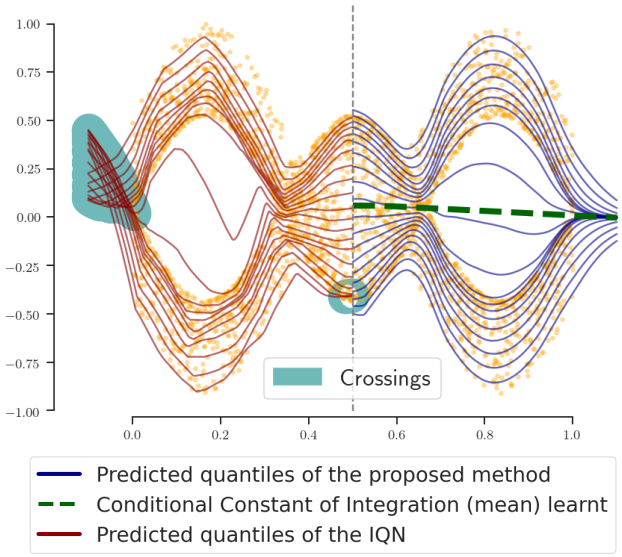


Figure 5.1: Synthetic data set presented in Brando et al. (2022a) showing that CheNet avoids crossing quantiles (right), unlike IQN (left).

Bondell et al. (2010); Koenker and Hallock (2001); Tagasovska and Lopez-Paz (2018) introduced in the next Section 5.1. Nevertheless, since the model is not restricted to being partially monotonic - i.e. be a monotonic function with respect to τ and not over the other inputs - some quantiles may still cross.

In this Chapter, following the results presented in Brando et al. (2022a), we propose an extension to the literature models to build a partial monotonic neural network with respect to the quantile value by constraining the weights, named as PCDN, and, additionally, we will see how the CheNet (previously introduced in Section 4.3.3) constitutes a solution for this phenomenon. Importantly, as we will show in the experimental section, to avoid crossing quantiles do not imply to obtain a better quantile regression performance (based on Eq. (3.22)): To force avoiding crossing quantiles can represent an extra constrain that makes the model difficult to obtain a better approximation.

5.1 The limitation of standard quantile regression

Despite the minimisation problem of Eq. (3.23), the flexibility of selecting a single model that estimates different conditional quantiles given certain inputs $\boldsymbol{x}$ can lead to a situation where the meaning between the predicted quantiles is lost due to a crossing. More formally,

Definition 14 *Let* $f\colon [0,1] \times \mathbb{R}^D \to \mathbb{R}$ *be a conditional quantile function that predicts a certain quantile of a random variable* $y \in \mathbb{R}$ *given* $\boldsymbol{x} \in \mathbb{R}^D$*. If there are* $\tau_1 < \tau_2$ *in* $[0,1]$ *such that for some* $\boldsymbol{x} \in \mathbb{R}^D$*,*

$$f(\tau_1, \boldsymbol{x}) > f(\tau_2, \boldsymbol{x}),$$

then f *has the* crossing quantile *phenomenon.*

In this case, the interpretation of the quantiles produced by f is not valid, resulting in important modelling problems in downstream tasks or nonvalid response distributions (as shown in Koenker et al. (2017)).

To avoid the crossing quantile phenomenon in the conditional quantile function f, it is required to impose monotonicity with respect to τ, i.e. for all $\tau_1, \tau_2 \in [0,1]$ and $\boldsymbol{x} \in \mathbb{R}^D$, if $\tau_1 < \tau_2$, then $f(\tau_1, \boldsymbol{x}) < f(\tau_2, \boldsymbol{x})$.

Different approaches can be listed, and subsequently analysed, to handle this crossing quantile phenomenon:

The deep heteroscedastic Normal distribution (N) Similarly to Section 3.2.1, the conditional normal distribution will be considered. Given that this quantiles are calculated from the parametric conditional distribution as $F(\tau, \boldsymbol{x}) = \mu(\boldsymbol{x}) + \sigma(\boldsymbol{x})\sqrt{2} \cdot \text{erf}^{-1}(2\tau - 1), \tau \in (0,1)$ where τ is the desired quantile and erf^{-1}, they will not cross. We will denote this approach as *N*.

The Implicit Quantile Network (IQN) Following Section 3.3.2, we are going to consider the IQN model. In particular, Tagasovska and Lopez-Paz (2018) proposes to add a regularization term with respect to the derivative as

$\max\left(-\frac{\partial\phi(\boldsymbol{x},\tau)}{\partial\tau}\right)$ to alleviate the crossing quantile phenomena. We will note this alternative optimization process as *IQN-D*. On the other hand, we will consider another regularisation term that penalises quantiles when crossing quantile phenomena appears as $\max(0., \phi(\tau_1; \boldsymbol{x}) - \phi(\tau_2; \boldsymbol{x}))$ for each $\tau_1 < \tau_2$. We will refer to this approach as *IQN-P*.

The Partial Constrained Dense Network (PCDN) The main idea of that model is to force the *selected weights* described in the following Section to be only positive. This combined with only considering ReLU activation function (Glorot et al., 2011b) produces that a partial monotonic function can be build.

The Unconstrained Partial Monotonic Neural Network (UPMNN) Similarly to CheNet, UPMNN has two different functions to optimize: The ϕ and the K. In that case, K corresponds to the q_0 value as a general shift of all the quantiles. The predicted quantile function does not ensure that always will avoid crossing quantile phenomenon but will be added to the comparison to analyse their difference.

The Chebyshev Network (CheNet) CheNet can be defined as a partial monotonous function considering the implicit and explicit versions introduced in Section 4.3.3.3. Several alternative selections of C_0 described in Section 4.3.3.4 are considered for the following comparisons.

5.2 The partial constrained dense network

The construction of neural networks with a monotonously increasing function of the inputs is a problem that has been identified in the literature, see, for instance, Sill (1998). These solutions were designed for completely dense models (i.e. those where all the neural network layers are fully-connected). These approaches were based on the following principle: a dense model with positive weights and with monotonically increasing activation functions gives a monotonously increasing function with respect to the input. Although, it is a very easy implementation to

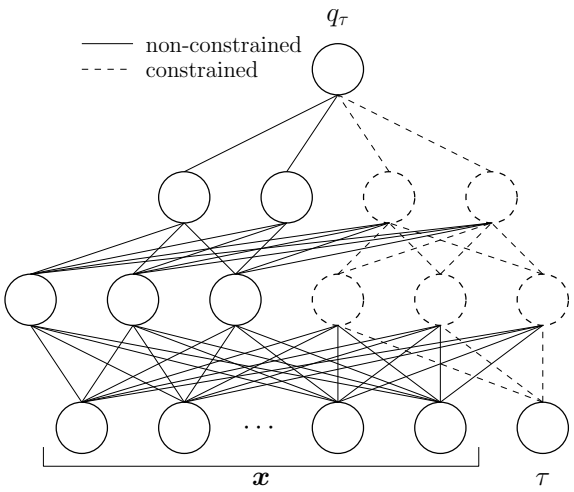


Figure 5.2: PCDN scheme.

carry out, it has been found that, at the practical level, the impossibility of having weights of different signs makes the learning process complicated Wehenkel and Louppe (2019).

According to the notation in the previous Section 5.1, to solve the crossing quantile phenomenon it is enough to define a monotonic function f with respect to the parameter τ (the quantile).

To built the model proposed in this section, we start with an IQN model, $\psi\colon [0,1] \times \mathbb{R}^D \to \mathbb{R}$, with only fully-connected layers by default.

Then, we select a desired part of the neurons for each hidden layer to ensure that ψ is monotonic with respect to τ; for instance, 50% of the neurons in each layer. These neurons are highlighted as dashed neurons in Figure 5.2.

Next, we force that these dashed selected neurons only have positive weights, their activation function are increasing and they are only subsequently connected to new layer dashed neurons or the last neuron, as shown in Figure 5.2.

Finally, in the first layer, we connect the input τ only to the dashed neurons of the first hidden layer, as we can see in Figure 5.2. Therefore, the changes in the τ input only affects the dashed neuron values, which behave as a monotonic function with respect to the propagated τ value. We refer to this monotonic version of the IQN as the Partial Constrained Dense Network (PCDN).

The PCDN has two important properties: On the one hand, the inputs $\boldsymbol{x}$ are connected to the output with restricted and non-restricted weights. On the other hand, the τ value is connected with the output of the model only with restricted neurons to ensure it is a partial monotonic increasing function with respect to τ. By applying PCDN we restrict the number of constrained weights and neurons to only selected part of the neural network. This model could be considered as an extension of other proposed models in the literature such as Monotone Composite Quantile Regression Neural Network (MCQRNN) (Cannon, 2018), which predicts a fixed number of quantiles.

5.3 Modelling the partial derivative with a neural network

Here we propose the Unconstrained Partial Monotonic Neural Network (UPMNN), an extension of the UMNN (see Reminder 4) to built a partial monotonic function. Once this is achieved, the model can be used to tackle the CQR problem (stated in Eq. (3.24)) considering the crossing quantile phenomenon.

Rather than building a monotonic function with respect to all the inputs - as in the UMNN - the UPMNN builds a function $\Phi\colon [0,1]\times\mathbb{R}^D \to \mathbb{R}$ so that it is only monotonic with respect to its first input variable $\tau \in [0,1]$. Once this function is defined, then it can be optimized using the stochastic CQR loss function designed for quantile functions (introduced in Eq. (3.25)).

Like the UMNN, the UPMNN learns a neural network, $\phi\colon [0,1]\times\mathbb{R}^D \to \mathbb{R}_+$, such that it is related to Φ by

$$\Phi(\tau;\boldsymbol{x}) \approx \Phi(0;\boldsymbol{x}) + \int_0^\tau \phi(t,\boldsymbol{x})\,dt \tag{5.1}$$

where the term $\Phi(0;\boldsymbol{x})$ corresponds to the conditional quantile function at $\tau = 0$.

The integral in Eq. (5.1) is computed via the Clenshaw-Curtis formula over the interval $[0,\tau]$. That formula, considering the quantile input, consists in the following steps:

1. First, we fix an *even* integer d, called degree, and we define the so-called nodes depending on τ:

$$\bar{t}_k^d(\tau) :- \frac{\tau}{2}\cos\left(\frac{\pi k}{d}\right) + \frac{\tau}{2}, \quad 0 \leq k \leq d. \tag{5.2}$$

   These nodes have the property that $\bar{t}_k^{2d}(\tau) = \bar{t}_{k/2}^d(\tau)$, which means that half of them can be reused when the value of d is doubled.

2. Second, we compute the quantities $\bar{c}_j(\tau, \boldsymbol{x})$ defined by

$$\bar{c}_j(\tau, \boldsymbol{x}) :- \sum_{k=0}^{d} \phi(\bar{t}_k^d(\tau), \boldsymbol{x}) \cos\left(\frac{j\pi k}{d}\right), \qquad 0 \leq j \leq d, \tag{5.3}$$

   which is just a matrix-vector multiplication. Here, we can use an algorithm, known as the Discrete Cosine Transform of type 1 (DCT-I), which performs the matrix-vector multiplication in Eq. (5.3) with a complexity $\Theta(d \log d)$ rather than a standard $\Theta(d^2)$ procedure. In general, this algorithm produces the unnormalized coefficients defined in Eq. (5.3). To normalize them with respect to the degree, we must apply a factor, for instance, $2/d$. Thus, let us redefine the quantities

$$\bar{c}_j(\tau, \boldsymbol{x}) \leftarrow \frac{2}{d}\bar{c}_j(\tau, \boldsymbol{x}).$$

3. Finally, $\Phi(\tau; \boldsymbol{x})$ in Eq. (5.1) is approximated by $P(\tau; \boldsymbol{x}, d)$ such that

$$P(0, \boldsymbol{x}; d) = \Phi(0; \boldsymbol{x}),$$

   i.e. all the constant of integration in Def. 7 - the sum of $K(\boldsymbol{x})$ and $K_w(\boldsymbol{x})$ - or in the constant of integration in Eq. (5.1) is the conditional quantile $\tau = 0$. All of the above leads us to the final Clenshaw-Curtis expression used in the UPMNN,

$$P(\tau, \boldsymbol{x}; d) = \tau\left(\frac{\bar{c}_0(\tau, \boldsymbol{x})}{2} - \sum_{k=1}^{d/2} \frac{\bar{c}_{2k}(\tau, \boldsymbol{x})}{4k^2 - 1}\right) + \Phi(0; \boldsymbol{x}). \tag{5.4}$$

Note that Eq. (5.4) has a τ dependency on all the coefficients of $\bar{c}_k(\tau, \boldsymbol{x})$, which comes from the fact that the P depends on τ, because the nodes in Eq. (5.2) also originally depend on the quantile τ. The resulting contribution - the main model - will avoid this dependency, which will be crucial to learn ϕ_w and K_v generically, as we will see hereafter.

5.4 The CheNet as a partial monotonic solution

5.4.1 Rate of convergence of the Chebyshev expansion

In Chebyshev series theory (Majidian, 2017; Trefethen, 2008), if a function ϕ_w is of class C^1 in $[-1, 1]$, then its Chebyshev expansion converges absolutely and uniformly to ϕ_w in $[-1, 1]$. In case that ϕ_w is in $C^k([-1, 1])$, then its Chebyshev coefficients c_ℓ verifies that $|c_\ell| = o(1/\ell^k)$. Moreover, if ϕ_w is now analytic, its Chebyshev expansion will have an exponential rate, i.e. $|c_\ell| = o(\exp(-\rho\ell))$ for some $\rho > 0$ denoting the radius strip in the complex plane whose strip contains all the Chebyshev coefficients c_ℓ. Therefore, smoother mappings are going to require less Cheybshev coefficients. To have a kind of measure about the accuracy of the approximation $p \approx \phi$, one needs to check the decay rate of the Chebyshev coefficients c_ℓ. Most of the times, it will be enough to monitor the last two coefficients $c_{d-1}(\boldsymbol{x})$, and $c_{d-2}(\boldsymbol{x})$ in absolute value.

Due to the discretisation in the root mesh $\{t_k\}_{k=0}^{d-1}$, which acts as a linear transformation, the evaluation of $P(t_k; \boldsymbol{x})$ will have roundoff error (i.e. machine precision) with respect to the integral of $\int_0^{t_k} \phi_w(t; \boldsymbol{x})\, dt$ and for the other values not in the mesh, its (absolute) error will be bounded by the aforementioned convergence rate.

5.4.2 Ensure monotonicity for all quantiles

There is one last detail that it should be analysed to be totally confident that the function $P(\tau; \boldsymbol{x}) = f(\tau, \boldsymbol{x})$ is monotonic with respect τ. It is required to consider that p is an approximation of ϕ_w, i.e. $p(\tau; \boldsymbol{x}) \approx \phi_w(\tau; \boldsymbol{x})$. Thus, although the

ϕ_w function has been forced to be strictly positive, the approached Chebyshev polynomial p may not be. According to Section 5.4.1, we are certain that for the roots values, $\{t_k\}_{k=0}^{d-1}$, the Chebyshev estimation has a negligible error. However, in the middle points between the roots it is important to be careful.

We will divide the solution according to whether the model chosen is implicit or explicit.

Implicit CheNet (dynamic degree): As the degree is not fixed, once it is detected that there is a point that does not meet the condition of montonicity we can augment the degree, d, as much as we want (without retraining the model, just evaluating P in other points) and we are certain that we will always end up finding a degree value such that the error will again be negligible, based on the convergence presented in Section 5.4.1. Consequently, the Implicit CheNet version can produce always a monotonic function.

Explicit CheNet (static degree): Once the appropriate degree is known, the model with that value can be considered a monotonic function. However, it is important to note that although the error will be small, the possibility of P ceasing of being monotonic exists in all non-root points. In fact, this solution implies to increase to d the number of outputs and, consequently, the number of parameters to learn.

5.5 Results and comparison

The following results will be analysed regarding several metrics related to the number of times each model has a crossing and analysing how the number of crossings decreases as bigger is the degree of the CheNet. Additionally, the performance of the compared models will be analysed using tables of sections before but highlighting the differences when the model ensure partial monotonicity or not.

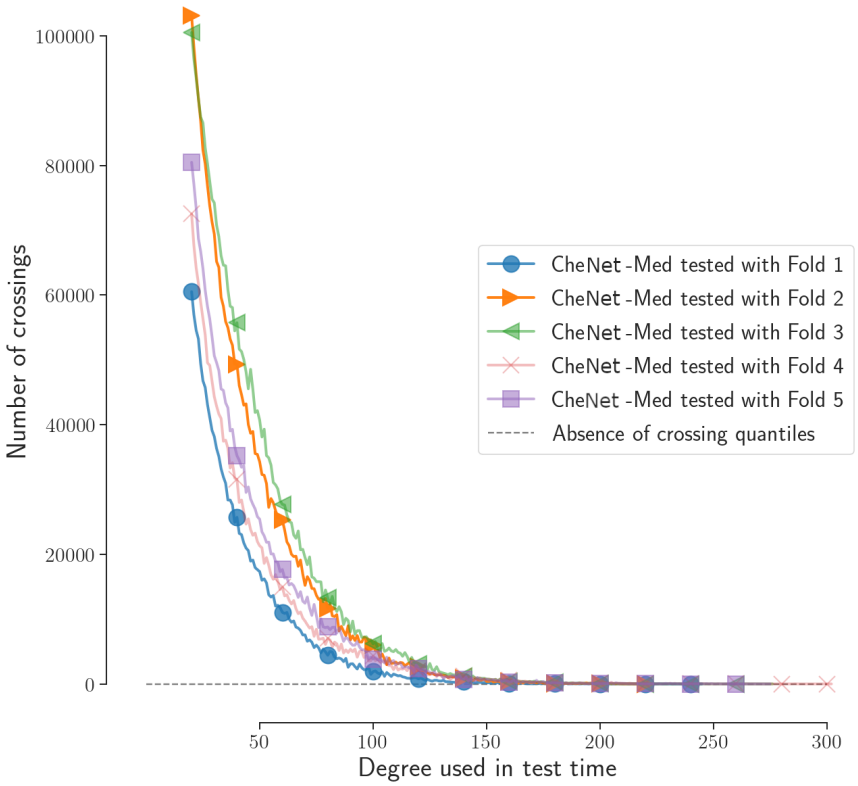


Figure 5.3: Correlation between the number of crossing quantiles and the degree used in test time for the worst data set in terms of crossings (the protein data set) using all CheNet-Med models trained with $d = 20$. Each line is displayed until it has four consecutive values of 0.

Number of crossing analysis Table 5.1 shows the minimum and maximum number of crossing quantiles for each model taking into account all the folds. As expected, the N model does not have crossing quantiles since it is derived from the normal quantile function. The PCDN never has any crossing quantiles, as stated in corresponding Section 5.2. Furthermore, none of the CheNet models have crossing quantiles if a sufficiently high degree is used, which can be decided in test time when these models are trained in the Implicit version or evaluating the internals roots in both versions (as shown in Section 4.3.3.3).

Table 5.1: Minimum and maximum, [min, max], of the crossing quantile numbers over all the test folds proposed in Hernández-Lobato and Adams (2015a). "*" denotes the number of crossing quantiles for high enough degree during inference or evaluating the roots.

	Housing	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht
N	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**
IQN	**[0, 0]**	[0, 1416]	[75, 3007]	[0, 834]	[1673, 104689]	[0, 4058]	[57554, 87096]	**[0, 0]**	[281, 5935]
IQN-P	**[0, 0]**	**[0, 0]**	[11, 2583]	[0, 782]	[1072, 123594]	[0, 2539]	[42461, 77603]	**[0, 0]**	[22, 2852]
IQN-D	[0, 55]	[0, 1541]	[733, 8259]	[0, 3959]	[6246, 263553]	[175, 72813]	[68210, 113867]	[0, 2131]	[931, 7632]
PCDN	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**	**[0, 0]**
UPMNN	**[0, 0]**	[0, 1201]	[2152, 22760]	**[0, 0]**	[0, 0]	[0, 433]	[2090, 11409]	**[0, 0]**	[150, 2630]
CheNet-q_0	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***
CheNet-q_1	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***
CheNet-Med	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***
CheNet-Mean	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***	**[0, 0]***

Table 5.2: Minimum and maximum degree used, [min, max], over all the test folds proposed in Hernández-Lobato and Adams (2015a) to obtain 0 crossing quantiles in test time.

	Housing	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht
CheNet-q_0	[20, 20]	[20, 40]	[40, 90]	[20, 20]	[20, 20]	[20, 40]	[180, 380]	[20, 20]	[20, 80]
CheNet-q_1	[20, 20]	[20, 40]	[20, 80]	[20, 20]	[20, 20]	[20, 40]	[130, 230]	[20, 20]	[30, 70]
CheNet-Med	[20, 30]	[20, 180]	[20, 350]	[20, 20]	[20, 20]	[20, 200]	[250, 300]	[20, 20]	[20, 40]
CheNet-Mean	[20, 20]	[20, 20]	[20, 210]	[20, 20]	[20, 20]	[20, 40]	[160, 190]	[20, 20]	[20, 20]

Asymptotic monotonic guarantees of CheNet Figure 5.3 depicts the statement presented in Section 5.4.2 for the worst data set in terms of number of crossing quantiles according to Table 5.1. For all the UCI data set presented in Brando et al. (2022a) and Brando et al. (2022b), we computed the minimum and maximum number of required degree to arrive to 0 crossing quantiles in the each test fold. That information is reported in Table 5.2. In this analysis the degrees start from $d = 20$ with an increment of 10 each time until there is no crossing quantiles in the test fold. For a qualitative analysis, we refer to the Figure 5.3. As we can see in Figure 5.3, the higher the degree evaluated in test time for a certain input data x, the more likely it is to have no crossing quantiles. All the tested folds reach 0 crossing quantiles.

Additionally, Table 5.2 contains the minimum and maximum degree used to

satisfy the 0 number of crossing quantiles during 4 degrees consecutively for each UCI dataset used in the previous results of Table 5.1.

Table 5.3: Mean and variance, mean ± variance, over all the up to 20 test folds proposed in Hernández-Lobato and Adams (2015a) of the quantile regression loss, see Eq. 3.24.

	Housing	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht
N	**88.72 ± 16.62**	168.45 ± 16.39	58.68 ± 18.99	2.53 ± 0.11	0.02 ± 0.00	**111.70 ± 2.66**	120.41 ± 2.43	**18.22 ± 1.31**	97.46 ± 59.04
IQN	**84.87 ± 19.41**	**138.80 ± 15.92**	**12.31 ± 1.64**	**2.25 ± 0.07**	0.02 ± 0.00	**109.45 ± 3.14**	**105.89 ± 1.80**	**18.34 ± 1.34**	**14.63 ± 4.69**
IQN-P	**85.13 ± 17.96**	**150.25 ± 15.03**	**13.02 ± 1.87**	**2.23 ± 0.09**	0.02 ± 0.00	**113.52 ± 2.42**	109.38 ± 1.75	**18.58 ± 1.29**	**16.40 ± 5.56**
IQN-D	**112.42 ± 22.21**	205.89 ± 29.44	46.72 ± 11.94	3.11 ± 0.17	0.03 ± 0.01	154.14 ± 5.78	153.00 ± 5.41	25.03 ± 2.53	**20.95 ± 10.54**
PCDN	**88.41 ± 20.25**	**147.40 ± 18.10**	**13.08 ± 1.66**	**2.31 ± 0.09**	0.02 ± 0.00	**113.94 ± 2.99**	111.26 ± 0.75	**18.42 ± 1.29**	**15.00 ± 4.88**
UPMNN	**83.85 ± 18.02**	**142.51 ± 14.90**	**14.45 ± 2.30**	**2.29 ± 0.07**	**0.01 ± 0.00**	**109.49 ± 3.52**	**105.60 ± 0.91**	**18.38 ± 1.41**	**14.56 ± 4.18**
(I) CheNet-Med	**84.79 ± 18.73**	**145.27 ± 14.54**	**19.20 ± 4.42**	**2.28 ± 0.07**	0.02 ± 0.00	**111.38 ± 3.00**	**107.33 ± 1.44**	**18.32 ± 1.14**	**16.74 ± 5.48**
(I) CheNet-Mean	**85.23 ± 17.91**	**142.03 ± 16.22**	**15.74 ± 3.96**	**2.30 ± 0.07**	0.02 ± 0.00	**111.72 ± 2.73**	108.20 ± 1.31	**18.40 ± 1.20**	**16.89 ± 5.40**
(E)CheNet-Med	**83.75 ± 18.41**	**142.11 ± 16.92**	**16.31 ± 3.57**	**2.26 ± 0.08**	**0.02 ± 0.01**	**111.18 ± 2.54**	**107.37 ± 1.68**	**18.23 ± 1.29**	**15.18 ± 4.89**
(E)CheNet-Mean	**84.81 ± 19.39**	**142.56 ± 13.45**	**14.89 ± 1.58**	**2.27 ± 0.06**	**0.02 ± 0.01**	**111.21 ± 2.83**	**107.68 ± 0.48**	**18.20 ± 1.33**	**16.43 ± 4.81**
(I) CheNet-q_0	**84.09 ± 19.01**	**142.48 ± 15.76**	**14.59 ± 1.81**	**2.30 ± 0.06**	**0.01 ± 0.00**	**109.36 ± 2.73**	**105.23 ± 1.55**	**18.42 ± 1.24**	**15.33 ± 4.70**
(I) CheNet-q_1	**91.67 ± 20.25**	**146.62 ± 20.25**	**17.76 ± 2.31**	**2.28 ± 0.05**	0.02 ± 0.00	**109.15 ± 2.91**	**108.07 ± 1.09**	**18.70 ± 1.15**	**15.73 ± 4.50**
(E)CheNet-q_0	**83.71 ± 18.76**	**138.34 ± 15.40**	**13.18 ± 1.74**	**2.20 ± 0.06**	0.02 ± 0.00	**108.46 ± 2.71**	**104.16 ± 1.45**	**18.27 ± 1.28**	**13.75 ± 4.77**
(E)CheNet-q_1	**88.96 ± 18.27**	**143.68 ± 16.54**	**16.92 ± 2.60**	**2.21 ± 0.06**	0.02 ± 0.00	**109.34 ± 2.77**	**106.34 ± 1.20**	**18.64 ± 1.55**	**16.13 ± 4.61**

Comparing Quantile Regression Performance As we saw, not all models proposed in this work ensure monotonicity. Although ideally this property should be satisfied for a model that perfectly fits the conditional density distribution $p(y \mid \boldsymbol{x})$ in a discrete manner, the reality is that imposing this property can make it more difficult the fitting process, resulting in a model with worse performance. To evaluate this point, we can look at the results are reported in Table 5.3 again, where the quality of the predicted quantiles is evaluated regardless of crossing quantiles because we calculate the sum of the quantile regression loss function of 100 random quantiles for each of the models in the test set. We can conclude that the non-restricted models are not clearly better than the restricted ones proposed in this work. Thus, avoiding the crossing quantile phenomenon does not imply a clear loss in terms of performance, regarding the compared models, and it will be a property to be imposed only when the problem to be solved requires it.

5.6 Conclusions

Quantile Regression (QR) is an extensions of the common regression methods to estimate the conditional density distribution of the response variable, $p(y \mid x)$, in a discrete manner (or in a point-wise way) and without assuming any parametric distribution. The distributional freedom of QR is a desired property but when a single QR-model tries to approximate several quantiles at once, the order between the predicted quantiles matters. When these quantiles are not predicted in an increasing way considering its quantile value, τ, then the predicted distribution is invalid. This phenomenon is called "crossing quantile". In this chapter, we introduced several methods that solves this issue illustrated in Figure 5.1.

Apart from the introduction of the Partial Constrained Dense Network (PCDN), another main contributions of this chapter is to show how the Chebyshev Network (CheNet) (initially presented in Section 4.3.3) can be applied to solve the crossing quantile phenomenon. In particular, CheNet is a deep learning model to approximate the partial derivative of an ending quantile function with respect to the quantile input. This partial derivative is imposed to be positive, thus, the ending function will be partially monotonic as desired. In Section 5.4, we described how to calculate this derivative by using a Chebyshev Polynomial approximation ensuring the monotonicity of the quantile function up to any desired precision.

Having computationally guarantees of the monotonicity of these models, our final goal was to verify if the imposed restrictions would adversely affect the performance of the model compared with a non-ensured quantile regression model. As we saw in Section 5.5, the proposed models outperforms or maintains the performance - in terms of likelihood estimation - and yields non-quantile crossing forecasts with respect the other compared models.

Overall, the proposed models constitute generic deep learning wrappers for any neural architecture to build partial monotonic quantile functions avoiding the crossing quantile phenomenon.

Chapter 6

UNCERTAINTY VISUALIZATION AND EVALUATION

The generic goal of visualization is to effectively and accurately communicate data (Bonneau et al., 2014). In recent times, technological advances lead to increase the data acquisition and their quality, this emphasize the importance of developing tools and methods to manage large volumes of data and build human-understandable processes to extract insights from them. All of this, recognizes and accelerates the research and work on visualization field (Rosling, 2006).

Particularly, one of the more challenging aspects of data visualization is the representation of uncertainty (Wilke, 2019). When we see data points drawn in a specific location in a plot, we tend to interpret it as a precise representation of the true data value. It is difficult to conceive that most of the times there is a non-deterministic scenario where several sources of uncertainty can affect our visualization. As we saw in Chapter 2, nearly every data set contains this ambiguity (or even the model chosen for the visualization can introduce it). Consequently, the way we choose to represent this uncertainty can produce a major difference in how decisions are made based on these visualizations or, generically, how our audience perceive the model forecasting or data.

To highlight the importance of representing uncertainty in regular decisions

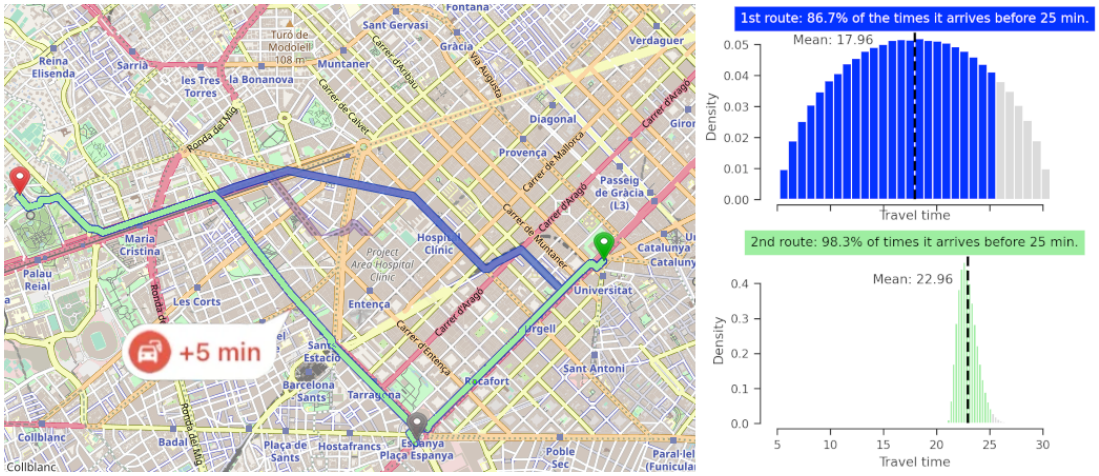


Figure 6.1: There are two routes. The blue one is shorter in expectation but when we must arrive before the 25 minutes we should take the green one.

making, we can consider a taxi driver which requires to select a certain route as described in Duerr et al. (2020). In this context, let us imagine a client asks the taxi driver that they needs to reach their destination in less than 25 minutes. To simplify the problem, we consider that there are two possible routes to the destination as shown in Figure 6.1: One that takes approximately 18 minutes on average and is shorter and the second that takes 23 minutes on average and is longer in distance. Based on the information of the distance and the average minutes of trip, the optimal decision would be the first option. However, as shown in the Figure 6.1, if we observe the distribution of the travel time we can see that, in the second option, the probability of arriving before 25 minutes is 98.3% that is higher than the 86.7% of the first option, which would lead the taxi driver to take the second route.

In the previous example, we have a typical case where the visualization of the information is crucial for the decision-making process. Nevertheless, the typical route predictors only indicate us the average travel time, which prevent us to know if there exist a considerable variability in the travel time that can change our decision. In fact, this idea connects with the importance of designing descriptors that avoids strong assumptions regarding the shown distributions as we did with

UMAL in Section 3.2.3, quantile regression in Section 3.3 or CheNet in Section 4.3.3. Therefore, as we highlighted in these previous sections, a central variability descriptor as uncertainty information might not even be enough if we are tackling a multi-modal or heterogeneous scenarios and, thus, we should consider to provide richer descriptors as we will see in this chapter.

In fact, most of these procedures to select which can be a better visualization would be linked with an evaluation metric, for instance, for the model selection. Eventually, measuring metrics should also consider the uncertainty to avoid misleading predilections (e.g. choose models with highest accuracy without considering the quality of its failures). In the following sections, we are going to enter into the details in some of them.

6.1 Describing each source of uncertainty

In Chapter 2, we introduced a procedure to split the uncertainty sources depending on the associated probability it has, which are obtained using the chain rule. Nowadays, this distinction presented in Eq. 2.2 is no commonly accepted and produces an open-debate regarding the nomenclature or proper names of these uncertainty types as we discussed in Section 2.1. Although the main goal of this work is to tackle aleatoric uncertainty, in order to clearify the differences with other kinds o uncertainty, in this sections, we will consider an expanded version of Eq. 2.2 to delve into the importance of having different representations for each part of this equation that corresponds to an specific type of uncertainty source. The equation will be

$$p(x^*, y^*, X, Y) \approx p(X)p(x^* \mid X) \int_{\mathbb{M}} p(M \mid X, Y, x^*) \cdot p(y^* \mid X, Y, M, x^*) dM, \quad (6.1)$$

where x^* is the new evaluated point with their corresponding new label to be predicted y^*, $p(X)$ corresponds to the measurement-error, $p(x^* \mid X)$ is the manifold uncertainty, $p(M \mid X, Y, x^*)$ is the epistemic uncertainty and $p(y^* \mid X, Y, M, x^*)$ is the aleatoric uncertainty, which was described in Figure 2.1. Fol-

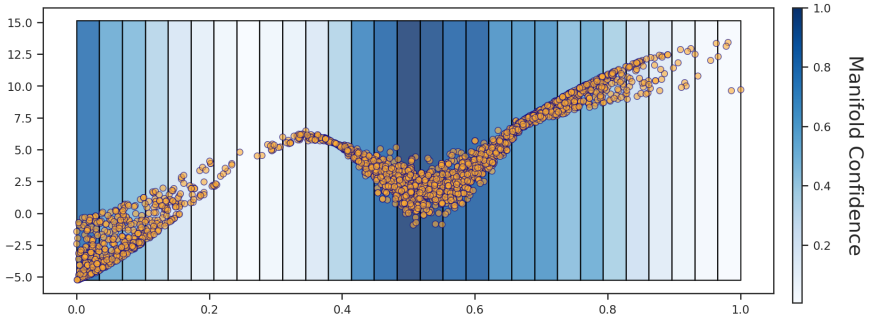


Figure 6.2: The Manifold uncertainty is captured by modelling $p(X)$. This value is shown in the background in blue.

lowing we will propose several ways to visualize them.

6.1.1 Measurement-error and manifold uncertainty visualization

When we were modelling $p(X)$, according to Chapter 2 notation, this includes the measurement-error, here denoted also as $p(X)$, and the manifold uncertainty, $p(x^* \mid X)$, which they have different meanings. Importantly, this combined uncertainty do not consider the variable to be predicted, Y, nor the model to perform such prediction, characterized by M. Therefore, we should be careful when we represent this uncertainty in a standard regression plot (e.g. most of the plots presented in Chapters 3, 4 and 5), where the horizontal axis is some input variable and the vertical axis is the predicted values, due to $p(X)$ not depend on Y. Consequently, one way to represent this uncertainty can be using the background colour as represented in Figure 6.2. In that case, each horizontal value has a different background colour where purpler zones correspond to high confidence X values while zones with lower X confidence will be bluish.

Analysing Figure 6.2 we can see that zones where there are less data points have a whitish colour (e.g. between 0.2 and 0.4 or between 0.8 and 1) that can also be produced by a measurement-error, which corresponds to an irreducible variability of the input variable. Furthermore, we should highlight that the $p(X)$

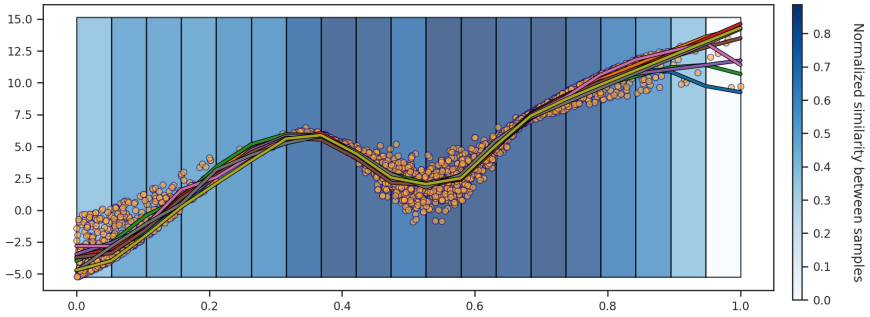


Figure 6.3: Each NN component of the ensemble is approximating the conditional median. The discrepancy on those components encodes the epistemic uncertainty. In blue, the normalized standard deviation is shown in the background.

value is independent of the conditional variability $p(Y \mid X)$, as we can observe, for instance, between the 0.4 and the 0.6 points.

6.1.2 Epistemic uncertainty representation

Epistemic uncertainty, $p(M \mid X)$, which corresponds to the uncertainty related to selecting a certain family of models $\mathbb{M}$ (see Figure 2.2) is a similar scenario than the Manifold uncertainty: The new response random variable y^* is not involved in this uncertainty. However, here the prediction of each model characterized by M is usually approximating some statistic of Y given X. This last detail produces several ways of visualizing this uncertainty depending on what is approximating each model but we should be careful to distinguish between the epistemic and aleatoric approximation part. To do it, here we will consider only point-wise approximator models, e.g. a model that is predicting the conditional median.

At the end, considering a certain family $\mathbb{M}$ of such models is to consider an ensemble with a finite - or not - number of components or models. Their discrepancy refers to the epistemic uncertainty we are capturing. Therefore, one way to visualize its discrepancy is to plot each prediction separately, as it is shown in Figure 6.3.

Comparing Figure 6.3 with Figure 6.2, we can see clearly the difference be-

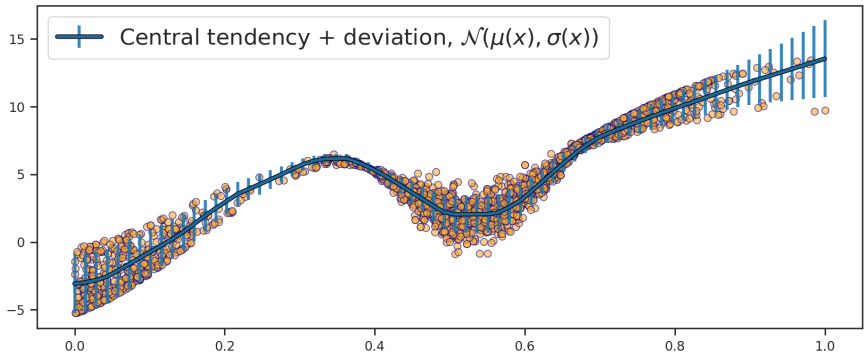


Figure 6.4: Conditional normal distribution where their parameters are approximated using a NN, as Section 3.2.1 shows. Aleatoric uncertainty is captured as central deviation from the mean.

tween capturing the Manifold or Epistemic uncertainty: For instance, in the horizontal interval from 0.2 to 0.4, the behaviour of both uncertainties are completely different. This is because the density of $p(X)$ is small in such interval but, differently, the approximated conditional medians of the ensemble are producing a similar forecast given the previous and posterior shape of the data is clearly defined (and the consequence behaviour that use to perform NN models). This fact could tend to change when X is high dimensional but, if the ensemble is naively approximated, we do not have any guarantee that the discrepancy will be always higher in zones where $p(X)$ is lower using such NN models.

Importantly, similarly to the Manifold case of Section 6.1.1, it is worth to highlight that high conditional variability zones between the response variable and the input one, such as in the horizontal interval from 0.4 to 0.6, does not imply to having an epistemic discrepancy if the approximated statistic is clearly defined. Therefore, we should need to model aleatoric uncertainty to detect this extra new source of uncertainty.

6.1.3 Aleatoric uncertainty visualization

The main source of uncertainty analysed in this work is the aleatoric uncertainty. Following Section 2.1 and Chapter 3, this kind of uncertainty is focused on mod-

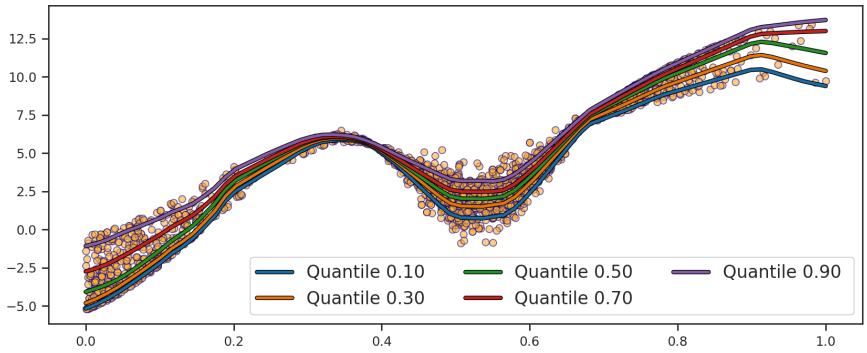


Figure 6.5: Free distributional quantiles approximated using quantile regression with a NN, as Section 3.3 shows. Aleatoric uncertainty is captured as discrete distributional cumulative points.

elling the variability of the response variable, Y, given a fixed X and M, denoted as $p(y^* \mid x^*, X, Y, M)$ in Figure 2.2 and shown in Eq. 6.1.

Unlike previous uncertainty types, visualizing aleatoric uncertainty has a direct impact on the response variable to be modelled, therefore, this uncertainty can be represented without the vertical bars used in Figure 6.2 and 6.3 because now it depends on the vertical axis values. Additionally, this uncertainty is irreducible, therefore, our goal will be to show the distributional shape to design shape-tailored techniques. This is why it is important to avoid strong assumptions regarding to the conditional distribution, such as symmetry or unimodality, if we do not have clear evidences that they cannot harm the forecasting procedure, as we analysed in Chapter 3, but even here for improving the visualization tasks.

When standard symmetric and unimodal approaches are considered, the result can be shown in Figure 6.4. As it is shown, non symmetrical areas (i.e. before the horizontal 0.3 points and after the 0.7 one) are not captured, thus, not reported using this approach. As we discussed in Section 3.2.1 and Section 3.3, richer descriptors than central tendency with variance can provide us decisive information. In this line, approximating the quantiles by means of quantile regression, as it is shown in Figure 6.5, allows to consider some upper/lower- bound quantiles as well as central ones to summarize the shape of the distribution in a point-wise

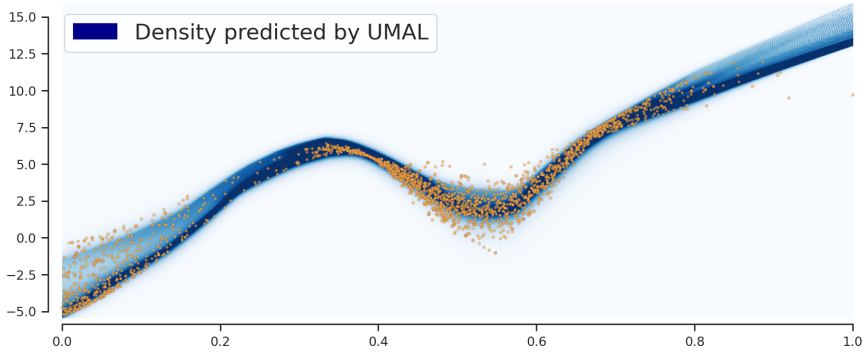


Figure 6.6: Conditional distribution approximated using UMAL, as Section 3.2.3 introduces. Aleatoric uncertainty is captured as the approximated likelihood.

way. However, quantiles constitutes a discrete approximation of the conditional distribution and, in some applications such as the described in Section 3.2.3, the estimation of the likelihood could be beneficial not only to take certain decisions but also to be combined with the previous uncertainty types as we will see in the next section. To provide a richer estimation of the likelihood, in Figure 6.6 we can see a UMAL forecast of the previously presented data set, where blueish areas are the ones that has higher likelihood.

On the whole, considering Figures 6.4, 6.5 and 6.6 we can observe that lower likelihood points are those where the conditional variability is higher. Therefore, between $[0., 0.2]$, $[0.4, 0.6]$ and $[0.8, 1]$. This behaviour contrast with the presented in previous Section 6.1.1 and 6.1.2 as we will discuss in the next section when an integrated approach will be designed.

Importantly, isolated aletoric uncertainty fixes a certain model parametrized by M. This can be seen as one of the components of the ensemble in the previous epistemic Section 6.1.2 and, based on this idea, we can build an integration procedure to visualize all the presented uncertainty sources as follows.

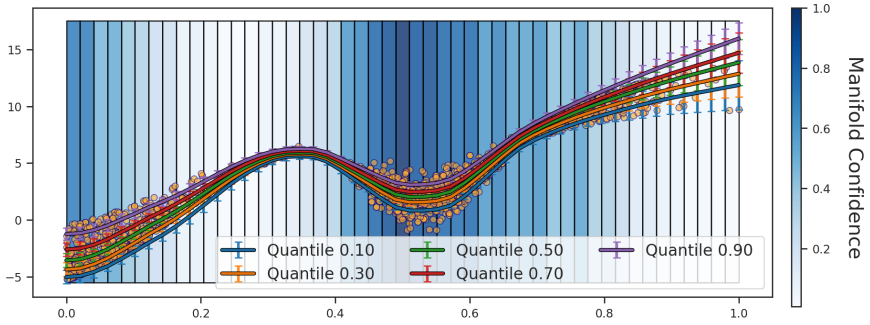


Figure 6.7: Discrete Integrated uncertainty visualization based on Sections 6.1.1, 6.1.2 and 6.1.3. Aleatoric uncertainty is captured using quantiles, epistemic one considers their mean and standard deviation and Manifold one as density bars.

6.1.4 Integrated uncertainty representation

Proposing an integrated procedure to represent all the uncertainty types is useful to synthesize all this complex information in a single plot. Based on the previously introduced visualization types, we can represent the presented Measurement-error, Manifold, Epistemic and Aleatoric uncertainty in a discrete way as it is shown in Figure 6.7. Particularly, the aleatoric uncertainty is represented as quantiles in that case. Consequently, we have a set of quantiles for each model M considered in the ensemble. Assuming they are independent samples, we can compute their mean and variance for each conditional quantile and represent them as error bars as shown in Figure 6.7.

Moreover, if all the uncertainties are expressed as probabilities, this integrated visualization can be combined in an unique probability using the Eq. 6.1, introduced in Chapter 2 and in Figure 2.2, which can provide us a continuous visualization technique that combines all the uncertainty sources by representing $p(X, Y)$, shown in Figure 6.8. This representation displays the confidence in all the uncertainty sources using the $p(X, Y)$ estimated information, which includes modelization of the residual variability, outlier detection, model uncertainty and conditional irreducible uncertainty.

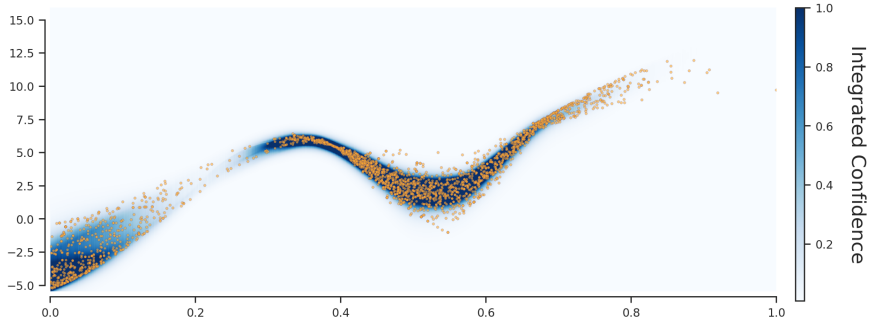


Figure 6.8: Continuous Integrated uncertainty visualization based on Sections 6.1.1, 6.1.2 and 6.1.3. All uncertainties are considered as probabilities.

6.2 Qualitative check of the reported uncertainty

Once we have techniques to report each kind of uncertainty as it was described in Section 6.1, now we require a process to ensure the decisions we can make, using that information, are reliable. This process is connected with the characterization of the predicted uncertainty quality. In other words, we need to be able to judge how confident we can be in scenarios forecast as doubtful or safe. Additionally, another key factor in evaluating the reported uncertainty is to detect which of a set of predictive systems are producing a better forecast of the uncertainty to, for instance, propose improvements of the evaluated models.

Next, we will describe different graphical evaluation methods used and developed during the present thesis and in Brando et al. (2022b, 2018a, 2019b, 2020) for characterize the main source of uncertainty of this work, the aleatoric one.

6.2.1 Error-retention curve for checking score quality

The goal of the thesis is mainly regression problems to forecast a single dimensional output. In this context, following the same scheme to build the uncertainty modelling of a "black box" presented in Section 4.1, our forecasting system has two different outputs: a prediction of the desired value - denoted as $\hat{y}$ - and an associated uncertainty measure - obtained from a function denoted as ψ - that

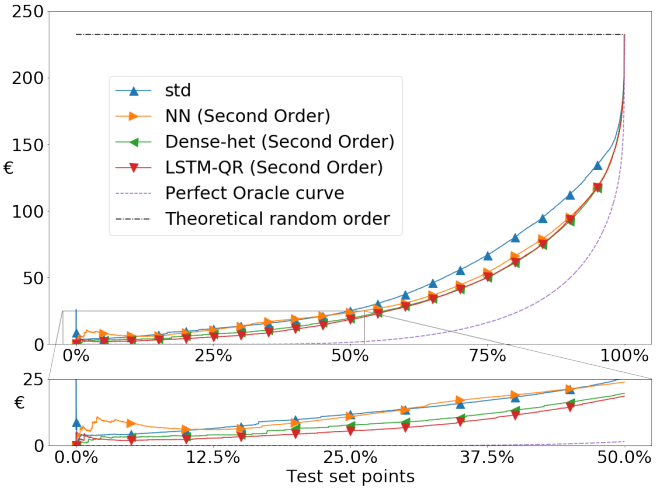


Figure 6.9: An example of an Error versus Retention Curve (ERC).

can be used for the confidence-based decision making process or even to order from less to high safe predictions. Helpfully, the order is uniquely defined in such single dimensional output space, which implies we can use a one dimensional confidence measure (denoted as the "uncertainty score" in Section 4.2) to sort from more to less confident points and see if the empirical actual error satisfies this order.

The Error versus Retention Curve (ERC), firstly introduced in Section 4.2.4.3, has the goal of showing how properly sorted are the predictions if we consider the predicted uncertainty score versus a perfect sorting process (denoted as the "oracle uncertainty score") that is computed a posteriori using the real error as follows,

$$ERC_d\left(\{y\}_{i=1}^N, \{\hat{y}\}_{i=1}^N, \kappa\right) = \frac{\sum_{i=1}^{N} d(y_i, \hat{y}_i) \cdot \mathbb{1}[\psi(\mathbf{x_i}) < \kappa]}{\sum_{i=1}^{N} \mathbb{1}[\psi(\mathbf{x_i}) < \kappa]} \tag{6.2}$$

where $\{y\}_{i=1}^N$ and $\{\hat{y}\}_{i=1}^N$ are the N real and predicted values respectively, the $\mathbb{1}[p]$ is the indicator function that verifies the condition p, the κ is the uncertainty score value selected as threshold and the d is any desired error metric to be evaluated. Consequently, depending on the selected error metric this affects the

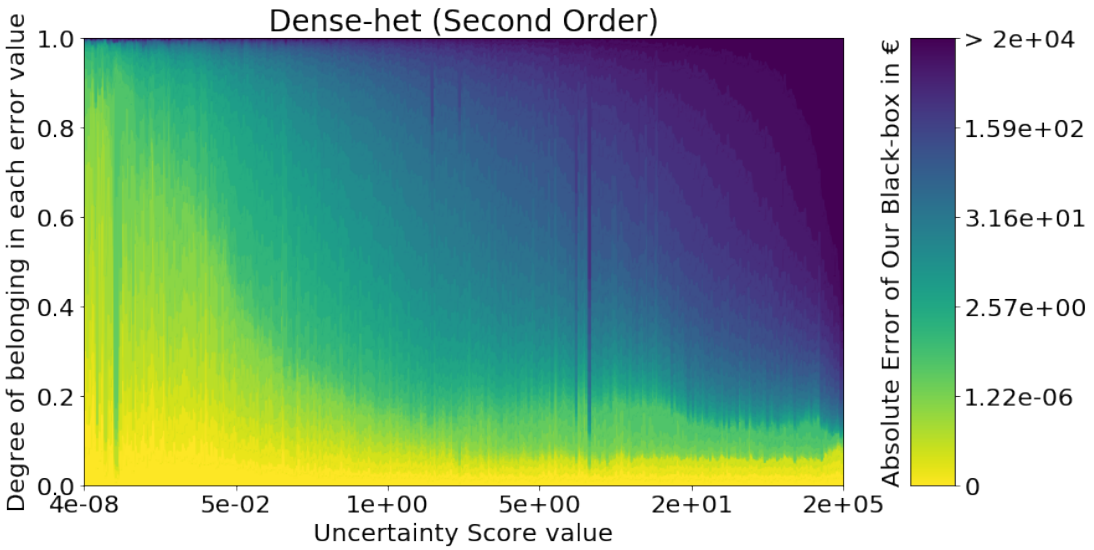


Figure 6.10: Density plot between the uncertainty score value and their corresponding Average Absolute Error produced by the Our *In Production* black-box when it is predicting in the Financial forecasting problem.

shape of this curve. In Figure 6.9, we can see how this plot is obtained considering a MAE. An extension of this plot was shown in Figure 4.2 of Section 4.2.4.3.

6.2.2 Error-retention density plot for score inaccuracies

The goal of the Error versus Retention Density (ERD) plot is to show an overview of how errors are distributed when each predicted uncertainty score is considered. Differently than ERC, each horizontal point of the ERD plot do not corresponds only to a summary statistic (e.g. conditional mean) of all the errors with a certain uncertainty score value but the ERD visualizes the degree of belonging in each error value based on a common legend of colours. This legend is defined considering all the absolute error values of the predictive model versus the real value to be predicted of a test set. For instance, in Figure 6.10 we can see that lower uncertainty score values (the first horizontal values) have small part of higher errors (blueish color) but this is increased as more higher is the uncertainty score value.

Similarly to ERC, the ERD plot changes depending on the selected error

metric (e.g alternately to the MAE, it can be considered the MSE, MAPE, MPE, RMSPE as it was introduced in Section 4.2.4.3).

6.2.3 Calibration curve to verify probabilities

Reminder 5 (The concept of calibration) *Nowadays, humans rely on computer produced scores to make important decisions in our lives. Whether a medical operation should be performed, the score for the granting or not a credit or even the score of predicting rainy days, these numbers play an important role in how we make decisions. However, what if these scores are not actually accurate? Do the model's predictions provides us the probability of rain or a good forecast to maximize the evaluated loss in the optimization process? This is where calibration comes in.*
Calibration is the process of adjusting probability scores so that they more accurately reflect its probabilistic reality. For example, consider a weather forecaster that reports a score of 0.7 *out of* 1 *points to rain tomorrow. This might seem like a good score, but without calibration it is impossible to know how good it actually is because it do not has an interpretable meaning. With calibration, however, we can transform this score into probabilities and use them more effectively in decision making. In this case, the probability that we should have taken an umbrella would be* 70%*, while the probability that it would not be necessary would be* 30%*.*

In the present work, we are considering a single dimensional response variable. Consequently, conditional quantiles that are defined in such space correspond to a real value or threshold that divides the response variable distribution into two parts with a certain probability below and above. This probability is directly the goal of a calibrated quantile. Thus, a properly estimated distribution would have their quantiles calibrated, which ensures the proper interpretation of the forescasted probability.

In this context, a calibration curve will be the representation of how well probabilities are forecasted compare to the empirically measured probabilities over a new set of data, which is not used for training or validation steps following

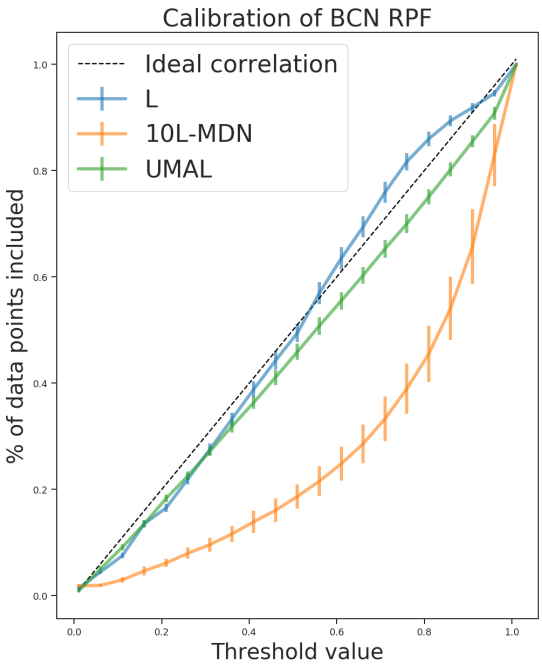


Figure 6.11: An example of a set of calibration curves.

Section 2.2 notation.

In Figure 6.11, we can see the curve that is computed as follows,

Definition 15 (Calibration curve) *Given a certain quantile $\theta \in [0, 1]$ and a set of data $(X, Y) = \{(x_i, y_i)\}_{i=1}^{N}$ that corresponds to samples of a certain distribution $p(X, Y)$ (following notation introduced in Chapter 2), now, we can be in two situations:*

- *Our predictive model forecasts a conditional probability (e.g. by predicting the conditional parameters of a certain parametric distribution). First, we compute the θ quantile of the predicted conditional probability. Secondly, we evaluate if the real value to be predicted is above or bellow the computed quantile. Finally, we repeat this process for all the test set points and consider the tan-per-cent of times the real value is equal or below the computed quantile as the vertical value that defines the calibration curve for the θ point.*
- *Our predictive model forecasts a conditional θ-quantile: For all the points*

inside the test set, compute the number of times the predicted quantile is above of the real value to be predicted and directly use this value as the vertical value that defines the calibration curve for the θ point.

The closest the computed calibration curve is to the real "ideal correlation" curve, the more similar to the empirical probability results is the forecast.

Once the calibration performance is obtained - and its deficiencies are detected - a re-calibration step can be carried out by using techniques such as temperature scaling Platt et al. (1999), which was shown by Guo et al. (2017) to lead to well-calibrated predictions on i.i.d. test sets. Importantly, this post-hoc calibration process can fail under even a mild shift in the data distribution (Ovadia et al., 2019), which is a widely prevalent situation in real-world applications and this encourage the research on the corresponding source of uncertainty (as discussed in Section 2.1).

6.3 Quantitative rating of the reported uncertainty

Differently than graphical evaluations that are important to quickly infer the prediction correctness, numerical evaluations are also important because they allow for more precise information regarding the model performance. This level of precision is often necessary in business contexts (Brando et al., 2018a) where small differences in certain points can have big impacts on profits and losses. Furthermore, by focusing in certain values, the comparison between several models performance can be simplified as we can see in Table 2.1.

Following we will present several numerical methods used and designed during the previous works (Brando et al., 2022b, 2018a, 2019b, 2020) for the numerical evaluation of the predicted uncertainty.

6.3.1 Ordering score index for evaluating sorting quality

Once we have defined the ERC, which qualitatively evaluates the sorting quality using the predicted uncertainty score, the goal of the ordering score index is to quantitatively summarize the difference between the area under the perfect oracle

curve of the ERC, here denoted as ψ_o, versus the evaluated system, denoted as ψ_k, as follows

$$\mathcal{S}(\psi_k) = 100\left(1 - \frac{A(\psi_k) - A(\psi_o)}{\delta_k - A(\psi_o)}\right), \tag{6.3}$$

where $\delta_k = \left[\left(\frac{1}{N}\sum_{i=1}^{N}|\psi_k(\mathbf{z}_i) - y_i|\right), \underbrace{\ldots}_{\text{Repeat } N \text{ times}}\right]$ is a vector with N times the same value, which is the generic mean absolute value of all the evaluated points. This vector corresponds to a theoretically random sorting process performance. On the other hand, $A(v)$ denotes the area of the ERC of the function v. On the whole, as it was introduced in Section 4.2.4.3, the ordering score index is near 100 when it is close to the perfect order and take lower values as more different it is, arriving to zero or even negative values due to reasons related to stochastic noise.

6.3.2 Quantile regression as a generic quantile metric

Importantly, the quantile regression formula,

$$\mathcal{L}(X, Y) = \mathbb{E}\left[\int_0^1 \Big(Y - \Phi_w(\tau, X)\Big) \cdot \Big(\tau - \mathbb{1}[Y < \Phi_w(\tau, X)]\Big)\, d\tau\right], \tag{6.4}$$

can be used as a generic evaluation metric to see how all the quantiles are properly distributed, unlike the likelihood evaluation as we saw in Section 3.3. However, we should take care that this metric do not considers if the predicted quantiles cross or not but only if they, individually, are close to the real quantile, as discussed in Chapter 5 introducing the crossing quantile phenomenon.

6.3.3 Calibration area to verify the trust on probabilities

Finally, the calibration curve can have an analogous comparison from the predicted calibration curve and the perfectly calibrated curve shown in Figure 6.12. As we show in that figure, the area between these two curves can be used as a sum-

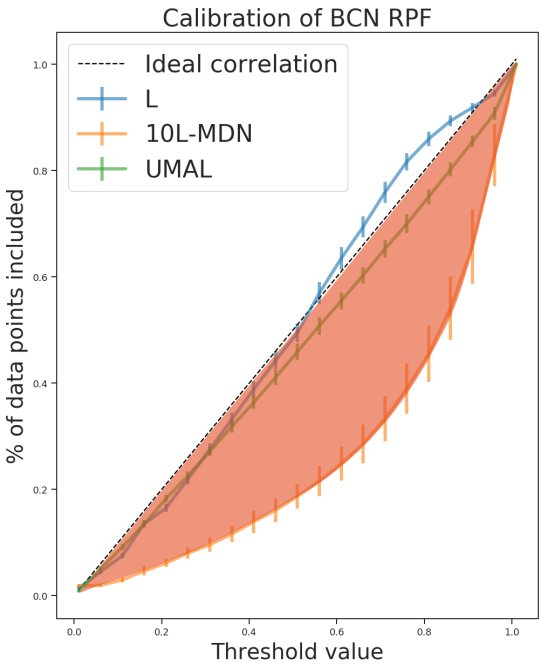


Figure 6.12: The calibration area corresponds to the area value of the reddish highlighted zone which is bounded by the ideal calibration curve and the produced by the evaluated model.

mary of how well calibrated are the forecasted quantiles, generically. Importantly, this calibration curve can be obtained for any probabilistic model (including parametric or distributional-free models).

6.4 Conclusions

This chapter has highlighted the importance of properly representing and measuring the uncertainty in any forecasting scenario using the uncertainty modelling techniques proposed in the present thesis.

Reporting accurately the uncertainty is strictly connected to the visualization field, as one of the more challenging open-problems, and its specially crucial for any further decision making decision process to provide trustable forecasting systems.

As it was described in Section 6.1, the source of the uncertainty characterize

the procedure to visualize and measure itself. In this section, a disentanglement between all the presented uncertainty sources - introduced in Chapter 2 - were proposed by linking several literature methods regarding for the presented uncertainty types with the presented in the current thesis.

At the same time, the proposed approach launches the possibility to combine different uncertainty sources modelling techniques to provide an integrated procedure to visualize and measure the uncertainty from all their sources. The proposed approach conceives two ways to model uncertainty: from a discrete viewpoint or from a continuous one.

Additionally, the present chapter remarks the importance of qualitative metrics, as a manner to visually show the performance of the model and obtain a range of possibilities to overstretch and detect critical modelling failures of the uncertainty such as the proper calibration of the reported probabilities. Furthermore, these metrics were enriched with quantitative metrics, which are in charge of summarize the previous visualizations by providing uncertainty quality indices. Importantly, these indices constitute a suitable characteristic to determine which models are more reliable in terms of uncertainty modelling.

Chapter 7

Conclusions

Increasingly, more and more human decisions are based on the outputs of forecasting systems reaching the point where certain decisions are fully made by autonomous machines. Crucially, many of these decisions are based on their best guess at what will happen and the associated risks are implicitly neglected. However, by only considering the expected value of an event, the autonomous system may be overlooking important factors that can affect the outcome, which can be critical in certain scenarios. This is the initial motivation of the presented thesis.

In particular, the present thesis has had several parts: First of all, in Chapter 2, it has been motivated why it is important to model not only the predictions in a forecasting problem but also to take into account the uncertainty sources. As it is shown there, this is especially critical in real-world problems, where the cost of making a mistake can be considerable higher than obtaining a generic good performance. This motivation to solve real-world problems is completely aligned with the industrial nature of the current thesis, as described in Section 2.

Once the modelling of the uncertainty has been motivated, a formal procedure has been defined in Section 2.1 to mathematically identify the different sources of uncertainty involved in a prediction problem and expose the current nomenclature disagreement that exists in the literature to refer to uncertainty types. One of the more widely tackled uncertainty type is the known as epistemic uncertainty, which is discussed and contextualized in Section 2.3. In contrast, the commonly known as aleatoric uncertainty, - which refers to the variability of options given

the same input, - has been identified as the main need in the real problems of this industrial collaboration (Brando et al., 2018a) and, therefore, from then on it became the main objective of study of the current thesis: provide reliable aleatoric uncertainty models.

Along these motivational line, in Chapter 3, different techniques have been introduced from the literature in order to be able to model the aleatoric uncertainty, as well as presenting new techniques to improve the aleatoric uncertainty modelling Brando et al. (2019a,b); Muelas et al. (2020).

Subsequently, the aleatoric uncertainty modelling helps us to identify the need of developing uncertainty modelling techniques for black box systems presented in Chapter 4. Here the concept of black box systems is introduced as point-wise predictive systems which do not models uncertainty and we do not have access to its internals but can be evaluated as mathematical functions. In this way, the objective of modeling the uncertainty of a black box system is introduced as a solution for one industrial problem we had (Brando et al., 2020), which prompts the development of new models to tackle it (Brando et al., 2022b) using quantile regression methods.

The free-distributional property of quantile regression introduces a well-known literature problem that appears when different quantiles are estimated simultaneously and they can cross. This phenomenon was the main problem that motivates the Chapter 5, providing new techniques to prevent this phenomenon (Brando et al., 2022a).

Finally, in Chapter 6, a proposal to integrally visualize all the uncertainty sources is presented. Furthermore, different metrics to evaluate the uncertainty modelling quality of the models were presented. All these results come as a conclusions summary of the research developed during these years.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org. 65

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016. 50, 85, 111

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. 81

George A Anastassiou. Multivariate hyperbolic tangent neural network approximation. *Computers & Mathematics with Applications*, 61(4):809–821, 2011. 14

Javier Antoran, James Allingham, and José Miguel Hernández-Lobato. Depth

uncertainty in neural networks. *Advances in Neural Information Processing Systems*, 33:10620–10634, 2020. 25

Norman P Archer and Shouhong Wang. Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. *Decision Sciences*, 24(1):60–75, 1993. 102

Pierre Baldi and Peter J Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26:2814–2822, 2013. 23

Chris Bishop. *Pattern Recognition and Machine Learning*, chapter 3: Linear Models for Regression. Springer, 2016. 78

Christopher M Bishop. Mixture density networks. *Technical Report NCRG/4288*, 1994a. 13

Christopher M Bishop. Mixture density networks. 1994b. 17, 18, 36, 40, 41, 42, 43, 44, 76, 97

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 17, 18

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112 (518):859–877, 2017. 19, 20

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015. 19, 21

Howard D Bondell, Brian J Reich, and Huixia Wang. Noncrossing quantile regression curve estimation. *Biometrika*, 97(4):825–838, 2010. 122

Georges-Pierre Bonneau, Hans-Christian Hege, Chris R Johnson, Manuel M Oliveira, Kristin Potter, Penny Rheingans, and Thomas Schultz. Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization*, pages 3–27. Springer, 2014. 134

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 18, 50

A. Brando, J. Gimeno, Jose A. Rodríguez-Serrano, and J. Vitrià. Deep non-crossing quantiles through the partial derivative. *AISTATS*, 2022a. Proceedings of the 25th International Conference on Artificial Intelligence and Statistics. 5, 101, 104, 108, 114, 122, 131, 153, 176

A. Brando, J. Gimeno, Jose A. Rodríguez-Serrano, and J. Vitrià. The chebyshev network: Modelling aleatoric uncertainty through the quantile regression derivative. *arXiv*, 2022b. Work under evaluation. 5, 75, 112, 114, 131, 143, 148, 153

Axel Brando. Mixture density networks for distribution and uncertainty estimation, 2017. URL https://github.com/axelbrando/Mixture-Density-Networks-for-distribution-and-uncertainty-estimation. GitHub repository with a collection of Jupyter notebooks intended to solve a lot of problems related to MDN. 37, 42, 43, 44, 173

Axel Brando and Joan Llop. Summary video of the modelling heterogeneous distributions with an uncountable mixture of asymmetric laplacians article, 2019. URL https://vimeo.com/369179175. Vimeo's video in https://vimeo.com/369179175. 28, 172

Axel Brando, Jose A Rodríguez-Serrano, Mauricio Ciprian, Roberto Maestre, and Jordi Vitrià. Uncertainty modelling in deep networks: Forecasting short and noisy series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 325–340. Springer, 2018a. 4, 9, 22, 29, 30, 31, 35, 37, 38, 39, 43, 78, 82, 87, 98, 118, 143, 148, 153

Axel Brando, Jose A Rodríguez-Serrano, and Jordi Vitria. How to model all uncertainty sources using deep learning. https://w.tame.events/e/3177403, 2018b. Accessed: 2022-4-27. 4

Axel Brando, Jose A Rodríguez-Serrano, and Jordi Vitria. How to model aleatoric uncertainty using umal. https://w.tame.events/e/89635027, 2018c. Accessed: 2022-4-27. 4

Axel Brando, Jose A Rodríguez-Serrano, and Jordi Vitria. Large-scale machine learning for financial recommender systems. https://bgsmath.cat/event/maths-industry-4-0/, 2018d. Accessed: 2022-4-27. 4

Axel Brando, Jose A Rodríguez-Serrano, and Jordi Vitria. Bbva's expenses forecasting using deep learning. https://shorturl.at/asLQ6, 2018e. Accessed: 2022-4-27. 4, 9

Axel Brando, Jose A Rodríguez-Serrano, and Jordi Vitrià. Detecting unusual expense categories for financial advice apps. In *KDD*, pages Anchorage, USA. Workshop on Anomaly Detection in Finance, 2019a. 4, 9, 153

Axel Brando, Jose A Rodríguez-Serrano, Jordi Vitria, and Alberto Rubio. Modelling heterogeneous distributions with an uncountable mixture of asymmetric laplacians. *Neural Information Processing Systems (NeurIPS), Vancouver, Canada*, 2019b. 5, 28, 29, 37, 39, 44, 45, 46, 48, 49, 50, 56, 57, 62, 65, 68, 97, 113, 115, 117, 143, 148, 153, 172, 173, 175, 176

Axel Brando, Jose A Rodríguez-Serrano, Jordi Vitria, and Alberto Rubio. Github's umal repository. *Website [Internet].[cited 26 May 2020]. Available: https://github.com/BBVA/UMAL*, 2019c. 28

Axel Brando, Damia Torres, Jose A Rodriguez-Serrano, and Jordi Vitria. Building uncertainty models on top of black-box predictive apis. *IEEE Access*, 8:121344–121356, 2020. 5, 75, 76, 143, 148, 153, 174

Alex J Cannon. Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes. *Stochastic environmental research and risk assessment*, 32(11):3207–3225, 2018. 126

Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014. 17

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. 113

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 15

Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2017. 13

François Chollet et al. Keras (2015), 2019. 65, 85, 111

Mauricio Ciprian, Leonardo Baldassini, Luis Peinado, Teresa Correas, Roberto Maestre, Jose A Rodríguez-Serrano, Oriol Pujol, and Jordi Vitria. Evaluating uncertainty scores for deep regression networks in financial short time series forecasting. Workshop on Machine Learning for Spatiotemporal Forecasting, 2016. NIPS. 9

C. W. Clenshaw. A note on the summation of Chebyshev series. *Math. Tables Aids Comput.*, 9:118–120, 1955. ISSN 0891-6837. 105

C. W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numer. Math.*, 2:197–205, 1960. ISSN 0029-599X. doi: 10.1007/BF01386223. URL https://doi.org/10.1007/BF01386223. 102

Murray Cox. Inside airbnb: adding data to the debate. *Inside Airbnb [Internet].[cited 16 May 2019]. Available: http://insideairbnb.com*, 2019. 28, 66

Robert Culkin and Sanjiv R Das. Machine learning in finance: the case of deep learning for option pricing. *Journal of Investment Management*, 15(4):92–100, 2017. 6

Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1992. 72

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In *International Conference on Machine Learning*, pages 1104–1113, 2018a. 46, 57

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b. 53

Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010. 102

Cesar De Pablo, Axel Brando, Jose A Rodríguez-Serrano, and Jordi Vitria. Poster about uncertainty modelling in deep networks: Forecasting short and noisy series. MLSS Madrid 2018, 2018. Accessed: 2022-4-27. 4

Jan De Spiegeleer, Dilip B Madan, Sofie Reyners, and Wim Schoutens. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18(10):1635–1643, 2018. 6

Marc Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490, 2015. 81

Rahul C Deo. Machine learning in medicine. *Circulation*, 132(20):1920–1930, 2015. 6

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, 2009. 8

Nicki S Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 6326–6336, 2019. 43

Alberto Diez-Olivan, Javier Del Ser, Diego Galar, and Basilio Sierra. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0. *Information Fusion*, 50:92–111, 2019. 6

David Duvenaud Dougal Maclaurin and Matt Johnson. Autograd: Efficiently computes derivatives of numpy code. https://github.com/HIPS/autograd, 2016. Accessed: 2022-4-27. 18

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. 113

Oliver Duerr, Beate Sick, and Elvis Murina. *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability*. Manning Publications, 2020. 135

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 68

Tom Fomby. Scoring measures for prediction problems. *Department of Economics, Southern Methodist University, Dallas, TX*, 2008. 87

Andrew YK Foong, David R Burt, Yingzhen Li, and Richard E Turner. On the expressiveness of approximate inference in bayesian neural networks. *Advances in Neural Information Processing Systems*, 2020. 25

Ana Fred, Maria De Marsico, and Gabriella Sanniti di Baja. *Pattern Recognition Applications and Methods: 5th International Conference, ICPRAM 2016, Rome, Italy, February 24-26, 2016, Revised Selected Papers*, volume 10163. Springer, 2017. 46

Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969. 14

Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988. 15

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059, 2016. 19, 23, 116

Marco Geraci. Mixed-effects models using the normal and the laplace distributions: A 2×2 convolution scheme for applied research. *arXiv preprint arXiv:1712.07216*, 2017. 38

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Rectifier and softplus activation functions. the second one is a smooth version of the first. *IEEE Transactions on Systems Science and Cybernetics*, 2011a. 14

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323, 2011b. 124

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016a. http://www.deeplearningbook.org. 15, 18

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016b. 13, 14

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 148

Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander Van Esbroeck. Monotonic calibrated interpolated look-up tables. *The Journal of Machine Learning Research*, 17(1):3790–3836, 2016. 102

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. volume 405, pages 947–951. Nature Publishing Group, 2000. 14

Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995. 14

Lingxin Hao, Daniel Q Naiman, and Daniel Q Naiman. *Quantile regression*. Number 149. Sage, 2007. 80

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, pages 1861–1869, 2015a. 68, 131, 132

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015b. 115, 116

Alexander Hevelke and Julian Nida-Rümelin. Responsibility for crashes of autonomous vehicles: an ethical analysis. *Science and engineering ethics*, 21(3): 619–630, 2015. 6

Montse Hidalgo. Article in the retina section of the newspaper el país. https://elpais.com/retina/2019/12/20/innovacion/1576838697_328758.html, 2018. Accessed: 2022-4-27. 5

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 15, 66

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 15

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014. Section 2.3. 83

Boehringer Ingelheim. Kaggle Challgenge: Predicting a biological response, 2012. URL https://www.kaggle.com/c/bioresponse/rules. 83

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 43

Sreenivasa Rao Jammalamadaka and Tomasz J Kozubowski. New families of wrapped distributions for modeling skew circular data. *Communications in Statistics-Theory and Methods*, 33(9):2059–2074, 2004. 58

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 6

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. 8, 36, 97

Mohammad E Khan, Guillaume Bouchard, Kevin P Murphy, and Benjamin M Marlin. Variational bounds for mixed-data factor analysis. In *Advances in Neural Information Processing Systems*, pages 1108–1116, 2010. 47

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014. 22

Daniel Klotz, Frederik Kratzert, Martin Gauch, Alden Keefe Sampson, Johannes Brandstetter, Günter Klambauer, Sepp Hochreiter, and Grey Nearing. Uncertainty estimation with deep learning for rainfall–runoff modelling. *Hydrology and Earth System Sciences Discussions*, pages 1–32, 2021. 62

Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001. 52, 76, 80, 101, 122

Roger Koenker, Victor Chernozhukov, Xuming He, and Limin Peng. *Handbook of Quantile Regression*. CRC press, 2017. 121, 123

Tomasz J Kozubowski and Krzysztof Podgórski. A multivariate and asymmetric generalization of laplace distribution. *Computational Statistics*, 15(4):531–540, 2000. 58

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016. 24

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances*

in Neural Information Processing Systems, pages 6402–6413, 2017. 19, 68, 97, 116

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 15

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015. 13

Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. 113

Patrick Lin. Why ethics matters for autonomous cars. In *Autonomous driving*, pages 69–85. Springer, Berlin, Heidelberg, 2016. 6

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 68

Roberto Maestre, Rodríguez-Serrano, Jordi Nin, Axel Brando, Irene Unceta, Alberto Hernandez, Jose Carmona, Lorenzo Caggioni, and Stefan Hosein. Whitepaper google cloud and bbva. https://www.bbvadata.com/white_papers/advanced_ai.pdf, 2018. Accessed: 2022-4-27. 4

H. Majidian. On the decay rate of Chebyshev coefficients. *Appl. Numer. Math.*, 113:44–53, 2017. ISSN 0168-9274. doi: 10.1016/j.apnum.2016.11.004. URL https://doi.org/10.1016/j.apnum.2016.11.004. 128

Amy Maxmen. Self-driving car dilemmas reveal that moral choices are not universal. *Nature*, 562(7728):469–469, 2018. 6

K. E. Basford McLachlan G. J. Mixture models: Inference and applications to clustering. 1988. 40, 41

Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of Machine Learning Research*, 7(6), 2006. 53

Jairo Mejia, Roberto Maestre, Rodríguez-Serrano, Jordi Nin, Axel Brando, Irene Unceta, Alberto Hernandez, Jose Carmona, Lorenzo Caggioni, and Stefan Hosein. Delivering advanced artificial intelligence in the banking industry. https://shorturl.at/vzBXY, 2018. Accessed: 2022-4-27. 4, 29

José Mena, Axel Brando, Oriol Pujol, and Jordi Vitrià. Uncertainty estimation for black-box classification models: a use case for sentiment analysis. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 29–40. Springer, 2019. 4

Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016. 76

Rhiannon Michelmore, Marta Kwiatkowska, and Yarin Gal. Evaluating uncertainty quantification in end-to-end autonomous driving control. *arXiv preprint arXiv:1811.06817*, 2018. 6

Robert Moore and John DeNero. L1 and l2 regularization for multiclass hinge loss models. In *Symposium on machine learning in speech and language processing*, 2011. 16

David Muelas, Luis Peinado, Axel Brando, and Rodríguez-Serrano. Detection of balance anomalies with quantile regression: the power of non-symmetry. In *KDD*, pages Anchorage, USA. Workshop on Anomaly Detection in Finance, 2020. 5, 153

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 13

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107: 107501, 2020. 43

Frank Nielsen and Ke Sun. Guaranteed bounds on the kullback–leibler divergence of univariate mixtures. *IEEE Signal Processing Letters*, 23(11):1543–1546, 2016. 47

Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL http://neuralnetworksanddeeplearning.com. 18

Frank Noé, Gianni De Fabritiis, and Cecilia Clementi. Machine learning for protein folding and dynamics. *Current opinion in structural biology*, 60:77–84, 2020. 6

Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992. 43

Emanuele Olivetti. kaggle_pbr. https://github.com/emanuele/kaggle_pbr, 2012. 83, 84

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019. 148

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 50

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011. 76

Francisco C Pereira, Constantinos Antoniou, Joan Aguilar Fargas, and Moshe Ben-Akiva. A metamodel for estimating error bounds in real-time traffic prediction systems. *IEEE Transactions on Intelligent Transportation Systems*, 15 (3):1310–1322, 2014. 80

Hossein Pishro-Nik. Mean squared error (mse). *Website [Internet].[cited 19 Sep 2020]. Available: https://shorturl.at/tFGY4*, 2020. 17

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999. 148

Robert Gilmore Pontius, Olufunmilayo Thontteh, and Hao Chen. Components of information for multiple resolution comparison between maps that share a real variable. *Environmental and Ecological Statistics*, 15(2):111–142, 2008. 88

Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2931–2940. IEEE. 24

Alvin Rajkomar, Jeffrey Dean, and Isaac Kohane. Machine learning in medicine. *New England Journal of Medicine*, 380(14):1347–1358, 2019. 6

Carl Edward Rasmussen. A practical monte carlo implementation of bayesian learning. In *Advances in Neural Information Processing Systems*, pages 598–604, 1996. 46

Carl Edward Rasmussen et al. The infinite gaussian mixture model. In *NIPS*, volume 12, pages 554–560. Citeseer, 1999. 46

REE. Real-time demand and generation. Jun. 11, 2020. [Online]., 2020. Available: https://demanda.ree.es/visiona/peninsula/demanda/total. 82

H Rosling. Ted talk: The best stats you've ever seen, 2006. 134

Sebastian Ruder. An overview of gradient descent optimization algorithms. http://sebastianruder.com/optimizing-gradient-descent/, 2016. Accessed: 2022-4-27. 18

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1 (5):206–215, 2019. 72

Mark Schmidt, Glenn Fung, and Rmer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *European Conference on Machine Learning*, pages 286–297. Springer, 2007. 16

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*, pages 2503–2511, 2015. 72

Joseph Sill. Monotonic networks. In *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 661–667, 1998. 102, 124

Neural Smithing. Supervised learning in feedforward artificial neural networks, 1999. 15

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 23

Jack Stilgoe. Machine learning, social learning and the governance of self-driving cars. *Social studies of science*, 48(1):25–56, 2018. 6

Natasa Tagasovska and David Lopez-Paz. Frequentist uncertainty estimates for deep learning. *Bayesian Deep Learning workshop NeurIPS*, 2018. 46, 122, 123

Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. In *Advances in Neural Information Processing Systems*, pages 6417–6428, 2019. 24, 57, 97, 116

HR Thompson. Distribution of distance to nth neighbour in a population of randomly distributed individuals. *Ecology*, 37(2):391–394, 1956. 81

M. K. Titsias and F. J. R. Ruiz. Unbiased implicit variational inference. In *Artificial Intelligence and Statistics*, 2019. 46

L. N. Trefethen. Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Rev.*, 50(1):67–87, 2008. ISSN 0036-1445. doi: 10.1137/060659831. URL https://doi.org/10.1137/060659831. 128

Harinandan Tunga, Rounak Saha, and Samarjit Kar. A method of fully autonomous driving in self-driving cars based on machine learning and deep learning. *Intelligent Multi-modal Data Processing*, pages 131–156, 2021. 6

Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016. 72

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Simple and scalable epistemic uncertainty estimation using a single deep deterministic neural network. 2020. 24

Math Vault. List of probability and statistics symbols. *Website [Internet].[cited 26 May 2020]. Available: https://mathvault.ca/hub/higher-math/math-symbols/probability-statistics-symbols/*, 2020. 17

Yaoshu Wang, Chuan Xiao, Jianbin Qin, Xin Cao, Yifang Sun, Wei Wang, and Makoto Onizuka. Monotonic cardinality estimation of similarity selection: A deep learning approach. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1197–1212, 2020. 102

A. Wehenkel and G. Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*, 2019. 102, 125

Claus O Wilke. *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media, 2019. 134

Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005. 17

Edwin Bidwell Wilson. First and second laws of error. *Journal of the American Statistical Association*, 18(143):841–851, 1923. 38

Bingzhe Wu, Zhichao Liu, Zhihang Yuan, Guangyu Sun, and Charles Wu. Reducing overfitting in deep convolutional neural networks using redundancy regularizer. In *International Conference on Artificial Neural Networks*, pages 49–55. Springer, 2017. 16

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya R Gupta. Deep lattice networks and partial monotonic functions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2985–2993, 2017. 102

Keming Yu and Rana A Moyeed. Bayesian quantile regression. *Statistics & Probability Letters*, 54(4):437–447, 2001. 59

Wenjie Zhang, Hao Quan, and Dipti Srinivasan. Parallel and reliable probabilistic load forecasting via quantile regression forest and quantile determination. *Energy*, 160:810–819, 2018. 53

Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–4. IEEE, 2015. 36, 65

Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2):787–794, 2020. 15

List of Figures