University of Brasília
Institute of Exact Sciences
Department of Statistics

Master's Dissertation

Quantile-based Recalibration of Artificial Neural Networks

by

Ricardo Torres

Brasília, February 2023

Quantile-based Recalibration of Artificial Neural Networks

by

Ricardo Torres

Dissertation submitted to the Departament of Statistics at the University of Brasília, as part of the requirements required to obtain the Master Degree in Statistics.

Supervisor: Dr. Guilherme Souza Rodrigues

Brasília, February 2023

If you wish to make an apple pie from scratch, you must first invent the Universe.

(Carl Sagan)

Acknowledgments

Thanks to the faculty of PPGEST/UnB, my supervisor Dr. Guilherme Souza Rodrigues, all professors and colleagues who contributed to this work in any way. Also thanks to Prof. Thais Carvalho Valadares Rodrigues and Prof. Rafael Izbicki for accepting the invitation to be part of this work's evaluation committee.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo Expandido

RECALIBRAÇÃO DE REDES NEURAIS ARTIFICIAIS BASEADA EM QUANTIS

Redes neurais artificiais são ferramentas poderosas, amplamente utilizadas atualmente para previsão e modelagem de dados. Embora venham se tornando cada vez mais poderosas, melhorias recentes vêm impactando negativamente sua calibração em prol de maior capacidade de previsão, tornando sua real confiança difícil de avaliar (Guo et al., 2017) – ou seja, a acurácia de suas previsões não corresponde à sua estimativa de confiança. Para ilustrar, considere um evento que ocorre 90% das vezes em um experimento aleatório infinito. Uma previsão para tal evento, realizada por um modelo bem calibrado, deveria então cair dentro de um intervalo de 90% de credibilidade cerca de 90% das vezes.

Para mitigar este problema, propomos um método de pós-processamento baseado em quantis para recalibração de redes neurais. O método explora a informação contida nas probabilidades acumuladas, $p_i = \hat{F}_i(y_i|\mathbf{x}_i)$, onde $\mathbf{x}_i$ e y_i representam, respectivamente, o i-ésiamo vetor de variáveis explicativas e a i-ésiama variável dependente de um conjunto de recalibração. $\hat{F}_i$ denota a distribuição preditiva acumulada da rede neural. Por meio do método proposto, é possível obter amostras de *Monte Carlo* da distribuição preditiva recalibrada.

Foi observado por Guo et al. (2017) que o aumento de capacidade das redes neurais e a falta de regularização estão intimamente relacionadas à falta de calibração, que, por sua vez,

se relaciona com o sobreajuste da log-verossimilhança durante o treinamento. Mukhoti et al. (2020) observaram ainda que o aumento da confiança da rede em suas previsões incorretas para amostras de teste classificadas incorretamente é um dos principais sintomas de descalibração e propõem o uso da função de perda focal no lugar da função de entropia cruzada para contornar o problema. Várias técnicas de pós-processamento também foram introduzidas na tentativa de corrigir o problema da descalibração, como o método *temperature scaling*, proposto por Guo et al. (2017). Kuleshov, Fenner, and Ermon (2018) propuseram que seria possível obter um modelo calibrado a partir do ajuste de um algorítmo de regressão ao conjunto de recalibração composto pelas probabilidades acumuladas e pelas estimativas empíricas dos respectivos quantis de cada previsão feita pelo modelo descalibrado. Kumar, Liang, and Ma (2019), afirmando que os métodos anteriores são menos calibrados do que o reportado, introduzem o *scaling-binning calibrator*, reduzindo localmente a variância das previsões por meio do ajuste de uma função paramétrica.

O método de recalibração proposto neste trabalho utiliza o fato de que, para cada $i = 1, \ldots, n$ em uma amostra com n elementos, a probabilidade acumulada estimada da i-ésima previsão do modelo deve ter distribuição aproximadamente Uniforme(0, 1), possibilitando que, global ou localmente, a informação do histograma dessas medidas seja utilizada para detectar algum padrão de viés. Dessa forma, a partir da distribuição preditiva obtida pelo modelo, é possível obter amostras recalibradas usando um simples algoritmo de simulação estocástica que explora viéses locais do preditor, considerando uma vizinhança (no espaço de uma dada camada da rede neural) da amostra a ser prevista.

Neste trabalho são apresentados dois exemplos ilustrativos da aplicação e do desempenho da recalibração proposta. O primeiro considera um simples modelo heteroscedástico Gaussiano com apenas uma covariável, ao qual foram ajustados um modelo linear homoscedástico e uma rede neural simples. Neste exemplo, são comparados os efeitos da recalibração local em relação à global. No segundo exemplo, considera-se uma variável resposta com distribuição Gama com a superfície da média dada pela função de Rosenbrock, comumente utilizada como teste para otimização de algoritmos, ao qual é ajustada uma rede neural com camadas de *dropout* intermediárias. Neste exemplo, a incerteza da rede é estimada de duas formas comumente utilizadas na literatura: a primera por meio da *weight scaling inference rule* e a segunda pela técnica de *Monte Carlo Dropout*, introduzida por Gal and Ghahramani (2016), ficando claro neste exemplo o ganho de estimação da distribuição preditiva do primeiro método em relação ao segundo. Em ambos os exemplos, os modelos recalibrados obtiveram performance preditiva superior aos modelos descalibrados.

Seguindo os exemplos ilustrativos, é apresentado um estudo de simulação realizado para investigar os efeito do método em diferentes cenários, incluindo tamanho da rede, camada de recalibração, tamanho das amostras disponíveis para treinamento e a combinação de diferentes parâmetros de recalibração. O estudo é feito com base em um modelo altamente não-linear, Gaussiano, apresentado em Tran et al. (2020), no qual apenas dez de vinte covariáveis estão probabilisticamente relacionadas à variável resposta. Neste estudo, é possível identificar fatores importantes que afetam o uso de memória, o tempo de treinamento e de inferência e o desempenho preditivo dos modelos recalibrados.

Por fim, o método proposto é testado em um conjunto de dados reais. Nessa seção propõe-se uma análise do preço de cerca de 53.940 diamantes a partir de suas caracteristicas físicas individuais. Assumindo distribuição condicional dos preços $Y|X \sim Gama(\alpha, \mu)$, ajustou-se aos dados um modelo linear generalizado Gama com função de ligação logarítmica e uma rede neural com função perda definida pela log-verossimilhança negativa da distribuição Gama. A recalibração de ambos modelos foi capaz de aumentar a precisão das previsões e melhorar as métricas intervalares.

De modo geral, a recalibração afetou positivamente as métricas de desempenho preditiva analisadas e produziu uma estimativa mais precisa da incerteza do modelo, obtendo, assim, uma melhor aproximação do processo gerador de dados. Embora hajam trabalhos com objetivos similares, as soluções apresentadas na literatura tendem a considerar as previsões obtidas na saída da rede e realizar a calibração de forma global. O método de recalibração proposto por

este trabalho generaliza esse conceito, permitindo a calibração local em qualquer camada intermediária da rede. Por si só, essa diferença metodológica abre novas possibilidades de acesso das representações dos dados produzidas pela rede, além de permitir a correção de viéses específicos de cada região. Ainda, é possível que a implementação conjunta com outras técnicas, como a proposta por Yu et al. (2021), produza resultados ainda melhores permitindo, por exemplo, uma melhor aproximação da forma das distribuições.

Palavras-chave: Calibração. Quantificação de incerteza. Intervalo de confiança. Cobertura.

Abstract

Artificial neural networks (ANN) are powerful tools for prediction and data modeling. Although they are becoming ever more powerful, modern improvements have compromised their calibration in favor of enhanced prediction accuracy, thus making their true confidence harder to assess. To address this problem, we propose a new post-processing quantile-based method of recalibration for ANN. To illustrate the method's mechanics we present two toy examples. In both, recalibration reduced the Mean Squared Error over the original uncalibrated models and provided a better representation of the data generative model. To further investigate the effects of the proposed recalibration procedure, we also present a simulation study comparing various parameter configurations – the recalibration successfully improved performance over the base models in all scenarios under consideration. At last, we apply the proposed method to a problem of diamond price prediction, where it was also able to improve the overall model performance.

Keywords: Calibration. Uncertainty assessment. Confidence interval. Coverage.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Loosely inspired by the biological brain, artificial neural networks are powerful tools for prediction and data-modeling. Their success in pattern recognition made them widely used in many different tasks ranging from classification to forecasting. Neural networks with greater representation capacity are generally flexible enough to represent all sorts of non-linear relationships in the system being modeled. Due to modern improvement techniques though, neural networks are no longer well calibrated compared with what they used to be earlier in this century (Guo et al., 2017). That is to say, their accuracy does not match their confidence estimates.

Consider an event that occurs in 90% of an infinitely-large number of trials, for example. A well-calibrated confidence (or credibility) interval should assign a probability of 0.9 to that event. More formally, given $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$, for $t = 1, \ldots, N$, independent identically distributed (i.i.d.) realizations of *X*, *Y* random variables with conditional distribution $F(Y|\mathbf{X}_t)$, and a confidence level of p, we say a neural network is calibrated if (Kuleshov, Fenner, and Ermon, 2018):

$$\frac{\sum_{t=1}^{N} \mathbb{I}_{\{y_t \leq \hat{F}_t^{-1}(p)\}}(t)}{N} \to p, \forall p \in [0, 1] \tag{1.1}$$

as $N \to \infty$, where $\hat{F}$ is the neural network's cumulative predictive distribution and $\mathbb{I}_A(t)$

denotes the indicator function which equals 1 if $y_t \in A$ and 0 otherwise. Stating it simply, if a network is well-calibrated, its predictive and empirical cumulative distribution functions should match given a large enough data set. Moreover, a sufficient condition for calibration is (Kuleshov, Fenner, and Ermon, 2018):

$$\mathbb{P}(Y \leq \hat{F}_Y^{-1}(p)) = p, \forall p \in [0,1]. \tag{1.2}$$

This notion of calibration can be easily transposed to the discrete case in a classification setting: if, instead of $\mathbb{R}$, the sample space is a finite set of labels, the neural network is calibrated if it assigns probability p to an event that occurs approximately $(100p)\%$ of the time. For example, in binary classification, where $y_t \in \{0,1\}$, the neural network is said to be calibrated if

$$\frac{\sum_{t=1}^{N} y_t \mathbb{I}_{\{\hat{F}(y_t)=p\}}(t)}{\sum_{t=1}^{N} \mathbb{I}_{\{\hat{F}(y_t)=p\}}(t)} \to p, \forall p \in [0,1] \tag{1.3}$$

as $N \to \infty$, where $\hat{F}(y_t)$ denotes $\mathbb{P}(Y_t = 1|x_t)$ estimated by the model, and a sufficient condition for calibration in this context is (Kuleshov, Fenner, and Ermon, 2018):

$$\mathbb{P}(Y = 1|\hat{F}_Y(y_t) = p) = p, \forall p \in [0,1]. \tag{1.4}$$

The existing methods of calibration are mostly based on the presented definition, especially in the context of neural networks, and are also referred to as *recalibration* by some authors (Rodrigues, Prangle, and Sisson, 2018; Prangle et al., 2014).

Guo et al. (2017) observed that increased model capacity and lack of regularization are closely related to uncalibration, which in turn appear to be linked to overfitting during training. Mukhoti et al. (2020) further observed that an increase in the network's confidence about the wrongly classified test samples is one of the key symptoms of uncalibration, and proposed the use of *focal loss* instead of *cross-entropy loss* to obtain well-calibrated models. Many post-processing techniques have also been introduced to tackle the problem of uncalibration of neural network models. Guo et al. (2017) proposed *temperature scaling*, based on *Platt scaling* (Platt et al., 1999), which divides a network's logit by a scalar $T > 0$ then performs softmax, with no effect on its accuracy. Kuleshov, Fenner, and Ermon (2018) proposed estimating the cumulative distribution from Equation 1.2 by fitting a regression algorithm to the recalibration set $\{F_t(y_t), \hat{F}(y_t)\}_{t=1}^{T}$, with $\hat{F}(y_t)$ denoting the empirical estimate of $P(y_t \leq F_t^{-1}(p))$, to get a calibrated model. Kumar, Liang, and Ma (2019) stated that popular recalibration methods like *Platt scaling* and *temperature scaling* are actually less calibrated than reported and introduced the *scaling-binning calibrator* which bins a parametric function fitting to reduce variance and ensure calibration.

Although previous works have shown significant improvement on neural networks calibration, there are issues that could improve the results even further once addressed. First of all, there is the assumption that the pattern of bias is the same throughout the covariate-space of $\mathbf{X}$, when in fact the model might behave differently across regions. For instance, a model may perform well for typical values of $\mathbf{X}$, but systematically fail to predict *Y* correctly when the covariates are located in low-density regions. Another important aspect is that recalibration methods tend to address uncalibration based on the network's output predictions, even though errors in prediction could have potentially been propagated from an intermediate layer to the output layer. The intermediate (or even the input layer) may also provide a richer space for diagnosing local biases.

To address these issues, we propose a post-processing recalibration algorithm similar to that presented by Rodrigues, Prangle, and Sisson (2018). The latest was designed to recalibrate approximate posterior distributions in the context of Approximate Bayesian Computation. However, contrary to their strategy, we propose the calibration to be done in a network's arbitrary layer instead of the original input-space, which often helps to mitigate the so-called *curse of dimensionality*. Last but not least, for computational reasons, we propose the use of an efficient search algorithm, the approximate K-Nearest Neighbours (KNN) (Arya et al., 1998), to compute the weights assigned to observations of what we call the *recalibration set*.

This dissertation is organized as follows. In Chapter 2, we present a quantile-based method for recalibration of neural networks. In Chapter 3, we apply the proposed method to a non-linear Gaussian model and a Gamma model and discuss its impact on performance. In Chapter 4, we conduct a simulation study to evaluate how the proposed method fares in a variety of scenarios, including different recalibration configurations and network models. In Chapter 5 we test our method with a real dataset example. Lastly, we summarise the results in Chapter 6.

Chapter 2

Recalibration

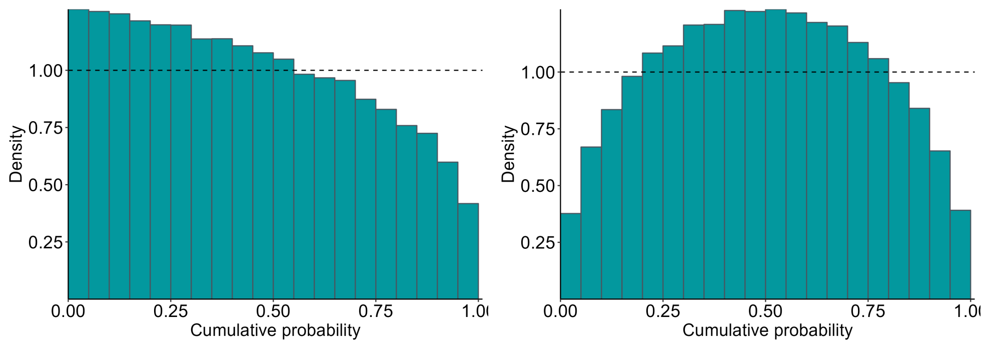
There are several model diagnostic techniques based on residuals – that is, that directly compare model predictions with observed values. Alternatively, one can evaluate model fitness by confronting observed values with their respective estimated conditional (or predictive) distributions. One long-established strategy for that is to analyse the histogram of the estimated distribution functions, each evaluated at the corresponding observation. To put it more precisely, let y_i be the i-th observation of the random response variable Y. The cumulative probability estimate of y_i is

$$p_i = \hat{F}_i(y_i|\mathbf{x}_i), \tag{2.1}$$

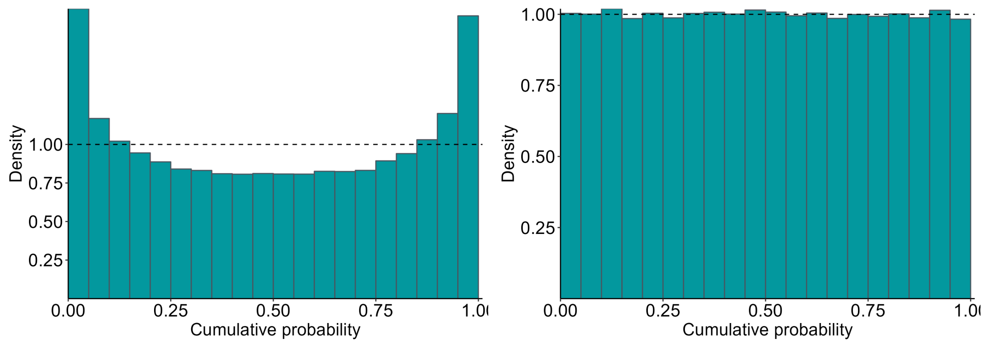
where $\hat{F}_i$ is the i-th conditional cumulative distribution function estimated by the model and $\mathbf{x}_i$ is the covariate vector of the i-th observation. This quantity is called P-value by some authors (Prangle et al., 2014; Rodrigues, Prangle, and Sisson, 2018).

Theorem 2.0.1 (Probability Integral Transformation)**.** (Bain and Engelhardt, 1987) Let $F_Y(y)$ be the cumulative distribution function (CDF) of a continuous random variable Y, then $U = F_Y(Y) \sim$ Uniform$(0, 1)$.

From Theorem 2.0.1, if the model is well specified, then $p_i \sim$ Uniform$(0, 1)$ for all i (Bain and Engelhardt, 1987). Our method consists of exploiting this coverage property to recalibrate the model predictions. If the histogram shows any particular pattern of bias, even if only locally, this piece of information can be directly used to derive a new predictive distribution which compensate, to some extent, this known, undesired, behavior. Based on the same principle, Yu et al. (2021) proposed a method for checking and adjusting approximate inference algorithms using the law of total variance and the tower property of conditional expectation. However, their approach only corrects for first and second order moments and it doesn't address errors related to the shape of an approximated distribution.



(a) A model that overestimates the median (b) A model that overestimates the variance



(c) A model that underestimates the variance (d) A possibly well specified model

Figure 2.1: Some possible patterns of cumulative probabilities histograms. (a) A model that overestimates the median of the true distribution, with most observations below the predicted $p = 0.5$. (b) A model that overestimates the variance, with most observations centered around the median of the predictive distribution. (c) A model that underestimates the variance, with most observations in the tails of the predictive distribution. (d) A model possibly well specified model, with observations equally distributed.

Figure 2.1 shows some typical cumulative probability histogram patterns and their respective interpretations. The model in Figure 2.1a overestimates the median, since most probabilities are located to the left ($p_i < 0.5$) of its respective predictive distribution median. In Figure 2.1b, since the histogram is concentrated around $p_i = 0.5$, and therefore there are less observations in the tails of the predictive distributions, the model overestimates the variance. The opposite can be seen in Figure 2.1c, where the model underestimates the variance. Figure 2.1d shows an approximately uniform histogram with no apparent bias. Notice that, while a non-uniform histogram indicates a poorly fitted model, a uniform histogram on itself does not imply proper calibration, since it does not guarantee uniformity for all i.

Algorithm 1 introduces our new post-processing method for recalibrating the predictive distributions of a probabilistic model built over a neural network. We assume an L-layers ANN has been previously fitted, with the predictive distribution defined according to the network's loss function. For example, if the Mean Squared Error (MSE) was adopted as the loss function, the neural network provides Maximum Likelihood Estimates (MLE) for the mean of a conditional homoscedastic Gaussian distribution. Other approaches, including the non-parametric use of *dropout layers* for estimating the predictive distributions, are discussed in Chapter 3.

Our recalibration method is composed of two stages: evaluating the predictive cumulative probabilities for each sample in the validation set and, then, performing the recalibration for each observation in the *new* set (a collection of samples from which the inputs are known and we want to predict the respective responses).

For each sample j we want to predict, we assign weights to the samples i, in the validation set, according to $||\mathbf{h}_{val}^{(i)} - \mathbf{h}_{new}^{(j)}||$ – that is, we ensure greater importance is given to samples which are similar (according to some pre-specified metric $||.||$), in terms of their activations h in the l-th layer. This reflects the principle that closer observations are usually more informative of the model's predictive capabilities (at that particular location). This procedure is known as *localization*. The size of the acceptance region, implied by the the number of observations considered in the recalibration, k, or by the acceptance fraction, plays a critical role in the recalibration performance. The smaller the region, the better the local bias is captured, at the

Algorithm 1 Neural Network-based Model Recalibration

```
 1: Input
      • Validation (or calibration) set, $\{\mathbf{x}_{val}^{(i)}, \mathbf{y}_{val}^{(i)}\}_{i=1}^{n}$, and new set, $\{\mathbf{x}_{new}^{(j)}\}_{j=1}^{m}$.
      • A neural network and its associated predictive distribution, $\hat{F}(\cdot \mid \mathbf{x})$.
      • A positive integer $l$ defining the network's layer where the samples are to be compared.
      • Neural network's outputs of the $l$-th layer on the validation set, $\{\mathbf{h}_{val}^{(i)}\}_{i=1}^{n}$.
      • A smoothing kernel $K_u(d)$ with scale parameter $u > 0$.
      • A positive integer $k$ defining the number of observations to be used for recalibration
        within KNN search.

    Cumulative probabilities
 2: for $i \leftarrow 1$ to $n$ do
 3:     Set $p_i = \hat{F}_i(\mathbf{y}_{val}^{(i)} | \mathbf{x}_{val}^{(i)})$.
 4: end for

    Recalibration
 5: for $j \leftarrow 1$ to $m$ do
 6:     Compute $\mathbf{h}_{new}^{(j)} = g(\mathbf{x}_{new}^{(j)})$, where $g$ represents the network's mapping from the input to
        the $l$-th layer.
 7:     Use the approximate KNN search method to identify the observations in the validation
        set for which the distances $||\mathbf{h}_{val}^{(i)} - \mathbf{h}_{new}^{(j)}||$ are within the $k$-smallest ones.
 8:     Compute the sample weight $w^{(i)} \propto K_u(||\mathbf{h}_{val}^{(i)} - \mathbf{h}_{new}^{(j)}||)$ for the $i$ indexes identified in
        Step 7 (that is, for which $w^{(i)} > 0$).
 9:     Set $\tilde{\mathbf{y}}_i^{(j)} = \hat{F}_j^{-1}(p_i | \mathbf{x}_{new}^{(j)})$.
10: end for

    Output
11: A set of weighted samples $\{(\tilde{\mathbf{y}}_i^{(j)}, w^{(i)})\}_{i=1}^{n}$ from the recalibrated predictive distribution

        $\tilde{F}_j(\cdot \mid \mathbf{x}_{new}^{(j)}),$

    for $j = 1, \ldots, m$.
```

cost of an increased error associated with the smaller number of *Monte Carlo* samples found in that region. Therefore, there is a trade-off to be considered when setting k. However, if one wishes to recalibrate a model globally, without considering local biases, it can be easily achieved by choosing a constant weighting function for all observations in the validation set, in Step 8.

The computational problem of identifying the closest samples to $\mathbf{h}_{new}^{(j)}$ in the validation set (Step 7, Algorithm 1) is, therefore, very important to the efficiency of the algorithm. This is an example of a general problem known in the computational literature as the *post office problem*. Due to the curse of dimensionality it becomes increasingly more difficult to find an efficient solution to this problem as the sample size and the number of dimensions of $\mathbf{X}$ increases. Given a set S of n points in M, a d-dimensional space, and $q \in M$ a single query point, a naive approach to the problem of finding the closest point in S to q (through linear search) takes $O(dn)$ time. While not quite memory-efficient, there are solutions to this approach that make use of a GPU upon calculating the distances to drastically speed up this process. This task becomes even more costly when trying to find the k-closest points of S to a whole query set $\mathbf{Q}$. An efficient alternative for large datasets is the use of approximate methods such as the *K-Nearest Neighbour* search. Arya et al., 1998, shows that, given $\epsilon > 0$, the k $(1+\epsilon)$-approximate nearest neighbours of q can be computed in $O(kd \log n)$ time. Recent works in the problem of identifying approximate nearest neighbors, such as the one proposed by Chen et al. (2021), can compute a solution to this problem with billion-scale data sets twice as fast as the state-of-the-art searching algorithms. In the subsequent chapters in this work, we use the Approximate Nearest Neighbor approach and the Euclidean norm. The influence of the ϵ approximation parameter in recalibration is tested, along with other parameters, in chapter 4.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ be a vector of cumulative probabilities, as defined in Step 3. The j-th recalibrated predicted value, $\hat{y}_j$, can then be obtained by taking the weighted average of the recalibrated predictive distribution's Monte Carlo samples, as given by

$$\hat{y}_j = \sum_{i=1}^{n} w^{(i)} \tilde{\mathbf{y}}_i^{(j)}, \tag{2.2}$$

where $\tilde{\mathbf{y}}_i^{(j)} = \hat{F}_j^{-1}(p_i|\mathbf{x}_{new}^{(j)})$ (see Step 9). Interval predictions and estimates of other features of the recalibrated predictive distributions (e.g. its variance) can also be easily derived. Notice that only k samples in $\{(\tilde{\mathbf{y}}_i^{(j)}, w^{(i)})\}_{i=1}^{n}$ have positive weights. Therefore, in practice, this output set can be safely reduced to store only the samples for which $w^{(i)} > 0$.

One big change that comes with the proposed method is that it can be used in any layer of a neural network, not just the output layer. Let $l = 1, \ldots, L$ denote the indexes of the network layers, then if $l = 1$ recalibration is done in the input layer and if $l = L$ recalibration is done in the output layer. Many well known methods implicitly recalibrate on $l = L$, such as Guo et al. (2017) and Kuleshov, Fenner, and Ermon (2018). Let's say we would like to recalibrate the network's predictive distribution according to the representations learned in the layer $l = L-1$. In order to perform recalibration, we then need to retrieve the data from the layers $l = L-1$ and $l = L$, for every observation in the validation set. In Chapter 4 we further investigate the effect of choosing different layers. The resulting dataset - the recalibration set - is composed of $1 + c + d$ columns, with the first column being the vector of cumulative probabilities, c being the number of neurons in the layer $L-1$ and d being the number of columns of the network's output (as can be seen in Chapter 5).

If performed in the covariate space, recalibration may require additional solutions to deal with the dimensionality of **x**. At the other end of the spectrum, recalibrating on the output layer does not guarantee the selected observations are in fact *close*, since only the target variable is being taken into account – two samples far apart in the covariate space can lead to similar predictions. Recalibrating on an intermediate layer may, therefore, provide two related benefits: the dimensionality can be easily controled when setting the ANN's architecture and the distances are evaluated over a convenient representation (learned by the ANN itself) of the original data.

Although the calibrated distribution is generally more reliable than the ones estimated by the models, a considerable disadvantage of quantile-based recalibration techniques is the loss of parameter interpretability. This, however, is not an issue in the context of neural networks since the network weights aren't easily interpretable in most cases.

The scale and measurement unit of the features can influence which of those becomes more relevant. For example, if we wish to recalibrate a model that considers two variables, age and height, the latter can become more or less relevant depending on its scale (centimeters, decimeters, meters, etc). Normalizing the variables is frequently a convenient solution.

Another issue mentioned before is the well known *curse of dimensionality*. As the dimensionality of $\mathbf{X}$ increases it can become very difficult to calculate the distances between the observations. Beyer et al., 1999, showed that, under certain conditions, the distance to the nearest neighbour observation approaches the distance to the furthest neighbour with the increase in dimensions. In other words, the contrast in distances between two points in the covariate space vanishes. Therefore, performing recalibration procedure can become very challenging in these circumstances.

Chapter 3

Toy Examples

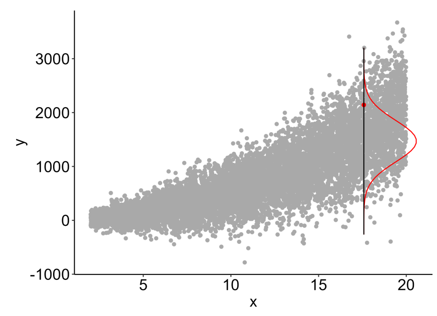
3.1 Heteroscedastic Gaussian Model

Consider $n = 100,000$ samples from the following Gaussian heteroscedastic quadratic model

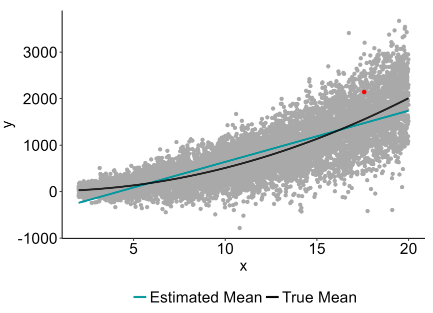
$$Y = 10 + 5X^2 + e \tag{3.1}$$

where $e \sim N(\mu = 0, \sigma = 30X)$ (Figure 3.1a) and $\mathbf{X} \sim U(2, 20)$, to which we fit the misspecified linear homoscedastic model $\hat{\mu_Y} = \beta_0 + \beta_1 X$. Figure 3.1b shows the fitted linear regression line in contrast with the actual mean curve. The mean and the standard deviation estimated by the linear model for the predictive distribution of the red-highlighted observation $(x, y) = (17.6, 2141.9)$, taken as an example, are $\hat{y} = 1469.6$ and $\hat{\sigma} = 385.2$, respectively. Figure 3.1c shows that, for a random variable with distribution $N(\mu = 1469.6, \sigma = 385.2)$, the highlighted observation has an associated cumulative probability of $p = 0.959$. The cumulative probability histogram of all observations in Figure 3.1d hints at the global bias of this model predictions.

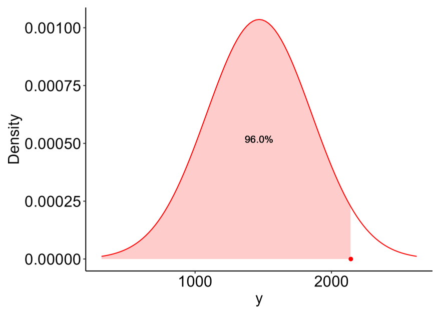
Figure 3.1 depicts the poor quality of the linear model fit due to its misspecification. Frosini's test rejected the null hypothesis of uniformly distributed p_i for all i with 5% significance level.



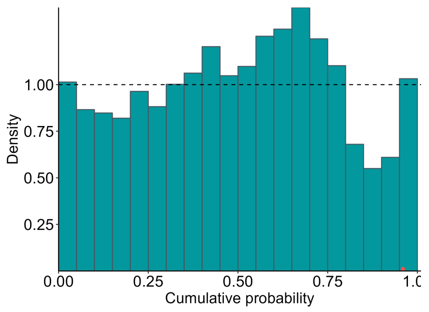
(a) Scatterplot of Y given X.



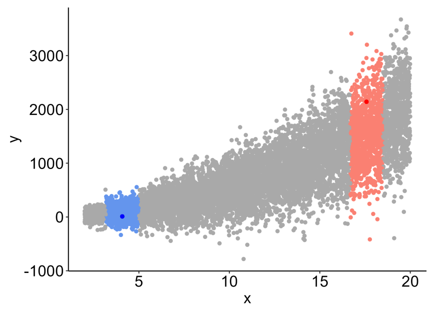
(b) Mean functions.



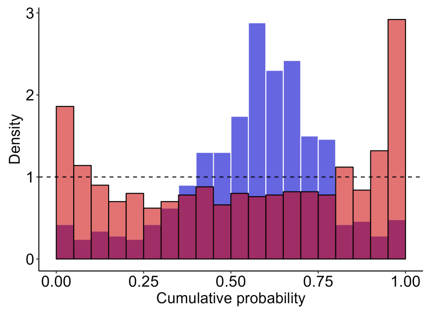
(c) Predictive density.



(d) Global probability histogram.



(e) Two distinct neighborhood regions.



(f) Local probability histograms.

Figure 3.1: (a) The true relationship between the response variable and the independent variable, with the position of the highlighted observed point in relation to the predictive distribution given by the model. (b) The true mean contrasted with the model's estimated mean. (c) The predictive density of the highlighted point and its cumulative probability. (d) The global cumulative probabilities histogram shows the model's global bias. (e) The neighborhood of the two highlighted points. (f) The cumulative probabilities histogram of the model in the two highlighted neighborhoods showing that the model presents two distinct bias patterns depending on the region.

The test's results can be seen in Table 3.1. In the global cumulative probability estimates histogram (Figure 3.1d), it is possible to see a combination of bias patterns indicating both model's variance underestimation and overestimation in distinct regions of the covariate space (Figures 3.1e and 3.1f). Besides that, there are clues of negative and positive biases in the distribution's mean estimations.

Test statistics B	P-value
2.1851	<0.001

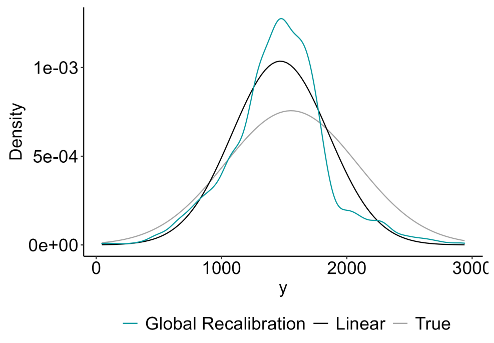
Table 3.1: Frosini's uniformity test of the p_i. The test indicates that the cumulative probabilities distribution is not Uniform and thus the model is not well calibrated.

In order to recalibrate the model, we split the dataset into training, validation and test sets by randomly taking $80,000$ to the training set and $10,000$ observations to the validation and test sets, respectively. In the context of recalibration, the validation set is used to compute the cumulative probability estimates vector, $\mathbf{p}$, and the test set is used to evaluate the performance. Figure 3.2 compares the linear model and the global recalibration. Figure 3.2a shows the models and the true probability density function of the highlighted observation in Figure 3.1a. Figure 3.2b shows the Kullback-Leibler divergence (KL divergence) between the models' estimated density and the true density for all observations. The Kullback-Leibler divergence (Kullback and Leibler, 1951) is given by,

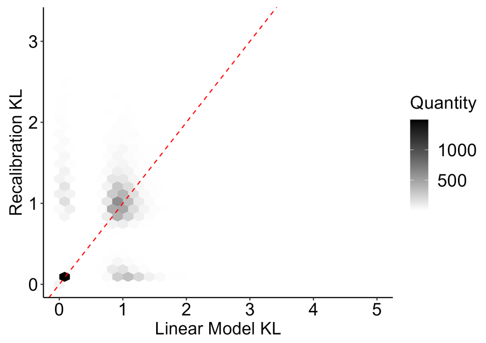
$$D_{KL}(P||Q) = \int P(y) \log\left(\frac{P(y)}{Q(y)}\right) dy, \tag{3.2}$$

where P and Q are density functions. A comparison between both models' fitted values and the true mean is shown in Figure 3.2c and a comparison between estimated and true standard deviations are shown in Figure 3.2d.

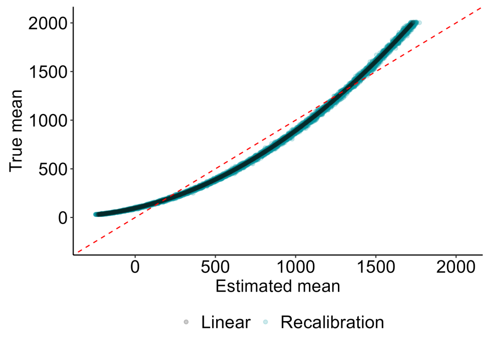
Overall the global recalibration offered no significant improvement over the linear model as seen in Figure 3.2. Table 3.2 shows that the average KL divergence of the global recalibrated model is approximately 3% smaller than the linear model, indicating a small gain in data



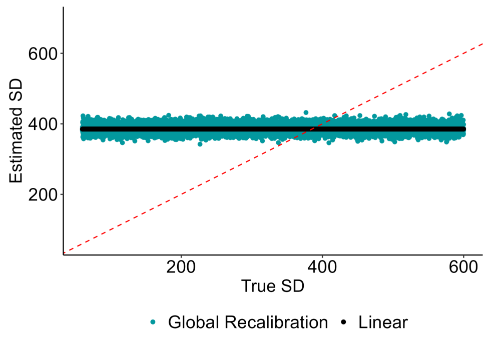
(a) Predictive density.



(b) Kullback-Leibler Divergence.



(c) True and estimated mean.



(d) True and estimated standard deviation.

Figure 3.2: (a) The predictive densities of the highlighted red point given by both models in contrast to the true density. (b) The Kullback-Leibler divergence of the recalibration in relation to the linear model. (c) Both models estimated means in relation to the true model. There seems to be no apparent difference between them. (d) Both models estimated standard deviations in relation to the true model.

distribution representation. Figure 3.2b indicates global recalibration offered estimation improvements in some regions of the prediction space while worsening them in others compared to the linear model. There is also little difference in the mean estimates between the models. Furthermore, both fitted models couldn't capture the true model's heteroscedasticity as seen in Figure 3.2d. Table 3.2 shows a slight loss in prediction capacity of the global recalibration when compared to the linear model.

Model	MSE	KL Divergence
Linear	14497.7	0.815
Global Recalibration	14676.8	0.806

Table 3.2: Comparison between the linear model and the global recalibration. Although there was a small reduction in the global recalibration KL divergence, there was no gain in terms of prediction accuracy.

Let us consider the two highlighted observations in Figure 3.1e. Looking at the histograms in Figure 3.1f we see that the true model's variance is underestimated in the red region while it is overestimated in the blue region. This happens because the linear model assumes homoscedasticity on the model's variance. Since there is different prediction bias in each region, these pieces of information can be added to the recalibrated model.

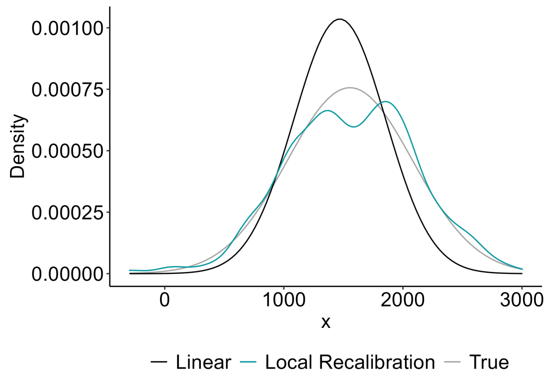
Next, in this example, we select and weigh the nearest neighbors using Epanechnikov's kernel,

$$k_u(d) = \frac{3}{4}\left(1 - \left(\frac{d_{ij}}{u}\right)^2\right), \tag{3.3}$$

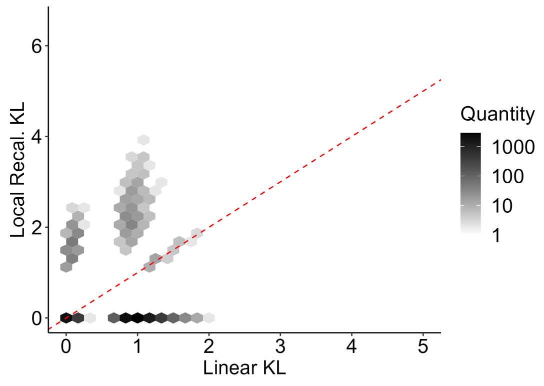
where u is chosen such that the 10% closest observations to the i-th one have positive weights. Also, in this case, d_{ij} is the Euclidean distance between the i-th and the j-th observations:

$$d_{ij} = \sqrt{(x_i - x_j)^2}. \tag{3.4}$$

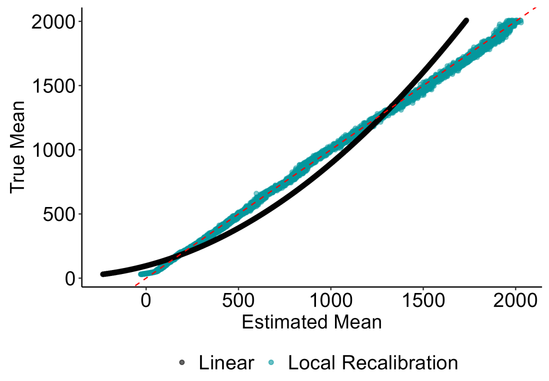
In Figure 3.3 we can see that local recalibration offered a substantial improvement over the linear model. Figure 3.3a shows that the local recalibration density estimator approximates



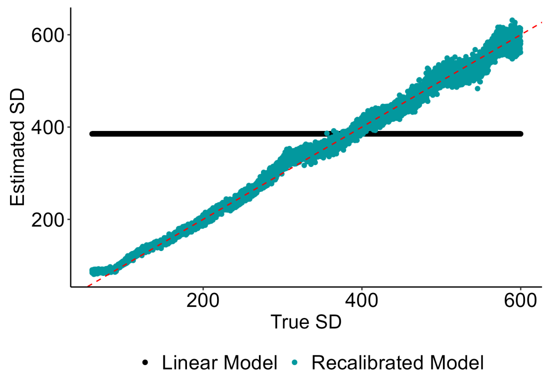
(a) Highlighted observation's density.



(b) Kullback-Leibler Divergence.



(c) True and estimated mean.



(d) True and estimated standard deviation.

Figure 3.3: (a) Predictive densities of the highlighted red point in comparison to the true density. It can be seen how well the recalibrated model approximates the density in relation to the linear model. (b) Kullback-Leibler divergence of the local recalibration in relation to the linear model. (c) Both models estimated means in relation to the true model. (d) Both models estimated standard deviation in relation to the true model.

the true density of the highlighted observation better than the linear model. Figure 3.3b shows information gained from local recalibration over the fitted linear model. In Figures 3.3c and 3.3d recalibrated estimates of the mean and the standard deviation align with the true values, showing that the locally recalibrated model captures the heteroscedasticity of the true model. Table 3.3 shows that recalibration mean squared error (MSE) is about 70% lower than the linear model, also the reduction in divergence reflects information gained from recalibration.

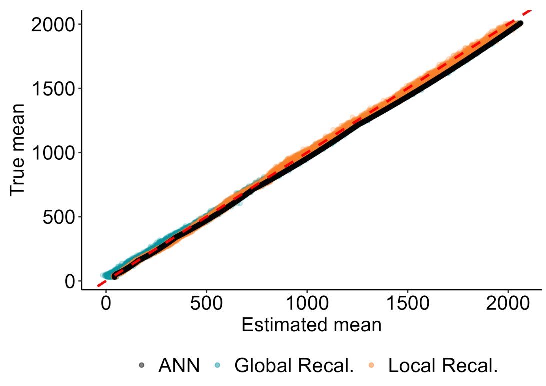
Model	MSE	K-L Divergence
Linear	14497.7	0.762
Local Recalibration	361.6	0.129

Table 3.3: Comparison between the linear model and the local recalibration. There is a very significant improvement in KL divergence and prediction accuracy from local recalibration in relation to the linear model.

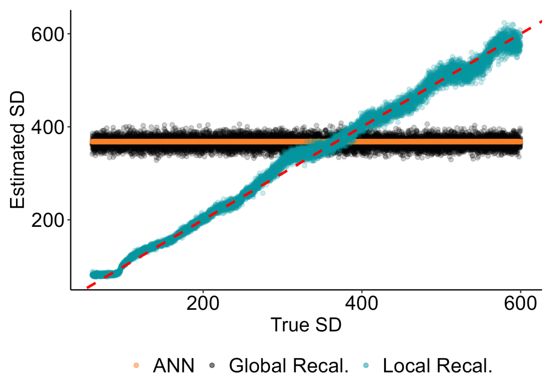
We then proceed to fit an arbitrary simple neural network model with two hidden layers, where the first layer has six neurons and the second one has two neurons, both having *ReLU* activation function. The output layer has a single neuron and linear activation function. We also adopted the mean squared error loss function, assuming the predictive distributions are Gaussian with the mean given by each prediction and variance equal to the mean squared error of the model on the validation data.

Next, we recalibrate the neural network model using an arbitrarily chosen proportion of 10% nearest neighbors to each sample observation in the penultimate layer. Figures 3.4a and 3.4b show the estimated mean and standard deviation, respectively, for both global and local recalibrated neural network models. It is possible to see that, while both models' mean estimates were similar, only the local recalibrated model captured the true heteroscedasticity. Table 3.4 shows the performance indicators for all the models considered. Although recalibration increased the MSE of the ANN, it also reduced the average KL divergence, showing that the recalibrated distribution offered a better approximation of the true distribution than the ANN.

The observed coverage in Table 3.4 represents the percentage of 95% confidence intervals that captured the respective observed value in the test dataset. At a first glance, all the con-



(a) True and estimated mean



(b) True and estimated standard deviation.

Figure 3.4: Comparison between ANN and recalibrated models. (a) Estimated means in relation to the true mean. (b) Estimated standard deviations in relation to the true model.

Model	MSE	K-L Divergence	Observed Coverage	sMIS
Linear	14497.7	0.815	93.76	2.717
Globally Recalibrated Linear	14676.8	0.806	94.83	2.698
Locally Recalibrated Linear	361.6	0.129	94.94	2.017
ANN	1457.9	0.864	93.36	2.636
Globally Recalibrated ANN	758.9	0.818	94.48	2.609
Locally Recalibrated ANN	342.1	0.191	94.99	2.013

Table 3.4: Performance comparison between linear and ANN models. There is clearly an improvement in terms of distribution approximation and interval prediction from recalibration in relation to the base models.

fidence intervals are close to their nominal confidence level. However, Figure 3.5 shows that only the local recalibration has evenly distributed predictions, meaning that the other models capture more observations where variance is overestimated. Although the original and globally recalibrated models miss approximately only 5% of the intervals on average, since their local coverage is greater than 5% for lower values of x and lower for higher values, it can be seen they still are uncalibrated. The standard Mean Interval Score (sMIS) (Gneiting and Raftery, 2007) in Table 3.4 is given by

$$IS_\alpha(i) = -\left((q_{\frac{\alpha}{2}} - q_{1-\frac{\alpha}{2}}) + \frac{2}{\alpha}(q_{\frac{\alpha}{2}} - y_i)\mathbb{I}_{\{y<q_{\frac{\alpha}{2}}\}}(i) + \frac{2}{\alpha}(y_i - q_{1-\frac{\alpha}{2}})\mathbb{I}_{\{y>q_{1-\frac{\alpha}{2}}\}}(i)\right), \qquad (3.5)$$

where $i = 1, \ldots, n$, $q_{\frac{\alpha}{2}}$ and $q_{1-\frac{\alpha}{2}}$ are the upper and lower interval quantiles, standardized by the validation set mean absolute values. sMIS directly compares prediction intervals by penalizing observations missed and rewarding narrower intervals. It can be seen that local recalibration performed best than the other models in terms of interval scores.

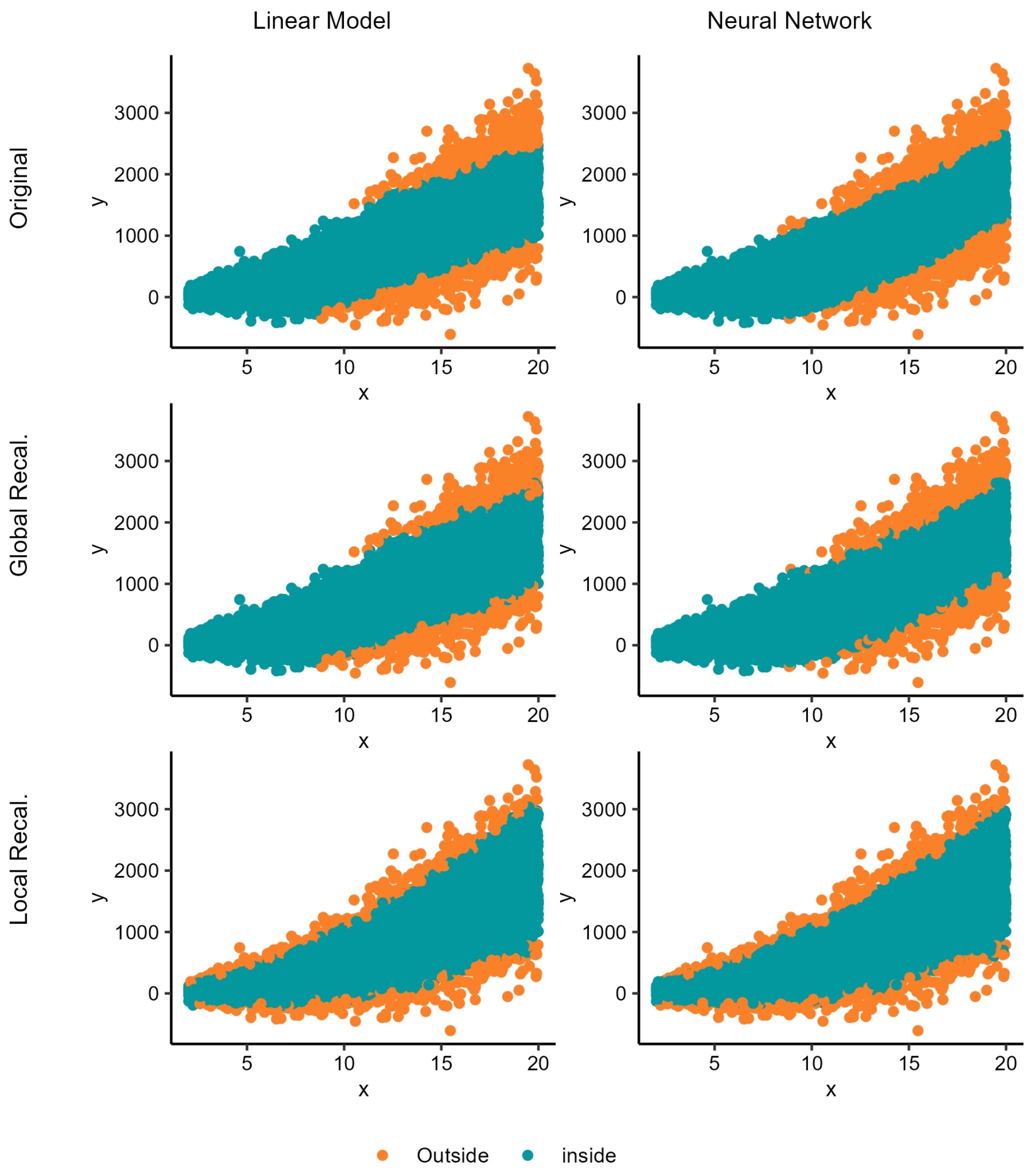


Figure 3.5: Interval coverage of each model in the covariate space. Local recalibration corrected regional bias and provided a more adequate coverage pattern.

3.2 Non-Linear Gamma Model

In this toy example we consider the Rosenbrock function, also known as Rosenbrock Vale, notorious for its wide use as a test function for optimization algorithms. This function is given by

$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2, \tag{3.6}$$

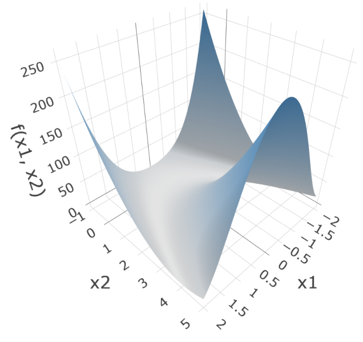
with a and b constants. Figures 3.6a and 3.6b show the function's surface in three and two dimensions, respectively.

We take the Rosenbrock function, within the intervals $x_1 \in (-2, 2)$ and $x_2 \in (-1, 5)$, with $a = 1$ and $b = 10$, as the conditional mean (μ) of a Gamma distributed random variable Y with shape parameter $\alpha = 100$ and scale parameter given by

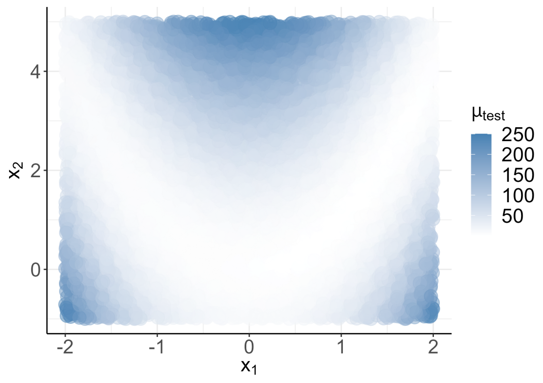
$$\theta = \frac{\mu}{\alpha}. \tag{3.7}$$

To simulate the relationship between the pair of independent variables and the response variable we generate $n = 100,000$ samples of $\mathbf{x} = (x_1, x_2)$ from the Uniform distribution such that $X_1 \sim Uniform(-2, 2)$ and $X_2 \sim Uniform(-1, 5)$. Then, we generate n samples of Y conditional to X_1 and X_2 so that $Y|X_1, X_2 \sim Gamma\Big(100, \frac{f(x_1,x_2)}{100}\Big)$. Therefore, for each pair (x_1, x_2), we have y sampled from a Gamma distribution with mean $f(x_1, x_2)$. After that we split the data into training (80%), validation (10%) and test (10%) data sets. Figure 3.6c shows the generated test data in three dimensions, where it can be seen that the, farther the observations are from the valley, the greater is the variability around the mean. Test data variability can also be seen in Figure 3.6d.

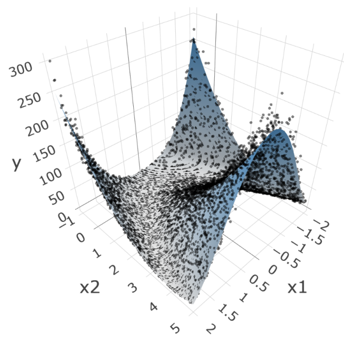
To estimate the mean function from the data, we fit an arbitrary neural network model to $log(y)$, consisting of four hidden dense layers (6400, 6400, 240, 240 neurons) with *ReLU* activation function and an output linear layer with a single neuron. We also added batch normalization layers and dropout layers with probability $p = 0.5$ between every hidden layer. The network



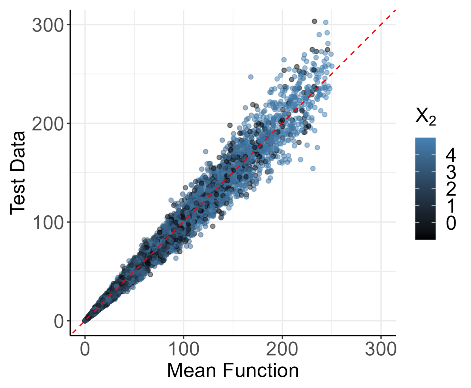
(a) Rosenbrock function surface.



(b) Rosenbrock function mapping.



(c) Test data surface.



(d) Test data variability.

Figure 3.6: (a) Rosenbrock function's surface in three dimensions. (b) Function values mapping in the covariate space. (c) Test data variability as a function of $\mathbf{x}$ values. (d) Test data variability versus the mean.

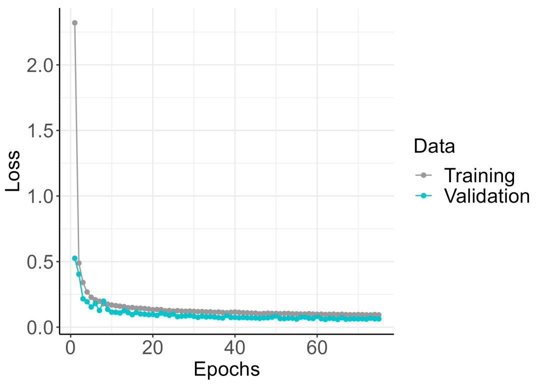
was trained for 75 epochs with a learning rate of 0.001, *ADAM* optimizer, MSE loss function and batches of size 100. The network loss function progression during training and validation can be seen in Figure 3.7a.

The use of dropout layers opens up two possibilities to estimate the network's uncertainty. The first approach consists of using the weight scaling inference rule (WSIR) to make a single prediction for every data point in the test set, assuming normality in the logarithmic scale of y. The second approach consists of using the dropout's randomly generated masks to obtain a sample of the network's predictions for every data point in the test set, following the methodology proposed by Gal and Ghahramani (2016) (Monte Carlo Dropout), without parametric assumptions on the predictive distribution.

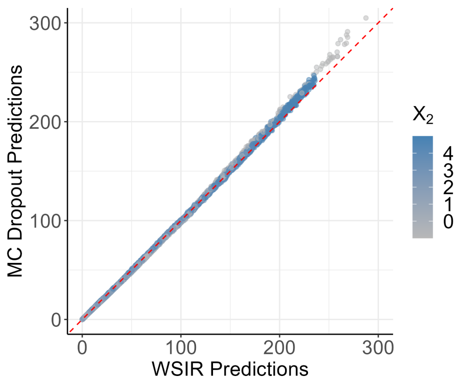
The logarithmic transformation takes the response variable from the interval $Y \in [0, \infty)$ to $\log(Y) \in (-\infty, \infty)$. Since the weight scaling inference rule averages the output of all dropout masks, in the first approach we can assume the network outputs the mean of a Normal distribution with variance equal to the validation set MSE. For every prediction, we generated a sample of size $1,000$ of the predictive distribution in the log-scale, then applied the exponential transformation, $e^{\hat{y}}$, to take the samples back to the interval $Y \in [0, \infty)$. By doing so, we got a Monte Carlo sample of the predictive distribution for every observation in the test set in its original scale.

The Monte Carlo Dropout (MC Dropout) approach uses the randomness intrinsic to the dropout technique to get samples of the network's predictions for each data point. At every iteration, each dropout layer generates a random mask that turns off some of the neurons according to the chosen dropout probability. We activated the dropout layers during the inference stage and generated samples of size $1,000$ for each observation in the test set (in the original scale). In both methods, the point estimates are the mean taken from the samples generated from the predictive distributions. Figure 3.7b compares predictions from the two methods. It can be seen that, for lower values of x_2, MC Dropout tends to return slightly higher predictions than WSIR method.

Since we have very high dimensionality in the network's layers weight space, we recalibrated both methods locating the nearest neighborhood of each observation in the input space. In this example, if we fix the number (k) of neighbors, the observations on the edge of the mean function's region will have a much greater neighborhood area than the ones in the center. To avoid that, for each observation in the test set, we selected the closest observations in the validation set based on a fixed maximum distance (equal to 0.5). Due to the non-parametric nature of the samples generated by the MC Dropout method, to directly compare both method's estimates, we calculated the cumulative probabilities empirically from the Monte Carlo samples



(a) Network loss function.



(b) Comparison between ANN methods.

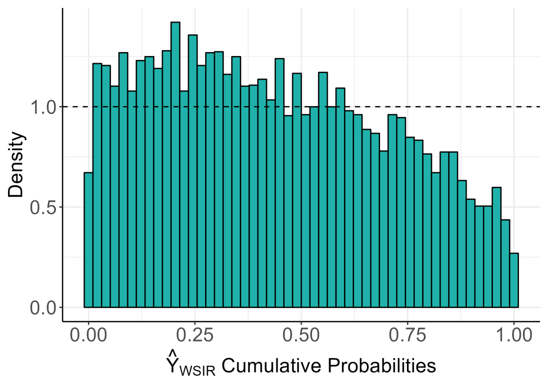
Figure 3.7: (a) Neural network loss function during training and validation. (b) Comparison between WSIR and MC Dropout predictions. MC Dropout tends to overestimate predictions for lower values of x_2 in relation to WSIR.

obtained from each method and generated unweighted samples of the recalibrated predictive distributions according to the weights defined by the Epanechnikov kernel. The cumulative probability histograms in Figures 3.8a and 3.8c indicate that both methods are not well calibrated globally. While the predictions made based on the WSIR predictive distribution appear to mainly overestimate the mean, the predictions of the MC Dropout method tend to overestimate the true model's variance. Figures 3.8b and 3.8d indicate both models are possibly well calibrated after the local recalibration procedure. Figures 3.9a and 3.9b show WSIR predictions and residuals in the covariate space. One of the major effects of recalibration on both ANN models was decreased prediction values for lower values of x_2, previously overestimated, as illustrated with the WSIR method in 3.9c. Figure 3.9d shows WSIR method's recalibration residuals.

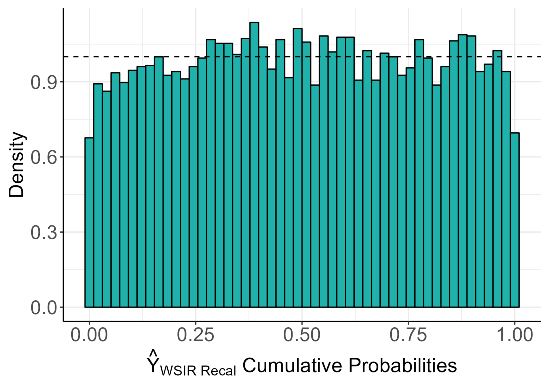
In summary, the analysis was carried out as follows:

1. Fit an ANN to the training data and validate in the validation data, in the log-scale;

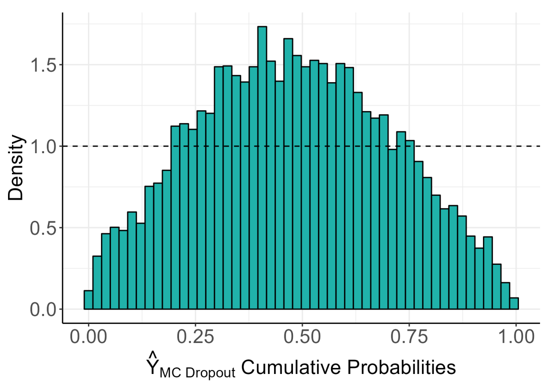
2. Generate samples of the normal predictive distribution for every test observation, using the WSIR. Exponentiate, then take the mean of each sample as the point prediction;



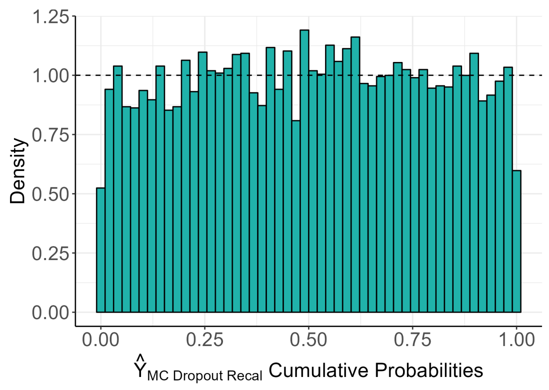
(a) WSIR cumulative probabilities.



(b) WSIR recalibration cumulative probabilities.



(c) MC Dropout cumulative probabilities.



(d) MC Dropout recalibration probabilities.

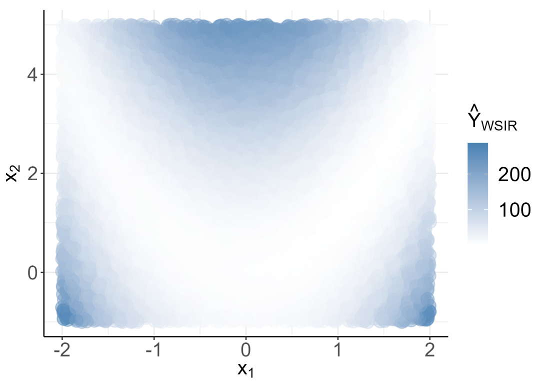
Figure 3.8: (a) WSIR predictions overestimate the mean. (b) WSIR cumulative probabilities after recalibration. (c) MC Dropout predictions overestimate the variance. (d) MC Dropout cumulative probabilities after recalibration.

3. Generate samples of the empirical ANN predictive distribution for every test observation using MC Dropout. Exponentiate the samples, then take the mean of each sample as the point prediction;

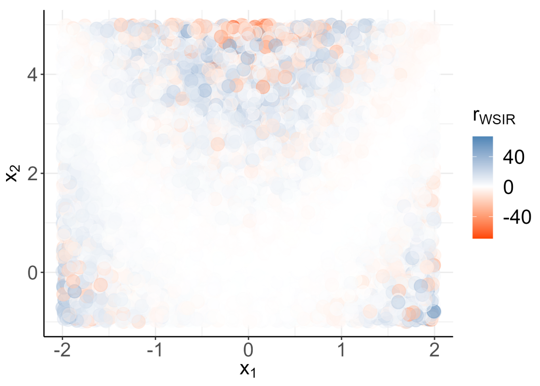
4. Recalibrate each method using Algorithm 1;

5. For each point in the test set, generate samples from the true model's distribution. Take the mean of each sample.

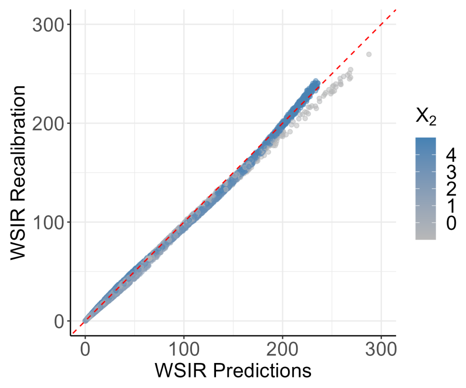
6. Calculate the MSE, confidence interval coverage and standardized mean interval score



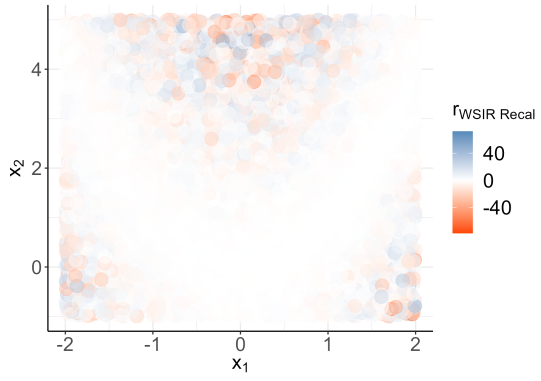
(a) WSIR predictions mapping.



(b) WSIR residuals mapping.



(c) WSIR predictions and recalibration.



(d) WSIR recalibration residuals.

Figure 3.9: (a) WSIR predictions in the covariate space. (b) WSIR residuals. (c) Recalibration effect on WSIR predictions, varying according to the x_2 space. (d) WSIR recalibration residuals.

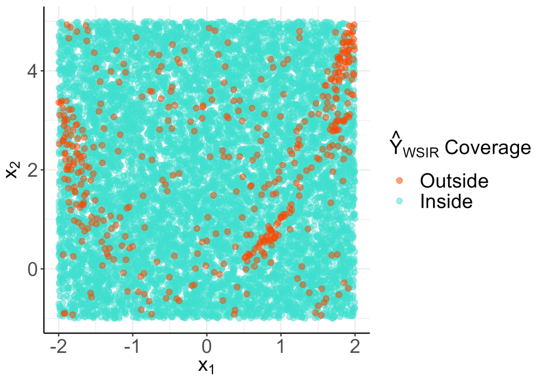
for all methods.

The performance of all methods was measured from samples obtained from their estimated predictive distributions and compared against samples taken from the true model's distribution. While the predictions taken from the WSIR presented smaller MSE value than the ones taken from the MC Dropout, the recalibrated methods resulted in the predictions closest to the true samples by a large margin, indicating recalibrated predictions were much closer to the original test data on average. In regards to confidence interval predictions, considering 95% confidence intervals, the largest coverage and interval score were presented by the MC Dropout method, which suggests this method generated too wide prediction intervals. The recalibrated methods generated confidence intervals coverage and score metrics very close to the true model. All performance metrics can be seen in Table 3.5.

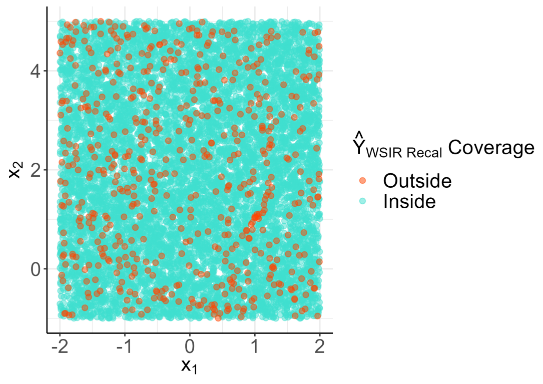
Model	MSE	Coverage	sMIS
WSIR ANN	71.2516	0.9548	0.5689
MC Dropout ANN	75.0649	0.9899	0.8745
Recalibrated WSIR ANN	60.5202	0.9432	0.4995
Recalibrated MC Dropout	60.4355	0.9468	0.4990
True model	59.6438	0.9462	0.4775

Table 3.5: Performance comparison between all methods.

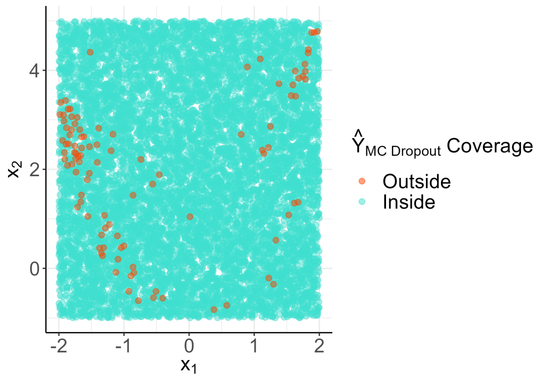
The coverage of all methods is shown in Figure 3.10, where light red points represent observations not captured by the prediction intervals. It can be seen that ANN predictions, despite getting close to the nominal coverage level, as was the case in the previous example, presented a very clear pattern, missing observations in specific areas. MC Dropout method's wide interval estimation is reflected in the interval coverage surface, showing a similar bias pattern to the first method while having a larger MSE. On the other hand, recalibration appears to have largely reduced local biases. In addition, after the recalibration (regardless of the base model), the coverage showed a much more evenly spread pattern throughout the covariate space, an indication that these models are well-calibrated.



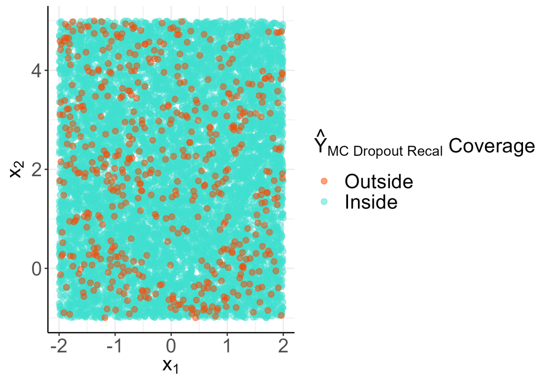
(a) WSIR coverage



(b) WSIR recalibration coverage



(c) MC Dropout coverage



(d) MC Dropout recalibration coverage

Figure 3.10: Confidence interval misses (light red points) throughout the covariate space. (a) WSIR. (b) WSIR with local recalibration. (c) MC Dropout. (d) MC Dropout with local recalibration.

Chapter 4

Simulation Study

In this analysis, we run a simulation study to investigate and quantify the effect of recalibration of neural network models under various conditions. We revisit the analysis in Tran et al. (2020) and consider the highly nonlinear model given by

$$y = 5 + 10x_1 + \frac{10}{x_2^2 + 1} + 5x_3x_4 + 2x_4 + 5x_4^2 + 5x_5 + 2x_6 + \frac{10}{x_7^2 + 1} + 5x_8x_9 + 5x_9^2 + 5x_{10} + \delta, \quad (4.1)$$

where $\delta \sim N(0, 1)$. We generate the variables $x_1, \ldots, x_{20}$, from which $x_{11}, \ldots, x_{20}$ are non-informative to the model, from a multivariate normal distribution with mean vector $\mathbf{0} = (0)_i$, $i = 1, \ldots, 20$, and covariance matrix $(0.5^{|i-j|})_{i,j}$, $i = 1, \ldots, 20, j = 1, \ldots, 20, i \neq j$. Data is simulated from this process in sets of sizes $N = 10^3$, 10^4, 10^5 and 10^6. Then, in each scenario, we randomly split the data into a training set, a validation set and a test set using 80%, 10% and 10%, respectively. We propose to fit and recalibrate two neural network models as well as compare recalibration performance with a K-Nearest Neighbor (KNN) regression model, due to methodological similarities.

Because this section compares the models' performances for each configuration under study, both recalibrations and the KNN regression models are fit with the validation set and evaluated with the test set, not being necessary to optimize KNN regression parameter values.

Apart from the number of neurons, the two neural networks considered are identical in architecture, both being composed of a 20 neurons input layer, followed by four hidden layers with *ReLU* activation function, then a linear output layer with a single neuron. The "small" network's hidden layers are all composed of 5 neurons, while the "big" network's first three hidden layers have 200 neurons and its last hidden layer has 5 neurons. Both networks are trained with the *Adam* optimizer and mean squared error loss function with the early stop callback. Due to computational and time constraints, both neural network model's learning rates were optimized in advance for every scenario considered in the simulation, with values ranging from 10^{-4} to 10^{-2}.

Both neural network models are recalibrated assuming normally distributed predictive distributions and considering Epanechnikov's kernel function to weight the recalibrated samples. To evaluate the effects of recalibration, both networks are recalibrated on the input layer, the fourth layer, the fifth layer and the output layer.

The networks are recalibrated with the 100, 500 and 1000 nearest neighbors, corresponding to different proportions of the validation and test data sets. The nearest neighbor search is conducted with three degrees of approximation: $\epsilon = 0$ (exact search), $\epsilon = 0.5$ and $\epsilon = 1$, where ϵ is the KNN search approximation parameter.

An algorithm directly comparable to the proposed recalibration method is the KNN regression, which we also apply considering the Epanechnikov's kernel function, the same KNN approximate search algorithm and the same set of values of the K (nearest samples) and ϵ parameters, for direct comparison. Due to memory and time constraints, both recalibrated models and KNN regression were fit in batches according to test set size, KNN regression being applied only to the first two scenarios of $N = 10^3$ and $N = 10^4$. In the last two scenarios, KNN regression computational cost was simply too high to be carried over with the available computational resources, taking alone up to two-thirds of the total simulation time, on average. As shown in the next set of graphic panels though, it is clear that, while KNN regression is comparable to the recalibration in terms of methodology, it does not compare in terms of computational efficiency and model performance.

The simulation study is run multiple times, each with a different seed. For each replication of the simulation study, we generate a data set with N observations of the independent variables $\mathbf{X}$ and the response variable Y, then we split it into training, validation and test sets. The neural network models are trained and validated with the generated data and then, with their predictions and weight sets in hand, we calculate the networks predictions cumulative probabilities and the target layers predictions of $\mathbf{X}$ validation and test sets. The recalibration predictions are then obtained for each target layer, nearest samples size and approximation value. At this moment, the KNN regression predictions are also obtained for each nearest sample size and approximation value.

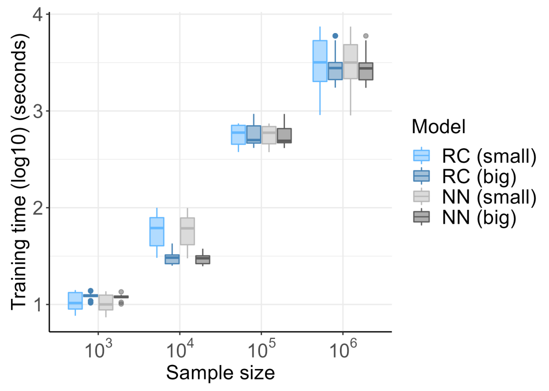
In this study, we split model fitting in two stages. The training stage consists of minimizing the neural network's loss functions, calculating the cumulative probabilities vector and the network's representations in the validation set.Since we are looking at the whole process until the final model is obtained, recalibration time in training stage consists of the networks training time plus the time taken to obtain recalibration data. Recalibration memory usage in training stage is composed of the networks' weights and the data necessary to obtain the nearest samples, the cumulative probabilities and the validation set representations. Since KNN regression doesn't need parameter optimization in this case, we consider its training time to be zero in all scenarios. Prediction stage consists of acquiring models predictions, which is mainly obtaining the nearest samples and kernel matrices for both methods. Memory usage in prediction stage consists of all data generated in this process and the data used to obtain the models predictions. For each sample size, we report the time and memory usage in training stage (training time and training memory), MSE, interval coverage, interval score, also time and memory usage in prediction stage (prediction time and prediction memory).

Figure 4.1a shows that recalibration (RC) does not significantly affect model training time as sample sizes increase. KNN regression training time is considered zero and thus is not represented. In all scenarios, training time increments scale with the neural networks (NN) time. However, it does affect inference (prediction) time as sample size increases, as seen in Figure 4.2a. Recalibration's target layer also doesn't seem to affect training time, as seen in Figure 4.1b. However, it appears to affect prediction time positively the higher the layer dimensions are, as shown in Figure 4.2c. The nearest samples size also increases prediction time – as it grows, the time taken by the KNN approximate search increases, as shown in Figure 4.2b. This is mitigated by the search level of approximation, with the exact search taking longer than a higher approximation level, as in Figure 4.2d.

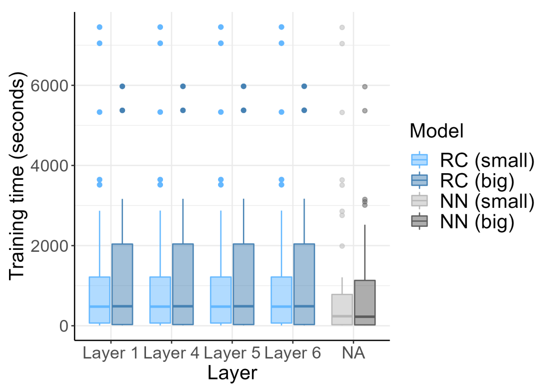
Data memory size increases proportionally to sample size and dimensionality both in the training and prediction stages (Figures 4.1c and 4.2e), while neural networks memory size doesn't increase at the same rate in the prediction stage. Figure 4.1d shows recalibration memory usage in the training stage in comparison to neural network training for each sample size, as a function of the target layer. Recalibration prediction memory size grows in $O(n)$, proportional to $n(2k + h + d)$, where n is the number of test observations, k is the number of nearest samples, h is the number of neurons in the target layer and d is the size of the network's output.

Two matrices of size $n \times k$ are generated from the approximate KNN search algorithm, one kernel matrix, obtained with the nearest samples distances to each test sample, and one matrix with the k-nearest samples indexes. The network output has size $n \times d$, where, in this simulation study, $d = 1$. Figure 4.2f shows how prediction memory usage is related to the main objects generated by recalibration, based on batch size. While other object sizes remain constant as batch size increases, the kernel matrix rapidly dominates recalibration's prediction memory usage. Both kernel matrix and indexes matrix become the most relevant objects time and memory-wise for large data sets.

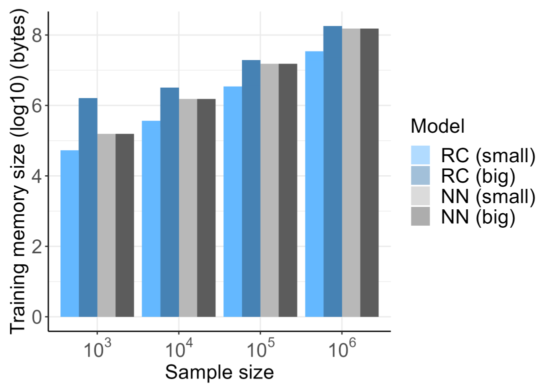
In Panel 4.3, the true model MSE (equal to 1) is represented by the golden dashed line. Figure 4.3a shows models MSE as sample size increases. Overall, recalibration improved the networks MSE on average, benefiting the most when there is plenty of data. It can be observed that even with more data provided, the smaller neural network's low capacity prevents it from improving model MSE. Even in this case, recalibration was able to improve performance, on



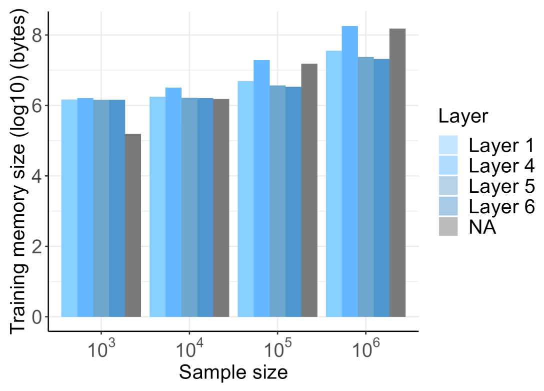
(a) Training time by data size.



(b) Training time by layers.



(c) Training memory size by data size.



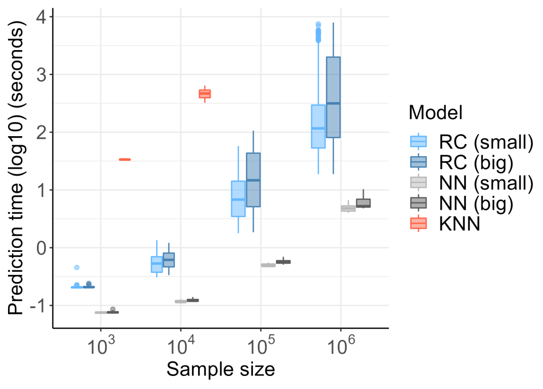
(d) Training memory by data size, by layers.

Figure 4.1: (a) Training time on each data set size scenario for neural networks (NN0 and recalibration (RC) models. (b) Recalibration training time on each NN target layer. (c) Training data memory size on each data set size. (d) Training data memory size for each sample size, stratified by target layer. NA represents neural network without recalibration.

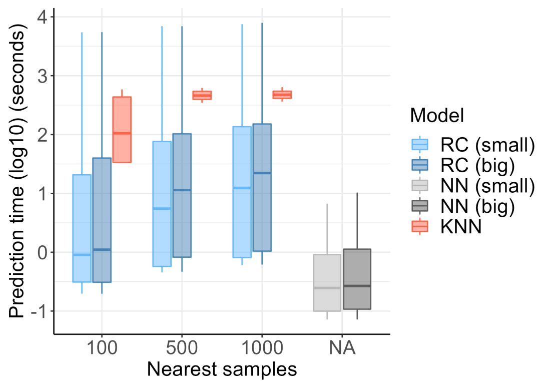
average. In Figure 4.3b, increasing the nearest samples size improved the biggest recalibrated model's MSE on average with diminishing effects as nearest samples size increases, meaning that increasing the proportion of nearest data won't add much information to the model predictions when there is already enough information, in terms of model capacity. The target layer also doesn't seem to influence much predictions accuracy, as Figure 4.3c shows, although it is relevant to prediction time. This fact suggests the information given by the nearest neighbors is roughly the same regardless of how it's been represented by the network, but the dimensionality of such space can speed up significantly prediction time. Approximation level also doesn't appear to affect MSE significantly. Figure 4.3d shows its effect for $N = 10^6$, excluding the KNN regression's MSE.

Figure 4.4a and Figure 4.4d show models performance on 95% confidence intervals for all sample sizes. With larger samples, neural networks interval coverage increases, surpassing nominal confidence level. Additionally, for small networks, interval scores remain constant on average, as sample size increases, while for big networks the scores decrease. This suggests that, again, the network's confidence interval performance is being limited by its capacity. Meanwhile, recalibration kept overall interval coverage below nominal level, on average, and improved interval scores in both cases, for all sample sizes, indicating that it successfully manages to improve interval prediction and interval width at the same time. While it does not seem to affect MSE, the target layer choice seem to slightly improve interval coverage as dimensionality decreases, as shown in Figure 4.4c. Interval coverage is also improved as nearest samples size increases, as in Figure 4.4b.

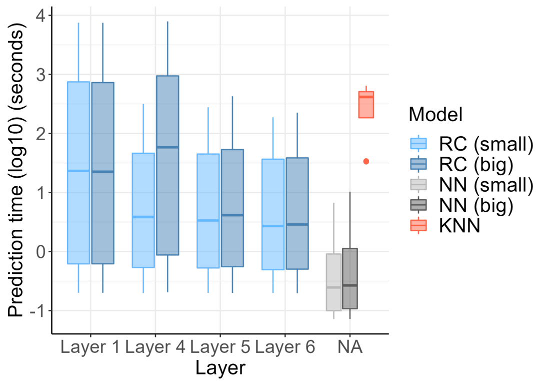
Specific effects of recalibration can be analysed when factoring different sample sizes. In the case when $N = 10^6$, Figures 4.5a, 4.5c and 4.5e show that decreasing nearest samples size improves MSE, as well as decreases prediction time substantially. On the other hand, increasing nearest samples size improves interval scores on average. MSE and interval score appear to increase with the choice of a target layer with reduced dimensionality, as shown in Figures 4.5b and 4.5d, in the case where $N = 10^6$. However, figure 4.5f show a substantial decrease of prediction time as layer dimensions decrease. While the smaller recalibrated model benefits from higher dimensionality, the bigger recalibrated model performs better on intermediate layers.



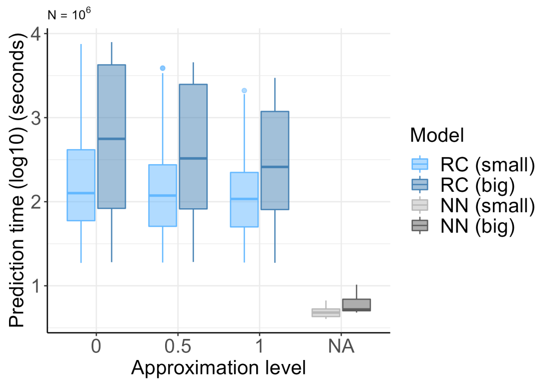
(a) Prediction time by data size.



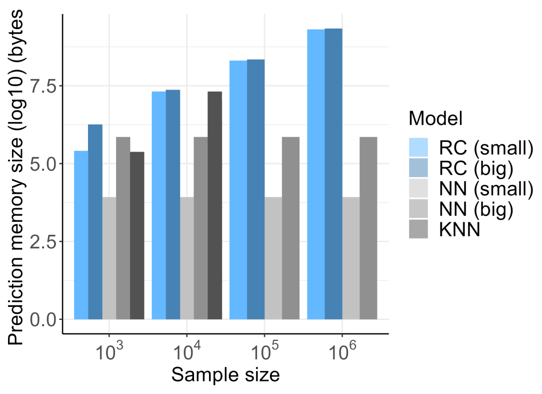
(b) Prediction time by nearest samples.



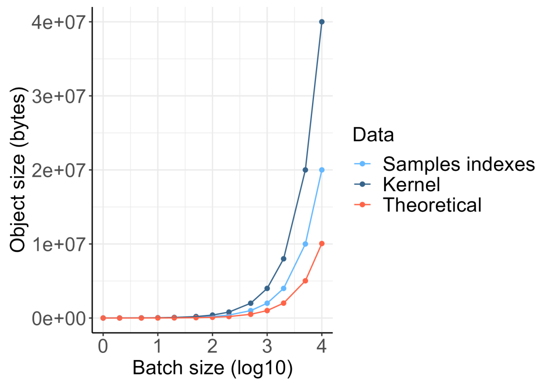
(c) Prediction time by layers.



(d) Prediction time by approximation.

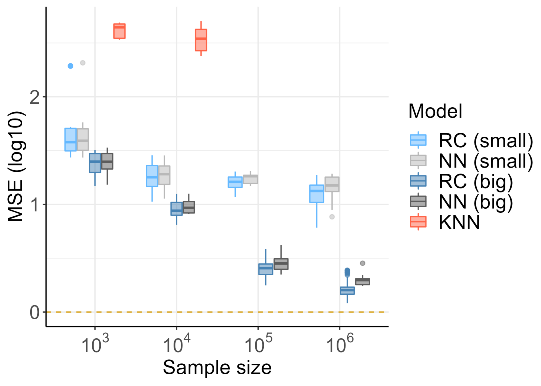


(e) Prediction memory size by data size.

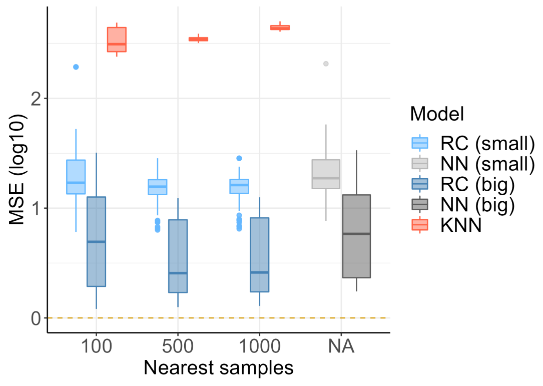


(f) Recalibration data size.

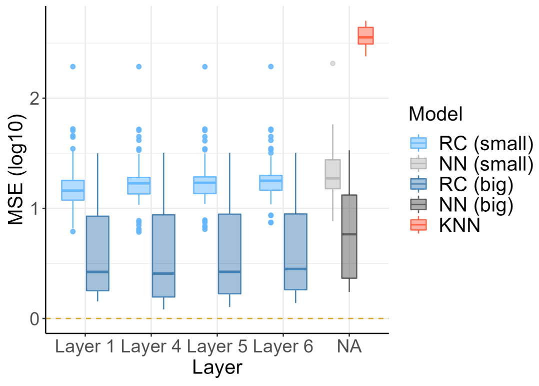
Figure 4.2: (a) Prediction time on each data set size scenario for neural networks (NN) recalibrations (RC) and KNN regression (KNN). (b) Recalibration prediction time on each data set size, for each target layer. (c) Recalibration prediction time on each ANN target layer. (d) Recalibration prediction time for each approximation level value, for $N = 10^6$. (e) Training data memory size on each data set size. (f) Recalibration nearest samples data size and kernel data size compared to the theoretical sizes.



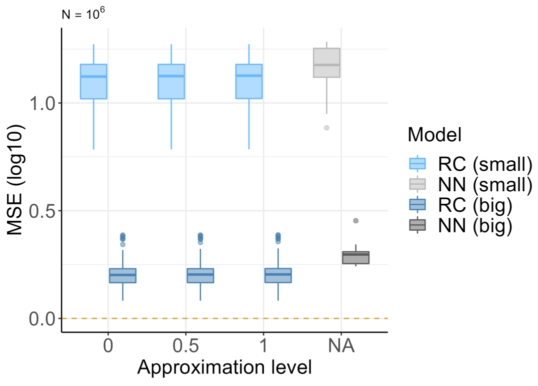
(a) MSE by data size.



(b) MSE by nearest samples.

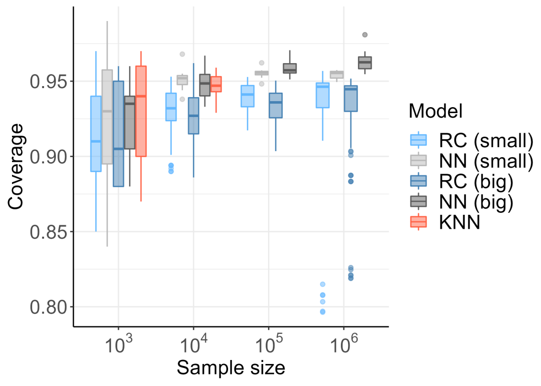


(c) MSE by layers.

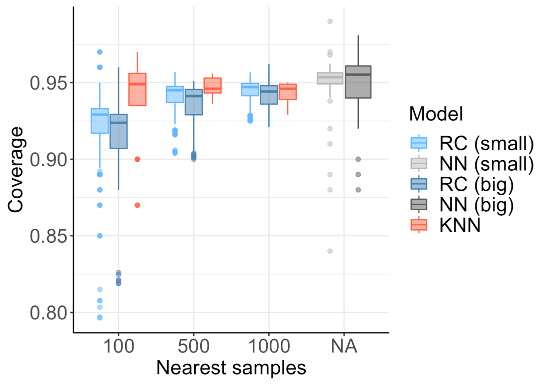


(d) MSE by approximation.

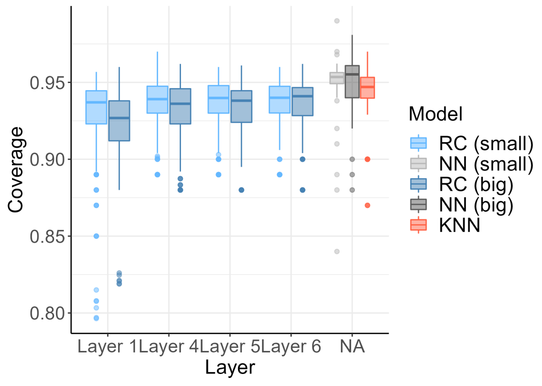
Figure 4.3: (a) MSE on each data set size scenario. (b) MSE on each nearest samples size, for all N sizes. (c) MSE on each ANN target layer, for all N sizes. (d) Recalibration MSE for each approximation level value, for $N = 10^6$.



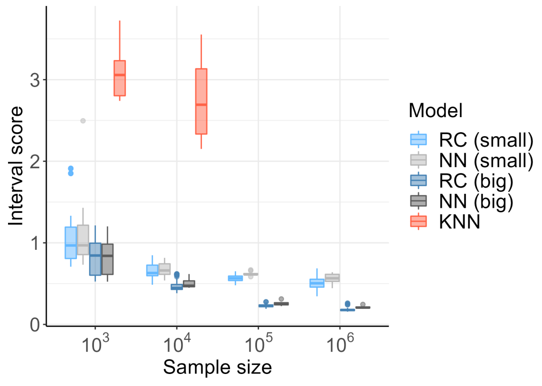
(a) Coverage by data size



(b) Coverage by nearest samples

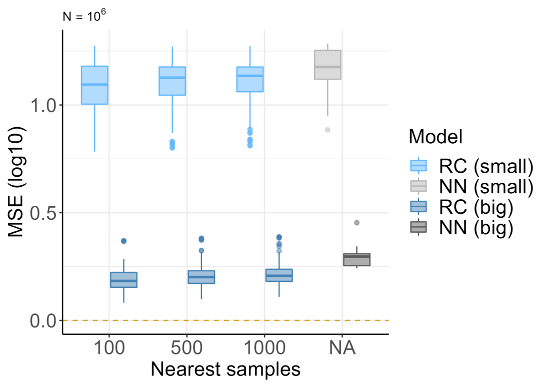


(c) Coverage by layers

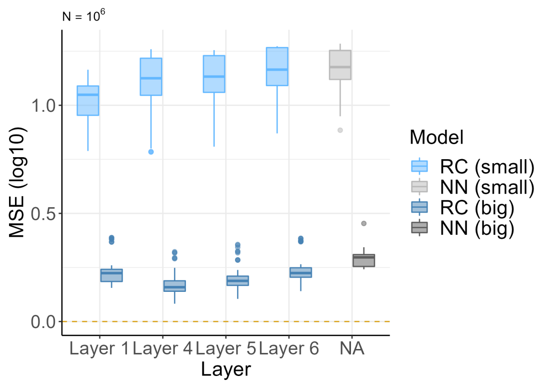


(d) Interval score by data size

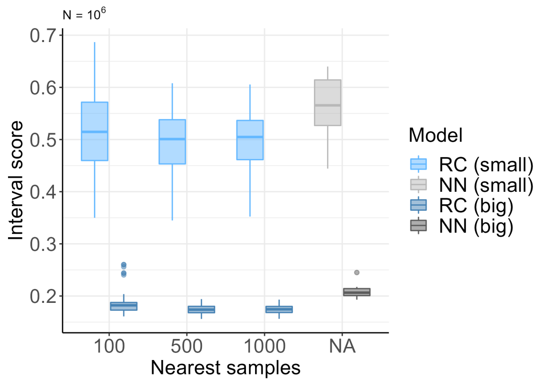
Figure 4.4: (a) Interval coverage on each data set size scenario. (b) Interval coverage on each nearest samples size. (c) Interval coverage on each ANN target layer. (d) Interval score on each data set size scenario.



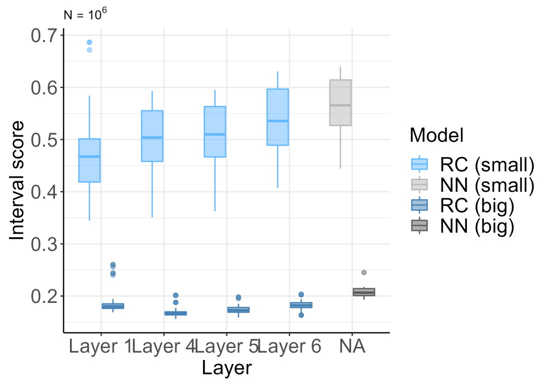
(a) MSE by nearest samples



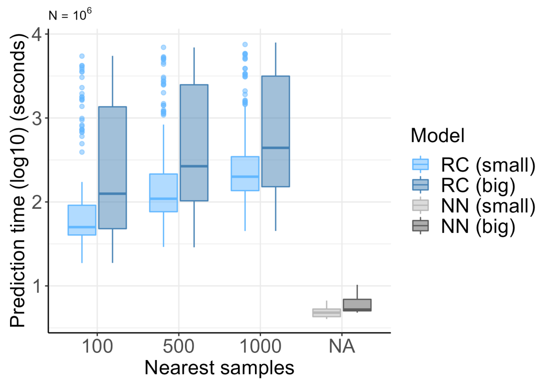
(b) MSE by layer



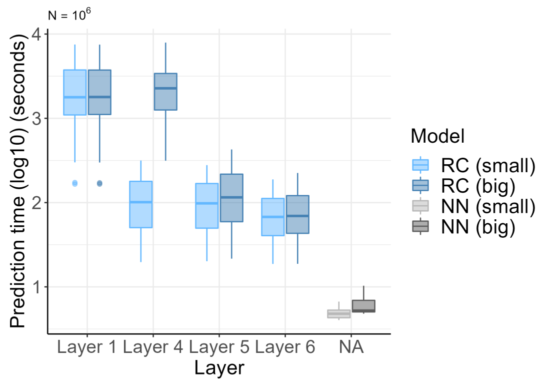
(c) Interval score by nearest samples



(d) Interval score by layer



(e) Prediction time by nearest samples



(f) Prediction time by layer

Figure 4.5: (a) MSE on each nearest sample size for $N = 10^6$. (b) MSE on each target layer for $N = 10^6$. (c) Interval score on each nearest sample size for $N = 10^6$. (d) Interval score on each target layer for $N = 10^6$. (e) Prediction time on each nearest sample size for $N = 10^6$. (f) Prediction time on each target layer for $N = 10^6$.

Chapter 5

Diamond Price Prediction

Diamonds are among the most valued materials extracted from nature. Desired not only as an adornment, but their physical properties - such as hardness and electrical conductivity - make them very desirable in many industries and scientific fields. All those different properties and applications, however, make their value hard to quantify. Polished diamond prices may vary widely depending on their individual characteristics, especially carat weight, color, cut and clarity. In this section we explore the data from $53,940$ diamonds from the data set *"diamonds"* available in the *ggplot2* R package (Wickham, 2016). This dataset lists several individual characteristics as variables, such as price in US dollars, carat weight, quality of the cut, diamond color, diamond clarity, length in millimeters, width in millimeters, depth in millimeters and table width (width of the top of the diamond relative to the widest point).

Our interest is to model diamond prices conditional to their physical attributes. For this, we assume the response variable to be Gamma distributed, $Y|\mathbf{X} \sim Gamma(\alpha, \mu)$, with $\mu = \mathbb{E}(Y|\mathbf{X})$ the conditional mean and α the shape parameter. After cleaning up, the data was randomly assigned 70% to the training set, 20% to the validation set and 10% to the test set. For this analysis we consider two models - a generalized linear Gamma model (GLM) with logarithmic link function and a neural network model with negative Gamma log-likelihood loss function.

The neural network model architecture as well as both recalibrated models' target layer and nearest samples sizes are tuned and the final hyperparameter combinations are the ones that minimize loss, in the case of the network, and MSE in the validation set. The final network architecture is composed of an input layer, a hidden layer with 100 neurons and *ReLu* activation function, two hidden layers with 5 neurons and *ReLu* activation function and a forked output with two exponential layers, one for estimating the mean, μ, and the other with zero-constrained weights for estimating the shape parameter, α. Figure 5.1 shows the model's training and validation loss. The GLM is recalibrated in the input space with a neighborhood of 20% of the validation set and the neural network model is recalibrated in the output space with all information from the validation set.

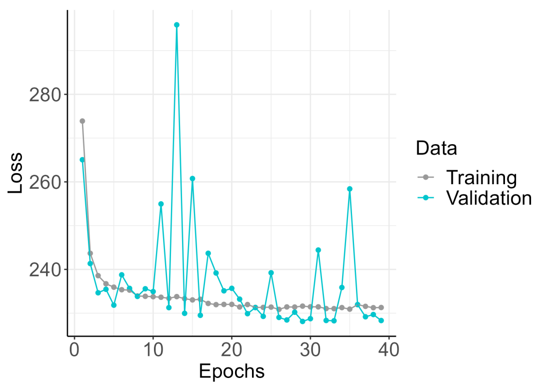
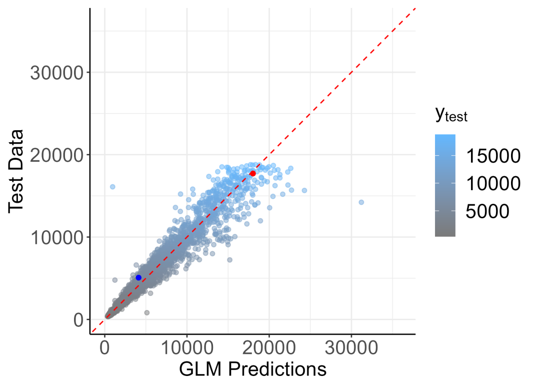


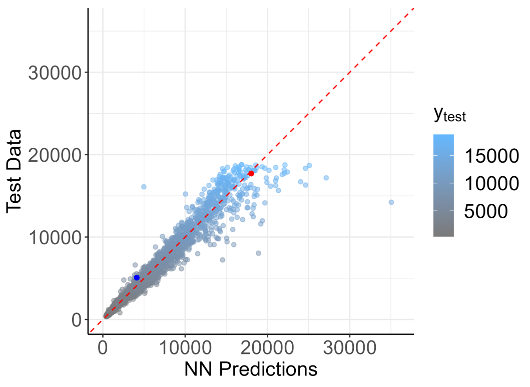
Figure 5.1: Neural network loss function during prediction and validation.

Figures 5.2a and 5.2b show greater dispersion for higher values of Y, meaning that both models perform worst in this interval compared to the lower values of Y. While both models overestimate predictions for higher values of Y, the neural network in general gives more precise estimations than GLM (Table 5.2). Two distinct points, $y^{(0)}_{test} = 4,113$ and $y^{(1)}_{test} = 18,026$, are highlighted in order to show model bias and recalibration effect in different regions of the dependent-variable space. Table 5.1 shows GLM and neural network estimates of the shape parameter and the exponential-family dispersion parameter, $\phi = \frac{1}{\alpha}$.

The cumulative probabilities histograms, in Figures 5.3a and 5.3b, show, respectively, GLM



(a) GLM predictions.



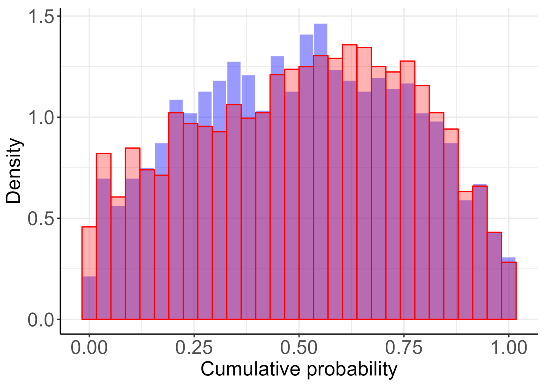
(b) NN predictions.

Figure 5.2: (a) GLM predictions variability in relation to the true data. (b) NN predictions variability in relation to the true data.

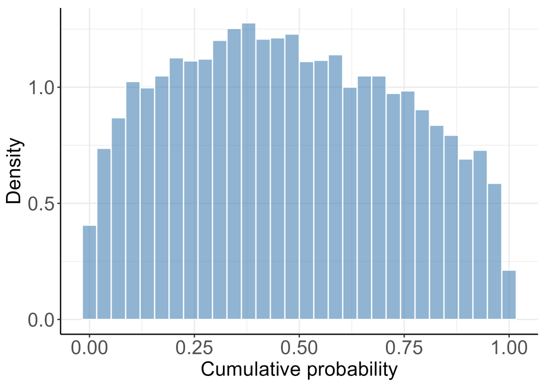
Parameter	GLM	ANN
Dispersion	0.0232	0.0175
Shape	43.1116	57.2838

Table 5.1: Dispersion (ϕ) and shape (α) parameters estimates.

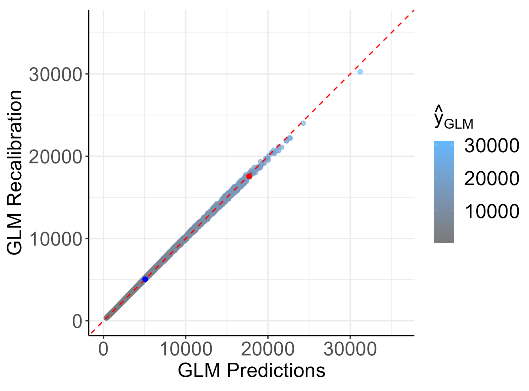
local estimation bias in the neighborhood of the highlighted points and global neural network estimation bias. The GLM cumulative probabilities show that model mostly overestimates variance for lower values of y while underestimating the median for higher values. On the other hand, the neural network globally overestimates the median at the same time it overestimates the variance of the distribution. In both cases, as Figures 5.3c and 5.3d show, the recalibration of the models produced lower predictions for higher values of y than the original models. Figures 5.3e and 5.3f compare the predictive distributions of all models for the highlighted points. It is shown that in all cases the recalibration effect appear to have decreased the variance as well as adjusted the distribution mean. Table 5.2 shows overall recalibration impact in both models. MSE metrics were improved with recalibration as well as interval metrics. Although GLM interval coverage presented a minor increase, its interval score was reduced, meaning the recalibration produced more accurate intervals.



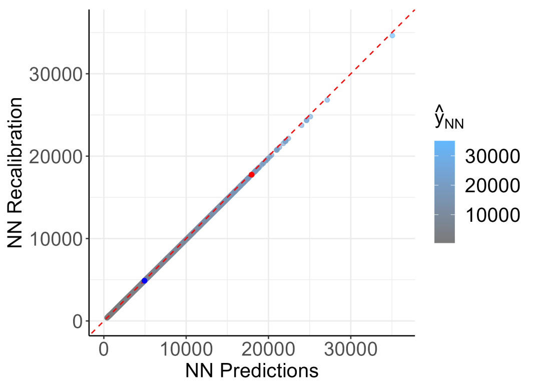
(a) GLM local cumulative probabilities.



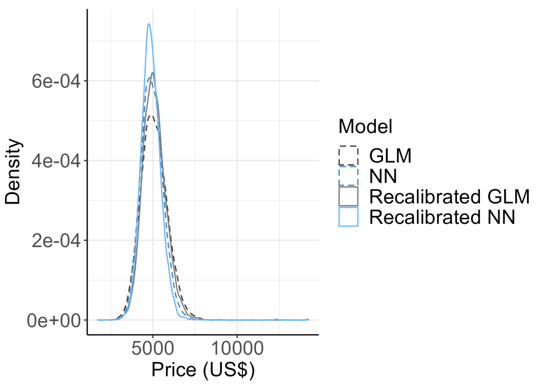
(b) NN global cumulative probabilities.



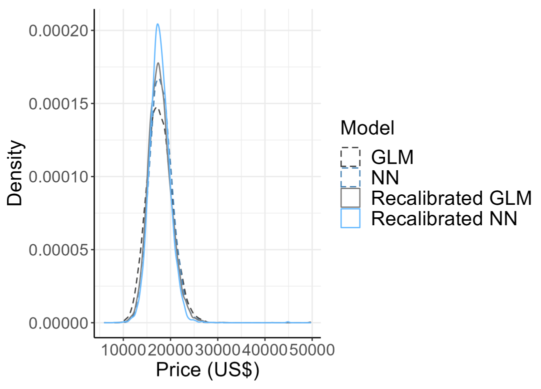
(c) GLM recalibration.



(d) NN recalibration.



(e) $y_{test}^{(0)}$ predictive distribution.



(f) $y_{test}^{(1)}$ predictive distribution.

Figure 5.3: (a) GLM local cumulative probabilities at $y_{test}^{(0)}$ (blue) and $y_{test}^{(1)}$ (red) neighbourhoods. (b) NN global cumulative probabilities. (c) Recalibration effect on GLM predictions. (d) Recalibration effect on NN predictions. (e) Estimated predictive distributions centered in $y_{test}^{(0)}$. (f) Estimated predictive distributions centered in $y_{test}^{(1)}$.

Model	MSE	Coverage	sMIS
GLM	758079.0	0.9737	0.7473
NN	722841.3	0.9749	0.7391
Recalibrated GLM	668665.0	0.9733	0.6792
Recalibrated NN	650431.6	0.9718	0.6651

Table 5.2: Performance comparison between all methods.

Chapter 6

Conclusion

This work presented a promising new method for (re)calibration of neural networks based on the cumulative probabilities of its predictive distributions. While there are many works with similar ideas, they mainly address the problem of calibration considering a network's output in its entirety, hence not capturing region-specific bias patterns. The proposed method generalizes this concept and recalibrates the networks locally, by selecting a proportion of the closest points to each new observation in any given layer.

The analyzed toy examples showed that the proposed method positively affected the Mean Squared Error, confidence intervals coverage and interval scores of uncalibrated models positively, generating better predictions and approximating the true distribution. Recalibrated models presented evenly distributed interval coverage whilst narrowing interval width, meaning they successfully corrected local prediction bias and variance estimation.

The simulation study investigates even further the effects of different parameter configurations in recalibration and compares its performance with the base models and the KNN regression model, which is very close to this work in terms of methodology. It was shown that, while training time is mostly affected by data dimensionality, prediction time also increases along with nearest samples size and decreases as target layer dimensionality and approximation level increase. Recalibration successfully provided better MSE and interval metrics in all cases presented. Overall, given the same conditions, recalibrated neural networks also proved to be much more efficient than the simple application of KNN regression. Against real data, recalibration was also shown to have positive effects on prediction precision and confidence interval metrics, adjusting the predictive distributions according to local bias.

This work uses two different methods to create probabilistic neural networks and generate the predictive distributions needed for recalibration. The first method consists of assuming a probabilistic model by choosing an appropriate loss function. One of the most popular choices of loss function in regression settings, the Mean Squared Error, assumes a Gaussian model. However, other distributions can be approximated, as was the case in Chapter 5 with the choice of the Gamma negative likelihood loss function. An empirical predictive distribution was also obtained from *Monte Carlo Dropout*, in Chapter 3, by activating the *dropout* layers during prediction. It is also possible to obtain a more precise empirical predictive distribution from quantile regression, by estimating the response variable's distribution conditional quantiles instead of its conditional mean – a method not explored in this work.

It might be possible to further improve recalibration shape approximations by applying corrections to the predictive distributions mean and variance beforehand, as proposed by Yu et al. (2021). While potentially greatly enhancing predictions and coverage, our method could become computationally expensive depending on the number of covariates and the choice of recalibration parameters. Quantile-based recalibration might contribute less in cases where it is possible (or feasible) to fit large, expensive models or there is access to a large enough data set.

References

Arya, Sunil et al. (1998). “An optimal algorithm for approximate nearest neighbor searching fixed dimensions”. *Journal of the ACM (JACM)* 45.6, pp. 891–923.

Bain, Lee J and Engelhardt, Max (1987). *Introduction to probability and mathematical statistics*. Brooks/Cole.

Beyer, Kevin et al. (1999). “When is “nearest neighbor” meaningful?” *International conference on database theory*. Springer, pp. 217–235.

Chen, Qi et al. (2021). “SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search”. *Advances in Neural Information Processing Systems* 34, pp. 5199–5212.

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

Gneiting, Tilmann and Raftery, Adrian E (2007). “Strictly proper scoring rules, prediction, and estimation”. *Journal of the American statistical Association* 102.477, pp. 359–378.

Guo, Chuan et al. (2017). “On calibration of modern neural networks”. *International Conference on Machine Learning*. PMLR, pp. 1321–1330.

Kuleshov, Volodymyr, Fenner, Nathan, and Ermon, Stefano (2018). “Accurate uncertainties for deep learning using calibrated regression”. *International Conference on Machine Learning*. PMLR, pp. 2796–2804.

Kullback, Solomon and Leibler, Richard A (1951). “On information and sufficiency”. *The annals of mathematical statistics* 22.1, pp. 79–86.

Kumar, Ananya, Liang, Percy, and Ma, Tengyu (2019). “Verified uncertainty calibration”. *arXiv preprint arXiv:1909.10155*.

Mukhoti, Jishnu et al. (2020). “Calibrating deep neural networks using focal loss”. *arXiv preprint arXiv:2002.09437*.

Platt, John et al. (1999). “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”.

Prangle, D et al. (2014). “Diagnostic tools for approximate Bayesian computation using the coverage property, Invited Paper”. *Australia and New Zealand Journal of Statistics* 56, pp. 309–329.

Rodrigues, GS, Prangle, Dennis, and Sisson, Scott A (2018). “Recalibration: A post-processing method for approximate Bayesian computation”. *Computational Statistics & Data Analysis* 126, pp. 53–66.

Tran, M-N et al. (2020). “Bayesian deep net GLM and GLMM”. *Journal of Computational and Graphical Statistics* 29.1, pp. 97–113.

Wickham, Hadley (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4. URL: `https://ggplot2.tidyverse.org`.

Yu, Xuejun et al. (2021). “Assessment and adjustment of approximate inference algorithms using the law of total variance”. *Journal of Computational and Graphical Statistics* 30.4, pp. 977–990.