Expected Pinball Loss For Inverse CDF Estimation

Anonymous authors
Paper under double-blind review

Abstract

We analyze and improve a recent strategy to train a quantile regression model by minimizing an expected pinball loss over all quantiles. We give an asymptotic convergence rate that shows that minimizing the expected pinball loss can be more efficient at estimating single quantiles than training with the standard pinball loss for that quantile, an insight that generalizes the known deficiencies of the sample quantile in the unconditioned setting. Then, to guarantee a legitimate inverse CDF, we propose using flexible deep lattice networks with a monotonicity constraint on the quantile input to guarantee non-crossing quantiles, and show lattice models can be regularized to the same location-scale family. Our analysis and experiments on simulated and real datasets show that the proposed method produces state-of-the-art legitimate inverse CDF estimates that are likely to be as good or better for specific target quantiles.

1 Introduction

Real world applications often seek estimates of the quantiles of a random variable. For example, a pizza place would like to tell customers that 90% of the time, their order will be delivered within 40 minutes. In addition, we may condition the estimate on features. For example, conditioned on the time-of-day being 5pm, and the order being for twenty pizzas, the conditional estimate is that 90% of the time their order will arrive in less than 73 minutes.

When there are no features, let random variable $Y \in \mathbb{R}$ have cumulative distribution function (CDF) F, and for $\tau \in (0, 1)$, the τ-quantile of Y is defined as $q_\tau = F^{-1}(\tau) = \inf\{q : P(Y \leq q) \geq \tau\}$. In the *conditional* setting, the pair of random variables $(X, Y) \in \mathbb{R}^D \times \mathbb{R}$, and the *conditional* τ-quantile of Y for feature vector X is defined as $q_\tau(x) = \inf\{q : P(Y \leq q|X = x) \geq \tau\}$.

Quantile regression takes as training data a set of n pairs $(x, y) \in \mathbb{R}^D \times \mathbb{R}$ from a joint distribution over (X, Y), and forms an estimator for one or more of the conditional quantiles of Y for any value of X. A standard objective to train a model to estimate the quantile for τ is to minimize the *pinball loss* (Koenker & Bassett, 1978), $L_\tau(y, \hat{y}) = \max(\tau(y - \hat{y}), (\tau - 1)(y - \hat{y}))$ for $y, \hat{y} \in \mathbb{R}$. In the unconditioned case with no features X, the training data is simply a set of n scalar values $\{y_i\}_{i=1}^n$, and minimizing the pinball loss has the satisfying property that it selects the empirical quantile of the training set (Chernozhukov, 2005).

Recent work by Tagasovska & Lopez-Paz (2019) proposed training one deep neural network (DNN) model $f(x, \tau)$ that takes τ as an input, and is trained to minimize an *expected* pinball loss, where τ is drawn from the uniform distribution. That work is important because it provides *simultaneous quantile regression* of the entire inverse CDF. However, that work left open a couple of important theoretical and practical issues that we address in this paper.

The first issue is whether one pays a price in estimation accuracy for a model $f(x, \tau)$ that can predict any quantile, compared to a model trained specifically for one target quantile. Surprisingly, we show theoretically with a convergence rate analysis that learning the full inverse CDF model $f(x, \tau)$ can produce a more efficient estimator for a single quantile than minimizing the pinball loss for just a single quantile, depending on the true distribution, quantile, and function class. Our real-world experiments confirm that in practice one does often win on single quantile estimates by training with the *expected* pinball loss, though the full inverse CDF model $f(\tau; x)$ may take a bit longer to train and a bit more memory to store.

The second issue we address is that training a DNN model with the expected pinball loss as proposed by Tagasovska & Lopez-Paz (2019) may fail to produce a legitimate inverse CDF because it does not guarantee *non-crossing quantiles*: that any two quantile estimates satisfy the common sense requirement that $q_\tau(x) \geq q_{\tau'}(x)$ for $\tau \geq \tau'$ at every x. For example, a model that does not guarantee non-crossing quantiles could tell a customer there is a 90% chance their delivery will arrive within 38 minutes, but an 80% chance it will arrive within 40 minutes. Such a model may make a customer confused and distrust the model, worrying it is broken or buggy.

Embarrassing quantile-crossing mistakes are known to happen often in quantile regression in general (He, 1997; Koenker & Bassett, 1978). Tagasovska & Lopez-Paz (2019) hypothesized that training with an expected pinball loss induces smoothing across τ that would reduce non-crossing. However, we show experimentally (Table 1) that a flexible model like a DNN optimized to minimize the expected pinball loss easily suffers a problematic amount of quantile crossing. To address non-crossing without losing model flexibility, we propose using deep lattice networks (DLNs) with a monotonicity shape constraint on τ to guarantee non-crossing quantiles, thus providing legitimate inverse CDF estimates. Additionally, we show that the DLN function class is amenable to two additional kinds of useful regularization for quantile regression: restricting the learned inverse CDF to a location-scale family, and restricting other features to have only monotonic effect on the predictions.

Next, in Section 2 we formalize the problem of producing legitimate inverse CDF estimates with quantile regression. Then in Section 3 we dig into what is known about beating the pinball loss by smoothing over multiple quantile estimates, and we give a new asymptotic convergence rate for minimizing the expected pinball loss. In Section 4 we propose using DLNs as the model architecture. Experiments in Section 5 on simulations and real-world data show that the proposed non-crossing DLNs provide competitive, trust-worthy estimates.

2 A Legitimate Inverse CDF

Let $\{(x_i, y_i)\}_{i=1}^n$ be a training set where each $x_i \in \mathbb{R}^D$ is a feature vector and $y_i \in \mathbb{R}$ is a corresponding label. Denote the inverse CDF estimator $f(x, \tau; \theta)$, where $\tau \in (0, 1)$ is an auxiliary feature that specifies the quantile of interest, and the model $f : \mathbb{R}^{D+1} \to \mathbb{R}$ is parameterized by $\theta \in \mathbb{R}^m$.

Recall the standard pinball loss given τ is $L_\tau(y, \hat{y}) = \max(\tau(y-\hat{y}), (\tau-1)(y-\hat{y}))$. Let $\mathcal{T} \sim \text{unif}(0, 1)$ denote a random τ so we can minimize the *expected* pinball loss over $\mathcal{T}$ to train $f(x, \tau; \theta)$ (Tagasovska & Lopez-Paz, 2019). In addition, we constrain the empirical loss minimization with non-crossing constraints, producing the following constrained training objective:

$$\min_\theta \mathbb{E}_\mathcal{T} \sum_{i=1}^n L_\mathcal{T}(y_i, f(x_i, \mathcal{T}; \theta)) \tag{1}$$

$$\text{s.t. } f(x, \tau^+; \theta) \geq f(x, \tau^-; \theta) \, \forall x \in \mathbb{R}^D, \tau^+ \geq \tau^-. \tag{2}$$

The idea of training one model that can predict multiple quantiles by minimizing a sum of pinball losses with non-crossing constraints dates back to at least the 2006 work of Takeuchi et al. (2006) for a *pre-specified, discrete set* of quantiles. Other prior work used a similar mechanism for a *pre-specified discrete set* of quantiles, and for *more-restrictive* function classes that are monotonic on τ, such as linear models (Bondell et al., 2010), and two-layer *monotonic* neural networks (Cannon, 2018), which are known to have limited flexibility (Daniels & Velikova, 2010). Monotonic DNN models with more layers (Minin et al., 2010; Lang, 2005) or min-max layers (Sill, 1998) can theoretically provide universal approximations, but have not performed as well experimentally as DLNs (You et al., 2017). Monotonic neural nets have also been proposed for estimating a *CDF* (Chilinski & Silva, 2018)). Training only for (1) without (2) (Tagasovska & Lopez-Paz, 2019) does not guarantee a legitimate inverse CDF because the estimated quantiles can cross.

There are two main prior approaches to estimating a *complete* legitimate inverse CDF. The first is nonparametric. For example, k-NN can be extended to predict quantiles by taking the quantiles rather than the

mean from within a neighborhood (Bhattacharya & Gangopadhyay, 1990). Similarly, quantile regression forests (Meinshausen, 2006) use random forests leaf nodes to generate local empirical quantiles.

The second strategy is to predict a parametric inverse CDF for any x. Traditionally these methods have been fairly rigid. He (1997) developed a method to fit a shared but learned location-scale family across x. Yan et al. (2018) proposed a modified 4-parameter Gaussian whose skew and variance was dependent on x. Recently, Gasthaus et al. (2019) proposed the *spline quantile function DNN* (SQF-DNN), which is a DNN model that takes an input x, and outputs the parameters for a monotonic piecewise-linear quantile function on τ. SQF-DNN can fit any continuous bounded distribution, given sufficient output parameters, and because for any x the output is a monotonic function on τ, guarantees no quantile crossing. Though Gasthaus et al. (2019) focused on RNNs, their framework is easy to apply to the generic quantile regression setting as well, which we do in our experiments.

3 Comparison To The Single Quantile Pinball Loss

We show through simulations and convergence rate analysis that minimizing the expected pinball loss can be better at estimating the τ-quantile than minimizing the pinball loss just for τ, and that these positive results are foreshadowed by classic results for unconditional quantile estimates. This section focuses on the unconditioned case (no features x), we will return to the problem given features x in Section 4.

3.1 Deficiency Of The Sample Quantile

Given only samples from Y, minimizing a pinball loss produces the corresponding sample quantile (Chernozhukov, 2005). However, it has long been known that simply taking the sample quantile is not necessarily the best quantile estimator, and in particular that it can be beaten by strategies that smooth multiple sample quantiles (Harrell & Davis, 1982; Kaigh & Lachenbruch, 1982; David & Steinberg, 1986; Reiss, 1980; Falk, 1984).

One important classic result is the Lehmann-Schaffe theorem, which says that if the data are known to be from a uniform $[a, b]$ distribution, then the average of the sample min and sample max, rather than the sample median, is the minimum variance unbiased estimator for the median.

Even if we do not know anything about the true distribution, general-purpose methods that smooth multiple quantiles of the data when estimating a particular quantile have proven effective. In fact, the widely used Harrell-Davis estimator is a weighted average of all the sample order statistics where the weights depend on τ, and the estimator for the median is asymptotically equivalent to computing the mean of bootstrapped medians (Harrell & Davis, 1982).

The prior work above suggests that there can be a better training loss than τ-specific pinball loss for the general problem of quantile regression given features - one that increases efficiency by better smoothing across the quantiles. Indeed, by minimizing the expected pinball loss as in (1), the estimate of any quantile is affected by the other quantiles (given some smoothness in f). We provide some further intuition about how it smooths in Appendix A.4.

3.2 Comparison to Harrell-Davis

To emphasize the similarity and value of minimizing the expected pinball loss, we present a comparison to the classic Harell-Davis estimator, which only works in the unconditioned setting of estimating $f(\tau; \theta)$. Without x, the proposed DLN function class simply reduces to a piecewise-linear function (PLF) $f(\tau; \theta)$, where the parameters θ are the values of $f(\tau; \theta)$ at K keypoints pinned at the K sample quantiles of the data. This simple DLN is trained on (1), and the non-crossing constraints (2) require the fitted PLF to be monotonically-increasing.

We simulate Y drawn from an exponential distribution. The results are shown in Fig. 1 (a) for predicting the median given $N = 51$ training samples. As expected, the sample median (green line) is not as good as the smoothed Harell-Davis estimate (orange line). The performance of the proposed DLN trained to

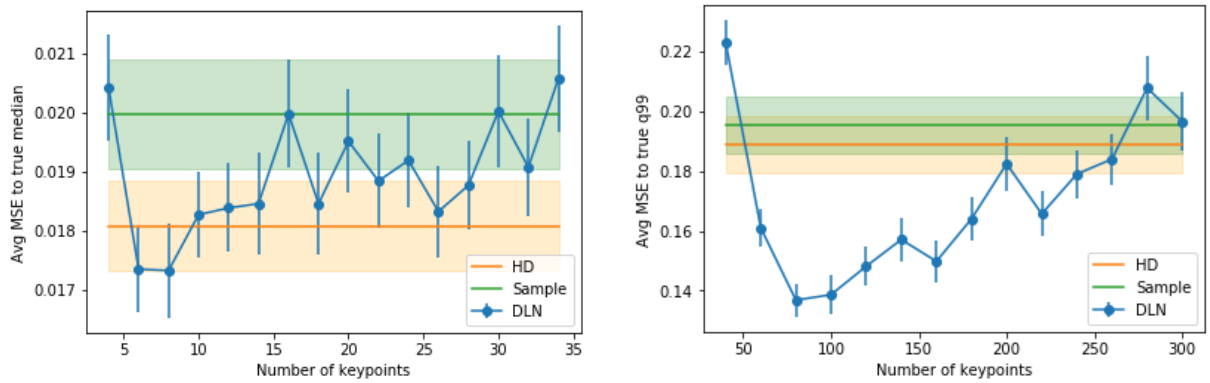


Figure 1: Estimating the quantiles of an exponential distribution (unconditioned). For estimating the median (left, N=51 data points) and 99th quantile (right, N=505), we compare the efficiency of the sample quantile, Harrell-Davis estimator, and learned DLN(here, a PLF) over τ trained with an expected pinball loss. The x-axes show the effect of different numbers of keypoints K in the PLF (the DLN), with more keypoints corresponding to a more flexible function over τ. The y-axis shows the MSE to the true target quantile. Error bars are computed over 5,000 repeats.

minimize expected pinball loss depends on the model flexibility allowed (as expressed by the hyperparamter K keypoints in the PLF) is given to the model $f(x, \tau)$, but over a wide range of model flexibility, the DLN performs comparably to the Harrell-Davis estimator, and better if K is cross-validated judiciously.

More impressively, in Fig. 1 (b) we show the results for estimating the the tail $\tau = 0.99$ from $N = 505$ training samples. Our proposal statistically significantly beats the Harrell-Davis method for a broad range of model flexibility (that is, number of keypoints in the fitted PLF).

The Harrell-Davis and related estimators that only apply to the unconditioned case, whereas training DLNs as per (1) and (2) extend seamlessly to *conditional* quantile estimation with any number of x features, which we return to in Sections 4 and 5.

3.3 Convergence Rate

We give a new asymptotic convergence rate for estimating the full inverse CDF by minimizing the expected pinball loss (1), optionally with a monotonicity constraint (2), and compare that to the convergence rate of single-quantile estimation learned by minimizing an individual pinball loss.

We show that depending on the underlying data distribution and the function class chosen to estimate the inverse CDF, the simultaneous estimation can produce a more efficient estimator for a single target quantile than minimizing the pinball loss for only that quantile.

For tractability, we work in the setting where there are no features x to condition on, only a set of training samples $\{y_i\}$, and we are interested in estimating the quantile $F^{-1}(\tau)$ for a random variable $Y \in \mathbb{R}$ given IID samples $\{y_i\}_{i=1}^n$. Such analysis is relevant to the conditional case because for function classes that are flexible over x, they intuitively will have similar dynamics at each value of x as the unconditioned case we work with here; Section 5.4 provides some experimental evidence.

3.3.1 Review of Single Pinball Loss Convergence Rate

We start with the well-understood case of single-quantile estimation. Let $\theta_\tau^* = \arg\min_\theta E_Y[L_\tau(Y, \theta)]$ be the true minimizer for the corresponding pinball loss and $\hat{\theta}_\tau^{(n)} = \arg\min_\theta \sum_{i=1}^n [L_\tau(y_i, \theta)]$ be the empirical minimizer over our data. (In our context, $\theta^*(\tau)$ is the true τ quantile of Y and $\hat{\theta}_\tau^{(n)}$ is the τ sample quantile of the data (Chernozhukov, 2005).) The asymptotic convergence rate for this single quantile estimate is

given by,

$$\sqrt{n}(\hat{\theta}_\tau^{(n)} - \theta_\tau^*) \xrightarrow{d} \mathcal{N}\left(0, \frac{\tau(1-\tau)}{p(\theta_\tau^*)^2}\right), \tag{3}$$

where $p(y)$ is the density for Y (Koenker, 2005).

3.3.2 Expected Pinball Loss Convergence Rate

Next, consider learning the full inverse CDF $f(\tau;\theta)$ parameterized by $\theta \in \Theta \subseteq \mathbb{R}^m$ by minimizing the expected pinball loss:

$$\theta^* = \arg\min_{\theta\in\Theta} E_{\mathcal{T}}[E_Y[L_{\mathcal{T}}(Y, f(\mathcal{T};\theta))]],$$

where $\mathcal{T} \in (0,1)$ is a random variable drawn from some distribution independently of Y. Note that the monotonicity constraints in Equation (2) can be wrapped in the function class f and feasible parameter set Θ. Let $\hat{\theta}^{(n)}$ be the corresponding minimizer under an empirical distribution over Y with samples $\{y_i\}_{i=1}^n$. In Theorem 1 below, we present a general asymptotic convergence rate for $\hat{\theta}^{(n)}$ that depends on the properties of the parametric function f and the distribution of $\mathcal{T}$. A natural choice for the distribution of $\mathcal{T}$ is Unif$(0,1)$, but in fact, depending on the distribution of Y and properties of the function class $f(\tau;\theta)$, other distributions could lead to lower asymptotic variance.

Theorem 1. *Suppose $f(\tau;\theta)$ is well specified: there exists $\theta^* \in \Theta \subseteq \mathbb{R}^m$ such that $f(\tau;\theta^*) = F^{-1}(\tau)$ for all $\tau \in (0,1)$. Suppose further that the function $\theta \mapsto f(\tau;\theta)$ is continuous, locally Lipschitz, and twice differentiable at $\theta = \theta^*$ for all $\tau \in (0,1)$. Then,*

$$\sqrt{n}(f(\tau;\hat{\theta}^{(n)}) - f(\tau;\theta^*)) \xrightarrow{d} \mathcal{N}\left(0, \nabla_\theta f(\tau;\theta^*)^\top Q^{-1} V (Q^{-1})^\top \nabla_\theta f(\tau;\theta^*)\right) \tag{4}$$

where $Q = E_{\mathcal{T}}[p(f(\mathcal{T};\theta^*))\Gamma(\mathcal{T})]$, $V = E_{\mathcal{T}}[\mathcal{T}(1-\mathcal{T})\Gamma(\mathcal{T})]$, with $\Gamma(\tau) = \nabla_\theta f(\tau;\theta^*)\nabla_\theta f(\tau;\theta^*)^\top$.

Given a specific distribution of $\mathcal{T}$, a function class of the inverse CDF f, and a data distribution Y, we can directly compare the asymptotic convergence rate of the direct quantile estimates (Equation (3)) to that of quantile estimates from the learned inverse CDF (Equation (4)). We illustrate this below with several examples.

3.3.3 Example 1: Single Parameter Uniform

To build intuition for the implications of Theorem 1, we consider a centered uniform distribution $Y \sim$ Unif$(-\frac{\alpha}{2}, \frac{\alpha}{2})$. Let $f(\tau;\theta) = \theta(\tau - \frac{1}{2})$. Expanding the asymptotic variance term in Theorem 1,

$$Q^{-1}V(Q^{-1})^\top = \frac{\alpha^2 E_{\mathcal{T}}[-\mathcal{T}^4 + 2\mathcal{T}^3 - \frac{5}{4}\mathcal{T}^2 + \frac{1}{4}\mathcal{T}]}{E_{\mathcal{T}}[\mathcal{T}^2 - \mathcal{T} + \frac{1}{4}]^2}.$$

This depends on the first four moments of the distribution of $\mathcal{T}$, the choice of which is up to the algorithm designer. For the natural choice of $\mathcal{T} \sim$ Unif$(0,1)$, the asymptotic variance for estimating a specific τ_0-quantile using the function $f(\tau_0;\hat{\theta}^{(n)}) = \hat{\theta}^{(n)}(\tau_0 - \frac{1}{2})$ is $1.2\alpha^2(\tau_0 - \frac{1}{2})^2$. This variance is lowest when $\tau_0 = 0.5$, and increases to $0.3\alpha^2$ at the most extreme quantiles when $\tau = 0$ or $\tau = 1$.

We next compare this to estimating the τ_0-quantile using the single pinball loss L_{τ_0}. Equation (3) shows that the estimate $\hat{\theta}_{\tau_0}^{(n)}$ has asymptotic variance $\alpha^2\tau_0(1-\tau_0)$. This shows that depending on the exact value of τ_0, the single quantile estimate could be less or more efficient than the estimate resulting from evaluating the function $f(\tau_0;\hat{\theta}^{(n)})$. The single quantile estimate is more efficient for the extreme quantiles, whereas the inverse CDF estimate is more efficient for estimating the median. Intuitively, the inverse CDF median estimate benefits from fitting the inverse CDF function to the surrounding data points, which is reminiscent of classic quantile smoothing.

3.3.4 Example 2: Two-Parameter Uniform

We next consider a two-parameter uniform distribution example: for $Y \sim$ Unif(α,β), $p(y) = \frac{1}{\beta-\alpha}$. Let $\theta \in \mathbb{R}^2$ with $f(\tau;\theta) = \theta_0 + \theta_1\tau$. That is, we learn a two-parameter line as our inverse CDF. We then have

$\nabla_\theta f(\tau;\theta) = \begin{bmatrix}1\\ \tau\end{bmatrix}$ and can therefore write the Q and V matrices from Theorem 1 as

$$Q = \frac{1}{\beta-\alpha} E_{\mathcal{T}} \begin{bmatrix}1 & \mathcal{T}\\ \mathcal{T} & \mathcal{T}^2\end{bmatrix}$$
$$V = E_{\mathcal{T}} \begin{bmatrix}\mathcal{T}-\mathcal{T}^2 & \mathcal{T}^2-\mathcal{T}^3\\ \mathcal{T}^2-\mathcal{T}^3 & \mathcal{T}^3-\mathcal{T}^4\end{bmatrix}.$$

Evaluating each of these matrices for $\mathcal{T} \sim \text{Unif}(0,1)$ we arrive at

$$Q^{-1}V(Q^{-1})^\top = (\beta-\alpha)^2 \begin{bmatrix}\frac{7}{15} & -\frac{3}{5}\\ -\frac{3}{5} & \frac{6}{5}\end{bmatrix},$$

and therefore an asymptotic variance for a particular τ_0-quantile estimate $f(\tau_0;\hat{\theta}^{(n)}) = \hat{\theta}_0 + \hat{\theta}_1\tau_0$ of $(\beta-\alpha)^2(\frac{7}{15} + \frac{6}{5}\tau_0^2 - \frac{6}{5}\tau_0)$. This is a quadratic function that finds its minimum at $\tau = 0.5$ and is highest (on the unit interval) at $\tau = 0$ and $\tau = 1$.

We can compare it to the convergence for single-pinball regression of $(\beta-\alpha)^2\tau(1-\tau)$, which is a quadratic function that attains its *maximum* at $\tau = 0.5$.

As we see in Figure 2, the variance is smaller for the expected pinball regression for quantiles between roughly $\tau = 0.3$ and $\tau = 0.7$. The opposite is true for the extreme quantiles.

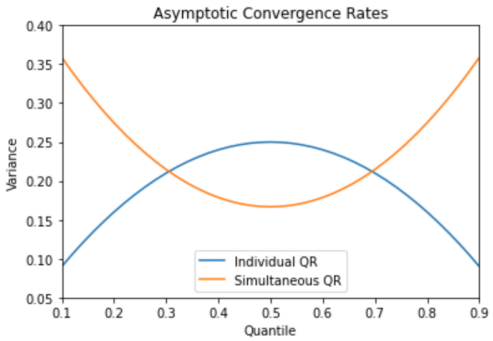


Figure 2: Plots of the asymptotic convergence variances for the expected pinball regression (orange) and individual single-pinball regressions (blue) for different quantiles, when we assume that the true distribution is Uniform(α,β), the expected pinball loss regression learns a linear model, and we assume $\beta - \alpha = 1$. Relaxing the last assumption would scale the y-axis but not change the intersection points

Overall, these examples illustrate the sometimes counterintuitive fact that it can be more asymptotically efficient to estimate a quantile through learning a full inverse CDF than it would be to just estimate the single quantile alone. In learning the full inverse CDF, the algorithm designer also gets to make a key choice in the distribution of $\mathcal{T}$ used in the learning objective. The examples above showed how the asymptotic convergence rates depend on the moments of $\mathcal{T}$, and we empirically revisit the choice of the distribution of $\mathcal{T}$ in experiments Section 5.

4 DLNs for Legitimate but Flexible Inverse CDFs

We propose using DLNs (You et al., 2017) as the function class in (1) because of their flexibility and because they efficiently enable three key regularization strategies: non-crossing constraints, restricting the learned distribution to location-scale families, and monotonic features, as explained below.

4.1 Review of Deep Lattice Networks

Deep lattice networks build upon a truly ancient strategy for approximating functions over a bounded domain (Campbell-Kelly et al., 2003): record the function's values on a regular grid of inputs, and then one can

linearly interpolate between the recorded values. Generalizing to multiple dimensions, a lattice function is a multi-dimensional look-up table whose parameters are the look-up table values, and the function is produced by linearly interpolating the look-up table values surrounding an input point. In the special case of a one-dimensional domain, a lattice is just a PLF. This strategy is very old, but good papers to understand the basics of lattice functions are Garcia et al. (2012) for the spline perspective, and Gupta et al. (2016) for the machine learning perspective (which

Mathematically, a lattice function $f(x) : \mathbf{R}^D \rightarrow \mathbf{R}$ can be expressed $f(x) = \theta^T \psi(x)$, where the parameters $\theta \in \mathbf{R}^p$ are the look-up table values, and the nonlinear transformation $\psi(x) \in \mathbf{R}^p$ maps an input $x \in \mathbf{R}^D$ to the ultra-sparse vector of linear interpolation weights over the look-up table values.

Garcia & Gupta (2009) showed that lattice models can be trained using a standard empirical risk minimization framework. Lattices can approximate any continuous bounded function if they are sampled on a regular grid with enough knots.

A key reason to use lattice functions in an application like quantile regression is that the regular structure of their parameters makes them amenable to shape constraints. Here, the constraint (2) requires $f(x, \tau; \theta)$ to be monotonically increasing in τ, which is imposed efficiently with sparse linear inequality constraints on the lattice parameters corresponding to forcing any neighboring gridpoints in the direction of τ to be increasing (Gupta et al., 2016). To more efficiently represent functions, lattice models are usually architected with a first layer that separately calibrates each input with a learned one-dimensional piecewise linear function (PLF) whose knots need-not be regularly-spaced referred to as a calibrator, then fuses the calibrated inputs together with a coarser multi-dimensional lattice (Garcia et al., 2012; Gupta et al., 2016).

The main drawback to lattices is that the number of lattice parameters needed to approximate a D-dimensional functions blows up exponentially as $O(2^D)$. This problem is handled by making an ensemble of calibrated lattices (Canini et al., 2016; Gupta et al., 2018; 2020) much as one makes a random forest, and these calibration and lattice layers can be mixed with other layers like linear layers, and cascaded ad infinitum, forming arbitrarily *deep lattice networks* (DLNs) (You et al., 2017; Cotter et al., 2019). Importantly, as long as each layer of a DLN is trained to respect a monotonic feature like τ, the entire DLN will be monotonic in τ. More generally, one could use unrestricted ReLU or embedding layers for the first few model layers on x, then fuse in τ later with τ-monotonic DLN layers.

Our DLN's are built with the standard architectures offered by the open-source TensorFlow Lattice library (TensorFlow Blog Post, 2020). The TensorFlow Lattice library enables solving the constrained optimization problem (1) and (2) with Dykstra's projection algorithm. In our experiments, we found our DLN models trained in a similar amount of time as the DNN and SQF-DNN models.

4.2 Location-Scale Family Models

We show that by different architecture choices for the DLN model used can regularize the estimated inverse CDF in semantically-meaningful ways, similar to prior more rigid distributional approaches. As a simple example, if one constrains the DLN architecture to not have any interactions between τ and any of the x features, then one learns a regression model with homoskedastic errors. As another example, the basic two-layer DLN called a *calibrated lattice* model (Gupta et al., 2016) described in Section 4.1 can be constrained to learn distributions across x that come from a shared, learned, location-scale family:

Lemma: Let $f(x, \tau)$ be a calibrated lattice model with piece-wise linear calibrator $c(\tau) : [0, 1] \rightarrow [0, 1]$ for τ, and let there be two knots in the regular grid in the direction of τ, and the lattice parameters are interpolated with standard multilinear interpolation. Then $f(x, \tau)$ represents an inverse CDF function $F^{-1}(y|x)$ where the estimated distribution for every x is from the same location-scale family as the calibrator $c(\tau)$.

The proof is in the Appendix. Note the number of keypoints in $c(\tau)$ controls the complexity of the learned base distribution, allowing the model to approximate location-scale families like the Gaussian, gamma, or Pareto distributions. Then in the second layer of f, the number of knots in the lattice over the $c(\tau)$ input controls how much the distribution should be allowed to vary across x. Two knots in the lattice limits us to a shared location-scale family, as noted above, while three knots in the lattice gives an extra degree of

freedom to shrink or stretch one side of the distribution differently across x. More lattice knots across τ, or ensembling of lattices, or more layers can all be used to move the model towards full generality.

4.3 Other Monotonic Features

Using DLNs also enables imposing monotonicity on the features x if domain knowledge says they should have a monotonic impact on the estimates (Gupta et al., 2016). For example, a pizza delivery time estimator might treat constrain an input quantifying how bad traffic is to only increase estimated quantiles (that is, worse traffic never makes the estimated delivery time shorter).

Our experience with real-world applications of quantile estimation is that real-world quantile estimates do often use features that are past measurements (or otherwise strong correlates) to predict the future distribution of measurements. For example, in the real-world Puzzles experiment in Section 5, we predict the quantiles of how long a club member will keep a rented puzzle, and one of the features is how long they kept their last puzzle. We constrain that input to have a monotonically positive effect on the estimate. (However, earlier hold-times are not necessarily monotonically indicative of the next future hold-time, because they may indicate a strongly decreasing trend over time). Domain knowledge about the relationship between the model inputs-and-ouput that can be captured with monotonicity constraints is a tuning-free, semantically-meaningful regularizer that can improve test performance and increase model explainability (Gupta et al., 2016), and where applicable, even help capture ethical rules (Wang & Gupta, 2020).

5 Experiments

Using simulations and real data, we first show that training a DLN that is monotonic in τ to satisfy both (1) and (2) works well compared to training a DNN for (1) as done in Tagasovska & Lopez-Paz (2019) or the SQF-DNN parametric approach of Gasthaus et al. (2019); the results are in Tables 1 and 2. Then we use the proposed τ-monotonic DLN models to compare training to minimize expected pinball loss versus training to minimize the pinball loss for a specific τ; the results are in Tables 4 and 5.

Bolded results in tables indicate that the metric is not statistically significantly different from the best metric among the models being compared, using an unpaired t-test.

5.1 Model Training Details

All hyperparameters were optimized on validation sets. We used Keras models in TensorFlow 2.2 for the unrestricted DNN comparisons that optimize (1) (Tagasovska & Lopez-Paz, 2019), and the SQF-DNN (Gasthaus et al., 2019), which optimizes the same objective in a different manner while also guaranteeing non-crossing quantile estimates. For DLNs, we used the TensorFlow Lattice library (TensorFlow Blog Post, 2020). For all DNN and DLN experiments, we use the Adam optimizer (Kingma & Ba, 2015) with its default learning rate of 0.001, except where noted. For DNN models, we validated the number of hidden layers and the hidden dimension. For the SQF-DNN, we also validated the number of distribution keypoints. For the smaller DLN models, we used the common two-layer calibrated lattice architecture (Gupta et al., 2016) and validated over its number of calibration keypoints and lattice vertices. For the larger datasets, we used an ensemble of calibrated lattices for the DLN model, and then also validated over the the number and dimensionality of the base models in the ensemble (Canini et al., 2016). For both DLNs and DNNs, we additionally validated over the number of training epochs. Training the different methods considered took roughly equally long.

5.2 Benchmark and Real Datasets Used

We tested on three publicly available datasets and one proprietary dataset for real-world problems. A co-lab will be made available.

Air Quality: The Beijing Multi-Site Air-Quality dataset from UCI (Zhang et al., 2017) contains hourly air quality data from 12 monitoring regions around Beijing. We predict the quantiles of the PM2.5 concentration from $D = 7$ features: temperature, pressure, dew point, rain, wind speed, region, and wind direction. The

DLN model is an ensemble of two-layer calibrated lattice models. We split the data by time (not IID) with earlier examples forming a training set of size 252,481, later examples a validation set of size 84,145, and most recent examples a test set of size 84,145.

Puzzles: This is a dataset from *(name withheld for blind review)*, a private library for wooden jigsaw puzzles. We predict how long a library member will hold a borrowed puzzle given their $D = 5$ previous hold-times. The DLN model is an ensemble of two-layer calibrated lattice models. Based on their domain knowledge, we also constrain the DLN output to be monotonically increasing in the most-recent-past hold-time feature. The 936 train and 235 validation examples are IID from past data, while the 210 test samples are the most recent samples (not IID). The anonymized dataset is publicly available on their website *(website withheld for blind review)*.

Traffic: This is a proprietary dataset from a large internet services company *(name withheld for blind review)* for estimating the time to drive a route. The DLN is a two-layer calibrated lattice whose $D = 4$ inputs are 1 categorical and 3 continuous features. We used 1,000 examples each for training, validation, and testing, with the training examples occurring earlier in time than the validation and test examples (not IID).

Wine: We used the Wine Reviews dataset from Kaggle (Bahri, 2018). We predict the quantiles of wine quality on a 100-point scale. We used $D = 42$ features, including price, a 1-d learned embedding for the country of origin (aka categorical calibration (Gupta et al., 2016)), and 40 Boolean features describing each wine. The DLN model is an ensemble of two-layer calibrated lattice models. The DLN output is constrained to be monotonically increasing in the *price* feature, as suggested in prior work (Gupta et al., 2018). The data was split IID with 84,641 examples for training, 12,091 for validation, and 24,184 for testing.

5.3 Model Architecture Experiments

We start by demonstrating the efficacy of using monotonic DLNs to predict the inverse CDF on simulations, and then on the real data.

5.3.1 Simulations

We used simulations from a recent quantile regression survey paper (Torossian et al., 2020) based on the sine, Griewank, Michalewicz, and Ackley functions with carefully designed noise distributions to represent a range of variances and skews across the respective input domains. We used 250 training examples for the 1-D sine and Michalewicz functions, 1,000 examples for the 2-D Griewank function, and 10,000 examples for the 9-D Ackley function.

Our metric is the average squared difference between the estimated and true quantile curves sampled at 99 uniform quantiles $\tau \in 0.01, \ldots, 0.99$, averaged over 100 repeats, and averaged over values of x across the domain (we use fine grids of 1,000 and 10,000 x values for the 1-D and 2-D experiments, respectively, and a random sample of 10,000 x-values for the 9-D experiment). We also report the fraction of test points for which at least two of their 99 quantiles crossed.

Results in Table 1 demonstrate that the proposed monotonic-in-τ DLNs are the best or statistically tied for the best across all four disparate simulations. The DNNs were trained with the same sampled expected pinball loss, and are sometimes close in performance, but suffer substantially from crossing quantiles. For example, on the sine-skew distribution, quantile-crossing was observed for the DNN model between at least two quantiles on 38% of test x values!

Spline quantile functions (Gasthaus et al., 2019) avoid crossing by construction, but their accuracy was much worse than the DLN accuracy on three of the four datasets, and not better on the fourth. In particular, the DLNs do much better at the Griewank and Ackley simulations, which have the lowest signal-to-noise ratios (Torossian et al., 2020).

5.3.2 Real Data

Table 2 compares these models on the four real datasets. We report the pinball loss averaged over $\tau in \{0.01, 0.02, \ldots, 0.99\}$, which is a standard metric for judging conditional quantile estimates, and one

Table 1: Simulations: Quantile MSE and percent crossing violations for $\tau \in \{0.01, 0.02, \ldots, 0.99\}$.

	Sine-skew (1,7)	Griewank	Michalewicz	Ackley
Model	MSE, *Crossing*	MSE, *Crossing*	MSE, *Crossing*	MSE, *Crossing*
DNN	$\mathbf{3.53 \pm 0.09}$, *38%*	1.29 ± 0.02, *5%*	0.311 ± 0.011, *12%*	237 ± 6, *0.3%*
SQF-DNN	5.68 ± 0.11, *0%*	1.05 ± 0.01, *0%*	$\mathbf{0.232 \pm 0.006}$, *0%*	667 ± 81, *0%*
DLN	$\mathbf{3.51 \pm 0.13}$, *0%*	$\mathbf{0.55 \pm 0.01}$, *0%*	$\mathbf{0.219 \pm 0.006}$, *0%*	$\mathbf{206 \pm 2}$, *0%*

Table 2: Real data experiments: Pinball loss on the test set, averaged over $\tau \in \{0.01, 0.02, \ldots, 0.99\}$.

Model	Air Quality	Puzzles	Traffic	Wine
DNN	18.041 ± 0.088	3.253 ± 0.024	0.0494 ± 0.00037	0.6603 ± 0.0088
SQF-DNN	$\mathbf{17.356 \pm 0.106}$	$\mathbf{3.108 \pm 0.044}$	0.0491 ± 0.00022	0.6552 ± 0.0037
DLN	$\mathbf{17.280 \pm 0.157}$	$\mathbf{3.092 \pm 0.020}$	$\mathbf{0.0479 \pm 0.00003}$	$\mathbf{0.6381 \pm 0.0005}$

which approximates the *continuous ranked probability score* metric used for measuring the quality of probabilistic forecasts (Gasthaus et al., 2019; Yan et al., 2018). The τ-monotonic DLNs performed the best or statistically similar to the best on three of the four problems, and was statistically significantly better than the DNNs trained with the same expected pinball loss for all four problems. Unlike the simulation results in Table 1, the SQF-DNN (monotonic by construction) performs notably better on these real datasets than the DNN in Table 2, but is never better than the DLN.

We also report the *absolute deviation calibration error* (Chung et al., 2021) in Table 3 (some authors use a squared variant (Kuleshov et al., 2018)), the formula is given in A.6. By that metric the DLN is stat. sig. better in 3 of the 4 cases than SQF-DNN, with substantial wins over SQF-DNN on Traffic and Wine.

Table 3: Real data experiments: Calibration on the test set, averaged over $\tau \in \{0.01, 0.02, \ldots, 0.99\}$.

Model	Air Quality	Puzzles	Traffic	Wine
DNN	0.0761 ± 0.0038	0.0826 ± 0.0031	0.0562 ± 0.0014	0.0253 ± 0.0010
SQF-DNN	0.0798 ± 0.0042	$\mathbf{0.0547 \pm 0.0031}$	0.0600 ± 0.0062	0.0671 ± 0.0083
DLN	$\mathbf{0.0588 \pm 0.0057}$	0.0681 ± 0.0052	$\mathbf{0.0210 \pm 0.0027}$	$\mathbf{0.0125 \pm 0.0031}$

5.4 Expected Pinball Loss Experiments

We next empirically show that training a full inverse CDF by minimizing the expected pinball loss is at least as good at estimating specific quantiles as training with the pinball loss just for those specific quantiles.

5.4.1 Simulations

We ran simulations on the 1D sine-skew function as per Torossian et al. (2020), with three noise scenarios: noise parameters $(a, b) = (1, 1)$ for symmetric low-noise, $(7, 7)$ for symmetric high-noise, and $(1, 7)$ for asymmetric high-noise (see Fig. 4 in the Appendix). These simulations are the simplest extension of the unconditioned context studied empirically and theoretically in Section 3 to the conditioned case, in that there is a single x feature, and enough data for a flexible quantile regression model to approximate the corresponding sample quantile of y within a small neighborhood of any value of x.

The results shown in Table 4 are the error in predicting the median given either $N = 100$ or $N = 1,000$ training samples. For both symmetric-noise cases $(1, 1)$ and $(7, 7)$, the model trained with an expected pinball loss produces much more accurate median estimates. This remains true as we increase the number of data points from 100 to 1000, at which point there are about 50 data points within each 0.1 region of $x \in [-1, 1]$.

Table 4: Sine-skew experiment with different sine-skew noise choices (1,1), (7,7) and (1,7). MSE between true median and estimated median, averaged over $x \sim \text{Unif}(-1, 1)$.

(a, b)	N	min $E_{\mathcal{T}}$	min single-τ loss
$(1, 1)$	100	$\mathbf{0.421 \pm 0.025}$	0.588 ± 0.036
$(1, 1)$	1,000	$\mathbf{0.050 \pm 0.003}$	0.067 ± 0.003
$(7, 7)$	100	$\mathbf{7.334 \pm 0.367}$	9.860 ± 0.560
$(7, 7)$	1,000	$\mathbf{0.967 \pm 0.058}$	1.470 ± 0.086
$(1, 7)$	100	$\mathbf{4.458 \pm 0.259}$	$\mathbf{4.416 \pm 0.386}$
$(1, 7)$	1,000	$\mathbf{0.451 \pm 0.037}$	$\mathbf{0.542 \pm 0.067}$

However, for the simulation with highly skewed noise $(1, 7)$ it is less useful to smooth across τ, and the expected and single pinball loss models perform similarly.

5.4.2 Real Data

In Table 5 we give the accuracy at predicting three specific target quantiles on the four real datasets by either training a full inverse CDF by minimizing $E_{\mathcal{T}}$ where $\mathcal{T}$ is drawn uniformly from $(0, 1)$; or training one model to minimize the discrete sum of the three pinball losses for the τs considered; or training three separate models that minimize each specific τ's pinball loss.

The test metric is computed on finite test sets, so the results are not as clean as the simulations where we can compare to the true quantiles.

Table 5: Effect on the accuracy of single quantile estimates. We compare: one model trained to minimize the expected pinball loss with respect to the uniform distribution over $\mathcal{T}$; one model trained to minimize the sum of the three pinball losses for the three quantiles shown; and three separate models trained with individual τ pinball losses. All models were DLNs with same number and dimensionality of base models as in Table 2, and monotonic on τ and the input features.

Air Quality:	Model	Pinball loss ($\tau = 0.5$)	Pinball loss ($\tau = 0.9$)	Pinball loss ($\tau = 0.99$)
	$\tau \sim \text{Unif}(0, 1)$	$\mathbf{23.576 \pm 0.047}$	$\mathbf{14.850 \pm 0.075}$	2.947 ± 0.025
	$\tau \sim$ Discrete	24.083 ± 0.073	15.439 ± 0.062	3.042 ± 0.012
	Single τ	$\mathbf{23.634 \pm 0.068}$	$\mathbf{14.908 \pm 0.092}$	$\mathbf{2.700 \pm 0.013}$
Puzzles:	Model	Pinball loss ($\tau = 0.5$)	Pinball loss ($\tau = 0.7$)	Pinball loss ($\tau = 0.9$)
	$\tau \sim \text{Unif}(0, 1)$	$\mathbf{4.173 \pm 0.013}$	$\mathbf{4.219 \pm 0.021}$	2.705 ± 0.029
	$\tau \sim$ Discrete	4.359 ± 0.014	$\mathbf{4.204 \pm 0.011}$	$\mathbf{2.614 \pm 0.010}$
	Single τ	4.293 ± 0.013	4.350 ± 0.022	2.708 ± 0.012
Traffic:	Model	Pinball loss ($\tau = 0.5$)	Pinball loss ($\tau = 0.9$)	Pinball loss ($\tau = 0.99$)
	$\tau \sim \text{Unif}(0, 1)$	$\mathbf{0.064386 \pm 0.00014}$	$\mathbf{0.040159 \pm 0.00018}$	$\mathbf{0.010645 \pm 0.00018}$
	$\tau \sim$ Discrete	$\mathbf{0.064578 \pm 0.00028}$	$\mathbf{0.039804 \pm 0.00014}$	0.011290 ± 0.00014
	Single τ	$\mathbf{0.064791 \pm 0.00012}$	$\mathbf{0.039900 \pm 0.00010}$	$\mathbf{0.01070 \pm 0.00018}$
Wine:	Model	Pinball loss ($\tau = 0.1$)	Pinball loss ($\tau = 0.5$)	Pinball loss ($\tau = 0.9$)
	$\tau \sim \text{Unif}(0, 1)$	0.4099 ± 0.0006	$\mathbf{0.8889 \pm 0.0017}$	0.3773 ± 0.0019
	$\tau \sim$ Discrete	0.4094 ± 0.0005	0.8891 ± 0.0010	0.3706 ± 0.0003
	Single τ	$\mathbf{0.4049 \pm 0.0004}$	$\mathbf{0.8867 \pm 0.0006}$	$\mathbf{0.3663 \pm 0.0003}$

The inverse CDF model is statistically significantly tied or better in every case for the median $\tau = 0.5$, which is the quantile we expected the smoothing of the expected pinball loss to be most helpful. We hypothesize that the win in median accuracy for Puzzles is due to the small size of the train set and high randomness of the truth, making that problem most helped by the regularization of τ-smoothing. Even for the more extreme quantiles of $\tau = 0.7$ and $\tau = 0.9$, the full inverse CDF model is statistically significantly better than training for those specific quantiles (though for $\tau = 0.9$, the model trained for three discrete quantiles is even better).

We expected the $E_{\mathcal{T}}$ loss to be least useful for extreme quantiles, since the smoothing will be more one-sided. Indeed, the wins for the single-quantile models are on more extreme quantiles: the 99th percentile for Air Quality, and the 10th and 90th percentile for Wine. For Wine, we hypothesize this is because it is a fairly easy regression problem with a large number of training samples and one continuous feature (wine price) that correlates highly with the label (wine quality), so the extra regularization is not helping. But for the large Traffic dataset, even the 99th percentile is statistically tied, and the single quantile model is worse even at the extreme quantiles for Puzzles. Overall, the results in Table 5 show that the full inverse CDFs are at least as good as the single-quantile models.

6 Conclusions

We gave theoretical and empirical evidence that training quantile regression models by minimizing the expected pinball loss can perform as well or better on any individual quantile than models trained for that quantile alone. We showed that minimizing the expected pinball loss is not sufficient to provide legitimate inverse CDF estimates, one must also produce a model whose quantiles are guaranteed to not cross. We showed that DLN models can be effectively constrained to be monotonic in τ and produce state-of-the-art quantile estimates. The non-crossing issue is important not just for theoretical reasons, but because it will be confusing to any non-expert end-user if a model's quantile estimates cross, eroding trust in that model, and by association, all AI.

A key open question is the choice of distribution of $\mathcal{T}$ when minimizing the expected pinball loss. Is a uniform distribution (as done here and in Tagasovska & Lopez-Paz (2019)) most efficient? If one is only interested in a single τ would a distribution more peaked around that τ produce better results?

Broader Impact Statement

We hope the broader impact of this work to be positive in increasing the deserved trustworthiness of AI by promoting quantile regression models with no crossing-quantiles. The alternative of quantile estimates that cross will undoubtedly erode public confidence in AI. For example, prior work would launch a quantile regression model that predicts there is a 90 percent chance your pizza will be delivered in 30 minutes, and that there is an 80 percent chance it will be delivered in 34 minutes. Such models are unnecessarily confusing and untrustworthy for non-experts, and using them in practice may then reduce the ability of worthy AI to have a positive impact on the world.

References

D. Bahri. Wine reviews. *Kaggle*, 2018. URL `https://www.kaggle.com/dbahri/wine-ratings`.

P. K. Bhattacharya and A. K. Gangopadhyay. Kernel and nearest-neighbor estimation of a conditional quantile. *The Annals of Statistics*, 1990.

H. D. Bondell, B. J. Reich, and H. Wang. Non-crossing quantile regression curve estimation. *Biometrika*, 2010.

M. Campbell-Kelly, M. Croaken, and E. Robson. *The History of Mathematical Tables: From Sumer to Spreadsheets*. Oxford University Press, Oxford, England, 2003.

K. Canini, A. Cotter, M. M. Fard, M. R. Gupta, and J. Pfeifer. Fast and flexible monotonic functions with ensembles of lattices. *NeurIPS*, 2016.

A. J. Cannon. Non-crossing nonlinear regression quantiles. *Stochastic Environmental Research and Risk Assessment*, 32:3207–3225, 2018.

V. Chernozhukov. Extremal quantile regression. *Annals of Statistics*, 33(2), 2005.

P. Chilinski and R. Silva. Neural likelihoods via cumulative distribution functions. *arXiv*, 2018.

Y. Chung, W. Neiswanger, I. Char, and J. Schneider. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *NeurIPS*, 2021.

A. Cotter, M. R. Gupta, H. Jiang, E. Louidor, J. Muller, T. Narayan, S. Wang, and T. Zhu. Shape constraints for set functions. *ICML*, 2019.

H. Daniels and M. Velikova. Monotone and partially monotone neural networks. *IEEE Trans. Neural Networks*, 21(6):906–917, 2010.

C. E. David and S. M. Steinberg. Quantile estimation. In *Encyclopedia of Statistical Sciences*, volume 7. Wiley, New York, 1986.

M. Falk. Relative deficiency of kernel type estimators of quantiles. *Annals of Statistics*, 12, 1984.

E. K. Garcia and M. R. Gupta. Lattice regression. *NeurIPS*, 2009.

E. K. Garcia, R. Arora, and M. R. Gupta. Optimized regression for efficient function evaluation. *IEEE Trans. Image Processing*, 21(9):4128–4140, September 2012.

J. Gasthaus, K. Benidis, Y. Wang, S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski. Probabilistic forecasting with spline quantile function RNNs. *AIStats*, 2019.

M. R. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. Canini, A. Mangylov, W. Moczydlowski, and A. Van Esbroeck. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research (JMLR)*, 17(109):1–47, 2016. URL `http://jmlr.org/papers/v17/15-243.html`.

M. R. Gupta, D. Bahri, A. Cotter, and K. Canini. Diminishing returns shape constraints for interpretability and regularization. *NeurIPS*, 2018.

M. R. Gupta, E. Louidor, O. Mangylov, N. Morioka, T. Narayan, and S. Zhao. Multidimensional shape constraints. *ICML*, 2020.

F. E. Harrell and C. E. Davis. A new distribution-free quantile estimator. *Biometrika*, 1982.

X. He. Quantile curves without crossing. *The American Statistician*, 51(2):186–192, 1997.

W. D. Kaigh and P. A. Lachenbruch. A generalized quantile estimator. *Communications in Statistics - Theory and Methods*, 1982.

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

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

R. Koenker and G. Bassett. Regression quantiles. In *Econometrica*, 1978.

V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. In *ICML*, 2018.

B. Lang. Monotonic multi-layer perceptron networks as universal approximators. *Artificial neural networks: formal models and their applications-ICANN*, 2005.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research (JMLR)*, 2006.

A. Minin, M. Velikova, B. Lang, and H. Daniels. Comparison of universal approximators incorporating partial monotonicity by structure. *Neural Networks*, 23(4):471–475, 2010.

R.-D. Reiss. Estimation of quantiles in certain nonparametric models. *Annals of Statistics*, 8, 1980.

J. Sill. Monotonic networks. *NeurIPS*, 1998.

N. Tagasovska and D. Lopez-Paz. Single-model uncertainties for deep learning. *NeurIPS*, 2019.

I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola. Nonparametric quantile estimation. *Journal of Machine Learning Research (JMLR)*, 2006.

TensorFlow Blog Post. TensorFlow Lattice: flexible, controlled, and interpretable ML, 2020. URL `https://blog.tensorflow.org/2020/02/tensorflow-lattice-flexible-\controlled-and-interpretable-ML.html`.

L. Torossian, V. Picheny, R. Faivre, and A. Garivier. A review on quantile regression for stochastic computer experiments. *Reliable Engineering & System Safety*, 2020.

A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

S. Wang and M. R. Gupta. Deontological ethics by monotonicity shape constraints. In *AIStats*, 2020.

X. Yan, W. Zhang, L. Ma, W. Liu, and Q. Wu. Parsimonious quantile regression of financial asset tail dynamics via sequential learning. *NeurIPS*, 2018.

S. You, K. Canini, D. Ding, J. Pfeifer, and M. R. Gupta. Deep lattice networks and partial monotonic functions. *NeurIPS*, 2017.

S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S.X. Chen. Cautionary tales on air-quality improvement in Beijing. *Proceedings of the Royal Society A*, 473, 2017.

A Appendix

A.1 Proof of Lemma

Proof: If a random variable Y conditioned on X belongs to the location-scale family then for $\tau \in (0,1)$ and $a \in \mathbb{R}$ and $b > 0$, it must hold that the conditional inverse CDF satisfies $F^{-1}_{Y|X=z}(\tau) = a + bF^{-1}_{Y|X=x}(\tau)$. Note that for $\tau \in (0,1)$, interpolating a lattice with two lattice vertices in the τ dimension yields the estimate $\hat{F}^{-1}_{Y|X=z}(\tau) = f(z,\tau) = f(z,0) + c(\tau)(f(z,1) - f(z,0))$. Thus mapping to the location-scale property, $a = f(z,0)$, $b = f(z,1) - f(z,0)$, and $\hat{F}^{-1}_{Y|X=x}(\tau) = c(\tau)$. Thus every estimated conditional inverse CDF $\hat{F}^{-1}_{Y|X=z}(\tau)$ is a translation and scaling of the piecewise linear function $c(\tau)$.

A.2 Proof of Theorem 1

Theorem 1. *Suppose $f(\tau;\theta)$ is well specified: there exists $\theta^* \in \mathbb{R}^m$ such that $f(\tau;\theta^*) = F^{-1}(\tau)$ for all $\tau \in (0,1)$. Suppose further that the function $\theta \mapsto f(\tau;\theta)$ is continuous, locally Lipschitz, and twice differentiable at $\theta = \theta^*$ for all $\tau \in (0,1)$. Then,*

$$\sqrt{n}(f(\tau;\hat{\theta}^{(n)}) - f(\tau;\theta^*)) \xrightarrow{d} \mathcal{N}\left(0, \nabla_\theta f(\tau;\theta^*)^\top Q^{-1} V (Q^{-1})^\top \nabla_\theta f(\tau;\theta^*)\right)$$

where,

$$Q = E_{\mathcal{T}}[p(f(\mathcal{T};\theta^*))\Gamma(\mathcal{T})], V = E_{\mathcal{T}}[\mathcal{T}(1-\mathcal{T})\Gamma(\mathcal{T})],$$

with $\Gamma(\tau) = \nabla_\theta f(\tau;\theta^*)\nabla_\theta f(\tau;\theta^*)^\top$.

Proof. Consider the loss function $\bar{L}(y, \theta) = E_{\mathcal{T}}[L_{\mathcal{T}}(y, f(\mathcal{T}; \theta))]$. Suppose that the risk $R(\theta) = E_Y[\bar{L}(Y, \theta)]$ has a unique minimizer $\theta^* = \arg\min_\theta R(\theta)$. Let $\hat{\theta}^{(n)}$ minimize the risk with an empirical distribution over Y with IID samples $\{y_i\}_{i=1}^n$, $\hat{\theta}^{(n)} = \arg\min_\theta \frac{1}{n}\sum_{i=1}^n \bar{L}(y_i, \theta)$. Suppose further that Y is a continuous random variable with density $p(y)$, and $\mathcal{T}$ is independent of Y.

We apply Theorem 5.23 from van der Vaart (1998)[1], which says that under some regularity conditions,

$$\sqrt{n}(\hat{\theta}^{(n)} - \theta^*) = -\nabla^2 R(\theta^*)^{-1} \cdot \frac{1}{\sqrt{n}} \sum_{i=1}^n \nabla_\theta \bar{L}(y_i, \theta^*) + o_P(1),$$

where $o_P(1)$ refers to a sequence of random variables that converges in probability to 0: $X_n = o_P(1)$ if and only if $X_n \xrightarrow{P} 0$. Specifically, this theorem requires the following conditions:

(i) $\hat{\theta}^{(n)} \xrightarrow{P} \theta^*$: true under the central limit theorem.

(ii) $\frac{1}{n}\sum_{i=1}^n \bar{L}(y_i, \hat{\theta}^{(n)}) \le \inf_\theta \frac{1}{n}\sum_{i=1}^n \bar{L}(y_i, \theta) + o_P(1/n)$: true by optimality of $\hat{\theta}^{(n)}$.

(iii) The loss $\bar{L}(y, \theta)$ is locally Lipschitz near $\theta = \theta^*$. This holds as long as $f(\tau; \theta)$ is locally Lipschitz near $\theta = \theta^*$.

(iv) For P-almost all y, the function $\theta \mapsto \bar{L}(y, \theta)$ is differentiable at $\theta = \theta^*$.

(v) $R(\theta)$ is twice differentiable at θ^* with positive definite Hessian $\nabla^2 R(\theta^*) \succ 0$.

To verify condition (iv), note that

$$\nabla_\theta \bar{L}(y, \theta^*) = \nabla_\theta E_{\mathcal{T}}[L_{\mathcal{T}}(y, f(\mathcal{T}; \theta^*)] = E_{\mathcal{T}}[\nabla_\theta L_{\mathcal{T}}(y, f(\mathcal{T}; \theta^*))].$$

Since Y is a continuous random variable with a density near $f(\tau; \theta^*)$, $\nabla_\theta L_\tau(y, f(\tau, \theta^*))$ exists for P-almost every y as long as $f(\tau, \theta)$ is also differentiable at $\theta = \theta^*$. This is because for continuous Y, the τ-quantile is unique, and the derivative $\nabla_\gamma L_\tau(y, \gamma)$ exists with probability 1. The full gradient is:

$$\nabla_\theta L_\tau(y, f(\tau; \theta^*)) = ((1-\tau)\mathbb{1}(f(\tau; \theta^*) \ge y) - \tau \mathbb{1}(f(\tau; \theta^*) \le y))\, \nabla_\theta f(\tau; \theta^*).$$

Therefore, $\nabla_\theta \bar{L}(y, \theta^*) = E_{\mathcal{T}}[\nabla_\theta L_{\mathcal{T}}(y, f(\mathcal{T}; \theta^*))]$ is also well defined and exists for P-almost every y.

To verify condition (v), we explicitly compute the Hessian:

$$\begin{aligned}
\nabla R(\theta^*) &= E_Y[\nabla_\theta \bar{L}(Y, \theta^*)] \\
&= E_Y[E_{\mathcal{T}}[\nabla_\theta L_{\mathcal{T}}(Y, f(\mathcal{T}; \theta^*))]] \\
&= E_{\mathcal{T}}[E_Y[\nabla_\theta L_{\mathcal{T}}(Y, f(\mathcal{T}; \theta^*))]] \quad \text{since } \mathcal{T} \perp\!\!\!\perp Y \\
&= E_{\mathcal{T}}[((1-\mathcal{T})P(f(\mathcal{T}; \theta^*) \ge Y) - \mathcal{T}P(f(\mathcal{T}; \theta^*) \le Y))\, \nabla_\theta f(\mathcal{T}; \theta^*)] \\
&= E_{\mathcal{T}}[(P(Y \le f(\mathcal{T}; \theta^*)) - \mathcal{T})\nabla_\theta f(\mathcal{T}; \theta^*)] \\
&= E_{\mathcal{T}}[(F(f(\mathcal{T}; \theta^*)) - \mathcal{T})\nabla_\theta f(\mathcal{T}; \theta^*)].
\end{aligned}$$

$$\begin{aligned}
\nabla^2 R(\theta^*) &= E_{\mathcal{T}}[\nabla_\theta \{(F(f(\mathcal{T}; \theta^*)) - \mathcal{T})\nabla_\theta f(\mathcal{T}; \theta^*)\}] \\
&= E_{\mathcal{T}}[\nabla_\theta f(\mathcal{T}; \theta^*) F'(f(\mathcal{T}; \theta^*)) \nabla_\theta f(\mathcal{T}; \theta^*)^\top + (F(f(\mathcal{T}; \theta^*)) - \mathcal{T})\nabla_\theta^2 f(\mathcal{T}; \theta^*)] \\
&= E_{\mathcal{T}}[p(f(\mathcal{T}; \theta^*)) \nabla_\theta f(\mathcal{T}; \theta^*) \nabla_\theta f(\mathcal{T}; \theta^*)^\top] \\
&= Q \succ 0.
\end{aligned}$$

The third equality in the computation of the Hessian above follows by assumption that f is well specified, since $f(\tau; \theta^*) = F^{-1}(\tau)$, so $F(f(\tau; \theta^*)) - \tau = 0$.

[1] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

With all conditions verified, applying Theorem 5.23 from van der Vaart (1998) we have,

$$\sqrt{n}(\hat{\theta}^{(n)} - \theta^*) = -Q^{-1} \cdot \frac{1}{\sqrt{n}} \sum_{i=1}^{n} E_{\mathcal{T}}[\nabla_\theta L_{\mathcal{T}}(y_i, f(\mathcal{T}; \theta^*))] + o_P(1). \tag{5}$$

Focusing on the sum term,

$$\begin{aligned}
&\frac{1}{\sqrt{n}} \sum_{i=1}^{n} E_{\mathcal{T}}[\nabla_\theta L_{\mathcal{T}}(y_i, f(\mathcal{T}; \theta^*))] \\
&= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} E_{\mathcal{T}}\left[((1-\mathcal{T})\mathbb{1}(f(\mathcal{T}; \theta^*) \geq y_i) - \mathcal{T}\mathbb{1}(f(\mathcal{T}; \theta^*) \leq y_i))\, \nabla_\theta f(\mathcal{T}; \theta^*)\right] \\
&= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} E_{\mathcal{T}}\left[(\mathbb{1}(y_i \leq f(\mathcal{T}; \theta^*)) - \mathcal{T})\nabla_\theta f(\mathcal{T}; \theta^*)\right]
\end{aligned}$$

Since $\mathbb{1}(y_i \leq f(\mathcal{T}; \theta^*))$ are IID samples from Bernoulli($\mathcal{T}$) and $\mathcal{T}$ is independent of Y, the random variable $E_{\mathcal{T}}\left[(\mathbb{1}(Y \leq f(\mathcal{T}; \theta^*)) - \mathcal{T})\nabla_\theta f(\mathcal{T}; \theta^*)\right]$ has mean $\mathbf{0}$ and variance

$$V = E_{\mathcal{T}}[\mathcal{T}(1-\mathcal{T})\nabla_\theta f(\mathcal{T}; \theta^*)\nabla_\theta f(\mathcal{T}; \theta^*)^\top].$$

By the central limit theorem, the sum converges in distribution to $\mathcal{N}(\mathbf{0}, V)$. Combining this with Equation (5) yields,

$$\sqrt{n}(\hat{\theta}^{(n)} - \theta^*) \xrightarrow{d} \mathcal{N}(\mathbf{0}, Q^{-1}V(Q^{-1})^\top).$$

Applying the delta method with $f(\tau; \theta)$ as a function of θ for a fixed desired quantile τ, we have the final result,

$$\sqrt{n}(f(\tau; \hat{\theta}^{(n)}) - f(\tau; \theta^*)) \xrightarrow{d} \mathcal{N}\left(\mathbf{0}, \nabla_\theta f(\tau; \theta^*)^\top Q^{-1} V (Q^{-1})^\top \nabla_\theta f(\tau; \theta^*)\right).$$

□

A.3 Additional Examples Illustrating Theorem 1

Single parameter uniform anchored at 0. Suppose the true distribution of interest is $Y \sim \text{Unif}(0, \alpha)$. Let $f(\tau; \theta) = \theta\tau$, which correctly parameterizes the true linear inverse CDF. By Theorem 1, the asymptotic variance of $\sqrt{n}(\hat{\theta}^{(n)} - \theta^*)$ is

$$Q^{-1}V(Q^{-1})^\top = \alpha^2(E_{\mathcal{T}}[\mathcal{T}^3] - E_{\mathcal{T}}[\mathcal{T}^4])/E_{\mathcal{T}}[\mathcal{T}^2]^2.$$

Thus, the asymptotic variance depends on the distribution of $\mathcal{T}$ through its third and fourth moments, and depends on the distribution of Y through α. For $\mathcal{T} \sim \text{Unif}(0, 1)$, the exact asymptotic variance is $0.45\alpha^2$. To estimate a specific τ_0-quantile, we would evaluate the function $f(\tau_0; \hat{\theta}^{(n)}) = \hat{\theta}^{(n)}\tau_0$, which would have an asymptotic variance of $0.45\alpha^2\tau_0^2$.

A.4 A Connection To Regression On Sample Quantiles

We provide some more intuition here about how the training procedure we describe in this paper - learning a function $f(\tau)$ with an expected pinball loss - smooths sample quantiles in the case that there are no conditioning x-features. In particular, we can show that our proposed method (except with a uniform discrete rather than uniform continuous distribution for τ) is equivalent to a regression on the observed sample quantiles with a particular cost function.

Proposition 1. *Given a dataset $\{y_i\}_{i=1}^n$ of n elements, construct the dataset $\{\tau(i), y^{(i)}\}_{i=1}^n$ where $y^{(i)}$ represents the ith order statistic of the data and $\tau(i) = \frac{i-1}{n-1}$ is the sample quantile that order statistic represents. Then training a model with an expected pinball loss (1) where τ is drawn uniformly from the discrete distribution $\{\tau(i)\}_{i=1}^n$ is equivalent to learning a single-variable regression over the data points $\{\tau(i), y^{(i)}\}_{i=1}^n$.*

The cost function c for the regression at the specific point $(\tau(i), y^{(i)})$ *is* 0 *when* $f(\tau(i)) = y^{(i)}$ *and otherwise piecewise-linear with slope*

$$c'(f(\tau(i))) = \begin{cases} -\tau(i) - \sum_{j=1}^{i-1} \mathbb{1}(f(\tau(i)) < y^{(j)}) & \text{for } f(\tau_i) < y^{(i)} \\ (1-\tau(i)) + \sum_{j=i+1}^{n} \mathbb{1}(f(\tau(i)) > y^{(j)}) & \text{for } f(\tau_i) > y^{(i)} \end{cases}$$

Proof. The τ-pinball loss can be expressed as

$$L_\tau(y, f(\tau)) = \begin{cases} \tau(y - f(\tau)) & \text{if } f(\tau) < y \\ (1-\tau)(f(\tau) - y) & \text{if } f(\tau) \geq y \end{cases}$$

When training with an expected pinball loss over the discrete values of τ corresponding to the sample quantiles of the data, this means that the expected pinball loss is a function only of $f(\tau(i))$, or the quantile predictions of f at those sample quantiles. This is precisely the context of ordinary regression; the loss we face is a function only of the values our function takes at the x-coordinates of our data points, which here are set up to be exactly the $\{\tau(i)\}_{i=1}^n$.

Let us consider the pinball loss applied to $\tau(i)$. To optimize (1), we average the losses across our data points $\{y_i\}_{i=1}^n$; for this proof, we (equivalently) work with the sums instead. Let our regression-equivalent cost $c(f(\tau(i)))$ be equal to the relative increase in the $\tau(i)$ pinball loss we face, summed over the data points, compared to if we had predicted $f(\tau(i)) = y^{(i)}$.

By the definition of the $\tau(i)$ pinball loss, decreasing $f(\tau(i))$ by ϵ will mean that we face an additional cost of $\tau(i)\epsilon$ for every data point we are below and moving away from and recover $(1-\tau(i))\epsilon$ for every data point we are above and moving towards. Say we start at $f(\tau(i)) = y^{(i)}$ and choose an ϵ such that $f(\tau(i)) - \epsilon > y^{(i-1)}$. The point $y^{(i)}$ has $\tau(i)(n-1)$ data points below it and $(1-\tau(i))(n-1)$ data points above it by definition. So our marginal loss from decreasing $f(\tau(i))$ by ϵ is $\tau(i)(1-\tau(i))(n-1)\epsilon$ due to the points above $y^{(i)}$ while we recover $(1-\tau(i))\tau(i)(n-1)\epsilon$ due to the points below it. These are equal! But we also lose $\tau(i)\epsilon$ because we are now moving away from the point $y^{(i)}$ itself, making the slope of $c(f(\tau(i)))$ equal to $-\tau(i)$ in the region between $y^{(i-1)}$ and $y^{(i)}$. Each time we pass a neighboring data point, we start losing $\tau(i)\epsilon$ from that point instead of gaining $(1-\tau(i))\epsilon$, increasing the steepness of the slope by 1. This is exactly what we want to show: $c'(f(\tau(i)))$ is equal to $-\tau(i)$ minus the number of data points smaller than $y^{(i)}$ that $f(\tau(i))$ is smaller than.

The opposite case - analyzing what happens when we increase $f(\tau(i))$ by ϵ is essentially analagous. Increasing $f(\tau(i))$ by ϵ leads to an additional cost of $(1-\tau(i))\epsilon$ for every data point we are above and moving away from and a decreased cost of $\tau(i)\epsilon$ for every data point we are below and moving towards. This implies the slope of the cost function between $y^{(i)}$ and $y^{(i+1)}$ is $(1-\tau(i))$, as the cost accrued and recovered by the points above and below $y^{(i)}$ cancel out and we only have the net cost due to moving away from $y^{(i)}$ itself. And for the same reason described above, passing data points adds to the slope by 1. □

See Figure 3 for some example cost functions.

This cost function is nonstandard in a couple ways. First, it varies per point. And second, its steepness is a function of how many neighboring sample data points the prediction $f(\tau)$ is off by rather than just how far away it is from the sample quantile in an absolute sense.

Still, it otherwise follows the normal properties of a cost function, such as taking the value 0 when the prediction is exactly correct and (weakly) rising in a convex fashion as the prediction is too high or too low. This makes it clear why a sufficiently flexible function $f(\tau)$ will exactly pass through all of the sample quantiles and also why a more limited $f(\tau)$ will serve as a smoother, passing below some sample quantiles and above others.

Worked Example: Consider the dataset {1, 2, 5, 6, 8}. These correspond to the sample quantiles {0, 0.25, 0.5, 0.75, 1}. Figure 3 plots the sample quantiles on a graph, the least squares regression fit, and the expected pinball loss fit with linear $f(\tau)$. We can see that both are quite similar. For example, both estimate the median as being lower than the observed sample median. The figure also shows the regression-equivalent

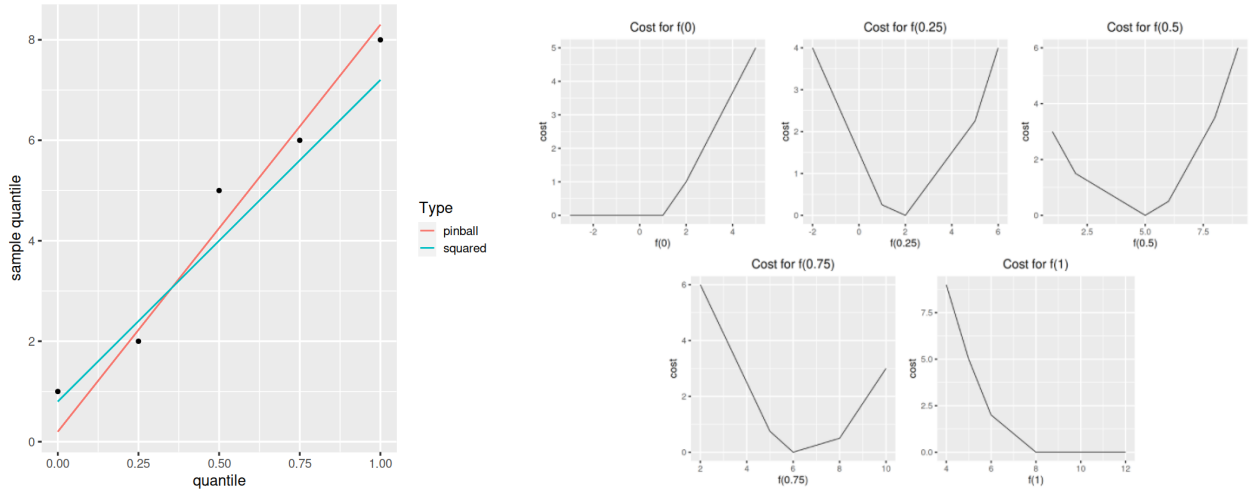
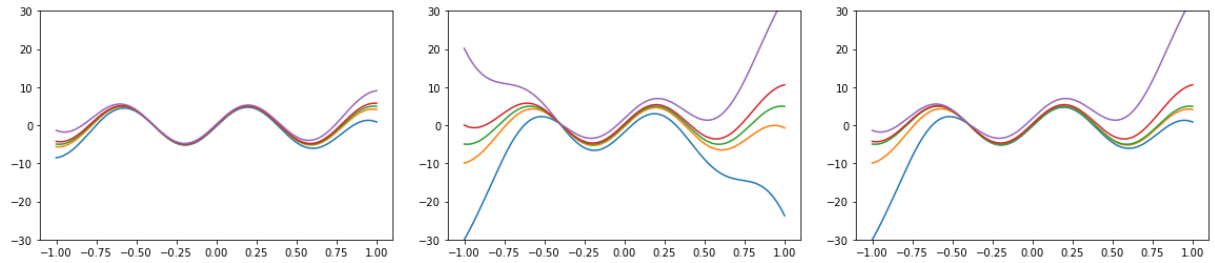


Figure 3: Given the dataset {1, 2, 5, 6, 8}, we show visually how learning $f(\tau)$ via the expected pinball loss with τ as a feature can be seen as a type of regression on the sample quantiles. For simplicity, we take $f(\tau)$ to be linear, which would be the correct functional form if the data are drawn from a uniform distribution. On the left, we plot the sample quantiles and compare the lines learned by a direct squared-loss regression on those data points (i.e. $\{\tau(i), y^{(i)}\}_{i=1}^{5}$) with the line learned by our method (i.e. learning $f(\tau)$ with the expected pinball loss). On the right, we show the equivalent costs imposed on predictions for each sample quantile if we reframed training $f(\tau)$ with an expected pinball loss over $\tau = \{0, 0.25, 0.5, 0.75, 1\}$ as a regression on sample quantiles. The cost is 0 when the prediction is exactly the corresponding sample quantile and weakly rising as it varies.

cost function for each of the quantile predictions, if we were to think of our method as a regression on sample quantiles.

A.5 Sine-skew Simulated Conditional Distribution

Fig. 4 illustrates the true quantiles of the sine-skew distribution with the different parameters used in our simulations.



(a) Sine-skew (1,1) (b) Sine-skew (7,7) (c) Sine-skew (1,7)

Figure 4: True quantiles of the sine-skew distribution with different noise parameters Torossian et al. (2020). The colored lines show the quantiles for $\tau = 0.1, 0.4, 0.5, 0.6, 0.9$.

A.6 Calibration Error Metric Definition

A predicted conditional inverse CDF $f(x, \tau; \theta)$ is *calibrated* if

$$P(Y \leq f(X, \tau; \theta)) = \tau.$$

A calibration error metric measuring the difference between $P(Y \leq f(X, \tau; \theta))$ and τ can be approximated over a finite dataset with data points $\{x_i, y_i\}_{i=1}^n$ and quantiles $\tau_1, ..., \tau_m$. We consider the absolute deviation:

$$\frac{1}{m} \sum_{j=1}^{m} \left| \left(\frac{1}{n} \sum_{i=1^n} \mathbb{1}(y_i \leq f(x_i, \tau_j; \theta)) \right) - \tau_j \right| \tag{6}$$

Others such as Kuleshov et al. (2018) have considered the squared deviation.

A.7 Hyperparameter Tuning Details

For each real data experiment, we tune hyperparameters over a validation dataset, where the validation dataset is selected according to Section 5.2. The tuning criterion was the validation pinball loss averaged over $\tau \in \{0.01, ..., 0.99\}$. For DLNs, the search spaces for real data experiments were the following:

- The number of calibration keypoints for the piecewise-linear calibration function over τ were tuned between $\{10, 20, 50, 100\}$. Note that as this number goes up, you get more model flexibility with respect to τ (only), which makes it effectively like just training separate models for each τ.
- The number of lattice keypoints for τ was tuned between $\{2, 3, 5, 7, 10\}$. That controls the flexibility of the interactions between τ and the other features x.
- Other feature calibration keypoints were tuned between $\{5, 10, 15, 20\}$, which are common values for DLNs.
- Step sizes were tuned between $\{0.001, 0.005, 0.01, 0.05, 0.1\}$
- minibatch sizes were tuned between $\{1000, 10000\}$.
- Number of steps was tuned between $\{100, 1000, 10000\}$.

To tune the DLN architecture for experiments with real data, we searched over the following hyperparameters:

- Air Quality: number of lattices $\in \{4, 8\}$,number of features per lattice $= 6$,
- Puzzles: number of lattices $\in \{8, 16, 32\}$,number of features per lattice $= 5$
- Traffic: single lattice which includes all features (not ensemble)
- Wine: number of lattices $\in \{100, 200, 400, 800\}$, number of features per lattice $\in \{2, 4, 8\}$.