Using Heteroscedastic Regression to Identify Model Bias

Anonymous Authors[1]

Abstract

Predictive uncertainty in a regression model arises due to three sources: aleatoric uncertainty, parameter uncertainty, and model inadequacy—sometimes called structural uncertainty. Most work has considered how to estimate parameter uncertainty, typically with Bayesian approaches. Estimating structural uncertainty, however, is much more difficult because it effectively corresponds to estimating the bias of the model. In this work, we investigate the utility of heteroscedastic regression for estimating the part of predictive uncertainty not captured by parameter uncertainty. We highlight two key properties: (1) the estimated variance per input provides an estimate of the aleatoric uncertainty and the structural uncertainty and (2) the optimization procedure naturally concentrates model capacity on a subset of the space, both reducing structural uncertainty in that subset and facilitating identification of what parts of the space can be well modelled. We design several synthetic experiments to elucidate these two properties, and show when heteroscedastic regression effectively models uncertainty due to model inadequacy.

1. Introduction

Over the past decade, neural networks have become the gold standard across a wide variety of applications in both regression and classification tasks (for instance, (Bengio et al., 2003; Hinton et al., 2012; Sermanet et al., 2014; Krizhevsky et al., 2017)). Under least squares regression, deep learning models output a point estimate of the mean of the conditional probability distribution for a given input, that is, $\hat{y} = \mathbb{E}[y|x]$. This output, however, does not tell us anything about the uncertainty of the prediction.

Capturing predictive uncertainty is important in many applications, particularly in high risk prediction tasks such as medical diagnostics (Yang et al., 2016) and autonomous vehicles (Kendall and Cipolla, 2016). Several recent papers have also demonstrated that uncertainty measures can be important in model-based reinforcement learning (model-based RL) (Kalweit and Boedecker, 2017; Kurutach et al., 2018; Abbas et al., 2020).

Predictive uncertainty in a learned regression model can stem from multiple sources. This paper discusses three primary sources of uncertainty: *aleatoric uncertainty* (stochasticity inherent in the data), *parameter uncertainty* (uncertainty about which parameters actually generated the data), and *model inadequacy* (the function class from which we have drawn our algorithm lacks the capacity to represent the true function of the data generating process).

Parameter uncertainty refers to uncertainty about the values of the parameters of our model, given a function class from which the model is drawn and all available data. Parameter uncertainty can be reduced through the collection of more data and eliminated in the presence of infinite data. There is a large body of work devoted to capturing parameter uncertainty in neural networks using Bayesian methods by approximating the posterior over parameters (MacKay, 1992; Neal, 1995; Hinton and van Camp, 1993; Barber and Bishop, 1998; Graves, 2011; Blundell et al., 2015; Gal and Ghahramani, 2016; Gal et al., 2017; Li and Gal, 2017). In large neural networks it is computationally intractable to compute the full posterior over parameters which has resulted in a significant body of research on techniques to approximate the posterior. Bayesian approximation methods have proved effective at capturing parameter uncertainty in neural networks, but they are not designed to capture aleatoric uncertainty or uncertainty due to model inadequacy.

An alternative approach to capturing parameter uncertainty is bootstrapping and model ensembling (Osband et al., 2016; Lakshminarayanan et al., 2017; Osband et al., 2018; Jain et al., 2020). Ensembling techniques work by training an ensemble of neural networks on independent samples of the data and using the empirical distribution over the parameters of the neural networks in the ensemble to estimate parameter uncertainty. For instance, the higher the variance in parameters of the networks, the higher the uncertainty. Similarly to Bayesian methods, these techniques have proved

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

reasonably effective at capturing parameter uncertainty, but do not reveal information regarding aleatoric uncertainty or model inadequacy.

Despite its importance, there has been less work devoted to capturing aleatoric uncertainty and uncertainty due to model inadequacy. Chryssolouris et al. (1996), Townsend and Tarassenko (1999), and Rivals and Personnaz (2000) used perturbation models to estimate parameter and aleatoric uncertainty. Perturbation models, however, require high dimensional weight covariance matrices making them difficult to use for neural networks with large numbers of weights (Zhang and Luh, 2005). Ding and He (2003) explored a different avenue to capturing uncertainty by applying Carroll and Ruppert (1988) regression transformation model. Their method is computationally expensive, however, and has not received much traction in neural network research. More recently Zhu and Laptev (2017) developed a technique to capture uncertainty due to model inadequacy using an encoder-decoder framework with Long Short Term Memory (LSTM) networks. This technique is restricted to time-series data.

The technique for capturing uncertainty that we analyze in this paper is heteroscedastic regression (Nix and Weigend, 1994; Nix and Weigend, 1995), a regression technique that attempts to learn estimates of both the mean and variance of the conditional probability distribution. Typically neural network regression objectives, e.g. least squares regression, assume homoscedasticity. That is, that the variability of the targets is assumed to be constant across all the data. In least squares regression we assume that the targets, Y_i, are a function of the inputs, X_i, and some unknown parameters θ, along with a noise or error term ϵ_i, i.e. $Y_i = f(X_i, \theta) + \epsilon_i$. Most regression models assume that the error term ϵ_i is independent of the inputs and is drawn independent and identically distributed (i.i.d.) from some distribution (in the case of least squares regression, a Gaussian distribution). While this assumption is mathematically convenient, it is not true in many applications. Heteroscedastic regression, on the other hand, instead assumes the noise to be input-dependent and the model tries to predict not only the mean of the conditional distribution, but also the variance.

Heteroscedastic regression is easy to implement and train and can take advantage of pre-existing deep learning frameworks and optimization techniques making it an attractive choice as a technique to capture uncertainty. Despite this, heteroscedastic regression has not been widely adopted in the machine learning community. We believe that this is at least partially due to a dearth of research rigorously examining the efficacy of the technique in the machine learning community.

Williams (1996) extended the original formulation of heteroscedastic regression to the multivariate case while Penny and Roberts (1997) used a technique inspired by heteroscedastic regression along with a committee of neural networks to attempt to capture uncertainty from all three sources. More recently Blum and François (2010) used the technique to approximate Bayesian inference. There has been some work investigating applications of heteroscedastic regression to various domains, such as Kendall and Gal (2017) to capture aleatoric uncertainty in computer vision, Ng et al. (2017) in predicting surgery times, and Abbas et al. (2020) to capture uncertainty due to model inadequacy in model based RL.

To our knowledge, however, there has not been a rigorous investigation into the soundness of heteroscedastic regression as a technique for capturing uncertainty due to model inadequacy and aleatoric uncertainty. This paper aims to fill that gap while also providing a comparison of heteroscedastic regression to least squares regression.

We provide an analysis of the optimal values of the estimates for the mean and variance of the conditional probability distribution based on the objective for heteroscedastic regression. We show that the Bayes Estimators for the mean and variance of $P(Y|X)$ are the conditional mean and the error due to model inadequacy and irreducible error, respectively. We then empirically investigate the effectiveness of heteroscedastic regression as a method for capturing uncertainty and providing robust estimates of the mean under a variety of assumptions.

2. Background

As mentioned above, in regression analysis we assume that the dependent variables are a function of the independent variables along with an additive noise term. The task is to learn this function $f_\theta : \mathbb{R} \to \mathbb{R}$ based on a given data set $\mathcal{D} = \{(x_i, y_i)_{i=1}^N\}$, where $x, y \in \mathbb{R}$ and f is parameterized by θ. Note that this can easily be extended to the multidimensional case where $\mathbf{x_i} \in \mathbb{R}^n$, $\mathbf{y_i} \in \mathbb{R}^m$, but for ease of exposition we consider the one-dimensional case here. In order to treat this task as an optimization problem, we pick an objective function to minimize.

In least squares regression, the objective which we seek to minimize is the sum of squares of the residuals, that is $L(\theta) = \sum_{i=1}^N (y_i - \hat{y}_i)^2$. The least squares objective corresponds to the maximum likelihood estimate under the assumption of input-independent, Gaussian error terms (Charnes et al., 1976).

The assumption of input-independent Gaussian error terms can equivalently be expressed as an assumption that the conditional distribution of the data generating process is Gaussian with a fixed variance for all inputs, $p(y|x) = \mathcal{N}(f_\mu(x), \sigma^2)$. From this perspective we can view the prediction $\hat{y} = f_{\hat{\mu}}(x)$ as an estimator of the mean of the conditional distribution, $P(Y|X)$, and we can use the mean-squared-error (MSE) of our estimator, $\mathbb{E}\left[\left(y_i - f_{\hat{\mu}}(x_i)\right)^2\right]$, as a measure of the generalzation error for our model.

MSE can be decomposed into three distinct sources of error (equation 1): *bias*, *variance*, and *irreducible error* (Geman et al., 1992). These three sources of error generally correspond to the three sources of predictive uncertainty mentioned earlier. Error due to bias results from model inadequacy, error due to variance is a result of parameter uncertainty, and irreducible error is due to aleatoric uncertainty. Both bias and variance can be reduced, although there is often a trade-off between these types of error in practice (Kohavi et al., 1996; Derumigny and Schmidt-Hieber, 2020), whereas irreducible error, as its name suggests, cannot be reduced as it is due to stochasticity inherent in the data generating process.

$$\begin{aligned}\mathbb{E}_{Y|X,\mathcal{D}}\left[\left(y - f_{\hat{\mu}}(x)\right)^2\right] &= \mathbb{E}_{Y|X}\left[\left(y - \mathbb{E}_{Y|X}[y|x]\right)^2\right] \\ &+ \mathbb{E}_{\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}[f_{\hat{\mu}}(x)] - f_{\hat{\mu}}(x)\right)^2\right] \\ &+ \left(\mathbb{E}_{Y|X}[y|x] - \mathbb{E}_{\mathcal{D}}[f_{\hat{\mu}}(x)]\right)^2\end{aligned} \tag{1}$$

The subscripts in the expectations indicate the source of randomness over which the expectation is being taken, where $\mathcal{D}$ refers to the data, as the estimator will vary depending on the data sampled. The first term in the decomposition is known as irreducible error, while the second and third terms represent variance and bias^2, respectively. Note that as more data is collected, the variance can be reduced, and in the limit, eliminated entirely.

The assumption of homoscedastic variance is often not true in practice and a relaxation of this assumption could enable a network to capture information about the variance in the data generating process. In heteroscedastic regression, we instead assume the variance of the Gaussian data generating process is dependent on the input, just like the mean. This gives us the new conditional probability distribution $p(y|x) = \mathcal{N}(f_\mu(x), f_{\sigma^2}(x))$ where now both $f_{\hat{\mu}}$ and $f_{\hat{\sigma}^2}$ are estimators of the mean and variance, respectively.

Maximizing the log-likelihood of equation (2) under these new assumptions leads to a different objective than that in least squares regression. This new objective is given in equation (3) below. Looking at this new objective we can see that the predicted variance, $f_{\hat{\sigma}^2}(x)$ acts as a kind of regularizer on the objective. In the additive term, we can see that the model is penalized logarithmically for predicting high variance. More interestingly, however, the squared residuals for the mean prediction are scaled by the inverse of the predicted variance. This means that when the regression model is unable to learn good estimates of the mean in certain regions of the inputs, the loss will be minimized by predicting higher variance. Intuitively this suggests that in regions of the data with high bias or irreducible error, the model should output a large $f_{\hat{\sigma}^2}(x)$, possibly capturing uncertainty due to model inadequacy and aleatoric uncertainty.

$$p(y|x) = \mathcal{N}(f_\mu(x), f_{\sigma^2}(x)) \tag{2}$$

$$L_i(\theta) = \frac{(y_i - f_{\hat{\mu}}(x_i))^2}{2 f_{\hat{\sigma}^2}(x_i)} + \frac{1}{2}\log f_{\hat{\sigma}^2}(x_i) \tag{3}$$

3. Optimal Solution for Heteroscedastic Regression

For least squares regression it is a well-established fact that the optimal predictor, in terms of minimizing the expected cost (also known as the Bayes estimator), is the mean of the conditional distribution, that is $f^*_{\hat{\mu}}(x) = \mathbb{E}[y|x]$.

We want to consider the optimal values for $f_{\hat{\sigma}^2}$ and $f_{\hat{\mu}}$ for every possible value of x. We denote the optimal values by $f^*_{\hat{\sigma}^2}$ and $f^*_{\hat{\mu}}$, respectively. For the sake of notational simplicity, we denote $f_{\hat{\sigma}^2}(x) = \hat{\sigma}^2$ and $f_{\hat{\mu}}(x) = \hat{\mu}$ in the following derivation. We perform our analysis with respect to the data generating distribution rather than with respect to a distribution over data samples.

$$\begin{aligned}f^*_{\hat{\sigma}^2}(x) &= \arg\min_{\hat{\sigma}^2} \mathbb{E}\left[C(\hat{\sigma}^2, \hat{\mu}, y)\right] \\ f^*_{\hat{\mu}}(x) &= \arg\min_{\hat{\mu}} \mathbb{E}\left[C(\hat{\sigma}^2, \hat{\mu}, y)\right]\end{aligned}$$

To solve for $f^*_{\hat{\sigma}^2}$ and $f^*_{\hat{\mu}}$ we use the expected cost as our objective.

$$\begin{aligned}\mathcal{L}(\hat{\sigma}^2, \hat{\mu}, y) &= \int_X p(x) \int_Y C(\hat{\sigma}^2, \hat{\mu}, y) p(y|x)\, dy\, dx \\ &= \int_X p(x) \int_Y \left(\frac{(\hat{\mu} - y)^2}{2\hat{\sigma}^2} + \frac{1}{2}\log \hat{\sigma}^2\right) p(y|x)\, dy\, dx\end{aligned}$$

To minimize this objective we only need to consider the inner integral. Taking the gradient, setting it equal to zero, and solving for $f^*_{\hat{\sigma}^2}$ and $f^*_{\hat{\mu}}$ we get the following.

$$\begin{aligned}\frac{\partial \mathcal{L}(\hat{\sigma}^2, \hat{\mu})}{\partial \hat{\sigma}^2} &= \int_Y \left(\frac{(\hat{\mu} - y)^2}{-2\hat{\sigma}^2} + \frac{1}{2\hat{\sigma}^2}\right) p(y|x)\, dy = 0 \\ \Rightarrow \hat{\sigma}^2 &= \int_Y (\hat{\mu} - y)^2 p(y|x)\, dy \\ \hat{\sigma}^2 &= \mathbb{E}\left[(\hat{\mu} - y)^2 | x\right]\end{aligned}$$

and

$$\frac{\partial \mathcal{L}(\hat{\sigma}^2, \hat{\mu})}{\partial \hat{\mu}} = \int_Y \left(\frac{(\hat{\mu} - y)}{\hat{\sigma}^2} \right) p(y|x)\, dy = 0$$
$$\Rightarrow \hat{\mu} = \int_Y y p(y|x)\, dy$$
$$\hat{\mu} = \mathbb{E}\left[y|x\right]$$

Despite having a different objective, the optimal value for heteroscedastic regression, $f^*_{\hat{\mu}}(x) = \mathbb{E}\left[y|x\right]$, is the same as in least squares regression. The optimal value for $f_{\hat{\sigma}^2}$ on the other hand is the squared residuals of the $f_{\hat{\mu}}$ network, $f^*_{\hat{\sigma}^2}(x) = \mathbb{E}\left[(f_{\hat{\mu}}(x) - y)^2|x\right]$.

The optimal solution for $f_{\hat{\sigma}^2}$ can be decomposed in a manner similar to the decomposition of MSE given in the background. We can see from this decomposition that in the infinite data regime, or for fixed a dataset, $f^*_{\hat{\sigma}^2}$ captures bias and irreducible error (equation (4)), demonstrating that heteroscedastic regression is a principled technique for capturing uncertainty due to model inadequacy and aleatoric uncertainty.

It is also worth noting that these optimal values do not depend on the data generating distribution being Gaussian, suggesting that heteroscedastic regression can still be a sound technique for capturing uncertainty even when the data is not normally distributed.

$$\mathbb{E}[(f_{\hat{\mu}}(x) - y)^2|x] = \overbrace{(f_{\hat{\mu}}(x) - \mathbb{E}[y|x])^2}^{bias^2} + \underbrace{\mathbb{E}[(\mathbb{E}[y|x] - y)^2|x]}_{\text{irreducible error}} \quad (4)$$

In the case of non-linear neural networks, it is not possible to derive a closed form solution for these optimal estimators, but if we restrict $f_{\hat{\mu}}$ to be linear in x, i.e. $f_{\hat{\mu}}(x_i) = w_\mu x_i$ where $w_\mu \in \mathbb{R}$, we can express $f^*_{\hat{\mu}}$ via a closed form solution. In this case, the optimal solution for $f_{\hat{\mu}}$ is that of weighted least squares regression and the closed form solution is shown below in equation (5), where $\Sigma \in \mathbb{R}^{N\times N}$, with $\frac{1}{f_{\hat{\sigma}^2}(x_i)}$ on the diagonal.

$$w_\mu = (X^T \Sigma X)^{-1} X^T \Sigma Y, \quad \text{where} \quad X, Y \in \mathbb{R}^{N\times 1} \quad (5)$$

4. Experiments

In this section, we perform several experiments to empirically validate the soundness of heteroscedastic regression as a technique for capturing uncertainty due to model inadequacy and aleatoric uncertainty. We use synthetic datasets in order to precisely investigate how heteroscedastic regression performs under different noise and function approximation regimes.

We also provide a comparison of the quality of the estimates of the conditional mean between heteroscedastic regression and least squares regression. We again use synthetic datasets to enable a precise analysis and develop a metric for assessing the quality of a regression model in order to facilitate our comparison of the two techniques.

4.1. Experimental Setup

We use two separate neural networks (i.e. they do not share weights) to learn $f_{\hat{\mu}}$ and $f_{\hat{\sigma}^2}$. The exact architecture in terms of number of hidden layers, units, and activation functions differs depending on the experiment, so we specify these details before explaining the individual experiments. To ensure the predicted variance, $f_{\hat{\sigma}^2}(x)$, is positive, we use soft-plus, $\ln(e^x + 1)$, as the last layer's activation function. For numerical stability, we added 10^{-6} to the predicted variance.

All experiments were run 30 times with each run consisting of 800 epochs. We use 2000 data points for each experiment. To choose the best learning rate for each of the experiments we performed a parameter sweep over 6 different step sizes $(2^{-5}, 2^{-7}, 2^{-9}, 2^{-11}, 2^{-13}, 2^{-15})$ and chose value that resulted in the best performance using the area under the curve (AUC) over the last 100 epochs. The networks were trained with mini-batch SGD with *Adam* as an optimizer with β_1 and β_2 equal to 0.9 and 0.999, respectively. The batch size was 128 for all experiments.

4.2. Empirical Verification of Optimal $f_{\hat{\sigma}^2}$

In the first set of experiments, we investigate learning $f_{\hat{\sigma}^2}$ while holding the $f_{\hat{\mu}}$ network fixed. The purpose of this experiment is to provide empirical validation for the optimal value of $f_{\hat{\sigma}^2}$. The analysis of section 3 suggests that a $f_{\hat{\sigma}^2}$ network of sufficient capacity should converge to $f^*_{\hat{\sigma}^2}(x) = \mathbb{E}\left[(f_{\hat{\mu}}(x) - y)^2|x\right]$ and should be able to learn this error whether it is due bias as a result of an $f_{\hat{\mu}}$ of insufficient capacity or irreducible error. The $f_{\hat{\sigma}^2}$ network had two hidden layers with 64 hidden units each and a ReLU activation function.

We designed four different experiments, each with their corresponding synthetic dataset, to analyze scenarios when uncertainty is due to model inadequacy, aleatoric uncertainty, or both. For each of the below, x was sampled from the interval $(0, 4)$. $U(0, 3)$ is the uniform distribution.

1. $y = 2x + \epsilon_x$, where $\epsilon_x \sim \mathcal{N}(0, 0.5x)$
2. $y = 2x + \begin{cases} x+1, & \text{for } x \in [2,3] \\ 0, & \text{else} \end{cases}$
3. $y = 2x + \epsilon_x$, where $\epsilon_x \sim \mathcal{N}\big(1, \sin(2x)\big)$

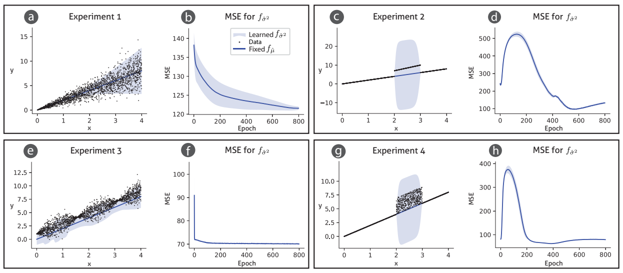

Figure 1. Plots (a), (c), (e), and (g) show the learned $f_{\hat{\sigma}^2}$, the fixed $f_{\hat{\mu}}$, and the data points. Plots (b), (d), (f), and (h) give the corresponding learning curves for each experiment. The shaded area in these plots represents the standard error.

4. $$y = 2x + \begin{cases} \epsilon_x \sim U(0,3), & \text{for } x \in [2,3] \\ 0, & \text{else} \end{cases}$$

For each of the experiments we fixed $f_{\hat{\mu}}(x) = 2x$. This decision was made to control the sources of error in each experiment. In experiment 1 the only source of uncertainty is aleatoric uncertainty, resulting in irreducible error. Experiment 2, on the other hand, is designed so that the only source of uncertainty due to model inadequacy as for $x \in (2,3)$ the $f_{\hat{\mu}}$ has not correctly modelled the data, resulting in a biased model. The data generating processes in experiments 3 and 4 exhibit both types of uncertainty, resulting in error due to bias and irreducible error.

The results of the experiments are shown in figure 1. The learned $f_{\hat{\sigma}^2}$ for each experiment are shown in the left hand plot while the corresponding learning curves are shown in the right hand plot for each of the 4 experiments. The shaded area in plots (a), (c), (e), and (g) represent the learned $f_{\hat{\sigma}^2}$ at the end of training. The experiments demonstrate that, as the analysis in section 3 predicted, the $f_{\hat{\sigma}^2}$ network learns the error from the $f_{\hat{\mu}}$ predictions whether this error is due to bias, irreducible error, or both, thus providing an accurate estimate of aleatoric uncertainty and structural uncertainty due to model inadequacy.

The loss shown in the learning curves is calculated as $\frac{1}{n}\sum_{i=1}^{n}[f_{\hat{\sigma}^2}(x) - (f_{\hat{\mu}}(x) - y)^2]^2$ where n is the number of data points. That is, it is the MSE of the variance network's predictions of the squared errors of the mean network. With a decaying learning rate and enough training examples, this loss would converge to zero.

4.3. Threshold Metric

In order to compare the performance of heteroscedastic regression with least squares regression, we need a meaningful metric to measure performance. It is not necessarily obvious what this metric should be. One option is to use the MSE between the true mean, f_{μ}, and the learned $f_{\hat{\mu}}$, but MSE gives equal weight to every data point, which does not necessarily provide a good measure of performance. For instance, if the noise in the data is input-dependent, we would expect to see higher MSE in regions of higher variance; however, because there is simply more noise in the data in these regions, high MSE does not necessarily correspond to poor performance of the learned $f_{\hat{\mu}}$. In fact, an algorithm that achieves lower MSE could well do a poorer job of approximating the true underlying function of the data generating process if it overfits to regions of high variance, compared to an algorithm that correctly recognizes the variance in these regions and succeeds in modelling the true underlying function.

To address this, we introduce two metrics: hard-threshold (equation 6) and soft-threshold (equation 7), to measure algorithm performance. The intuition behind these metrics is that in some cases we might prefer to have an optimal or near optimal estimate for a subset of the data and a poor estimate on other regions of the sampling domain, rather than having a sub-optimal answer for all the data.

For the hard threshold, we calculate the Euclidean distance ($d(x,y) = \|f_{\hat{\mu}}(x) - y\|_2^2$) between the predictions and the true value for each data point. If the distance is lower than some fixed threshold, our metric is equal to one for that data point (meaning the prediction is useful), and zero otherwise (equation 6). For the soft threshold metric, we again use the Euclidean distance as an input to our soft-threshold function (equation 7) which outputs a real number in the range of $(0,1]$, where 0 represents a poor prediction and 1 represents a perfect prediction.

Both of these threshold functions have a hyper-parameter η that changes the strictness of the functions. For the hard-threshold function, the lower the η the stricter the function (i.e. the prediction must be closer to the true value to be considered successful) whereas a lower η corresponds to a relaxation of the threshold for the soft-threshold function. We take the average of the values of the threshold functions over all the data points and use it as a measure of performance for the $f_{\hat{\mu}}$ model. The σ below is the sigmoid function.

$$\text{Hard-Threshold}(d(x,y),\eta) = \begin{cases} 1, & \text{if } d(x,y) \le \eta \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$$\text{Soft-Threshold}(d(x,y),\eta) = 2\Big(1 - \sigma\big(\eta d(x,y)\big)\Big) \quad (7)$$

4.4. Comparison of Heteroscedastic and Least Squares Regression

The second set of experiments investigates the performance of heteroscedastic regression compared to least squares regression under several regimes chosen to explore the effect of model capacity and sources of uncertainty. We examine two scenarios, one in which $f_{\hat{\sigma}^2}$ has sufficient capacity to approximate the true error of the $f_{\hat{\mu}}$ network and a second in which the network has insufficient capacity to learn the true error. For all of the following experiments the $f_{\hat{\mu}}$ network and $f_{\hat{\sigma}^2}$ network are trained simultaneously on the heteroscedastic regression objective.

We compare the performance of the heteroscedastic and least squares regression models using MSE and our two threshold metrics. It is important to note that because least squares regression is being trained on an MSE objective, it should always achieve lower MSE than heteroscedastic regression, but as explained above, this is not necessarily a good measure of performance.

4.4.1. Sufficient Network Capacity for $f_{\hat{\sigma}^2}$

In this scenario, the capacity of the $f_{\hat{\sigma}^2}$ network is sufficient to approximate the true error from the learned $f_{\hat{\mu}}$ network. The $f_{\hat{\mu}}$ network for both heteroscedastic and least squares regression, however, is restricted to the class of linear functions. Similar to the above experiments, we again design synthetic datasets with sources of error coming exclusively due to model inadequacy, exclusively due to irreducible error, or both. The $f_{\hat{\sigma}^2}$ for these experiments network had two hidden layers with 64 hidden units each and a ReLU activation function.

Model inadequacy as the sole source of error: For this case, we designed two experiments each with a corresponding dataset defined as follows.

1. $y = \frac{1}{2}x^2, \quad x \in (-4, 4)$

2. $y = 2x + \begin{cases} x+1, & \text{for } x \in [2,3] \\ 0, & \text{for } x \in (0,2) \cup (3,4) \end{cases}$

The results of these two experiments are given in figures 2 and 3. The learned models using heteroscedastic and least squares regression are shown in plots (a) and (d) as the blue and red lines, respectively. As in the experiments with a fixed $f_{\hat{\mu}}$, the $f_{\hat{\sigma}^2}$ network successfully learns an accurate approximation of the error due to model bias.

Due to the weighting from $f_{\hat{\sigma}^2}(x)$ in the objective for heteroscedastic regression, the $f_{\hat{\mu}}$ network learns a different function than the network in least squares regression despite having an identical architecture. The least squares model tries to minimize the error of the learned mean for the whole input space equally, while the $f_{\hat{\mu}}$ network from heteroscedastic regression focuses its resources on areas of the data where it is able to learn a good function approximation. As a result, the MSE (plots (b) and (e)) is lower for least squares regression.

When we compare the two regression techniques using our hard and soft threshold metrics we see a different story.

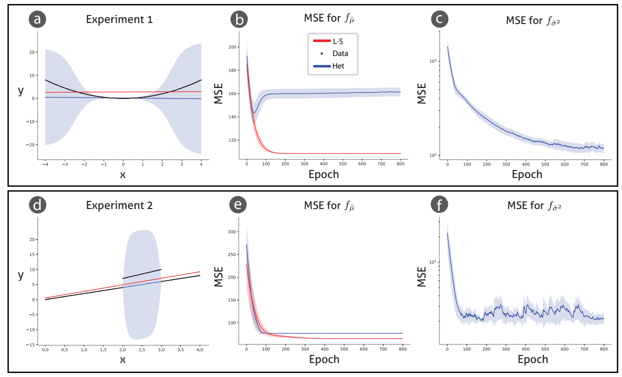

Figure 2. Plots (a), (d) show the learned models and data. The learned $f_{\hat{\sigma}^2}$ is the shaded region. Plots (b) and (e) show the learning curve for heteroscedastic and least squares models. Plots (c) and (f) show the learning curve for $f_{\hat{\sigma}^2}$. The shaded area is the standard error.

Heteroscedastic regression consistently outperforms least squares regression on over 120 different threshold values for both the hard and soft thresholds. This is because by concentrating resources on areas of the input space where the $f_{\hat{\mu}}$ network is able to achieve lower error, the model is able to avoid overfitting to uncertainty caused by model misspecification. Hence, not only is heteroscedastic regression capturing meaningful information about the certainty of the predictions, but it is, at least by certain metrics, actually learning better predictions about the mean than least squares regression.

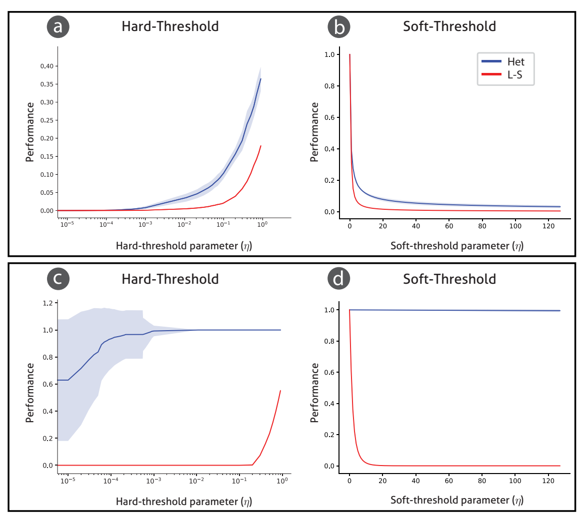

Figure 3. Plots (a) and (c) show performance according to the hard threshold metric, while plots (b) and (d) are for the soft threshold metric. The shaded area is the standard error. The first and second rows show the results of the first and second experiment, respectively.

Aleatoric uncertainty as the sole source of error: To examine the case of aleatoric uncertainty as the sole source of error we designed one experiment using the same synthetic data generation process as in an earlier experiment.

$$y \;=\; 2x + \mathcal{N}(0, 0.5x), \quad x \in (0, 4)$$

The $f_{\hat{\mu}}$ network is again linear and has an identical architecture for both heteroscedastic and least squares regression. The underlying function of the data generating process is linear and the heteroscedastic noise is distributed symmetrically about this function, so we would expect both heteroscedastic and least squares regression to successfully learn to approximate this function.

The plots of the learned functions and evaluation metrics can be found in the appendix in the interest of space because the models did in fact perform exactly as expected. Both heteroscedastic and least squares regression do indeed learn the underlying function $y = 2x$ and achieve the same MSE. The $f_{\hat{\sigma}^2}$ network also successfully learns the irreducible error. The two techniques are also nearly identical according to our threshold metrics, but heteroscedastic regression is more stable over the 30 runs. In this scenario, we observe that the two regression techniques perform equally well, but that heteroscedastic regression has learned an accurate estimate of the uncertainty of the model, thus providing us with more information regarding the distribution of the data and a measure of confidence in the predictions from the $f_{\hat{\mu}}$ network.

Model Inadequacy and Aleatoric Uncertainty as Sources of Error: To investigate this scenario we again re-used a data generating process from an earlier experiment.

$$y \;=\; 2x + \begin{cases} \epsilon_x \sim U(0, 3), & \text{for } x \in [2, 3] \\ 0, & \text{for } x \in (0, 2) \cup (3, 4) \end{cases}$$

For $x \in [2, 3]$ the regression models are not able to adequately fit the data as it would require a piecewise function to learn $\mathbb{E}[y|x]$ in this region and, additionally, there is a large amount of irreducible error in this interval. The results of this experiment are given in figure 4. Least squares and heteroscedastic regression learn slightly different functions as least squares regression tries to minimize the MSE over the whole input space whereas heteroscedastic regression focuses its resources on the intervals $(0, 2)$ and $(3, 4)$ where it is able to learn the true underlying function exactly.

The $f_{\hat{\sigma}^2}$ network successfully learns to model the squared errors of the $f_{\hat{\mu}}$ network. It is worth noting that even though the $f_{\hat{\mu}}$ network consistently predicts a value too small in the interval $[2, 3]$, the $f_{\hat{\sigma}^2}$ network is learning the squared errors, so its predictions are distributed symmetrically about the learned mean.

Comparing heteroscedastic and least squares regression in this regime according to our three metrics, we once again see that least squares achieve a slightly lower MSE, but heteroscedastic regression outperforms least squares according to the two threshold metrics.

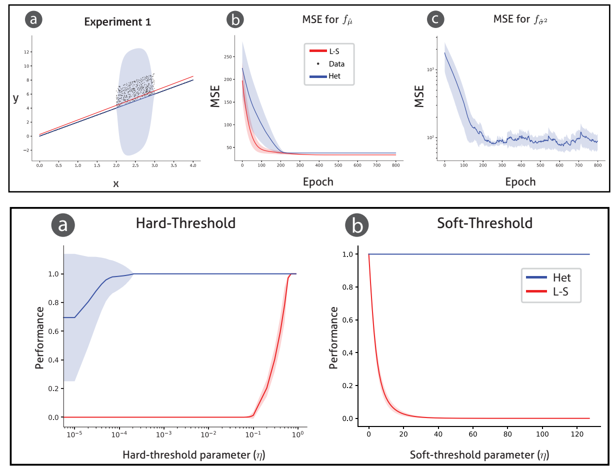

Figure 4. First row: Plot (a) shows the learned models and data. The learned $f_{\hat{\sigma}^2}$ is the shaded region. Plots (b) shows the learning curve for heteroscedastic and least squares models. Plots (c) shows the learning curve for $f_{\hat{\sigma}^2}$. The shaded area is the standard error. Second row: Plot (a) shows performance according to the hard threshold metric, while plot (b) is for the soft threshold metric. The shaded area is the standard error.

4.4.2. Insufficient Network Capacity for $f_{\hat{\sigma}^2}$

Up to this point, all of the experiments have considered scenarios where $f_{\hat{\sigma}^2}$ is drawn from a function class with sufficient capacity to learn the error from the $f_{\hat{\mu}}$ network. This is obviously not necessarily always true. To investigate the impact of having an $f_{\hat{\sigma}^2}$ of insufficient capacity, we designed an experiment in which the $f_{\hat{\mu}}$ network has sufficient capacity to learn the conditional mean of the data, but $f_{\hat{\sigma}^2}$ is unable to properly model the error.

This has the potential to have adverse effects on the performance of heteroscedastic regression because the objective function is weighted by $\frac{1}{f_{\hat{\sigma}^2}}$. Poor predictions of $f_{\hat{\sigma}^2}$ could result in a misallocation of resources from the $f_{\hat{\mu}}$ network. In order to avoid a feedback loop where a prediction of high variance leads to a poor prediction of the mean, further entrenching the prediction of high variance, one can add a constant to the predictions from the $f_{\hat{\sigma}^2}$ network and decay this constant over time. To this end, we added a constant of 5 x $10^{\frac{-\text{epoch}\#}{100}}$ to the predictions of the $f_{\hat{\sigma}^2}$ network.

In our experiment $f_{\hat{\sigma}^2}$ is linear, but the noise in the data is non-linear. We used five different architectures for $f_{\hat{\mu}}$. Each $f_{\hat{\mu}}$ network has two hidden layers, but differing numbers of hidden units increasing in multiples of four from 8 to 32 hidden units. As in previous experiments, the least squares regression architectures are identical to that of the

$f_{\hat{\mu}}$ networks. The data generating process is given below.

$$y = \sin(5x) + 2 + \begin{cases} \epsilon_x \sim U(-2, 2), & \text{for } x \in (2, 4) \\ 0, & \text{else} \end{cases}$$

The result of this experiment is shown in figure 5 while plots of evaluation metrics can be found in the appendix. The linear $f_{\hat{\sigma}^2}$ is unable to correctly model the error, resulting in predictions of relatively uniform variance across the full input space for the smaller $f_{\hat{\mu}}$ networks, and an exploding variance prediction for the $f_{\hat{\mu}}$ network with 32 hidden units.

The poor estimates of variance also have adverse effects on the predictions of the mean for heteroscedastic regression. For least squares regression, the network is able to learn a good approximation of the conditional mean of the data (the sin curve) for all network sizes, whereas the $f_{\hat{\mu}}$ network for the heteroscedastic regression model does not learn a really good approximation until the network has at least 24 hidden units. According to our threshold metrics, least squares outperforms heteroscedastic regression in each case, except for networks with 32 hidden units, in which case the performance is almost identical.

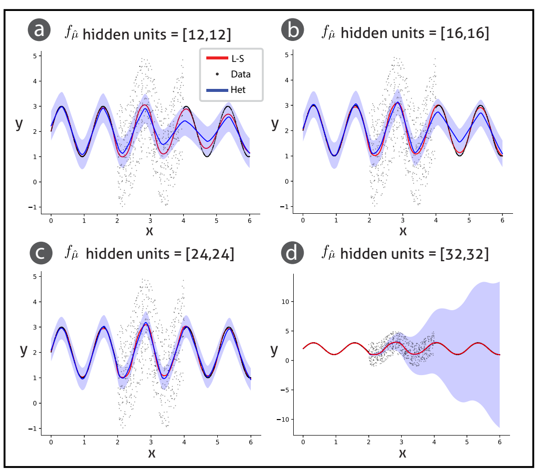

Figure 5. Plots show the learned models and data. The learned $f_{\hat{\sigma}^2}$ is the shaded region.

These results suggest that in scenarios where the $f_{\hat{\sigma}^2}$ network does not have sufficient capacity to model the error it can result in poor predictions from $f_{\hat{\sigma}^2}$, as well adversely effecting the predictions of $f_{\hat{\mu}}$ network.

4.4.3. Homoscedastic Noise

As a final piece of analysis, we investigated the performance of heteroscedastic and least squares regression in the presence of homoscedastic noise. Details of this experiment can be found in the appendix, but, interestingly, heteroscedastic regression was able to perform as least as well as least squares according to our threshold metrics while also providing accurate estimates of the uncertainty. This is despite least squares regression explicitly assuming homoscedasticity of the data.

5. Conclusion

This paper has attempted to provide a rigorous analysis of heteroscedastic regression as a technique for capturing uncertainty. Previous research has found success in applying heteroscedastic regression to machine learning problems, but we do not believe that there has been a robust investigation of the type of uncertainty captured by the technique, nor the scenarios in which it may or may not be effective. To this end, we designed experiments to evaluate heteroscedastic regression in a variety of regimes in which we were able to precisely control the sources of uncertainty in the model.

Our experiments demonstrate that heteroscedastic regression is effective at capturing uncertainty due to model inadequacy as well as aleatoric uncertainty. When the $f_{\hat{\sigma}^2}$ network has sufficient capacity to model this uncertainty it converges to an estimate of the model bias plus irreducible error. In addition to providing a sound technique to capture uncertainty, heteroscedastic regression has the potential to improve estimates of the conditional mean under certain conditions, as it prevents the regression model from overfitting to noise by weighting the loss function.

When the $f_{\hat{\sigma}^2}$ network does not have sufficient capacity to model the uncertainty, the technique is much less effective and is likely to perform worse than least squares regression. Initializing the network with a large constant that is added to its predictions and decaying this constant over time helps to alleviate the problems caused by poor predictions $f_{\hat{\sigma}^2}(x)$, but heteroscedastic regression still consistently performed worse than least squares regression, especially for smaller $f_{\hat{\mu}}$ networks. Finally, we investigated the performance of heteroscedastic regression in the presence of homoscedastic noise and found that it is able to perform as well as least squares regression while also providing robust estimates of the predictive uncertainty of the model.

We hope that this analysis encourages researchers and practitioners to further explore the use of heteroscedastic regression as a technique for capturing predictive uncertainty and learning robust regression models under a variety of noise regimes. We feel that it is a neglected and underutilized technique that complements other more well-studied techniques for capturing parameter uncertainty, while also being easy to implement within existing frameworks.

References

Abbas, Z., Sokota, S., Talvitie, E., and White, M. (2020). Selective dyna-style planning under limited model capacity. In *International Conference on Machine Learning*, pages 1–10.

Barber, D. and Bishop, C. M. (1998). Ensemble learning in bayesian neural networks. In *Neural Networks and Machine Learning*, pages 215–237.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Blum, M. G. and François, O. (2010). Non-linear regression models for approximate bayesian computation. *Statistics and Computing*, 20(1):63–73.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622.

Carroll, R. J. and Ruppert, D. (1988). *Transformation and Weighting in Regression*, volume 30. CRC Press.

Charnes, A., Frome, E. L., and Yu, P. L. (1976). The equivalence of generalized least squares and maximum likelihood estimates in the exponential family. *Journal of the American Statistical Association*, 71(353):169–171.

Chryssolouris, G., Lee, M., and Ramsey, A. (1996). Confidence interval prediction for neural network models. *IEEE Transactions on Neural Networks*, 7(1):229–232.

Derumigny, A. and Schmidt-Hieber, J. (2020). On lower bounds for the bias-variance trade-off. *arXiv preprint arXiv:2006.00278*.

Ding, A. A. and He, X. (2003). Backpropagation of pseudo-errors: neural networks that are adaptive to heterogeneous noise. *IEEE Transactions on Neural Networks*, 14(2):253–262.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

Gal, Y., Hron, J., and Kendall, A. (2017). Concrete dropout. In *Advances in Neural Information Processing Systems*, volume 30, pages 3581–3590.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, page 5–13.

Jain, S., Liu, G., Mueller, J., and Gifford, D. (2020). Maximizing overall diversity for improved uncertainty estimates in deep ensembles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4264–4271.

Kalweit, G. and Boedecker, J. (2017). Uncertainty-driven imagination for continuous deep reinforcement learning. In *Conference on Robot Learning*, pages 195–206.

Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE International Conference on Robotics and Automation*, pages 4762–4769.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5574–5584.

Kohavi, R., Wolpert, D. H., et al. (1996). Bias plus variance decomposition for zero-one loss functions. In *International Conference on Machine Learning*, volume 96, pages 275–83.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. (2018). Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

Li, Y. and Gal, Y. (2017). Dropout inference in bayesian neural networks with alpha-divergences. In *International Conference on Machine Learning*, pages 2052–2061.

MacKay, D. J. (1992). Bayesian methods for adaptive models. PhD Dissertation, California Institute of Technology.

Neal, R. (1995). Bayesian learning for neural networks. PhD thesis, Department of Computer Science, University of Toronto.

Ng, N. H., Gabriel, R. A., McAuley, J., Elkan, C., and Lipton, Z. C. (2017). Predicting surgery duration with neural heteroscedastic regression. In *Machine Learning for Healthcare Conference*, pages 100–111.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks*, volume 1.

Nix, D. A. and Weigend, A. S. (1995). Learning local error bars for nonlinear regression. In *Advances in Neural Information Processing Systems*, pages 489–496.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems*, 29:4026–4034.

Penny, W. D. and Roberts, S. J. (1997). Neural network predictions with error bars. *Dept. of Electrical and Electronic Engineering*.

Rivals, I. and Personnaz, L. (2000). Construction of confidence intervals for neural networks based on least squares estimation. *Neural Networks*, 13(4-5):463–484.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*.

Townsend, N. W. and Tarassenko, L. (1999). Estimations of error bounds for neural-network function approximators. *IEEE Transactions on Neural Networks*, 10(2):217–230.

Williams, P. M. (1996). Using neural networks to model conditional multivariate densities. *Neural Computation*, 8(4):843–854.

Yang, X., Kwitt, R., and Niethammer, M. (2016). Fast predictive image registration. In *Deep Learning and Data Labeling for Medical Applications*, pages 48–57.

Zhang, L. and Luh, P. B. (2005). Neural network-based market clearing price prediction and confidence interval estimation with an improved extended kalman filter method. *IEEE Transactions on Power Systems*, 20(1):59–66.

Zhu, L. and Laptev, N. (2017). Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops*, pages 103–110.