An implicit function learning approach for parametric modal regression

Yangchen Pan, Ehsan Imani
Univ. of Alberta & Amii
{pan6,imani}@ualberta.ca

Amir-massoud Farahmand
Vector Institute & Univ. of Toronto
farahmand@vectorinstitute.ai

Martha White
Univ. of Alberta
whitem@ualberta.ca

Abstract

For multi-valued functions—such as when the conditional distribution on targets given the inputs is multi-modal—standard regression approaches are not always desirable because they provide the conditional mean. Modal regression algorithms address this issue by instead finding the conditional mode(s). Most, however, are nonparametric approaches and so can be difficult to scale. Further, parametric approximators, like neural networks, facilitate learning complex relationships between inputs and targets. In this work, we propose a parametric modal regression algorithm. We use the implicit function theorem to develop an objective, for learning a joint function over inputs and targets. We empirically demonstrate on several synthetic problems that our method (i) can learn multi-valued functions and produce the conditional modes, (ii) scales well to high-dimensional inputs, and (iii) can even be more effective for certain uni-modal problems, particularly for high-frequency functions. We demonstrate that our method is competitive in a real-world modal regression problem and two regular regression datasets.

1 Introduction

The goal in regression is to find the relationship between the input (observation) variable $X \in \mathcal{X}$ and the output (response) $Y \in \mathcal{Y}$ variable, given samples of (X, Y). The underlying premise is that there exists an unknown underlying function $g^* : \mathcal{X} \mapsto \mathcal{Y}$ that maps the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. We only observe a noise-contaminated value of that function: sample (x, y) has $y = g^*(x) + \eta$ for some noise η. If the goal is to minimize expected squared error, it is well known that $\mathbb{E}[Y|x]$ is the optimal predictor (Bishop, 2006). It is common to use Generalized Linear Models (Nelder & Wedderburn, 1972), which attempt to estimate $\mathbb{E}[Y|x]$ for different uni-modal distribution choices for $p(y|x)$, such as Gaussian (l_2 regression) and Poisson (Poisson regression). For multi-modal distributions, however, predicting $\mathbb{E}[Y|x]$ may not be desirable, as it may correspond to rarely observed y that simply fall between two modes. Further, this predictor does not provide any useful information about the multiple modes.

Modal regression is designed for this problem, and though not widely studied in the general machine learning community, has been actively studied in statistics. Most of the methods are non-parametric, and assume a single mode Lee (1989); Lee & Kim (1998); Kemp & Silva (2012); Yu & Aristodemou (2012); Yao & Li (2014); Lv et al. (2014); Feng et al. (2017). The basic idea is to adjust target values towards their closest empirical conditional modes, based on a kernel density estimator. These methods rely on the chosen kernel and may have issues scaling to high-dimensional data due to issues in computing similarities in high-dimensional spaces. There is some recent work using quantile regression to estimate conditional modes (Ota et al., 2018), and though promising for a parametric approach, is restricted to linear quantile regression.

A parametric approach for modal regression would enable these estimators to benefit from the advances in learning functions with neural networks. The most straightforward way to do so is to

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

learn a mixture distribution, such as with conditional mixture models with parameters learned by a neural network (Powell, 1987; Bishop, 1994; Williams, 1996; Husmeier, 1997; Husmeier & Taylor, 1998; Zen & Senior, 2014; Ellefsen et al., 2019). The conditional modes can typically be extracted from such models. Such a strategy, however, might be trying to solve a harder problem than is strictly needed. The actual goal is to simply identify the conditional modes, without accurately representing the full conditional distribution. Training procedures for the conditional distribution can be more complex. Methods like EM can be slow (Vlassis & Krose, 1999) and some approaches have opted to avoid this altogether by discretizing the target and learning a discrete distribution (Weigend & Srivastava, 1995; Feindt, 2004). Further, the mixture requires particular sensitive probabilistic choices to be made such as the number of components.

In this paper, we propose a new parametric modal regression approach, by developing an objective to learn a parameterized function $f(x, y)$ on both input feature and target/output. We use the Implicit Function Theorem (Munkres, 1991), which states that if we know the input-output relation in the form of an implicit function, then a general multi-valued function, under certain gradient conditions, can locally be converted to a single-valued function. We learn a function $f(x, y)$ that approximates such local functions, by enforcing the gradient conditions. We introduce a regularization, that enables the number of discovered modes to be controlled, and also prevents spurious modes. We empirically demonstrate that our method can effectively learn the conditional modes on several synthetic problems, and that it scales well when the input is made high-dimensional. We also show an interesting benefit that the joint representation learned over x and y appears to improve prediction performance even for uni-modal problems, for high frequency functions where the function values change quickly between nearby x. To test the approach on real data, we propose a novel strategy to use a regular regression dataset to test modal regression algorithms. We demonstrate the utility of our method on this real world modal regression dataset, as well two regular regression datasets.

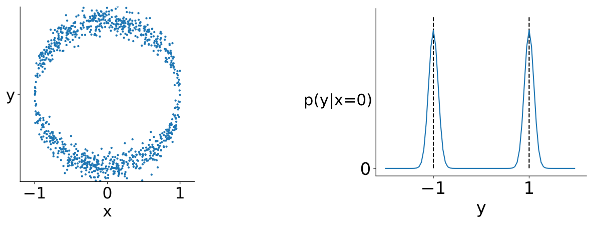
2 Problem Setting

We consider a standard learning setting where we observe a dataset of n samples, $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^n$. Instead of the standard regression problem, however, we tackle the modal regression problem. The goal in modal regression is to find the set of conditional modes

$$M(x) = \left\{ y : \frac{\partial p(x,y)}{\partial y} = 0, \ \frac{\partial^2 p(x,y)}{\partial y^2} < 0 \right\}$$

where in general $M(x)$ is a multi-valued function. Consider the example in Figure 1. For $x = 0$, the two conditional modes are $y_1 = -1.0$ and $y_2 = 1.0$.

The standard approaches to find conditional modes involve learning $p(y|x)$ or using non-parametric methods to directly estimate the conditional modes. For example, for a conditional Gaussian Mixture Model, a relatively effective approximation of these modes are the means of the conditional Gaussians. More generally, to get precise estimates, non-parametric algorithms are used, like the mean-shift algorithm (Yizong Cheng, 1995). These algorithms attempt to cluster points based on x and y, to find these conditional modes. We refer readers to Chen (2018); Chen et al. (2014) for a detailed review.



(a) The Circle dataset (b) Distribution $p(y|x = 0)$

Figure 1: (a) The Circle ataset is a synthetic dataset generated by uniformly sampling $x \in (-1, 1)$, and then sampling y from $0.5\mathcal{N}(\sqrt{1-x^2}, 0.1^2) + 0.5\mathcal{N}(-\sqrt{1-x^2}, 0.1^2)$. (b) The bimodal conditional distribution over y, for $x = 0$.

3 An implicit function learning approach

In this section, we develop an objective to facilitate learning parametric functions for modal regression. The idea is to directly learn a parameterized function $f(x, y)$ instead of a function taking only x as input. The approach allows for a variable number of conditional modes for each x. Further, it allows us to take advantage of general parametric function approximators, like neural networks, to identify these modal manifolds that capture the relationship between the conditional modes and x.

3.1 Implicit Function Learning Objective

Consider learning an $f(x, y)$ such that $f(x, y) = 0$ for all conditional modes and non-zero otherwise. Such a strategy—finding $f(x, y) = 0$ for all conditional modes—is flexible in that it allows for a different number of conditional modes for each x. The difficulty with learning such an f, particularly under noisy data, is constraining it to be zero for conditional modes y_j and non-zero otherwise. To obtain meaningful conditional modes $y_1, \ldots, y_{m_x}$ for x, the y around each y_j should be described by the same mapping $g_j(x)$. The existence of such g_j is guaranteed by the Implicit Function Theorem (Munkres, 1991)[1] as described below, under one condition on f.

Implicit Function Theorem: Let $f : \mathbb{R}^d \times \mathbb{R}^k \mapsto \mathbb{R}^k$ be a continuously differentiable function. Fix a point $(a, b) \in \mathbb{R}^d \times \mathbb{R}^k$ such that $f(a, b) = \mathbf{0}$, for $\mathbf{0} \in \mathbb{R}^k$. If the Jacobian matrix J, where the element in the ith row and jth column is $J_{[ij]} = \frac{\partial f(a,b)[i]}{\partial y[j]}$, has nonzero determinant, then there exists open sets $\mathcal{U}_a, \mathcal{V}_b$ containing a, b respectively, s.t. $\forall x \in \mathcal{U}_a$, $\exists$ an unique $y \in \mathcal{V}_b$ satisfying $f(x, y) = \mathbf{0}$. That is, $\exists g : \mathcal{U}_a \mapsto \mathcal{V}_b$. Furthermore, such a function $g(\cdot)$ is continuously differentiable and its derivative can be found by differentiating $f(x, g(x)) = \mathbf{0}$.

We focus on the one dimensional case (i.e. $k = 1$) in this work. The theorem states that if we know the relationship between independent variable x and dependent variable y in the form of implicit function $f(x, y) = 0$, then under certain conditions, we can guarantee the existence of some function defined locally to express y given x. For example, a circle on two dimensional plane can be expressed as $\{(x, y)|x^2 + y^2 = 1\}$, but there is no definite expression (single-valued function) for y in terms of x. However, given a specific point on the circle (x_0, y_0) $(y_0 \neq 0)$, there exists an explicit function defined locally around (x_0, y_0) to express y in terms of x. For example, at $x_0 = 0$, we have two local functions g_1 and g_2: for the positive quadrant we have $y_0 = g_1(x_0) = \sqrt{1 - x_0^2}$ and for the negative quadrant we have $y_0 = g_2(x_0) = -\sqrt{1 - x_0^2}$. Notice that at $y_0 = 0$, the condition required by the implicit function theorem is not satisfied: $\frac{\partial(x^2+y^2-1)}{\partial y} = 2y = 0$ at $y_0 = 0$, and so is not invertible.

Obtaining such smooth local functions g enables us to find these smooth modal manifolds. The conditional modes $g_1, \ldots, g_{m_x}$ satisfy $f(x, g_j(x)) = 0$ and $\frac{\partial f(x,g_j(x))}{\partial y} \neq 0$, where $m_x \in \mathbb{N}$ is the number of conditional modes for x. When training f, we can attempt to satisfy both conditions to ensure existence of the g_j. The gradient condition ensures that for y locally around $f(x, g_j(x))$, we have $f(x, y) \neq 0$. This encourages the other requirement that $f(x, y)$ be non-zero for the y that are not conditional modes at least within the neighborhood of the true mode. We include more discussion about avoiding spurious modes in Section 3.2.

In summary, given a training set $\{(x_i, y_i)\}_{i=1:n}$, we want to push $f(x_i, y_i) = 0$ and $\frac{\partial f(x_i,y_i)}{\partial y} \neq 0$. This naturally leads to the below modal regression objective:

$$L(\theta) \overset{\text{def}}{=} \sum_{i=1}^{n} l_\theta(x_i, y_i) \qquad \text{where} \qquad l_\theta(x, y) \overset{\text{def}}{=} f_\theta(x, y)^2 + \left(\frac{\partial f_\theta(x, y)}{\partial y} + 1\right)^2 \tag{1}$$

We could add hyperparameters for the term constraining the derivative, both in terms of how much we weight it in the objective as well as the constant used to push $\frac{\partial f_\theta(x,y)}{\partial y}$ away from zero. We opt for this simpler choice, with no such hyperparameters. Using a probabilistic perspective, we justify the choice of using square for $f_\theta(x, y)$ and for pushing the partial derivative to -1.

Probabilistic interpretation. Conventional statistical learning takes the perspective that the training samples are realized random variables drawn from some unknown probability distribution $(X, Y) \sim \mathbb{P} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$. For example, we can assume the conditional distribution $p(y|x)$ is Gaussian, resulting in the conventional regression objective where we minimize the mean squared error. Our objective can also been seen as having an underlying probabilistic assumption, but with a weaker assumption.

[1]The initial version of the theorem was originally proposed by Augustin-Louis Cauchy (1789–1857) and the generalization to functions of any number of real-variables was done by Ulisse Dini (1845–1918).

Consider the following data generation process. We sample x from some unknown marginal, $p(x)$, then sample a mixture component j from $p(j|x)$, and then sample the target y from $p(y|x, j)$. To formally derive our objective, we introduce an oracle: $j : \mathcal{X} \times \mathcal{Y} \mapsto [m_x]$. This function takes a sample (x, y) as input, and returns the corresponding component from which the y value got sampled: $j(x, y)$ is the index of the corresponding Gaussian component. Such a function is not accessible in practice and can only be used for our interpretation here.

We assume that the noise around each conditional mode is Gaussian. More precisely, define $\epsilon(X, Y) \overset{\text{def}}{=} g_{j(X,Y)}(X) - Y$. Our goal is to approximate $\epsilon(x, y)$ with parameterized function $f_\theta(x, y)$ for parameters θ. We **assume**

$$\epsilon(X, Y) \sim \mathcal{N}(\mu = 0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\epsilon(X, Y)^2}{2\sigma^2}\right) \tag{2}$$

Note that this is a **weaker assumption** than assuming $p(y|x)$ is Gaussian. For example, $p(y|x)$ can be a mixture of Gaussians. Under our probabilistic assumption, we can estimate θ by maximum likelihood estimation, giving the objective

$$\arg\min_\theta -\sum_{i=1}^{n} \ln\left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{f_\theta(x_i, y_i)^2}{2\sigma^2}\right)\right] = \arg\min_\theta \sum_{i=1}^{n} f_\theta(x_i, y_i)^2.$$

Additionally, we then want to satisfy the constraint on the derivative of f_θ. For an observed (x, y), we have that for some $j \in [m_x]$,

$$\frac{\partial \epsilon(x, y)}{\partial y} = \frac{\partial (g_j(x) - y)}{\partial y} = -1. \tag{3}$$

Therefore, when learning f, we encourage $\frac{\partial f_\theta(x,y)}{\partial y} = -1$ for all observed y — as required by the implicit function theorem. In summary, our goal is to minimize the negative log likelihood of f_θ, which approximates a zero-mean Gaussian random variable, under this constraint which we encourage with a quadratic penalty term. This gives the above objective 1.

Predicting modes. In modal regression, it is common to either want to find the highest likelihood mode, or to extract all the modes to get a set of predictions. Most existing methods focus only on finding the single most likely mode since they typically have the assumption that the mode is unique (Wang et al., 2017; Feng et al., 2017). The extension to the parametric setting, where we learn a surface characterizing the multi-modal structure, enables us to more obviously handle both cases.

Assume we are given test point x^*. We define two sets

$$S_{global}(x^*) = \{y \mid y = \arg\min_y l_\theta(x^*, y)\} \text{ and } S_{local}(x^*) = \left\{y \,\middle|\, \frac{\partial l_\theta(x^*, y)}{\partial y} = 0, \frac{\partial^2 l_\theta(x^*, y)}{\partial^2 y} > 0\right\}$$

For the first case, where we are only interested in the modes with highest likelihood, we need to find the set $S_{global}(x^*)$. For the second case, where the goal is to extract all conditional modes, we need find the set $S_{local}(x^*)$. To understand why these sets are appropriate, notice that we use l_θ instead of trying simply to find y such that $f_\theta(x^*, y) = 0$. There are two reasons for this: $l_\theta(x^*, y)$ only selects for y that satisfy the implicit function conditions and it also reflects the likelihood of y. For the first point, we need to ensure there is an implicit function at the testing point which can map from the input space to the target space.

The second point is more nuanced. Notice that minimizing l_θ could return all the modes, not just the most likely ones, and so $S_{local}(x)$ and $S_{global}(x)$ should presumably return similar sets. We know that $S_{global}(x^*) \subseteq S_{local}(x^*)$, but ideally we want S_{global} to only consist of the most likely mode. Hence we further constrain the surface with another regularization term, which we introduce in the next section 3.2. In that section, we discuss how, under this regularization, the training points with lower likelihood should have a larger residual, leading to different local loss values. Consequently, $\arg\min_y l_\theta(x^*, y)$ should be unique, or at least should consist of a smaller set of more likely modes.

Finding either of these sets requires a search, or an optimization. In all our experiments, we opted for the simple strategy of searching over 200 evenly spaced values in the range of y to get the two set of predictions. That is, we implement

$$S_{global}(x^*) \approx \{y_j || l_\theta(x^*, y_j) - \min_{i \in [200]} l_\theta(x^*, y_i)| < 10^{-5}, j \in [200]\}$$

$$S_{local}(x^*) \approx \{y_j | l_\theta(x^*, y_j) < l_\theta(x^*, y_{j+1}), l_\theta(x^*, y_j) < l_\theta(x^*, y_{j-1}), j \in \{2, ..., 199\}\}.$$

Connection to energy-based models. It is worth mentioning that an alternative formulation is possible by combining energy-based models with a margin loss (LeCun et al., 2006). Such model predicts an energy for a pair (x, y). The training process pulls down the energy of pairs of (x, y) in the dataset while the margin constraint avoids the trivial solution of predicting low energy all across the input space. In our method, it is the gradient condition in the implicit function theorem that prevents the trivial solution of predicting a mode everywhere. We may still want to ensure that $l_\theta(x^*, y)$ does not become small where there is no mode, as otherwise our method may predict spurious modes. We address this issue in the next section, with a regularization that we motivate theoretically and empirically.

3.2 Higher Order Regularizers to Control the Number of Modes

We first introduce Theorem 1, which motivates our regularization method for avoiding spurious modes. See Appendix A.1 for the proof.

Theorem 1. *Let $f(x)$ be a continuously differentiable function s.t. $|f''(x)| \leq u$ for some $u > 0$. Let a be such that $f(a) = \epsilon, f'(a) = k$. Then, $\nexists b$ such that $0 < |b - a| < \frac{2|k|}{u}$, $f(b) = \epsilon, f'(b) = k$.*

We can consider a in this theorem as a true mode and hence $\epsilon = 0, k = 1$. This theorem is telling us that within $\frac{2}{u}$ to a, there is no other point b can make $f_\theta(x^*, b) = 0$, $\frac{\partial f_\theta(x_i, y_i)}{\partial y} = -1$ if $|\frac{\partial^2 f_\theta(a,y)}{\partial^2 y}| \leq u$ for any y in the target space. The smaller the u is, the farther away the spurious mode has to be. Hence, this theorem tells that the spurious mode should be mostly eliminated by small enough u. This suggests a regularization to penalize the second order derivative magnitude:

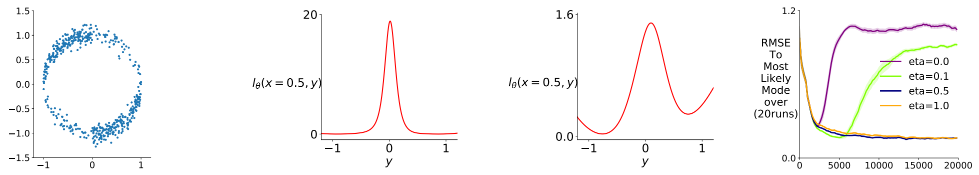
$$L_\eta(\theta) \stackrel{\text{def}}{=} \sum_{i=1}^{n} f_\theta(x_i, y_i)^2 + \left(\frac{\partial f_\theta(x_i, y_i)}{\partial y} + 1\right)^2 + \eta \left(\frac{\partial^2 f_\theta(x_i, y_i)}{\partial^2 y}\right)^2 \tag{4}$$

As we increase the weight η for the regularizer, the optimization is restricted in the number of modes it can identify. The lowest error choice according to the objective is to find the modes with highest likelihood. Note that this regularizer is not the second order derivative wrt the parameters; it is wrt to the targets. Therefore, it does not introduce significantly more computational cost to compute, and has the same order of computation as the original objective.

The original objective (with $\eta = 0$) actually already has some of the same regularizing affects, due to the regularizer on the gradient w.r.t. y. We provide some theoretical support that even with $\eta = 0$, the function is likely to avoid spurious modes in Appendix A.1. However, the original objective does not allow for a mechanism to control this effect, whereas η with this second order regularizer provides a tunable knob to more stringently enforce this to be the case. Further, it also allows the user to find a surface that learns only the highest likelihood modes, by picking larger η. In some cases, even if a mode exists in a multimodal distribution—and so it is not spurious—it may be preferred to ignore it if it has low probability of occurring.

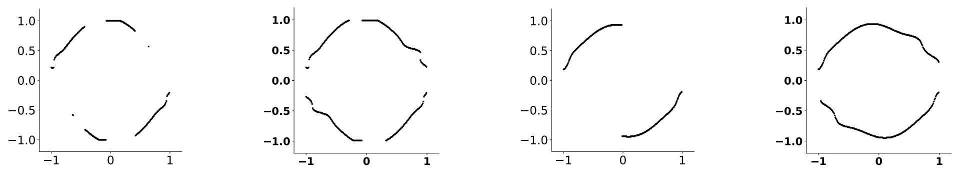
We empirically test the utility of this regularizer for identifying the modes with higher likelihood, and for removing any spurious modes. We generate a training set by uniformly sampling $x \in [-1, 1]$ and training target y by sampling $y \sim 0.8\mathcal{N}(\sqrt{1-x^2}, 0.1^2) + 0.2\mathcal{N}(-\sqrt{1-x^2}, 0.1^2)$ if $x < 0$ and $y \sim 0.2\mathcal{N}(\sqrt{1-x^2}, 0.1^2) + 0.8\mathcal{N}(-\sqrt{1-x^2}, 0.1^2)$ if $x \geq 0$. That is, the highest likelihood mode is positive if $x < 0$ and is negative if $x > 0$. We generate 4k such points for training and another 1k such points without adding noise to the target for testing.

In Figure 2, we visualize the learned function $l_\theta(0.5, y)$ by training our implicit learning objective with and without regularization respectively. For most points x^*, both functions return an accurate set of modes. But, at the more difficult location, in-between when the likelihoods of the modes switches, $x^* = 0.5$, there are more clear differences. At $x^* = 0.5$, the two true conditional modes with high and low likelihood are $\approx -0.87, 0.87$ respectively. Figure 2(b) shows that without the regularization, $\arg\min_y l_\theta(0.5, y)$ possibly returns spurious modes, with larger flat regions; this means we cannot distinguish between the modes with high and low likelihood. In contrast, Figure 2(c) shows that with $\eta = 1$ we identify both the high likelihood mode and the low one by looking at the two local minima at around ± 0.87 of the loss function l_θ trained with the above regularization. Further, the highest likelihood mode has a smaller $l_\theta(0.5, y)$, and would indeed be the only point in $S_{global}(x^*)$ and correctly identified as the highest likelihood mode. We additionally show the global and local predictions with and without the regularization in Figure 3.



(a) Training Set (b) $\eta = 0$ (c) $\eta = 1$ (d) learning curves

Figure 2: (a) shows the training set; (b) shows the loss function $l_\theta(x^* = 0.5, y)$ trained without regularization; (c) shows the one trained with $\eta = 1$. (d) shows the learning curves for $\eta \in \{0.0, 0.1, 0.5, 1.0\}$ in terms of the root mean squared error (RMSE) as a function of number of mini-batch updates on the testing set. The RMSE is computed between randomly picked mode from $S_{global}(\cdot)$ and the corresponding true most likely mode.



(a) $\eta = 0, S_{global}$ (b) $\eta = 0, S_{local}$ (c) $\eta = 0.5, S_{global}$ (d) $\eta = 0.5, S_{local}$

Figure 3: (a) and (b) shows the predictions of evenly spaced points $x \in \{-1, -0.99, -0.98, ..., 0.99, 1.0\}$ from $S_{global}(\cdot), S_{local}(\cdot)$ learned without using regularization. (c) and (d) shows those points learned with $\eta = 0.5$. (c) shows that the regularizer clearly enables prediction of the most likely mode, whereas (a) has many spurious modes. Looking at both (c) and (d), a decision maker using these predictions could both know the highest likelihood mode (given in (c)) as well as the full set of modes (given in (d)).

Despite some of the issues highlighted in Figure 2 for $\eta = 0$, visualizing the actual predictions made under the original objective reflects a more positive outcome. In Figure 3(a)(b), one can see that the original objective 1 does provide useful predictions even without the regularization; it only fails to identify the highest likelihood mode. The simpler algorithm with $\eta = 0$ looks like a promising choice, if multiple modes are desired, whereas we suggest the use of regularization if the goal is to predict a unique highest likelihood mode. We empirically observe that in many cases, our algorithm is able to achieve competitive performance with other baselines even without using the regularization. We assume $\eta = 0$ for the remainder of this work.

4 The properties of implicit function learning

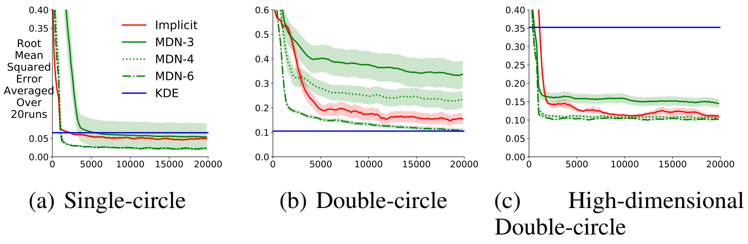
In this section, we empirically investigate the properties of our learning objective. First, on several synthetic Circle datasets, we show the utility of implicit function learning for handling the case where the highest likelihood mode is not unique. Note it is common to assume the existence of a unique mode, to avoid the difficulties of this setting. Second, we demonstrate that our objective allows us to leverage the representational power of neural networks, by testing it on high-frequency data.

4.1 Comparison on Simple Synthetic Datasets

Datasets. We empirically study the performance of our algorithm on the following datasets. **Single-circle:** The training set contains 4k samples and is generated by the process described in the caption of Figure 1. **Double-circle:** The same number of training points, 4000, are randomly sampled from two circles (i.e. $\{(x, y)|x^2 + y^2 = 1\}, \{(x, y)|x^2 + y^2 = 4\}$) and the targets are contaminated by the same Gaussian noise as in the single-circle dataset. This is a challenging dataset where $p(y|x)$ can be considered as a *piece-wise* mixture of Gaussian: there are four components on $x \in (-1, 1)$ and two components on $x \in (-2, -1) \cup (1, 2)$. The purpose of using this dataset is to examine how the performances of different algorithms change when we increase the number of modes. **High dimensional double-circle.** We further increase the difficulty of learning by projecting the one dimensional feature value to 128 dimensional binary feature through tile coding: $\phi : [-2, 2] \mapsto \{0, 1\}^{128}$. The purpose of using this dataset is to examine how different algorithms can scale to high dimensional inputs.

Algorithm Evaluation. We compute RMSE between a randomly picked mode from $S_{global}(\cdot)$ and the closest true mode to it. This evaluation method reflects a practical setting where any of the most likely modes would be acceptable.

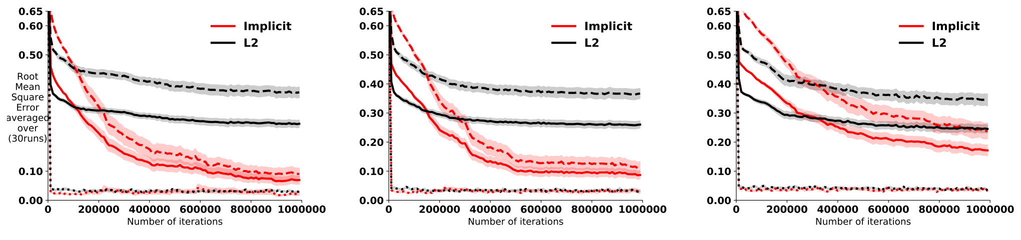
Algorithm Description. We compare to Kernel Density Estimation (**KDE**) and Mixture Density Networks (**MDN**). KDE is a non-parametric approach to learn a distribution, which has strong theoretical guarantees for representing distributions. It can, however, be quite expensive in terms of both computation and storage. We use KDE because several existing nonparametric modal regression models are proved to converge to the corresponding KDE model (Yao & Li, 2014; Wang et al., 2017; Feng et al., 2017); it therefore acts as an idealized competitor. MDN (Bishop, 1994) learns a conditional Gaussian mixture, with neural networks, by maximizing likelihood. For both methods, we use a fine grid search—200 evenly spaced values in the target space—to find a mode given by the KDE distribution or MDN distribution: $\hat{y} = \arg\max_y \hat{p}(y|x)$. For both our algorithm (**Implicit**) and MDN, we use a 16×16 tanh units NN and train by stochastic gradient descent with mini-batch size 128. We optimize both algorithms by sweeping the learning rate from $\{0.01, 0.001, 0.0001\}$.



(a) Single-circle (b) Double-circle (c) High-dimensional Double-circle

Figure 4: (a)(b)(c) show testing RMSE as a function of training steps. MDN-3 indicates MDN trained with 3 components. All results are averaged over 20 random seeds and the shaded area indicates standard error.

The learning curves on the above three datasets are shown in Figure 4. We plot the RMSE as a function of number of mini-batch updates for the two parametric methods MDN and Implicit. KDE is shown as a constant line since it directly uses the whole training set as input. MDN performs poorly with only two components, even when the true number of modes is two; therefore, we include 3, 4 and 6 mixture components. From Figure 4, one can see that: 1) although (a) and (b) have low dimensional feature, MDN with only 3 and 4 components degrades significantly when we increase the number of modes of the training data from two to four. Furthermore, MDN shows significantly larger variance across runs, which we observe frequently across several experiments in our work. 2) KDE scales poorly with both the number of modes and input feature dimension. We also found that it is quite sensitive to the kernel type and bandwidth parameter. 3) Our algorithm Implicit achieves stable performances across all the datasets and performs as well as MDN on the high dimensional double-circle dataset. In contrast, MDN seems to no longer have advantage with a larger number of mixture components than 3 for the high dimensional binary features. We visualize the predictions of our algorithms on the single-circle and double-circle datasets in the Appendix A.3.



(a) No noise (b) $\sigma = 0.1$ (c) $\sigma = 0.2$

Figure 5: Figure (a)(b)(c) show performances of **Implicit (red)** and l_2 regression **L2 (black)** objective as we increase the Gaussian noise variance in the High Frequency dataset. We show the testing error measured by RMSE on entire testing set (**solid line**), on high frequency region (i.e. $x \in [-2.5, 0.0)$, **dashed line**) and on low frequency region ($x \in [0.0, 2.5]$, **dotted line**). The results are averaged over 30 random seeds. The targets in the testing set are true targets and hence are not noise-contaminated. We run 1 million iterations to make sure each algorithm is sufficiently trained and both early and late learning behaviour can be examined.

4.2 Robustness to high frequency data

The above circle example can be thought of as an extreme case where the underlying true function has extremely high frequency (i.e. when the input changes a little, there is a sharp change for the true target). In this section, we investigate if our modal regression algorithm does provide an advantage to handle such datasets. We do so by testing it on a uni-modal high-frequency dataset, and comparing the solution found by Implicit to standard ℓ_2 regression. We generate a synthetic dataset by uniformly sampling $x \in [-2.5, 2.5]$ and compute the targets by $y = \sin(8\pi x) + \xi, x \in [-2.5, 0)$ and $y = \sin(0.5\pi x) + \xi, x \in [0, 2.5]$, where ξ is zero-mean Gaussian noise with variance σ^2. This function has relatively high frequency when $x \in [-2.5, 0)$ and has a relatively low frequency when $x \in [0, 2.5]$ (see A.3 for more on this dataset). We use a small NN with 16×16 tanh units. The only difference to the NN for ℓ_2 is that Implicit has one more input unit, i.e. the target y. We perform extensive parameter sweeps to optimize its performance. Please see A.2 for any missing details.

Figure 5(a-c) shows the evaluation curve on testing set for the above two algorithms as the noise variance increases. Notice that, when noise $\xi \equiv 0$, our implicit function learning approach achieves a much lower error (at the order of 10^{-2}) than the l_2 regression does (at the order of 10^{-1}). As noise increases, the targets become less informative and hence our algorithm's performance decreases to be closer to the l_2 regression. Unsurprisingly, for both algorithms, the high frequency area is much more difficult to learn and is a dominant source of the testing error. After sufficient training, our algorithm can finally reduce the error of both the high and low frequency regions to a similar level. We include additional rseults in Appendix A.3, visualizing the learned networks.

5 Results on Real World Datasets

In this section, we first conduct a modal regression case study on a real world dataset. Additionally, because our method incorporates information from the target space, we hypothesize it should be beneficial even for regular regression tasks. We therefore also test its utility on two standard regression dataset. Appendix A.2 includes details for reproducing the experiments and dataset information.

5.1 Modal Regression: Predicting Insurance Cost

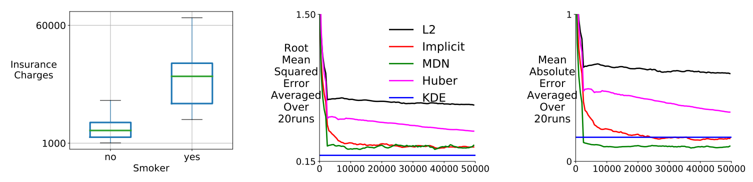
To the best of our knowledge, the modal regression literature rarely uses real world datasets. This is likely because it is difficult to evaluate the predictions from modal regression algorithms. We propose a simple way to construct a datatset from real data that is suitable for testing modal regression algorithms. The idea is to take a dataset with categorical features, that are related to the target, and permute the categorical variables to acquire new target values. Afterwards, we remove those categorical features from the dataset, so that each instance can have multiple possible target values.

We construct a modal regression dataset from the *Medical Cost Personal Dataset* (Lantz, 2013) using the following steps. 1) We determined that the `smoker` categorical variable is significantly relevant to the insurance charge. This can be seen from our Boxplot 6(a). 2) An ℓ_2 regression model is trained to predict the insurance charges. 3) We flip the `smoker` variable of all examples and query the trained ℓ_2 model to acquire a new target for each instance in the dataset. 4) We augment the original dataset with these new generated samples. 5) We remove the `smoker` variable from the dataset to form the modal regression dataset.

Each instance has two possible targets, and ℓ_2 regression can only produce one output; we therefore compute the error between the predicted value and the closest of the two modes. For Implicit and MDN, we compute the error by randomly picking a mode from $S_{global}(\cdot)$ as described in Section 4.1. We report both root mean squared error (RMSE) and mean absolute error (MAE). Figure 6(b)(c) shows the learning curves of different algorithms. It can be seen that: 1) both ℓ_2 and Huber regression perform significantly worse than our algorithm. This suggests that neither conditional mean nor median is a good predictor on this dataset, and provides evidence that our dataset generation strategy was meaningful. 2) The performance of MDN and KDE are inconsistent across the two error measures. 3) Our algorithm, Implicit, consistently achieves good performance, though it does take longer to learn the relationship.

5.2 Standard Regression Tasks

We finally test if our method can achieve comparable performance in standard regression problems. A natural choice for standard regression is to predict the unique mode, with S_{global}, as this provides one approach to robust regression. We include the Huber loss as a comparator to see if Implicit has such an effect. We additionally gauge a worst-case scenario, where we predict multiple modes with



(a) smoke v.s. charges (b) RMSE (c) MAE

Figure 6: (a) shows the boxplot of smoking v.s. insurance cost. (b)(c) shows testing error as a function of number of training steps. All results are averaged over 20 random seeds and the standard error is low.

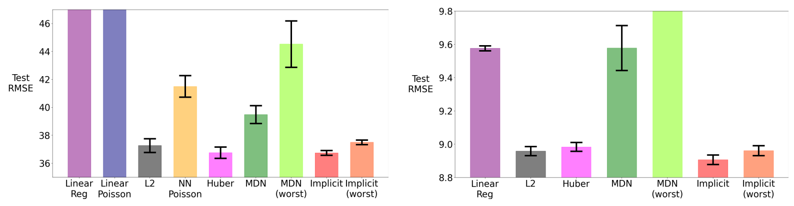
S_{local}, but are evaluated based on the furthest distance to the observed target in the dataset (even if in the true underlying data there might be a closer mode).

Algorithm evaluation. We report both RMSE and MAE on training and testing set respectively. All of our results are averaged over 5 runs and for each run, the data is randomly split into training and testing sets. For our algorithm, we use 64×64 tanh units NN. For the ℓ_2 and Huber regression, we use the same size NN and optimize over unit type ReLu and tanh. Huber regression is used as it is known to be more robust to some skewed or heavy-tailed data distributions.

Baselines and datasets are as follows. **LinearReg:** Ordinary linear regression, where the prediction is linear in term of input features. We use this algorithm as a weak baseline. **LinearPoisson:** The mean of the Poisson is parameterized by a linear function in term of input feature. **NNPoisson:** The mean of the Poisson is parameterized by a neural network (Fallah et al., 2009).

Figure 7 compares the test root mean squared error of different methods. The exact numbers and other error measures are presented in Appendix A.3. We show results on the **Bike sharing dataset** (Fanaee-T & Gama, 2013), where the target is Poisson distributed, in Figure 7(a). The prediction task is to predict counts of rental bikes in a specific hour given 114 features after preprocessing. In Figure 7(b), we show results on the **Song year dataset** (Bertin-Mahieux et al., 2011), where the task is to predict a song's release year by using audio features of the song. The target distribution that is clearly not Gaussian, though it is not clear which generalized linear model is appropriate.

Implicit achieves highly competitive performance on both datasets; although it does slightly worse than MDN on Bike sharing and slightly worse than Huber on Song year dataset when measured by MAE (see A.3). MDN frequently exhibits more variant performance than other algorithms, and its worst case prediction can be significantly worse than from $S_{global}(\cdot)$. In contrast, our algorithm seems to learn an accurate set S_{local}, which is only slightly worse that S_{global}.



(a) Bike sharing dataset (b) Song year prediction dataset

Figure 7: Test RMSE on two standard regression tasks. Error bars show standard errors, for 5 runs. For other error measures and more details, see Appendix A.3.

6 Conclusion and Discussion

The paper introduces a parametric modal regression approach, using the idea of implicit functions to identify local modes. We show that it can handle datasets where the conditional distribution $p(y|x)$ is multimodal, and is particularly useful when the underlying true mapping has a large bandwidth limit. We also illustrate that our algorithm achieves competitive performance on real world datasets for both modal regression and regular regression. This work highlights the feasibility, and potentially utility, of more widely using modal regression. However, a remaining potential barrier to such widespread use is the difficulty in evaluating modal regression approaches on real data. We provided a strategy to add modality to datasets for evaluation, but did not identify how to use and evaluate mode predictions for standard datasets, even though these datasets likely already have multiple modes. An important next step is to investigate how to evaluate a set of predicted modes, on standard regression datasets.

7 Broader Impact Discussion

This work is about parametric methods of modal regression. Non-parametric modal regression methods are typically studied in the statistics community; and there is yet little parametric modal regression algorithm suitable in deep learning setting. Hence, our work should be generally beneficial to the machine learning and statistics research community. Potential impact of this work in real world is likely to be further improvement of applicability of modal regression methods, which provide more information to decision makers than popular regression methods which attempt to learn conditional mean. The proposed objective may be also of high interest to the community studying energy-based models. We do not consider any specific application scenario as the goal of this work.

Acknowledgements

We would like to thank all anonymous reviewers for their helpful feedback. Particularly, we thank reviewer 2 and the meta-reviewer for careful reading our paper and providing suggestions to improve the presentation. We also thank Jincheng Mei for reading our theorems and provide suggestion to improve Theorem 1.

Funding Transparency Statement

We acknowledge the funding from the Canada CIFAR AI Chairs program and Alberta Machine Intelligence Institute.

References

Abadi, M., Agarwal, A., Barham, P., and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Batra, D., Yadollahpour, P., Guzman-Rivera, A., and Shakhnarovich, G. Diverse m-best solutions in markov random fields. In *Proceedings of the 12th European Conference on Computer Vision*, pp. 1–16, 2012.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *International Conference on Music Information Retrieval*, 2011.

Bishop, C. M. Mixture density networks. *Technical Report NCRG/94/001*, 1994.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Chen, Y.-C. Modal regression using kernel density estimation: A review. *Wiley Interdisciplinary Reviews: Computational Statistics*, pp. e1431, 2018.

Chen, Y.-C., Genovese, C. R., Tibshirani, R. J., and Wasserman, L. Nonparametric modal regression. *arXiv:1412.1716*, 2014.

Ellefsen, K. O., Martin, C. P., and Tørresen, J. How do mixture density rnns predict the future? *arXiv:1901.07859*, 2019.

Fallah, N., Gu, H., Mohammad, K., Seyyedsalehi, S. A., Nourijelyani, K., and Eshraghian, M. R. Nonlinear poisson regression using neural networks: a simulation study. *Neural Computing and Applications*, 18(8):939, Jul 2009.

Fanaee-T, H. and Gama, J. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pp. 1–15, 2013.

Feindt, M. A Neural Bayesian Estimator for Conditional Probability Densities. *arXiv:0402093*, 2004.

Feng, Y., Fan, J., and Suykens, J. A. K. A statistical learning approach to modal regression. 2017.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Husmeier, D. Modelling conditional probability densities with neural networks. 1997.

Husmeier, D. and Taylor, J. G. Neural networks for predicting conditional probability densities: Improved training scheme combining em and rvfl. *Neural Networks*, 11:89–116, 1998.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Jebara, T. and Pentland, A. P. *Discriminative, Generative and Imitative Learning*. PhD thesis, USA, 2002.

Kemp, G. C. and Silva, J. S. Regression towards the mode. *Journal of Econometrics*, 170(1):92 – 101, 2012.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Lantz, B. *Machine Learning with R*. Packt Publishing, 2013.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. 2006.

Lee, M.-j. Mode regression. *Journal of Econometrics*, pp. 337–349, 1989.

Lee, M.-J. and Kim, H. Semiparametric econometric estimators for a truncated regression model: a review with an extension. *Statistica Neerlandica*, 52(2):200–225, 1998.

Li, K., Peng, S., Zhang, T., and Malik, J. Multimodal image synthesis with conditional implicit maximum likelihood estimation, 2020.

Lv, Z., Zhu, H., and Yu, K. Robust variable selection for nonlinear models with diverging number of parameters. *Statistics & Probability Letters*, pp. 90–97, 2014.

Munkres, J. R. *Analysis On Manifolds*. Boca Raton: CRC Press, 1991.

Nelder, J. A. and Wedderburn, R. W. M. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, pp. 370–384, 1972.

Ota, H., Kato, K., and Hara, S. Quantile regression approach to conditional mode estimation. *arXiv:1811.05379*, 2018.

Powell, M. J. D. Algorithms for approximation. chapter Radial Basis Functions for Multivariable Interpolation: A Review, pp. 143–167. Clarendon Press, 1987.

Vlassis, N. and Krose, B. Mixture conditional density estimation with the em algorithm. In *International Conference on Artificial Neural Networks*, pp. 821–825, 1999.

Wang, X., Chen, H., Cai, W., Shen, D., and Huang, H. Regularized modal regression with applications in cognitive impairment prediction. In *Advances in Neural Information Processing Systems 30*, pp. 1448–1458. 2017.

Weigend, A. S. and Srivastava, A. N. Predicting conditional probability distributions: a connectionist approach. *International Journal of Neural Systems*, 06:109–118, 1995.

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

Yao, W. and Li, L. A new regression model: Modal linear regression. *Scandinavian Journal of Statistics*, 41(3):656–671, 2014.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

Yu, K. and Aristodemou, K. Bayesian mode regression. *arXiv:1208.0579*, 2012.

Zen, H. and Senior, A. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *International Conference on Acoustics, Speech and Signal Processing*, pp. 3844–3848, 2014.

A Appendix

The appendix includes the proof for Theorem 1 and additional theoretical discussions regarding spurious modes in Section A.1. We provide all experimental details for reproducible research in Section A.2 and provide additional empirical results in Section A.3. We also include a brief discussion of some possibly related areas in the end.

A.1 Proofs and Theoretical Insight

In this section, we provide mathematical proof for our Theorem 1. We then provide additional theoretical intuitions regarding why the spurious modes may not easily show up even without regularization by introducing Theorem 2 and Corollary 1. We conclude this section by some discussions on modal regression algorithm evaluation in Section A.1.3.

A.1.1 Proof for Theorem 1

Theorem 1. Let $f(x)$ be a continuously differentiable function s.t. $|f''(x)| \le u$ for some $u > 0$. Let a be such that $f(a) = \epsilon, f'(a) = k$. Then, $\nexists b$ such that $0 < |b - a| < \frac{2|k|}{u}, f(b) = \epsilon, f'(b) = k$.

Proof. Proof by contradiction. Assume $\exists b \ne a$, such that $|b - a| < \frac{2|k|}{u}, f(b) = \epsilon, f'(b) = k$. By applying first order Taylor expansion to $f(a)$, for some ξ, we have

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(\xi)}{2}(x - a)^2$$

Then,

$$\begin{aligned} f(b) &= f(a) + f'(a)(b - a) + \frac{f''(\xi)}{2}(b - a)^2 \\ |f'(a)(b - a)| &\le \frac{u(b - a)^2}{2} \\ |b - a| &\ge \frac{2|k|}{u} \end{aligned}$$

This contradicts with $|b - a| < \frac{2|k|}{u}$. This completes the proof. $\square$

A.1.2 More discussions regarding the spurious mode

We found that in practice, our method does find correct modes even without the regularization of penalizing the second order derivative, as we discussed in Section 3.2. We introduce below theorem to explain why a spurious mode cannot come out naturally.

Theorem 2. *Let $f(x)$ be a continuously differentiable function on the interval (a, b). Let $x_1 < x_2 \in (a, b)$ and $f(x_1) = f(x_2)$ and $f'(x_1)f'(x_2) > 0$ (i.e. the slopes have the same sign and are nonzero). Then $\exists x_0 \in (x_1, x_2)$ such that $f(x_1) = f(x_2) = f(x_0)$ and $f'(x_0)f'(x_1) \le 0, f'(x_0)f'(x_2) \le 0$.*

Proof. Without loss of generality, let $f'(x_1) < 0, f'(x_2) < 0$. The positive case can be proved in exactly the same way. We prove the existence of such x_0 in Part I and prove its derivative property in Part II.

Part I. We firstly show $\exists x_0 \in (x_1, x_2)$ such that $f(x_1) = f(x_2) = f(x_0)$. Since $f'(x_2) < 0$, one can choose a ϵ_1 such that $0 < \epsilon_1 < \frac{x_2 - x_1}{2}$ and $f(x_2 - \epsilon_1) > f(x_2)$. Similarly, since $f'(x_1) < 0$, one can choose a ϵ_2 such that $0 < \epsilon_2 < \frac{x_2 - x_1}{2}$ and $f(x_1 + \epsilon_2) < f(x_1) = f(x_2)$. Since the function is a continuously differentiable function, applying intermediate value theorem yields $\exists x_0 \in [x_1 + \epsilon_2, x_2 - \epsilon_1], s.t. f(x_0) = f(x_1) = f(x_2)$.

Part II. To show the second part $f'(x_0)f'(x_1) < 0, f'(x_0)f'(x_2) < 0$, consider two cases depending on whether such x_0 proved in part I is unique or not.

Case 1. When such x_0 is unique. Then it implies that there exists no other $x' \in (x_1, x_2), x \ne x_0$ such that $f(x') = f(x_1) = f(x_2) = f(x_0)$. We can prove by contradiction. Assume $f'(x_0)f'(x_1) > 0$.

Then we can replace x_1 by x_0 in **Part I**'s argument, and this gives another $x'_0 \in (x_0, x_2) \subset (x_1, x_2)$ and it contradicts with the prerequisite that x_0 is unique. Hence, $f'(x_0)f'(x_1) \leq 0$. Hence, we complete the proof of Case 1 when x_0 proved in Part I is unique.

Case 2. When such x_0 is not unique. Then there are multiple points: $x_1 < x'_0 < x'_1 < ... < x_2$ such that $f(x'_0) = f(x'_1) = f(x_1) = f(x_2) = f(x_0)$. Then the first case's result directly applies: one can choose x'_0, x'_1, i.e. the first closest point and the second closest point to x_1; then $f'(x'_0)f'(x_1) \leq 0$.

The above two parts complete the proof. □

This theorem immediately leads to the below corollary.

Corollary 1. *Let $f(x)$ be a continuously differentiable function on the interval (a, b). If there are, in total $n > 1, n \in \mathbb{Z}$ points: $a < x_1 < ... < x_n < b$ such that $f(x_1) = f(x_2) = ... = f(x_n)$ and they have the same sign of nonzero derivatives, then there must be another $n - 1$ points on (a, b) which have an identical function value with $f(x_i), i = 1, ..., n$ with different derivative signs.*

Proof. This is a direct consequence of applying the above Theorem 2 consecutively across (x_1, x_2), (x_2, x_3) etc. □

The corollary tells us that if we have two roots for $f(x) = 0$ on the interval (a, b) and the tangent lines on the two points have the same derivative direction, there must be another root between the two with different derivative sign. Consider the following simple example. We compose a training set by uniformly sampling 4000 data points from a circle: $\{(x, y)|x^2 + y^2 = 1\}$ and train our implicit function $f_\theta(x, y)$. Consider $f_\theta(0, y)$ as the function $f(x)$ in the above Theorem 2. Figure 8 shows the function $f_\theta(x = 0, y), y \in [-1.2, 1.2]$ after training. The correct target values at $x = 0$ should be $+1$ and -1. The slope of the function around each of these modes is optimally -1, to satisfy the regularizer $(\frac{\partial f_\theta(x^*,y)}{\partial y} + 1)^2$ which encourages a slope of -1 through all observed y. As one can see that, by Theorem 2, there is one root for $f_\theta(0, y) = 0$ at $y = 0$. However, the slope of the function here incurs a high cost for the derivative part.

Now, consider that if there is a wrong prediction $y' \in (-1, 1)$ but $f_\theta(0, y')$ is almost zero and $\frac{\partial f_\theta(0,y)}{\partial y}$ is very close to -1—it has the same sign as the other two *correct* predictions. By our Corollary 1, this indicates that there are another two roots for $f_\theta(0, y) = 0$ on $(-1, y'), (y', 1)$ respectively—which leads to two more waves on the interval $(-1, 1)$. That is, every time the objective gives one more incorrect prediction, the function $f_\theta(0, y)$ has two more waves. This would result in a high frequency function, which could be difficult to approximate even with training data. Hence, unless the training data reflects such a high frequency pattern, it is unlikely to choose to do so.

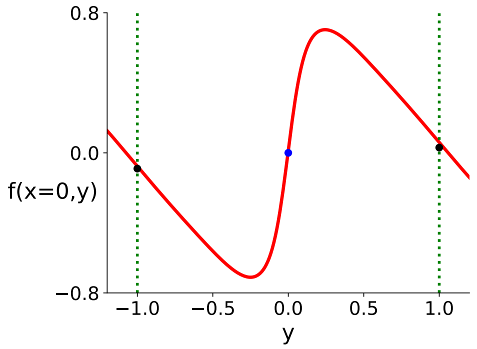


Figure 8: The trained function $f_\theta(x = 0, y)$. The **blue point** is $(0, 0)$; and the other two **black points** are the predicted points by $\arg\min_y f_\theta(0, y)^2 + (\frac{\partial f_\theta(0,y)}{\partial y} + 1)^2$.

A.1.3 Modal Regression Algorithm Evaluation

To our best knowledge, there is no common standard to evaluate modal regression algorithms when there are multiple true modes. In our main paper, we introduced two ways for evaluation. First, we randomly pick a predicted mode from $S_{global}(\cdot)$ and compute RMSE/MAE to the closest true mode. Second, on the real world datasets, we consider the worst case prediction from the set $S_{local}(\cdot)$.

We now introduce an arguably more formal, but less intuitive way to evaluate our predictions—Hausdorff distance, which is a natural way to compute distance between two sets. The distance is defined as:

$$d_{\mathrm{H}}(X,Y) = \max\left\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \right\} \tag{5}$$

where X, Y are two non-empty subsets of some metric space. It should be noted that, when $|Y| = 1$, then this distance is equivalent to computing the worst case prediction as introduced for the regular regression datasets Songyear and Bike sharing in Section 5. This is because $d_{\mathrm{H}}(X,Y)|_{Y=\{y\}} = \max\left\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \right\} = \max\left\{ \sup_{x\in X} d(x,y), \inf_{x\in X} d(x,y) \right\} = \sup_{x\in X} d(x,y)$. This also indicates that it is only suitable to use this method to evaluate distance between the set $S_{local}(\cdot)$ and the set of true modes when $|S_{local}(\cdot)| > 1$, because when $|S_{local}(\cdot)| = 1$, the evaluation becomes meaningless in that even if it predicts the correct highest likelihood mode, the evaluation will choose to compute its distance to some other true mode as long as the set contains a single prediction. In section A.3, we include learning curves in terms of Hausdorff distance between the set S_{local} and the set of true modes. We do not include the learning curve of Hausdorff distance based on S_{global} as this set frequently contains only a single prediction.

A.2 Reproduce experiments in the paper

In this section, we provide additional information about datasets we used and experimental details for reproducing all results in this paper. We introduce common settings below.

Common settings. Our implementation is based on Python 3.3.6. Our deep learning implementation is based on Tensorflow 1.14.0 (Abadi et al., 2015). All of our algorithms are trained by Adam optimizer (Kingma & Ba, 2015) with mini-batch size 128 and all neural networks are initialized by Xavier (Glorot & Bengio, 2010). For our implicit function learning algorithm, we use tanh units for all nodes in neural network. We search over 200 evenly spaced values for prediction for Implicit, MDN, and KDE. When evaluating the algorithms, the data is randomly split into training and testing sets and we evaluate the testing error every 200 training steps for each random seed unless otherwise specified. The core part of the code is shared at `https://github.com/yannickycpan/parametricmodalregression.git`.

A.2.1 Reproduce results from Section 3.2

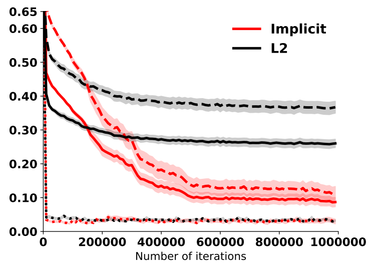
Algorithm implementation. We use a relatively larger neural network size than those used on other circle datasets to highlight the effect of our regularization methods. We use 64×64 tanh units neural network and for each regularization weight η value, and optimize learning rate from $\{0.01, 0.001, 0.0001\}$.

Optimal parameter choice. The best learning rate is 0.0001.

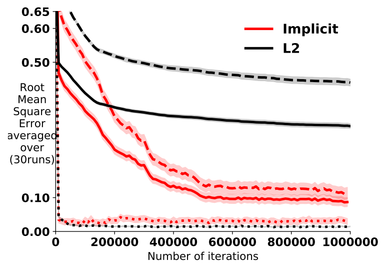
Dataset generation. We generate a training set by uniformly sampling $x \in [-1, 1]$ and training target y by sampling $y \sim 0.8\mathcal{N}(\sqrt{1-x^2}, 0.1^2) + 0.2\mathcal{N}(-\sqrt{1-x^2}, 0.1^2)$ if $x < 0$ and $y \sim 0.2\mathcal{N}(\sqrt{1-x^2}, 0.1^2) + 0.8\mathcal{N}(-\sqrt{1-x^2}, 0.1^2)$ if $x >= 0$. We generate 4k training data points and 1k testing points. For the purpose of plotting prediction in Figure 3, we use the numpy function numpy.arange$(-1, 1, 0.01)$ to generate evenly spaced x values.

A.2.2 Reproduce results from Section 4.1

Algorithm implementation. We keep the neural network size as 16×16 tanh units and sweep over learning rate from $\{0.01, 0.001, 0.0001\}$ in all circle experiments from Section 4.1. For mixture density network (**MDN**), the maximization is done by using MLE, the method described in the original paper (Bishop, 1994). For **KDE** model, we use the implementation from `https://github.com/statsmodels/statsmodels`. On the high-dimensional double-circle dataset, we consider the binary features as categorical. We input the whole training data to build KDE model. We use normal reference rule of thumb for bandwidth selection. For both KDE and MDN, we use grid search in the target space to find out the mode. That is, $\hat{y} = \arg\max_y \hat{p}(y|x) = \arg\max_y \hat{p}(x,y)$ given testing



(a) l_2 with learning rate 0.01



(b) l_2 with learning rate 0.001

Figure 9: In (a) we repeat the figure shown in previous Section 4.2.

point x. Note that the classic mean-shift algorithm for mode seeking attempts to do hill climbing (i.e. gradient ascent) on a KDE model with multiple initial values, which may still get a local optimum. We opt to directly search from 200 evenly spaced values in the target space to find out the mode given a KDE model.

Optimal parameter choice. The best learning rate is 0.01 for both MDN and Implicit except on double circle dataset, the optimal learning rate is 0.001 for Implicit on the high dimensional double circle dataset. Note that Implicit performs only slightly worse when learning rate is 0.01.

Dataset generation. Circle dataset is generated by uniformly sampling $x \in [-1, 1]$ first and then $y = \sqrt{1-x^2}$ or $y = -\sqrt{1-x^2}$ with equal probability. Double circle dataset is generated by uniformly sampling an angle $\alpha \in [0, 2\pi]$ then use polar expression to compute $x = r\cos\alpha, y = r\sin\alpha$ where $r = 1.0$ or $r = 2.0$ with equal probability. High dimensional double dataset is generated by mapping the original x to $\{0, 1\}^{128}$ dimensional space. We refer to `http://www.incompleteideas.net/tiles.html` for tile coding software. The setting of tile coding we used to generate feature is: memory size $= 128$, 8 tiles and 4 tilings.

A.2.3 Reproduce results from Section 4.2

Algorithm implementation. We use 16×16 tanh units neural network for both algorithm. We sweep over $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$ to optimize stepsize for both the l_2 regression and our algorithm, while we additionally sweep over hidden unit and output unit type for the l_2 regression from {tanh, relu} and found tanh to be better. The best parameter is chosen to optimize the second half of the learning curve, and the testing error is averaged over 30 random seeds. For the purpose of doing prediction, we use $S_{global}(\cdot)$.

Optimal parameter choice. The best learning rate chosen by the l_2 regression is 0.01 while our algorithm chooses 0.001. As a result, in Figure 9, we also plot the learning curve with learning rate 0.001 for the l_2 regression to make sure that the performance difference is not due to a slower learning rate of our algorithm.

Dataset generation. The dataset is generated by uniformly sampling $x \in [-2.5, 2.5]$ and then compute targets according to the equation:

$$y = \begin{cases} \sin(8\pi x) & x \in [-2.5, 0) \\ \sin(0.5\pi x) & x \in [0, 2.5] \end{cases}$$

Figure generation. To generate the learning curves from Figure 5 in Section 4.2 and Figure 9, we evaluate the testing error every 10k number of training iterations (i.e. mini-batch updates). We compute the testing errors on the entire testing set, on high frequency area ($x \le 0$) and low frequency area ($x \ge 0$) respectively.

A.2.4 Reproduce results from Section 5

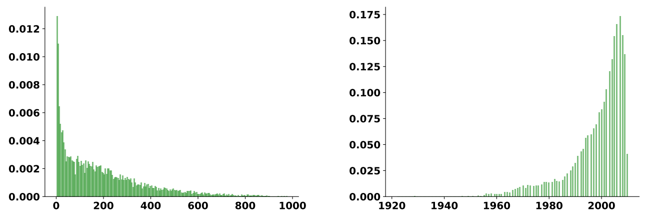
Algorithm implementation. We use 64×64 tanh units neural network for all algorithms except for NNPoisson where the Poisson mean is using linear activation. We optimize learning rate from

$\{0.01, 0.001, 0.0001, 0.00001\}$. For MDN, we set number of mixture components as 6. When we report the RMSE/MAE on those real world datasets, we choose the average of the best 5 consecutive evaluation testing errors to report. At each run, we evaluate the testing errors every 10k training steps (i.e. mini-batch updates) and we train each algorithm up to 200k steps.

Optimal parameter choice. All algorithms choose learning rate 0.0001 except PoissonReg chooses 0.01 on Bike sharing dataset.

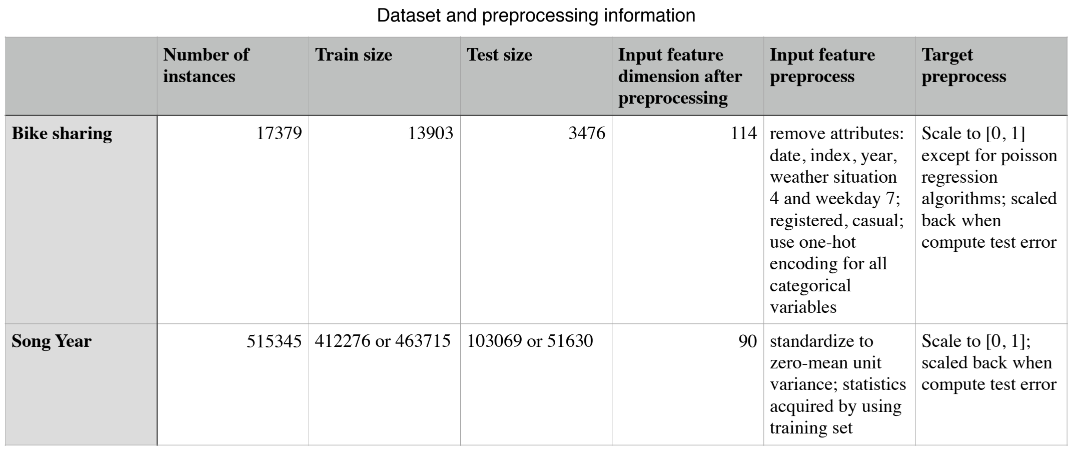
Dataset generation. The Medical Cost Personal Datasets by Lantz (2013) can be downloaded from `https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv`. We standardize the *age* and *bmi* variables and perform logarithmic transformation for the target variable *insurance charge* due to its wide range and large values; then we further scale the target variable to $[0, 1]$ when training. We report the errors after converting the predictions back to logarithmic scale. We upload the dataset processed by the steps described in Section 5 through anonymous file sharing and can be downloaded at `https://anonymousfiles.io/WkojSn5F/`, the last two columns are two possible targets/modes.

The bike sharing dataset (Fanaee-T & Gama, 2013) (`https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset`) and song year dataset (Bertin-Mahieux et al., 2011) (`https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd`) information are presented in figure 11. Note that the two datasets have very different target distributions as shown in Figure 10.



(a) Bike sharing dataset target distribution (b) Song year dataset target distribution

Figure 10: Bike sharing targets show a clear Poisson distribution while song year dataset's target distribution is not intuitive.



Dataset and preprocessing information

	Number of instances	Train size	Test size	Input feature dimension after preprocessing	Input feature preprocess	Target preprocess
Bike sharing	17379	13903	3476	114	remove attributes: date, index, year, weather situation 4 and weekday 7; registered, casual; use one-hot encoding for all categorical variables	Scale to [0, 1] except for poisson regression algorithms; scaled back when compute test error
Song Year	515345	412276 or 463715	103069 or 51630	90	standardize to zero-mean unit variance; statistics acquired by using training set	Scale to [0, 1]; scaled back when compute test error

Figure 11: Data preprocessing information.

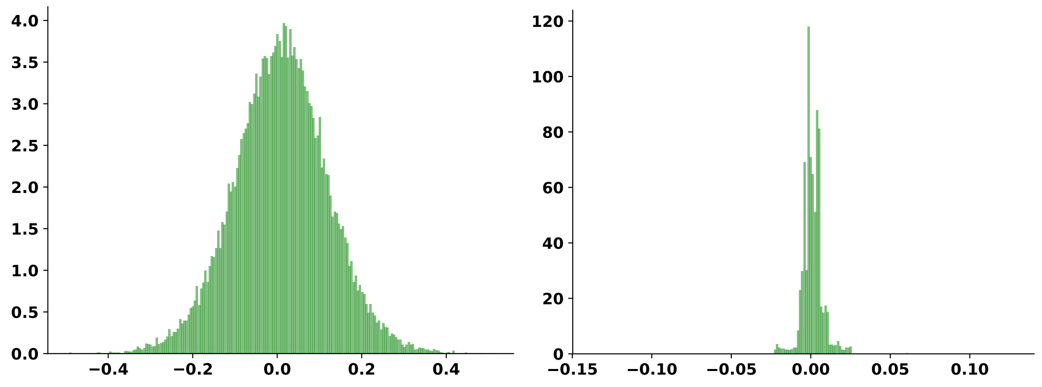
A.3 Additional experimental results

In this section, we provide the following additional experiments.

1. A.3.1: Visualization of the empirical density function $f_\theta(X, Y)$ after training.
2. A.3.2: Predictions on the single-circle and double circle datasets corresponding to learning curves from Section 4.1.
3. A.3.3: Learning curves in terms of Hausdorff distance for experiments in Section 4.1.
4. A.3.4: Additional experiments to show that our Implicit objective learns a finer neural network representation on the high frequency data as introduced in Section 4.2.
5. A.3.5: Learning curves in terms of Hausdorff distance for experiments on the Modal regression real world dataset in Section 5.
6. The concrete training and testing RMSE and MAE on all regular regression real world datasets.
7. A.3.7: Predictions on a classic inverse problem (Bishop, 1994) in A.3.7.

A.3.1 Examining the learned error function

It should be noted that our learning objective is based on the assumption that the error $\epsilon(X, Y) \sim \mathcal{N}(\mu = 0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\epsilon(X,Y)^2}{2\sigma^2}\right)$. As a sanity check, we visualize the empirical distribution of the trained $f_\theta(X, Y)$ by showing histograms constructed with all training samples. From Figure 12, one can see that the trained errors do look centering around zero. The one trained with a noise-contaminated dataset show a larger variance (i.e. heavier tail) than the one trained with a noise-free dataset.



(a) Targets in training set are noise-contaminated (b) Targets in training set are not noise-contaminated

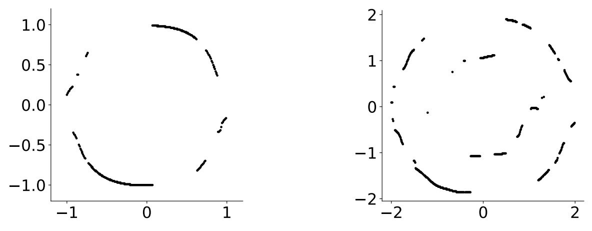
Figure 12: Figure (a) shows the empirical distribution of $f_\theta(X, Y)$ trained with adding zero-mean Gaussian noise with standard deviation $\sigma = 0.1$ and (b) shows the one trained without adding noise to the target.

A.3.2 Predictions on circle datasets

We further verify the effectiveness of our algorithm by plotting the predictions from $S_{global}(\cdot)$ of evenly spaced 200 x values from $[-1, 1]$ for the single-circle dataset and from $[-2, 2]$ for the double-circle dataset. We do not include predictions from S_{local} as the true modes have equal likelihood. The predictions are shown in Figure 13.

A.3.3 Hausdorff distance learning curves on circle datasets

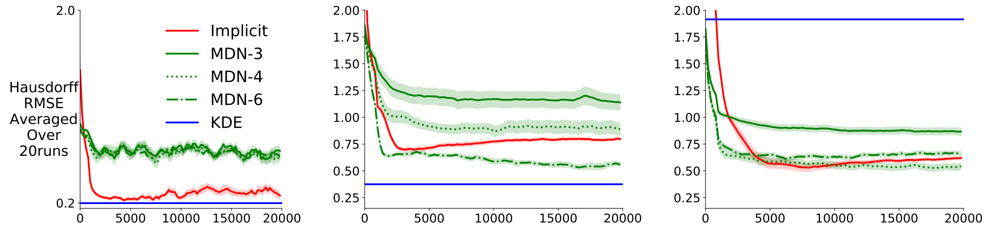
Figure 14 shows the result. One can see that: 1) similar to the result from our main paper, MDN is highly sensitive to number of mixture components and seems to have many spurious predictions even on the single circle dataset; 2) KDE is highly sensitive to datatype (i.e. numerical or categorical) and



(a) single-circle (b) double-circle

Figure 13: (a)(b) shows the predictions of our algorithm on the single-circle dataset by plotting $S_{global}(\cdot)$ for each x.

feature dimension; 3) our algorithm Implicit performs stably and reasonably well across different feature dimension and datatype.



(a) Single-circle (b) Double-circle (c) high dimension double-circle

Figure 14: (a)(b)(c) show testing Hausdorff RMSE as a function of training steps. MDN-3 indicates MDN trained with 3 components. All results are averaged over 20 random seeds and the shaded area indicates standard error.

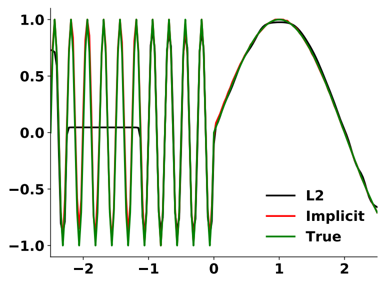
A.3.4 Examining the neural network representation

We further investigate the performance gain of our algorithm by examining the learned neural network representation. We plot the predictions in figure 15(a) and the corresponding NN representation through heatmap in figure 15(b). In a trained NN, we consider the output of the second hidden layer as the learned representation, and investigate its property by computing pairwise distances measured by l_2 norm between 161 different evenly spaced points on the domain $x \in [-2.5, 2.5]$. That is, a point (x, x') on the heatmap in figure 15(b) denotes the corresponding distance measured by l_2 norm between the NN representations of the two points (hence the heatmap shows symmetric pattern w.r.t. the diagonal). For our algorithm, the target input is given by minimizing our implicit learning objective.

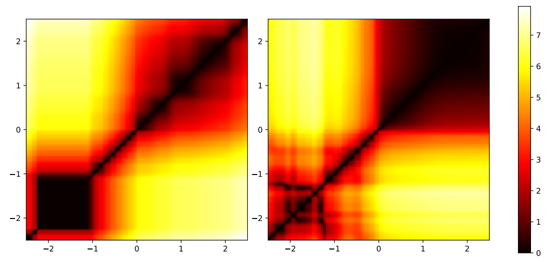
The representations provide some insight into why Implicit outperformed the l_2 regression. In figure 15(a), the l_2 regression fails to learn one part of the space around the interval $[-2.25, -1.1]$. This corresponds to the black area in the heatmap, implying that the l_2 distance between NN representations among those points are almost zero. Additionally, one can see that the heatmap of our approach shows a clearly high resolution on the high frequency area and a low resolution on the low frequency area, which coincides with our intuition for a good representation: in the high frequency region, the target value would change a lot when x changes a little, so we expect those points to have finer representations than those in low frequency region. This experiment shows that given the same NN size, our algorithm can better leverage the representation power of the NN.

A.3.5 Hausdorff distance learning curves for predicting insurance cost

Figure 16 shows the result. Note that Hausdorff-RMSE means that for each testing point, we use square error for $d(x, y)$ in (5) and then take the root of the mean Hausdorff distance of all testing



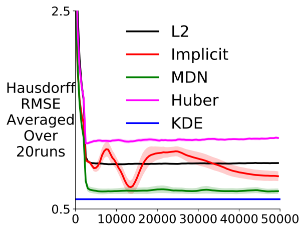
(a) Predicted functions



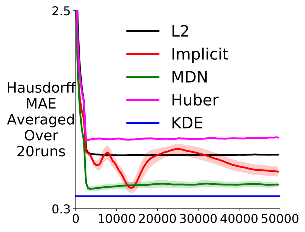
(b) Distance heatmap

Figure 15: (a) Approximated and true functions. (b) The distance matrix showed in heat map computed by hidden layer representation learned by **L2** (left) and **Implicit** (right) method.

points. Similarly, for Hausdorff-MAE, we use absolute difference for $d(x, y)$ then we take the mean Hausdorff distance of all testing points.



(a) RMSE



(b) MAE

Figure 16: Figure (a) (b) shows Hausdorff w.r.t. RMSE and MAE respectively. The results are averaged over 20 random seeds.

A.3.6 Detailed results on standard regression tasks

Tables 1 and 2 show the performance of the algorithms measured by root mean squared error and mean absolute error.

Table 1: Prediction errors on bike sharing dataset. All numbers are multiplied by 10^2.

Algorithms	Train RMSE	Train MAE	Test RMSE	Test MAE
LinearReg	10094.40(±13.60)	7517.64(±19.95)	10129.40(±59.26)	7504.22(± 44.20)
LinearPoisson	8798.26(±14.58)	5920.99(±13.66)	8864.90(±66.07)	5935.00(± 38.32)
NNPoisson	1620.46(±47.71)	1071.39(±29.55)	4150.03(±77.76)	2616.49(± 20.45)
L2	1919.74(±23.12)	1421.36(±21.35)	3726.40(±49.51)	2526.77(±27.89)
Huber	1914.34(±33.87)	1398.41(±21.11)	3675.03(±40.67)	2487.61(±11.05)
MDN	2888.06(±64.55)	1456.31(±76.21)	3948.48(±63.95)	**2298.47**(±36.37)
MDN(worst)	3506.32(±166.30)	1604.43(±35.90)	4452.52(±166.65)	2431.40(±41.31)
Implicit	2075.33(±7.01)	1504.63(±4.34)	**3674.08**(±16.82)	2419.54(±12.22)
Implicit(worst)	2154.70(±7.55)	1602.03(±5.35)	3749.95(±16.36)	2514.99(±13.51)

A.3.7 A classic inverse problem

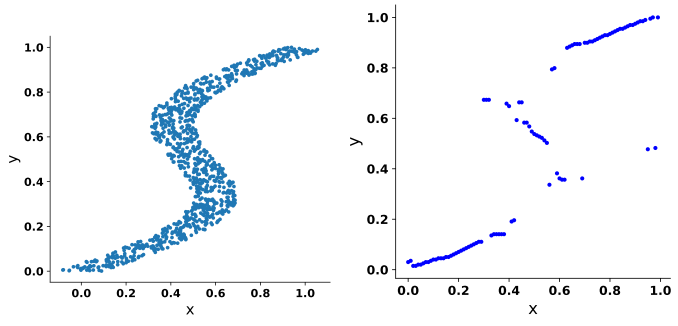
One important type of applications of multi-value function prediction is inverse problem. We now show additional results on a classical inverse function domain as used in (Bishop, 1994). The learning dataset is composed as following.

$$x = y + 0.3\sin(2\pi y) + \xi, y \in [0, 1] \tag{6}$$

Table 2: Prediction errors on song year dataset. All numbers are multiplied by 10^2.

Algorithms	Train RMSE	Train MAE	Test RMSE	Test MAE
LinearReg	956.40($\pm$0.37)	681.56($\pm$0.68)	957.56($\pm$1.49)	681.66($\pm$1.52)
L2	850.20($\pm$2.50)	590.18($\pm$1.63)	895.77($\pm$2.75)	608.42($\pm$1.03)
Huber	872.56($\pm$5.00)	569.49($\pm$1.75)	898.33($\pm$2.67)	**581.48**($\pm$1.37)
MDN	955.85($\pm$12.97)	605.58($\pm$4.02)	957.76($\pm$13.53)	615.57($\pm$4.37)
MDN(worst)	1699.48($\pm$239.09)	1212.82($\pm$230.93)	1700.11($\pm$240.09)	1215.60($\pm$231.33)
Implicit	876.71($\pm$1.88)	598.68($\pm$4.11)	**890.61**($\pm$2.87)	606.91($\pm$3.48)
Implicit(worst)	886.83($\pm$2.92)	604.16($\pm$2.08)	896.08($\pm$3.00)	612.25($\pm$2.53)

where ξ is a random variable representing noise with uniform distribution $U(-0.1, 0.1)$. We generate 80k training examples. In Figure 17, we plot the training dataset, and predictions by our implicit function learning algorithm with $(\arg\min_y f_\theta(x,y)^2 + (\frac{\partial f_\theta(x,y)}{\partial y} + 1)^2)$. We search over 200 evenly spaced ys in $[0,1]$ for 200 evenly spaced $x \in [0,1]$ to get points in the form of (x, y)s.



(a) Training data (b) Predictions by Implicit

Figure 17: Figure (a) shows what the training data looks like. (b) shows the predictions of our implicit learning approach.

A.4 Related Areas

We would like to include a brief discussion about the related areas to our parametric modal regression approach. For the purpose of estimating modes, generative models (Jebara & Pentland, 2002; Li et al., 2020) can be also used. However, learning a generative model may be a more difficult task. It is worth doing a rigorous comparison between our method and some classical generative models in terms of sample efficiency. M-best MAP inference (Batra et al., 2012) may be also adapted to predict conditional modes. A conceptually similar but distinct work is score matching (Hyvärinen, 2005), where the score function can be the derivative of the probability density function w.r.t. data and the score function evaluated at observations in the training set would be pushed to zero.