Controlled Direct Effect Priors for Bayesian Neural Networks

Jianzhun Du*
Harvard University
Cambridge, MA, USA
jzdu@g.harvard.edu

Andrew Slavin Ross*
Harvard University
Cambridge, MA, USA
andrew_ross@g.harvard.edu

Yonadav Shavit*
Harvard University
Cambridge, MA, USA
yonadav@g.harvard.edu

Finale Doshi-Velez
Harvard University
Cambridge, MA, USA
finale@seas.harvard.edu

Abstract

When training Bayesian neural networks (BNNs), practitioners may wish to place priors that encode external knowledge not necessarily present in the training data. We propose a method for imposing priors on the changes in network outputs after performing user-defined interventions on inputs, which we believe is flexible enough to encode many forms of domain knowledge. We connect our approach to the literature on causality, monotonicity, and invariance, then interrogate its behavior on a variety of toy and real-world datasets.

1 Introduction

In this work, we propose a method for incorporating external knowledge into the priors of Bayesian neural networks. Specifically, we focus on a particular class of knowledge: that our network f is either *monotonic* or *invariant* under a user-defined perturbation, which we formalize as a "guiding function" g. As a concrete example, imagine g is a function which returns the brightness of an image. A day vs. night model f_{day} should generally increase monotonically with brightness, while a cat vs. dog model f_{cat} should probably be invariant to changes in brightness. The novelty of our approach over existing methods [1, 2, 3] is that we are not restricted to imposing monotonicity or invariance with respect to individual input features (which in the invariance case would not be particularly interesting), but instead any nonlinear function of our inputs, which allows users to encode more complex forms of prior knowledge.

2 Approach

Our method is inspired by the notion of a "controlled direct effect" (CDE). [4] defines CDEs as measurements of "the sensitivity [of an outcome variable] to changes in [a set of variables] while all other factors in the analysis are held fixed." If a CDE is positive, this indicates the presence of a causal relationship between the variable change (often called a "transition") and the outcome; if it is 0, it indicates the absence of one. Although CDEs are difficult to estimate from real-world data due to the unobservability of counterfactuals, they are easy to compute if the outcome is simply the output of a neural network.

In this paper, we focus on placing priors over the CDEs of continuous transformations on inputs $x \in \mathbb{R}^D$ on binary classification or univariate regression models $\hat{y} = f(x) : \mathbb{R}^D \to \mathbb{R}$, meant to

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

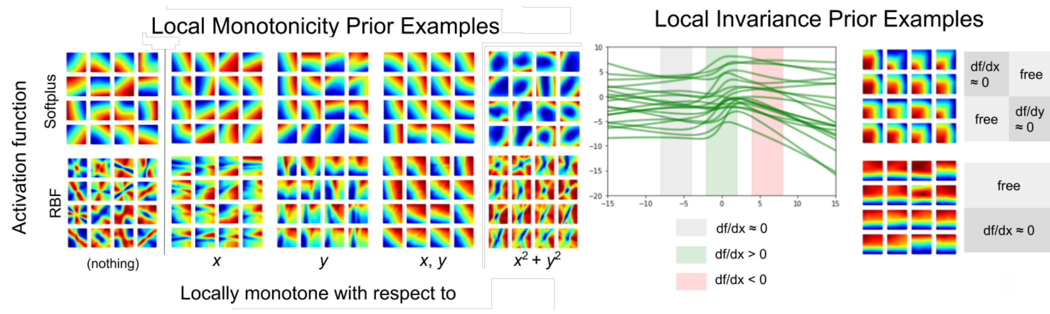

Figure 1: Left: 2D local monotonicity prior samples from 50-unit hidden layer BNNs with (RBF / softplus) activations and local monotonicity priors with respect to x, y, both x and y, and radial distance (a nonlinear function of the input features). Right: samples from 1D and 2D local invariance priors imposed over limited regions (indicated by the legends beside each set of plots).

estimate a set of targets y. In particular, assume we have a "guiding function" $g(x)$, which measures an aspect of interest without directly measuring any of the other confounding factors (even if its value over the dataset is correlated, e.g. if all the cats used to train f_{cat} were photographed in the dark). We recommend thinking of guiding functions as abstract features, which may simply be input dimensions but can be higher-level (e.g. expert-defined metrics, embedding components, or outputs of models that predict quantities other than $\hat{y}$).

Using this concept, we can define a transition

$$\mathtt{push}_\epsilon(x, g) \triangleq \underset{x' \in \Omega \cap B_\epsilon(x)}{\operatorname{argmax}} \; g(x'), \tag{1}$$

which moves x to a new point x' that maximizes $g(\cdot)$ subject to the constraint that x' is near x (within B_ϵ, an ϵ-ball under some distance measure) and that x' remains within the permitted input space Ω (possibly $\mathbb{R}^D$, or possibly a manifold embedded within it). The CDE of this transition is

$$\text{CDE}_\epsilon(f, g, x) = f(\mathtt{push}_\epsilon(x, g)) - f(x). \tag{2}$$

Using the CDE, we extend prior notions of monotonicity and invariance [5, 2, 6] (discussed further in Section A) to apply to guiding functions:

Definition 2.1 *f is* locally monotone *wrt. g if $\exists\, \epsilon > 0$ such that, for all $x \in \Omega$,* $\boxed{\text{CDE}_\epsilon(f, g, x) \geq 0.}$

Definition 2.2 *f is* locally invariant *wrt. g if $\exists\, \epsilon > 0$ such that, for all $x \in \Omega$,* $\boxed{\text{CDE}_\epsilon(f, g, x) = 0.}$

Approximation using gradients. Computing Equation 2 is difficult because it requires choosing ϵ and performing an inner optimization over g for each value of x. In Section B, we show that as $\epsilon \to 0$ (under certain smoothness assumptions on f and g, with locally unconstrained Ω),

$$\text{CDE}_\epsilon(f, g, x) \approx \epsilon \nabla f(x)^\intercal \nabla g(x) \propto \cos(\nabla f(x), \nabla g(x)) \tag{3}$$

Using this approximation, we can formulate ϵ-independent causal direct effect "error" functions

$$\begin{aligned} \mathtt{InvErr}(f, g, x) &\equiv \cos^2(\nabla f(x), \nabla g(x)), \\ \mathtt{MonoErr}(f, g, x) &\equiv \lfloor \cos(\nabla f(x), \nabla g(x)) \rfloor_-^2 \end{aligned} \tag{4}$$

which we incorporate into a prior on Bayesian neural networks f_W using

$$P(f_W) \propto \mathcal{N}(W|0, \sigma^2) \prod_m \exp\left\{-\lambda_m \mathbb{E}_{p_m(x)}\left[\mathtt{Err}_m(f_W, g_m, x)\right]\right\}, \tag{5}$$

where we impose each local monotonicity/invariance prior (indexed by m) over a separate input distribution p_m, which is often the training distribution but can be altered to include unlabeled data or to exclude areas where priors should not hold (allowing users to express knowledge only pertaining to certain regions). The log prior probability, which (up to a constant) is $\log \mathcal{N}(W|0, \sigma^2) - \sum_m \lambda_m \mathbb{E}_{p_m(x)}\left[\mathtt{Err}_m(f_W, g_m, x)\right]$, is relatively simple to compute, making it convenient for log loss optimization or variational inference with minibatches.

Samples from 1D and 2D examples of these kinds of priors are shown in Figure 1 (with additional toy regression and classification examples in Figure 7).

3 Experiments

We tested our method in several settings: a synthetic 3D regression example with nonlinear local invariance (Figures 4 and 5), the UCI Adult and Concrete datasets [7] with local monotonicity to individual features and nonlinear functions of them (Figure 6 and Table 1), and COMPAS [8] with local invariance to a race prediction model. For brevity, we focus on COMPAS but refer readers to the appendix for other results and implementation details.

COMPAS is a dataset for rearrest prediction of defendants from Broward County, Florida. The dataset has been shown to contain significant racial disparities, which tend to result in different predictive performances for white and black defendants [9]. Since discrimination in policing has been shown to lead to higher arrest rates for blacks disproportionate to rates of offense [10], practitioners may wish to counteract this bias with an invariance prior.

When trying to encode race with a guiding function $g(x)$, simply letting $g(x)$ be an input dimension will not work as we may simply pick up correlates in the data. Instead, we first train a logistic classifier $h(x)$ to predict race given x, and then set $g(x) = h(x)$. In setting a prior that $f(x)$ is locally invariant wrt. $g(x)$, we encourage a notion of individual fairness ("treat similar individuals similarly," [11]) with a kernel that considers individuals similar if they differ only by race (as imperfectly quantified by $g(x)$). This approach differs conceptually from others that try to enforce statistical parity.

In Figure 2, we illustrate the effect of increasing the invariance prior strength on the performance statistics of the MAP predictor. Overall, we find that local invariance with respect to race prediction models can reduce differences in model false-positive and false-negative rates for white and black defendants at an initially modest cost to accuracy. In Figure 3, we also show that these locally invariant models have qualitatively different feature importance distributions.

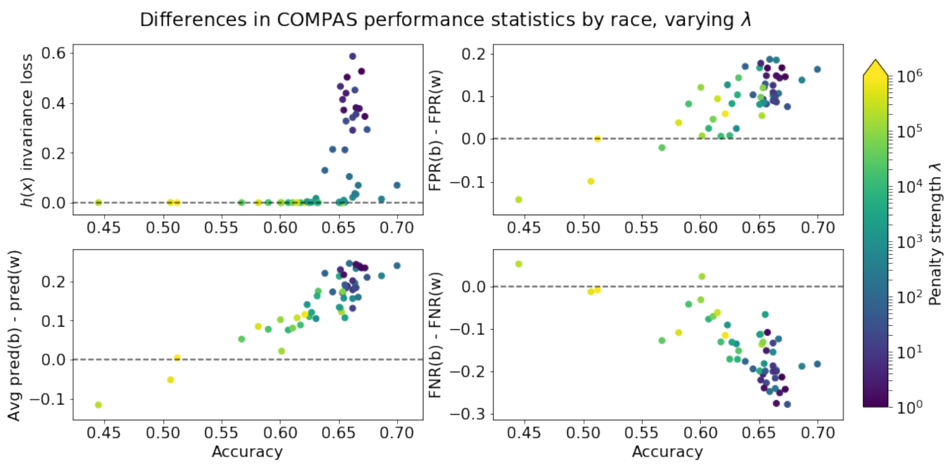

Figure 2: Prediction statistics vs. accuracy for a MAP predictor trained on COMPAS to predict recidivism, varying the invariance prior strength λ. **Top left**: mean invariance loss (see Eq. 4) decreases to 0 with λ at low initial cost of accuracy. **Bottom left**: the difference in mean prediction between black and white defendants (initially positive) decreases with λ. **Top and bottom right**: differences in false positive and false negative rates (initially biased towards FPs for black defendants and FNs for white defendants) fall to 0 with λ, suggesting model errors become more symmetric.

4 Discussion and Conclusion

In this paper, we proposed a method for incorporating external knowledge into a BNN prior if the knowledge can be framed as "local" monotonicity or invariance with respect to some guiding function g. Overall, encoding expert causal knowledge into a prior can be useful when it helps disambiguate equally valid hypotheses supported by the data (as in the 3D synthetic example in Section D.1), or counteract dataset bias (as in the COMPAS example).

One interesting finding from our experiments is that results for invariance priors were generally stronger than those for monotonicity. In our real-world monotonicity experiments, our data already supported our prior (Section D.2), so although our approach was never harmful, it also never helped. We observed similar results in preliminary experiments on other datasets from the non-Bayesian monotonicity literature. We speculate this may occur because BNNs often underfit [12], while avoiding overfitting is usually the motivation for monotonicity priors on datasets that are already inherently monotone [13]. Another possible reason for differences in effect strengths is that local invariance is an equality constraint, whereas monotonicity is a much less prescriptive inequality.

Overall, we believe expressing causal knowledge in terms of sensitivity to local changes is a promising approach. More broadly, we feel that better tools for relating causal knowledge to model priors will be essential for understanding and improving their generalization beyond limited training data.

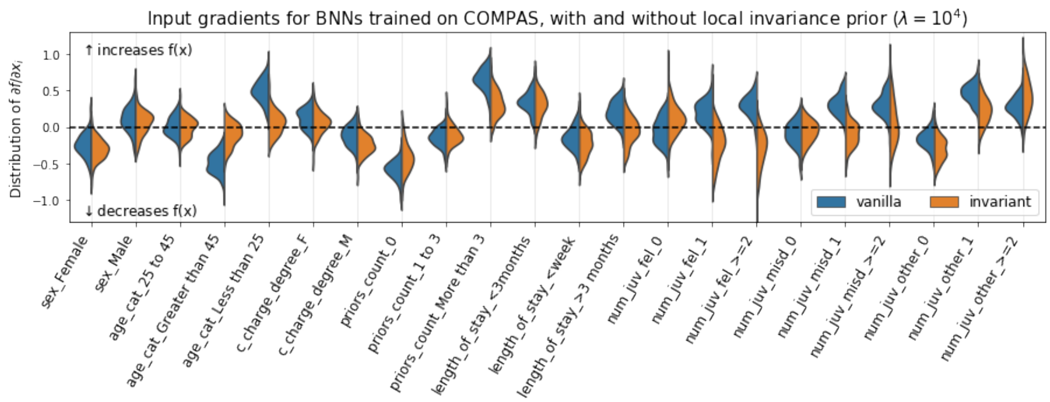

Figure 3: Distribution of MAP feature importances in COMPAS, with and without local invariance priors wrt. a separate race classifier. After incorporating the prior, some associations are dramatically different. For example, having no or few juvenile penalties (`num_juv_fel_{0,1}`) becomes exculpatory, while being older (`age_cat_25_to_45`) ceases to mitigate perceived risk, and having numerous non-felony/misdemeanor juvenile citations (`num_juv_other_>=2` is more heavily penalized.

References

[1] Joseph Sill and Yaser S Abu-Mostafa. Monotonicity hints. In *Advances in neural information processing systems*, pages 634–640, 1997.

[2] Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander Van Esbroeck. Monotonic calibrated interpolated look-up tables. *The Journal of Machine Learning Research*, 17(1):3790–3836, 2016.

[3] Marco Lorenzi and Maurizio Filippone. Constraining the dynamics of deep probabilistic models. *arXiv preprint arXiv:1802.05680*, 2018.

[4] Judea Pearl. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009. `https://projecteuclid.org/euclid.ssu/1255440554`.

[5] Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

[6] Eric Nalisnick and Padhraic Smyth. Learning priors for invariance. In *International Conference on Artificial Intelligence and Statistics*, pages 366–375, 2018.

[7] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[8] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. How we analyzed the COMPAS recidivism algorithm. *ProPublica*, 2016.

[9] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

[10] The Economist. Ticket to ride: Measuring racial bias in police forces. `https://www.economist.com/united-states/2017/06/22/measuring-racial-bias-in-police-forces`, 2017.

[11] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[12] Soumya Ghosh and Finale Doshi-Velez. Model selection in Bayesian neural networks via horseshoe priors. *arXiv preprint arXiv:1705.10388*, 2017.

[13] Mahdi Milani Fard, Kevin Canini, Andrew Cotter, Jan Pfeifer, and Maya Gupta. Fast and flexible monotonic functions with ensembles of lattices. In *Advances in Neural Information Processing Systems*, pages 2919–2927, 2016.

[14] Johan Kwisthout, Hans Bodlaender, and Gerard Tel. Local monotonicity in probabilistic networks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 548–559. Springer, 2007.

[15] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.

[16] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015.

[17] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[18] Alexandre Lacoste, Thomas Boquet, Negar Rostamzadeh, Boris Oreshki, Wonchang Chung, and David Krueger. Deep prior. *arXiv preprint arXiv:1712.05016*, 2017.

[19] Thomas Frerix, Daniel Cremers, and Matthias Nießner. Linear inequality constraints for neural network activations. *arXiv preprint arXiv:1902.01785*, 2019.

[20] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

[21] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Reverse-mode differentiation of native Python. In *ICML workshop on Automatic Machine Learning*, 2015.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[25] I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

[26] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

A Related Work

Methods of imposing hard or soft monotonicity constraints have been studied for neural networks [14, 5, 2, 1], though not always in a Bayesian context. [3], though applied to deep Gaussian processes, provide an excellent framework for expressing priors about both local monotonicity and invariance. However, all of these methods only consider monotonicity with respect to individual input dimensions.

Invariance to general (nonlinear, multidimensional) perturbations has recently been studied for Bayesian models [6]. However, most other attempts to incorporate negative knowledge either focus on techniques like enforcing global uncorrelatedness of $f(x)$ and $g(x)$ across predefined groups (e.g. demographic parity, [15]), or "censoring" group membership [16]. The disadvantage of these approaches over *local* invariance techniques is that they make it impossible to learn a perfectly accurate model if base rates differ between groups [9].

Other methods of incorporating domain knowledge into priors include distillation between neural networks and logical rules [17], function priors based on related tasks [18], and inequality constraints on network outputs [19]. Another approach is simply to learn residuals on top of expert-provided baselines. In this paper, we focus on the case where exact baselines are unavailable, but experts still have information about the effects of perturbations. Finally, our approach can be seen as a specialization of the recently introduced framework of attribution priors [20], though our formulation allows for the additional option that each CDE prior can be defined for a different input distribution.

B Gradient Approximation Details

Taylor expanding f and g around x, in the limit of $\epsilon \to 0$ and $\mathcal{B}_\epsilon(x) \cap \Omega \approx \mathcal{B}_\epsilon(x)$,

$$\begin{aligned}
\mathrm{CDE}_\epsilon(f, g, x) &= f(\texttt{push}_\epsilon(x, g)) - f(x) \\
&\approx f(x + \epsilon \nabla g(x)) - f(x) \\
&= \big(f(x) + \epsilon \nabla f(x)^\intercal \nabla g(x) + \mathcal{O}(\epsilon^2)\big) - f(x) \\
&\approx \epsilon \nabla f(x)^\intercal \nabla g(x) \\
&\propto \frac{\nabla f(x)^\intercal \nabla g(x)}{||\nabla f(x)||\,||\nabla g(x)||} = \cos(\nabla f(x), \nabla g(x)).
\end{aligned} \tag{6}$$

We opted to use the cosine approximation rather than the dot product because the dot product can be spuriously minimized by sending $\nabla f(x)$ to 0, which sometimes produced undesirable effects in preliminary experiments. However, using cosines does change the relative strengths of CDE-based penalties for different inputs (even if it does not change our objective's global minimum). Another option we did not explore would be to normalize ∇f but not ∇g, i.e. penalize $\nabla f^\intercal \nabla g / ||\nabla f||$. Note that in the special case of region-specific 1D invariance priors (e.g. the 1D example in Figure 1), we omit ∇f normalization since we actually want the gradient to approach 0.

C Training Details

For all experiments, we optimize our neural networks using Autograd [21], generally using a single 100-unit hidden layer with softplus activations (which often work better with gradient penalties, due to their smoothness). If finding a MAP solution, we use the Adam [22] optimizer with learning rate 10^{-2}. If running Hamiltonian Monte Carlo (HMC) [23] (e.g. for generating prior samples), we use L=10 leapfrog steps with step size ϵ tuned so that the acceptance rate converges to ≈0.7.

D Additional Experiments

D.1 Synthetic 3D Regression (Local Invariance)

In Figure 4 we present a 3D regression dataset with inputs on a subset $\mathcal{S}_{\text{train}}$ of $[-2, 2]^3$ and regression targets equal to $f(x, y, z) = x^3 + y^2$. However, on this subset, a different function $g(x, y, z) = y + z^3$ predicts the same targets to within 0.05. That is, $\mathcal{S}_{\text{train}} = \big\{(x, y, z) \in [-2, 2]^3 : |x^3 + y^2 - y - z^3| \leq 0.05\big\}$ In our test set, we evaluate over the entirety of $\mathcal{S}_{\text{test}} = [-2, 2]^3$.

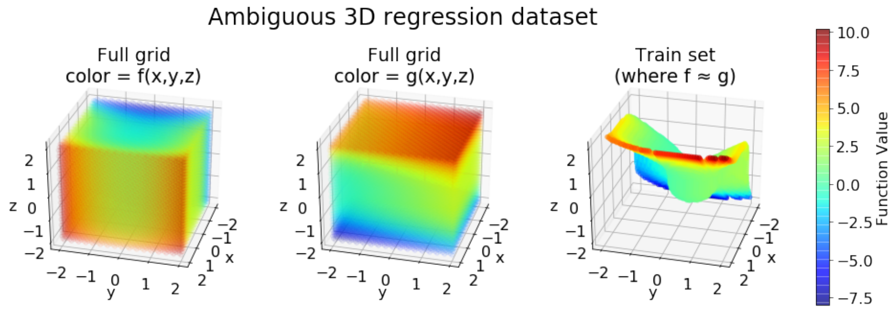

Figure 4: 3D ambiguous regression dataset. The true function generating the regression target is $f(x, y, z) = x^3 + y^2$ (left), but in the training region (right), a second function $g(x, y, z) = y + z^3$ (center) always returns a value within 0.05 of $f(x, y, z)$. The challenge is to train a model which generalizes outside the training region in accordance with f rather than g.

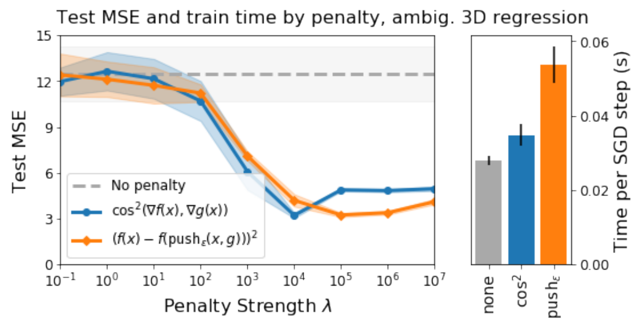

Figure 5: Comparison of MSE and train time on the 3D regression problem (over different local invariance formulations and λs). Shaded regions show standard deviations over 10 restarts. Both formulations reduce MSE, but the $\texttt{push}_\epsilon$ version is slower.

For this experiment, we test two formulations of our local invariance prior, one based on gradients (Equation 4) and one based on projected gradient descent (similar to [24]), which we expect to be a more exact approximation of $\text{CDE}_\epsilon(f, g, x)$. In that formulation, we run 25 steps of projected gradient descent within each outer optimization step (using Euclidean distance for B_ϵ with $\epsilon = 0.35$ and a step size of 0.2). As a baseline, we test a normal BNN trained with just the $\sigma^2 = 100$ Gaussian weight prior (which all methods share). Results by method are presented in Figure 5; the takeaway is that training with a local invariance penalty allows much better extrapolation to the full grid, and that cosine approximations perform on par with PGD.

D.2 UCI Adult and Concrete Datasets

We also evaluated how well our method performs in the monotonicity case on the ADULT and Concrete Compressive Strength datasets from the UCI repository [7]. The ADULT dataset is commonly used to test monotonic neural networks; we used the same setup as [13] (which gives us five guiding functions g that each return individual columns of x, with the column corresponding to female gender negated). Results are in Table 1.

Method	Train Accuracy	Test Accuracy
Logistic Regression	84.66 ± 0.00	84.65 ± 0.00
BNN, Gaussian Prior	85.81 ± 0.05	84.96 ± 0.06
BNN, Monotonicity per Eq. 4	85.84 ± 0.07	84.98 ± 0.07
BNN, Monotonicity per [1]	85.86 ± 0.05	85.05 ± 0.10

Table 1: On the ADULT dataset, monotonicity priors did not significantly change generalization performance, perhaps because baseline BNNs already learn to be locally monotone (Figure 6).

The Concrete dataset is less commonly considered in the monotonicity literature, but [25] provide a nonlinear expert baseline model proportional to $(x_{\text{water}}/(x_{\text{cement}} + x_{\text{slag}} + x_{\text{ash}}))^{-1.3} \cdot (\ln(x_{\text{age}}) + \text{const})$, which we use as our g. We hypothesized this dataset would be an interesting test case for local monotonicity with respect to a nonlinear guiding function. However, we did not see significant decreases in error by imposing local monotonicity priors. When we investigated our results further (Figure 6), we discovered that baseline neural networks had already learned to fully satisfy our local monotonicity priors, rendering them superfluous.

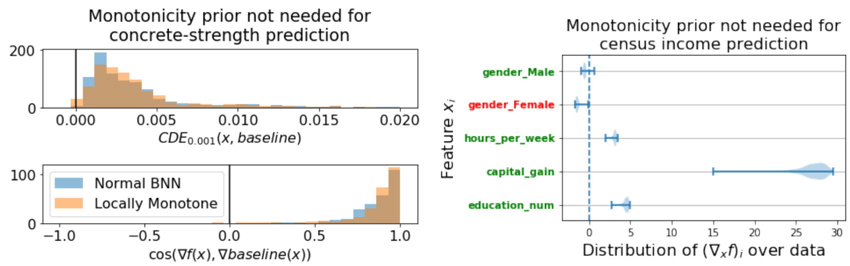

Figure 6: Superfluousness of local monotonicity priors on the concrete strength (left) and census income (right) datasets. On the concrete dataset, normal BNNs and the expert-provided baseline already show strong gradient and CDE alignment, while on the census dataset, local monotonicity already holds everywhere (except for gender_Male, whose gradients are very small, though this may not even be a true exception because gender_Male and gender_Female cannot change independently).

D.3 Additional Toy Examples

Finally, in Figure 7 we present additional toy examples to illustrate the kinds of priors and posteriors we can learn with our method.

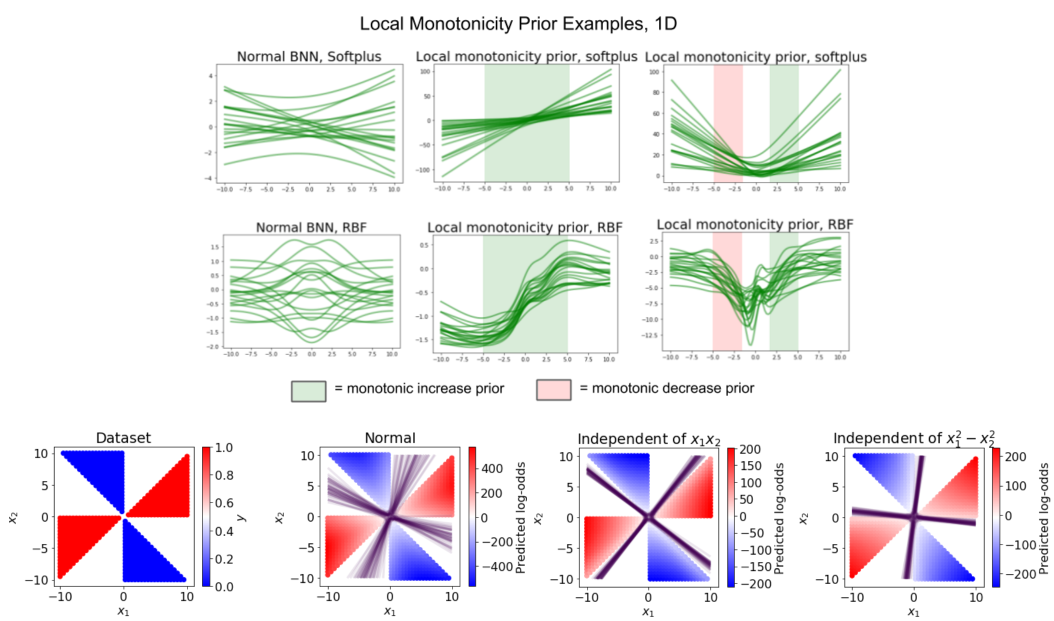

Figure 7: Top: Additional 1D locally monotone regression examples (computed via HMC and imposed by sampling over shaded regions). Bottom: 2D classification example where training data is restricted to the subset of $[-10, 10]^2$ where $x_1 x_2$ and $x_1^2 - x_2^2$ have equal signs (which determines y). We use softplus MLPs and plot posterior mean log-odds (in red/blue) and decision boundary samples (in purple), computed using Bayes-by-Backprop [26]. In the right two plots, we impose local invariance priors with respect to the two generating functions over training inputs, which lets us control decision boundary behavior despite the functions' nonlinearity.