PenDer: Incorporating Shape Constraints via Penalized Derivatives

Akhil Gupta,[1] **Lavanya Marla,**[1] **Ruoyu Sun,**[1]
Naman Shukla,[2] **Arinbjörn Kolbeinsson**[3]
[1]University of Illinois, Urbana-Champaign, IL, USA
[2]Deepair LLC, Dallas, TX, USA
[3]Imperial College London, London, UK
{akhilg3, lavanyam, ruoyus}@illinois.edu, naman@deepair.io, ak711@imperial.ac.uk

Abstract

When deploying machine learning models in the real-world, system designers may wish that models exhibit certain shape behavior, i.e., model outputs follow a particular shape with respect to input features. Trends such as monotonicity, convexity, diminishing or accelerating returns are some of the desired shapes. Presence of these shapes makes the model more interpretable for the system designers, and adequately fair for the customers. We notice that many such common shapes are related to derivatives, and propose a new approach, PenDer (Penalizing Derivatives), which incorporates these shape constraints by penalizing the derivatives. We further present an Augmented Lagrangian Method (ALM) to solve this constrained optimization problem. Experiments on three real-world datasets illustrate that even though both PenDer and state-of-the-art Lattice models achieve similar conformance to shape, PenDer captures better sensitivity of prediction with respect to intended features. We also demonstrate that PenDer achieves better test performance than Lattice while enforcing more desirable shape behavior.

Introduction

Questions around trust and interpretability in machine learning models are becoming increasingly relevant (Lipton 2016; Ribeiro, Singh, and Guestrin 2016; Doshi-Velez and Kim 2017) as these tools are being widely used for high-stakes decisions. Specifically, neural networks have shown tremendous success across critical domains such as healthcare (Shahid, Rappon, and Berta 2019), finance (Guresen, Kayakutlu, and Daim 2011; Nelson, Pereira, and de Oliveira 2017), pricing (Chiarazzo et al. 2014; Shukla et al. 2019), and law systems (Aikenhead 1996). Due to the significant societal implications involved, conformance to prior domain knowledge (Feelders 2000) and ethical notions, compatibility with business regulations, and model interpretability are actively desired by system designers. Meanwhile, users desire that the system is fair to their interests.

Even though real-world data has highly interactive features (Hall and Xue 2014), it is natural to possess *a priori* intuition about shape trends between a subset of input features and the output. Shape constraints are a classic way to characterize a function by whether its shape obeys certain properties (Groeneboom and Jongbloed 2014). In particular, 1-D shapes (Johnson and Jiang 2018) including monotonicity, convexity or concavity, diminishing or accelerating returns are fairly common in practice.

Shape behavior matching common beliefs improves users' trust and confidence in the AI-driven system and increases interpretability. For example, an ML model for screening loan applicants is expected to favor people with higher credit score, all other features remaining same: leading to accountability on the part of financial institutions while being fair to applicants. From a system designer's perspective, shape constraints lead to effective regularization and enhanced generalization to test data (Dugas et al. 2000). They make the model resilient to noisy data by virtue of control over model behavior (Gupta et al. 2018).

To facilitate the discussion, we call an input feature k an *intended* feature if the system designer requires the output to follow a specified shape with respect to this feature. Like Gupta et al. (2018), our notion of shape constraints is *ceteris paribus*, i.e. the shape holds for changes in a single feature k given other features remain constant. While constraints can be defined on pairs of features that complement each other (Cotter et al. 2019) giving rise to 2-D shape information, we focus on 1-D shape constraints in this work (Fig. 1).

In this work, we show how 1-D shape constraints can be incorporated via a simple regularization term without significantly compromising the predictive power of neural networks. More specifically, we propose a new approach that consists of two steps: first, we introduce a new formulation that includes constraints on the derivatives (of different orders) of the objective function; second, we solve this constrained problem by Augmented Lagrangian method (ALM). We call it PenDer (Penalizing Derivatives), since the resulting algorithm essentially penalizes the derivatives. We present experimental results on two open-source datasets and a real-world airline dataset, illustrating the effectiveness of PenDer in the following aspects:

- First, we show PenDer learns neural networks that achieve test performance close to standard neural nets while exhibiting desired shape behavior. In particular, our test performance is better than the state-of-the-art Lattice models (Gupta et al. 2018; Fard et al. 2016).
- Second, we demonstrate that Lattice models can be insen-

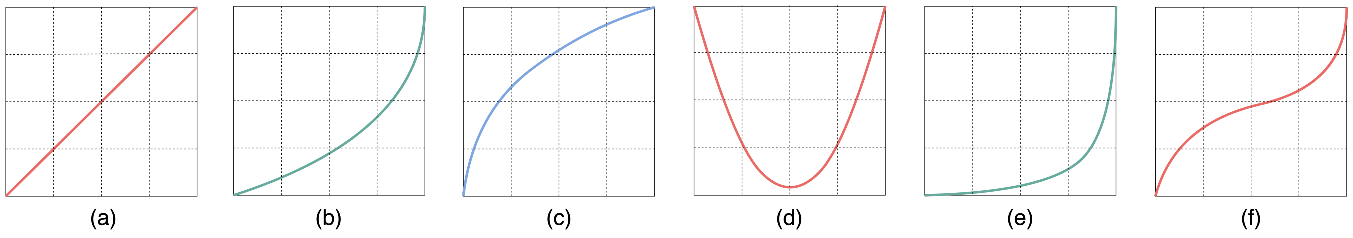

Figure 1: Illustration of some 1-D shape trends, where x-axis corresponds to the intended feature and y-axis is the corresponding change in output. We note that these common shapes are related to derivatives and devise a general framework in this work which incorporates such shape trends. Let f be the learned function, and we express these shapes in terms of derivatives of f. (a) Monotone ($f' > 0$), (b) Accelerating Returns ($f' > 0$, $f'' > 0$), (c) Diminishing Returns ($f' > 0$, $f'' < 0$), (d) Convex ($f'' > 0$), (e) and (f) correspond to higher-order shapes ($f'' > 0$, $f''' > 0$; and $f''' > 0$ respectively)

sitive to intended features, which is undesirable since that would not match the practical beliefs in many situations. In contrast, our method, PenDer, learns models that are sensitive to the intended features.

Motivating Example: Law School Admissions

Imagine that a law school admissions panel implements a neural network to predict the applicants' chances of acceptance. Assume the model has achieved good prediction accuracy. An ethics committee may want to examine if discriminatory behavior may still occur - one natural way is to check whether the system's behavior is consistent with common expectations. For instance, higher LSAT score should lead to higher chances of acceptance, and the committee may wonder whether the model exhibits such monotone behavior.

How to examine the existence of monotone behavior? In reality, any two applicants can differ in many features, and thus directly comparing their LSAT scores with admission outcomes (accepted or not) is not reasonable. A standard way of resolving this issue is to compare two *hypothetical* applicants, say, Adam and John, with John's LSAT score being higher than Adam's, and all other features such as undergraduate GPA remaining the same. We expect that John should have a higher chance of acceptance than Adam. For a systematic examination, for each existing applicant, we can generate a number of hypothetical applicants varying in the intended feature (here, LSAT score). The ML model generates predictions for these applicants, and we expect acceptance probabilities to increase with increase in LSAT scores.

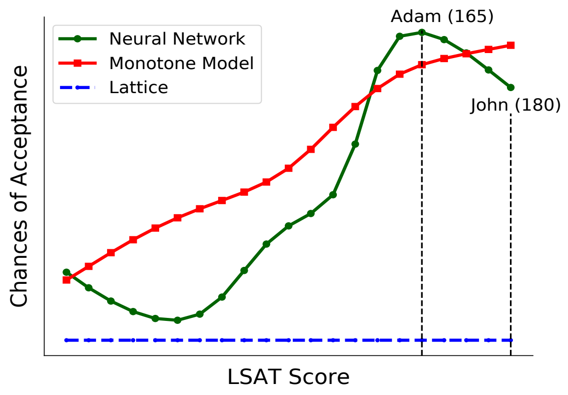

Figure 2: Comparison of predicted chances for hypothetical applicants from the Law School dataset. Suppose we generate 20 applicant profiles, differing only in LSAT scores. x-axis indicates the LSAT scores, and y-axis the predicted chances of acceptance. Green (round) and blue (dotted) curves represent the output of the neural network and Lattice respectively. The red (squared) curve indicates a more desirable monotone shape with a reasonable "slope".

We use an open-source dataset with law school numbers (Rankin 2020) to train a 4-layer neural network model and a Lattice model (Fard et al. 2016), see Fig. 2. The neural network learns an undesirable non-monotone trend between LSAT score and acceptance, which could be a concern for the designer. Lattice indeed learns a monotone trend; however, it learns a flat curve, indicating that the admission decision is *insensitive* to the LSAT score. This is contrary to expectations that LSAT score is an important factor for admission decisions (Brunet Marks and Moss 2016).

A more desirable shape for the committee is, for instance, the red (squared) curve indicated in Fig. 2. The unexpected behavior of Lattice in this example is one of the major motivations to explore alternative approaches. We provide formal metrics to distinguish the red (squared) curve from the two other curves in the section "Problem and Definitions".

Real-World Shape Constraints

We now discuss a few more real-world examples of shape behavior. Shape constraints play a critical role in fostering interpretability, fairness, and trust across these systems.

School Admissions In addition to the motivating example discussed above where students with higher standardized scores should have better chances, scholarship decisions is another similar setting. Machine learning model trained for determining percentage of tuition waiver should favor students from low-income families: acknowledging *favor the less fortunate* ethical pattern (Wang and Gupta 2020).

Finance The strong effect of credit scoring on profitability for financing companies is well-studied (Einav, Jenkins, and Levin 2013). For maximizing profits and trustworthiness of decisions, a loan approval model shall be monotone with credit score (when other factors are similar). Similarly, for credit card issuance, it would not be acceptable that a high-income applicant is rejected, whereas a low-income applicant with otherwise similar characteristics is accepted. Shape behavior is also prevalent across bond rating (Daniels and Kamp 1999) and option pricing (Gamarnik 1998).

Medical Diagnosis & Triage Medical studies have shown that cardiovascular disease risk increases with increase in cigarette consumption (Royston 2000) or dietary cholesterol (Zhong et al. 2019). Another study showed that higher levels of certain bio-markers are associated with increased prostate cancer chances (Ghosh 2007). This kind of prior knowledge can inform shape choices for ML-driven healthcare applications that greatly improve the interpretability of diagnosis.

Pricing Monotone behavior is also common in hedonic price models, where the price of a good depends on bundle of characteristics (Harrison and Rubinfeld 1978). Besides, the pricing models shall consider more complicated shape constraints. For instance, utility theory suggests that the purchase probability follows diminishing marginal returns – a price increase from \$20 to \$30 for a pair of jeans can reduce customer interest more compared to an increase from \$80 to \$90 (Mankiw 2011). Diminishing returns shape corresponds to concavity as well, so the pricing models shall consider it.

Related Work

Shape constraints are commonly enforced in neural network models by modifying either (i) the model architecture, or (ii) the objective loss function. Most past work has focused on monotonicity, with recent interest in shapes such as diminishing and accelerating returns (Gupta et al. 2018).

Architectural changes for monotonicity include connecting hidden layer nodes differently, imposing constraints on weights, or defining a different structure altogether (Archer and Wang 1993; Sill 1998; Kay and Ungar 2000; Lang 2005; Dugas et al. 2009; Daniels and Velikova 2010; Fard et al. 2016). In a recent work, You et al. (2017) proposed deep Lattice networks using a combination of linear calibrators and lattice layers for learning guaranteed monotone functions. Gupta et al. (2018) extended the Lattice models (Fard et al. 2016; You et al. 2017) to incorporate convex/concave shape constraints, outperforming previous methods.

The second class of methods modify the loss function. Although they are not expected to fully guarantee shape, they exploit the trade-off between empirical risk and shape error. Daniels and Kamp (1999) added a bias term as a penalty for negative weights, but their loss is effective only when the data is largely monotone. Sill and Abu-Mostafa (1997) regularized the objective function by penalizing the squared deviations from monotonicity for virtual pairs of input samples. Our work differs from MonoHints (Sill and Abu-Mostafa 1997) in a few aspects. First, they only consider monotonicity constraints, whereas our approach easily applies to any shape behavior related to derivatives (we explicitly consider first and second order derivatives in this work). Second, they penalize the finite differences, while we penalize the derivatives directly. As they introduce extra virtual data points for computing the finite differences, their method may require extra memory and FLOPs; in addition, the extra cost is even higher if they want to extend the finite difference method to higher-order derivatives. Third, they solve an unconstrained formulation directly while we optimize the constrained formulation via ALM, which is a more popular strategy in nonlinear programming.

Lattice Models Lattice models are the current state-of-the-art approach for enforcing shape constraints in neural networks (Gupta et al. 2018), with a publicly available TensorFlow package (You et al. 2019). However, Lattice has a few limitations. First, it may suffer from combinatorial explosion. Gupta et al. (2016) establish that the number of parameters in Lattice look-up table scale as K^D, when there are K keypoints along each intended feature and D intended features in total. As a result, it is challenging to apply Lattice with 10 intended features: with 20 keypoints, Lattice needs to process $20^{10} \approx 10^{13}$ (10 trillion) data points. This issue restricts the applicability of Lattice to large-scale problems. Second, from Fig. 2, we see that Lattice models lead to insensitive model behavior (flattened trends) with the intended feature, which can be a concern for system designers. We observe this behavior consistently with different datasets, as shown in the section "Experiments" and Appendix D.

Due to the limitations of existing models, including Lattice, we believe it is interesting to introduce new alternatives to handle shape constraints. In this work, we use Lattice as a major baseline as it achieves the SoTA performance on the general shape-constraining-task – other approaches either handle only monotonicity constraints or are inferior to Lattice (Gupta et al. 2018).

Problem and Definitions

Consider the general setting of a supervised learning problem with an underlying data distribution $P(x, y)$, where $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, and $P(x, y)$ is a certain distribution on $\mathbb{R}^d \times \mathbb{R}$. For simplicity, we consider scalar y, although our method can be extended to vectors. Consider a training set of n samples $\{(x_i, y_i)\}; i = 1, \ldots, n$ (and, test set with n_t samples). We want to find a function $f(x)$ in a class of possible functions $\mathcal{H}$ (e.g. a set of neural networks f_θ parameterized by θ) such that the prediction $f(x_i) \sim y_i, \forall i \in [1, n]$.

Besides accuracy, we want f to satisfy one or more shape constraints. We say f is *increasing* over the k-th feature (or, $x[k]$) if for any $u, v \in \mathbb{R}^d$ satisfying $u[k] \geq v[k]$ and $u[j] = v[j], \forall\, j \in [d] \setminus k$, we have $f(u) \geq f(v)$. Similarly, we say f is *decreasing* over $x[k]$ if $\geq$ is changed to $\leq$. We say f is *convex* over $x[k]$ if given u, v that satisfy previous conditions, we have $f'(u) \geq f'(v)$. And, we say f is *concave* over $x[k]$ if we have $f'(u) \leq f'(v)$. We refer to such properties as "shape-constrained" behavior. We say f is shape-constrained over the feature set $S \subseteq [d]$ if f is shape-constrained over every feature in S. Our goal is to learn the function f: shape-constrained over user-defined features S, where S corresponds to the prior domain knowledge enforced by the system designer.

For comparing different approaches, we utilize evaluation metrics that measure model's predictive performance, model's ability to conform to shape on test data, and sensitivity of the enforced shape. While the first two metrics are well-known, we now introduce our sensitivity metric.

Sensitivity: Lipschitz Metric

Fig. 2 shows that Lattice (Gupta et al. 2018) leads to a flattened shape, which though monotone, is not desirable, while the hypothetical red (squared) curve is more desirable. To distinguish the two curves, we use sensitivity as a metric. We define Lipschitz metric ($\mathcal{L}_k$) for the k-th feature as the average gradient of the predictions across the inflated test dataset, $\overline{X}$. This "inflated version" is obtained by perturbing the intended feature k to equally-spaced values in a certain range $[\tau_{\min}, \tau_{\max}]$ (fixed keypoints) for every test sample, while keeping all other features constant[‡]. A natural choice for $\tau_{\min}, \tau_{\max}$ is the minimal and maximal value of the intended feature in the entire dataset.

$$\mathcal{L}_k = \frac{1}{n_t} \sum_{i=1}^{n_t} \left\| \left(\frac{\partial f}{\partial \overline{x}_i[k]} \right) \right\|_2, \text{ where:} \quad \{\overline{x}_i[k] \in [\tau_{\min}, \tau_{\max}]\} \tag{1}$$

Zero sensitivity ($\mathcal{L}_k = 0$) is particularly undesirable since it indicates a flat trend line.

One may wonder whether we can treat the sensitivity as a shape constraint, similar to monotonicity and convexity. However, for different data points, the desired sensitivity metric may be different, and it is not easy to set sensitivity lower bounds for all data points. Therefore, it is hard to enforce sensitivity *a priori*. We treat it as *a posteriori* metric: compute $\mathcal{L}_k$ for every feature in S after the model is trained.

PenDer: Penalizing Derivatives

We now propose our approach, PenDer, that penalizes the derivatives of the objective function to enforce shape constraints. Many shape behaviors can be modeled by simple conditions on the derivatives, e.g., "a function is increasing monotone" means the first order derivative is non-negative, and convexity/concavity are determined by the sign of the second order derivative. Therefore, constraining the first or second order derivatives is sufficient for conformance to common 1-D shapes such as monotonicity and convexity. In PenDer, we add constraints to the standard loss function to form a constrained optimization problem, and then solve the constrained problem using the Augmented Lagrangian method (ALM) (Hestenes 1969). The idea of incorporating *a priori* knowledge with point-wise derivatives in the formulation is inspired from finite element analysis (Strang 1972; Wilmott et al. 1995) and the function classes presented by Dugas et al. (2009). We explain the details of PenDer below.

Suppose $S \subseteq [d]$ is the set of intended features. Let θ denote the collection of all parameters in the neural network $f_\theta(\cdot)$, and $F(\theta)$ be the standard empirical risk on the training data set, $d_{\text{train}} = \{x_i, y_i\}_{i=1}^n$. To embed prior shape knowledge, we add constraints on the derivative of the function f at every x_i in d_{train}, with respect to every feature in S. More specifically, we add the following constraints

$$c_i^r[k]: \quad \frac{\partial^r f_\theta(x_i)}{\partial x_i^r[k]} \geq 0 \ ; \ \forall i \in [1, n], \ \forall k \in S \tag{2}$$

to embed general shape behavior, where $c_i^r[k]$ denotes the non-negativity constraint on the r-th order partial derivative of f with respect to k-th feature at i-th training sample. We focus on 1st and 2nd order derivatives as the common shapes seem to only require 1st and 2nd order derivatives, but we note that our formulation allows for higher order constraints.

Setting $r = 1$ in the equation above, we obtain increasing monotone constraints for the k-th feature. For decreasing monotone, $\geq$ can be changed to $\leq$. Similarly, we can realize convexity and concavity constraints by setting $r = 2$ in the formulation above. Note that our constraints don't assume any dependency between shapes across various intended features i.e., the output could exhibit increasing behavior with respect to one feature and concavity with respect to another. Such flexibility is not easy to achieve for model-architecture-change-type methods. Let $\mathcal{C}$ denote the set of constraints. The form of $\mathcal{C}$ depends on the requirements; for instance, if we only consider *increasing monotone* then $\mathcal{C}$ consists of constraints from Eq. 2 for $r = 1$.

The constrained problem for shape-constrained f can be written as: $\min_\theta \ F(\theta)$ subject to $\mathcal{C}$. When using ALM to solve the constrained optimization problem, these constraints are moved to the objective function in the form of regularizers, to obtain the augmented Lagrangian function as follows:

$$\begin{aligned}\mathcal{L}_\mu(\theta; \lambda) = F(\theta) &+ \sum_{k \in S} \frac{\mu_k}{n} \sum_{i=1}^n \max(0, -c_i^r[k])^2 \\ &+ \sum_{k \in S} \frac{1}{n} \sum_{i=1}^n \lambda_{i,k} \cdot \max(0, -c_i^r[k])\end{aligned} \tag{3}$$

where, μ_k (> 0) refers to the penalty coefficient corresponding to the quadratic penalty term for the k-th feature, and $\lambda_{i,k}$ is the Lagrange multiplier (at $c_i^r[k]$). Note that the max operator with negative $c_i^r[k]$ corresponds to the constraint violation at $x_i[k]$.

The ALM works as follows[§]: the neural network parameters θ are updated by performing gradient descent on $\mathcal{L}_\mu(\theta; \lambda)$, and ($\mu_k$, $\lambda_{i,k}$) are updated via gradient ascent on $\mathcal{L}_\mu(\theta; \lambda)$; $\forall\, k \in S, i \in [1, n]$. We present the pseudocode for PenDer in Algorithm 1. We present a proof of convergence for inexact ALM methods in constrained convex cases, borrowed from Xu (2019), in Appendix A.

Experiments

In this section, we present empirical results for the proposed PenDer approach, in comparison to (i) deep neural network

[‡]This is similar to "one-pixel-perturbation" in adversarial robustness, with the key difference that adversarial robustness often means the output is invariant after a small pixel perturbation, while here we hope the output exhibits shape properties.

[§]Rigorously speaking, this is an inexact version of ALM (Sahin et al. 2019) because we do not update the primal variable via exact minimization of the augmented Lagrangian function.

Algorithm 1 PenDer: Penalizing Derivatives

```
1:  Initialize:
        $\theta^{(0)} \leftarrow$ Initial NN parameters
        $\mu_k^{(0)} \leftarrow$ Penalty coefficients
        $\lambda_{i,k}^{(0)} \leftarrow$ Langrange multipliers
        $T \leftarrow$ Maximum iterations
        $\rho \leftarrow$ Update multiplier for $\mu$
2:  for $t = 0, \ldots, T-1$ do
3:      $\theta^{(t+1)} \leftarrow$ updated parameters after $\min \mathcal{L}_\mu(\theta^{(t)}; \lambda)$
4:      if stopping criteria met then
5:          stop with approximate solution $\theta^{(t+1)}$
6:      for $k \in S$ do
7:          for $i = 1, \ldots, n$ do
8:              $\lambda_{i,k}^{(t+1)} \leftarrow \lambda_{i,k}^{(t)} + \mu_k^{(t)} \cdot \max(0, -(c_i^r)^{(t)}[k])$
9:          $\mu_k^{(t+1)} \leftarrow \rho \cdot \mu_k^{(t)}$
10: return $\theta^{(t+1)}$
```

(DNN) and (ii) Lattice model (Gupta et al. 2018; Fard et al. 2016) with shape constraints. In addition to quantitative comparison, we perform post-hoc analysis of the methods to gauge which one yields trends acceptable in society.

Evaluation Metrics

To compare the methods, we use three metrics: (i) test accuracy (classification) or mean absolute error (MAE, regression) for predictive performance, (ii) the monotonicity or convexity score for shape conformance, and (iii) the Lipschitz metric (Eq. 1) for sensitivity of the enforced shape.

While accuracy and MAE are reported on the test dataset, the other two shape-specific metrics are computed using model predictions on an "inflated" version ($\overline{X}$) of the test dataset. For simplicity, we discretize the intended feature k into fixed keypoints ($[\tau_{\min}, \tau_{\max}]$) and perturb the test dataset in this feature while keeping all other features constant. Note that the ground-truth output labels are not needed for computing shape metrics. We now define metrics for measuring the shape conformance.

Monotonicity and Convexity Score $(\mathcal{M}_k, \mathcal{C}_k)$ These metrics quantify whether the learned function f conforms to the shape enforced by the system designer. We leverage the finite difference method for approximating the derivative on $\overline{X}$ as we believe that finite differences are more intuitive for the end user. For instance, when predicting the impact of LSAT score on acceptance rate, the admissions committee can be given insights such as "5 more points lead to 2% higher chance of admission around a score of 170". Of course, the derivative serves a similar purpose, but because such insights are commonly discussed in the language of finite difference, we report that. Let Δ_+ denote the forward difference operator (Wilmott et al. 1995), and then Δ_+^r indicates the r-th order forward difference. For r-th order shape conformance, we define:

$$\gamma_k^r = \frac{1}{n_t} \sum_{i=1}^{n_t} \delta_i \,, \text{ where:} \tag{4}$$

$$\delta_i = \begin{cases} 1 & \text{if } \Delta_+^r f(x_j; \theta) \geq 0 \; \forall \; j \ni \{x_j[k] \in \mathbb{K}, \\ & \qquad x_j[p] = x_i[p] \; \forall \; p \neq k\} \\ 0 & \text{otherwise.} \end{cases}$$

Here, the latent variable δ is an indicator that measures the degree of conformance for the k^{th} feature across the dataset. *Ceteris paribus*, $x[k]$ takes values in $\mathbb{K}$ i.e. between $[\tau_{min}(x[k]), \tau_{max}(x[k])]$ which are the minimal and maximal values of $x[k]$ present in the entire dataset.

Similar to the previous section, we set $r = 1$ in Eq. 4 to define the Monotonicity Score ($\mathcal{M}_k$). Setting $r = 2$ describes the Convexity Score ($\mathcal{C}_k$) which measures conformance to convex/concave behavior. A score of 1 corresponds to a fully conforming function; higher the score, better the shape conformance. Note that γ_k^r and δ_i are defined with *increasing* and *convex* shape in mind. For *decreasing* or *concave* behavior, we change $\geq$ to $\leq$ in the definition of δ_i.

Datasets

We utilize three datasets in this work, two open-source: Law School Admissions (Rankin 2020) and Used Cars (Leka 2019), and a real-world proprietary airline ancillary dataset. Our major motivation for these datasets is that they have intended features whose shape is not learned by the unconstrained DNN. This was investigated using conformance metrics ($\mathcal{M}_k$ and $\mathcal{C}_k$) on the predictions from DNN (detailed analysis in Appendix B). In addition to the three datasets in this main text, we present empirical comparison of methods on two other open-source datasets: Wine Ratings used by Gupta et al. (2018), and Sberbank Russian Housing dataset (Sberbank 2017), in the Appendix E.

- **Law School Admissions** (Rankin 2020): A classification task to predict whether or not the applicant would get accepted to a particular law school. Features include LSAT score, GPA, race, gender, school type, and in-state status, among others. Acceptance probability is believed to be somewhat *increasing* over LSAT score.
- **Used Cars** (Leka 2019): Retrieved from eBay-Germany, this dataset contains data of used cars for resale. The regression task is to predict the price using the vehicle type, age of the car, horsepower, and mileage reading. We expect *diminishing marginal returns* in car price with increasing mileage. For example, a significant drop in price is expected initially between a 1K-mile-old car and a 10K-mile-old car, but the price gap between a 80K-mile-old car and a 90K-mile old car is relatively small. Thus, we want the function to be *decreasing* and *convex*.
- **Airline Ancillary**: Ancillary pricing is a sub-field within airline pricing (Shukla et al. 2019). We obtain a proprietary real-world dataset for ancillary pricing from a large airline, and want to predict the purchase probability of an ancillary (classification). The designers of the pricing recommender system want the probability of purchase to follow a *diminishing returns* (*decreasing* and *convex*) shape with ancillary price, as per utility theory.

We use 20% of the entire dataset for test analysis, and perform a random 80-20 split on the remaining data to create training and validation datasets. We use the validation data to tune hyperparameters, and report evaluation metrics on the test set.

Model Specifics and Hyperparameters

We train a 4-layer neural network (two hidden layers) for all our experiments (except Lattice). We tune hyperparameters such as optimizer, learning rate, and activation function on the DNN, and keep them same for PenDer to facilitate a fair comparison. Adam (Kingma and Ba 2015) outperforms SGD on all the datasets. We employ early stopping (no model improvement for 40 epochs) as our stopping criterion, and decay the learning rate by a factor of 10 when validation error reaches a plateau. Because piece-wise linear activation functions cannot be used for penalizing high-order derivatives, we use sigmoid activation for our datasets. While ReLU could be an option when penalizing only the 1st order derivatives, we find from our experiments that sigmoid works better in those cases as well. The models are trained on cross-entropy loss for classification, and mean squared loss for regression. For more details, refer to Appendix C. Note some method-specific parameters:

- PenDer: $\lambda_{i,k}$ are initialized at 0 and we initially choose a small value for $\mu_k = 0.1$. ρ determines the increase in μ and we tune it from $\{2, 5, 10\}$. While $\lambda_{i,k}$ are updated in each iteration, μ_k are updated only when the quadratic penalty loss component doesn't improve on the validation set. For datasets where the unconstrained DNN indicates low shape conformance, higher ρ was found to be effective as it penalizes the loss component for shape constraints aggressively as training progresses.

We consistently observe that PenDer converges in at most twice the number of iterations as the unconstrained DNN, implying that the addition of shape constraints does not affect the convergence speed too much. In practice, we find that training Lattice models to convergence takes at least twice as much as time as PenDer, and we expect this gap will quickly increase as the number of intended features increase (though we do not perform such experiments here).

Implementation and Reproducibility. We implement PenDer (Algorithm 1)[†] in TensorFlow (TF). Lattice is already provisioned in TF by You et al. (2019). We train our models on a 2.7 GHz Dual-Core processor. We replicate the experiments on each method and dataset using different seed values, and report the mean and deviation among performance and shape-specific metrics over 10 runs. Each run uses a different subset of the data and initializes NN parameters differently. Our implementation focuses on reproducibility, and we share the code to replicate model training.

Law School Admissions

For this classification task, we have 128K training, 32K validation, and 40K test examples (Rankin 2020). We predict the probability of application acceptance to law schools while enforcing *increasing* monotone shape on LSAT score. Table 1 presents the performance of the three methods with respect to our metrics. First, we observe from the DNN monotonicity scores $\mathcal{M}_k$ that the dataset is not inherently monotone in the intended feature as the score is low. Both Lattice and PenDer learn fully monotone functions ($\mathcal{M}_k = 1$) with a minor drop of 0.5% in test accuracy, thereby appropriately capturing merit-based ethics.

Model	Test Acc. (%)	$\mathcal{M}_k$ (LSAT)	$\mathcal{L}_k$ (LSAT)
DNN	75.24 ± 0.13	0.44 ± 0.07	0.14 ± 0.01
Lattice	74.87 ± 0.11	1	0
PenDer	74.87 ± 0.13	1	0.07

Table 1: Performance of methods: *increasing* constraint with LSAT score.

Though Lattice and PenDer look similar for the first two metrics, the Lipschitz metric marks the distinction in captured sensitivity. As Lattice structurally enforces the monotone behavior, it diminishes feature importance for LSAT score to zero aiming to increase test performance. The trained Lattice model is insensitive to the LSAT score ($\mathcal{L}_k = 0$) across the entire test dataset. Such behavior is unacceptable for the designer as LSAT score is an important parameter for applicant selection (Brunet Marks and Moss 2016) and a good model should exhibit sensitivity.

As an illustration, consider a sample (student profile) from the test dataset (Fig. 3). Varying the LSAT score from the minimum to maximum value ($[\tau_{\min} = 120, \tau_{\max} = 180]$) while keeping everything else fixed, we obtain the predicted probability trend from different models for the student - which should be *increasing* as expected. Plot shows that PenDer effectively makes the trend strictly increasing, whereas Lattice fits a flat line insensitive to LSAT score.

From Table 1, note that Lattice still generates a model with reasonably good test accuracy, indicating that we have tuned Lattice reasonably well. We include more plots with respect to other features in Appendix D to demonstrate that Lattice learns a good fit overall. However, it still leads to insensitive trends with respect to the intended LSAT score.

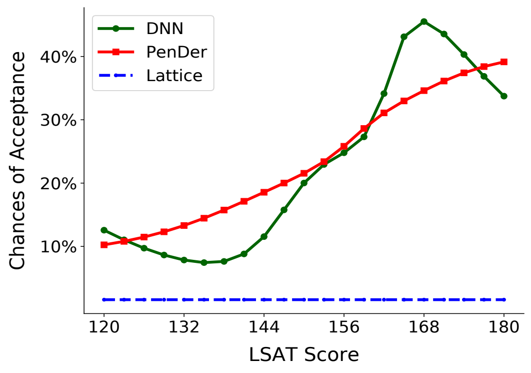

Figure 3: Comparison of models for a sample student application. PenDer results in a desirable *increasing* trend as compared to Lattice and DNN.

[†]Code and Appendix available within the GitHub repository at https://github.com/deepair-io/PenDer.

Used Cars - Ebay

For the used cars data from eBay (Leka 2019), we have 78K training, 19.5K validation, and 24.4K test samples. *Diminishing returns* (decreasing and convex) shape in price is enforced with respect to mileage. Our results in Table 2 use mean absolute error (MAE) to compare price prediction error in regression. Again, observe from DNN shape metrics ($\mathcal{M}_k$ and $\mathcal{C}_k$) that the dataset does not inherently adhere to the expected shape for mileage. However, Lattice and PenDer guarantee the shape behavior, but PenDer does so with a \$100 lower error which is a significant difference.

Model	Test MAE (\$)	$\mathcal{M}_k$	$\mathcal{C}_k$
DNN	$1,941 \pm 17$	0.29 ± 0.04	0.005 ± 0.005
Lattice	$2,080 \pm 30$	1	1
PenDer	$1,982 \pm 24$	1	1

Table 2: Performance of methods: *diminishing returns* constraint with mileage reading.

In terms of sensitivity, Lattice again suffers from insensitivity to the mileage while predicting the price ($\mathcal{L}_k = 0$ in Table 3). PenDer captures similar sensitivity as DNN, strengthening the claim that PenDer generates desirable models for the system designer. Across the three metrics, PenDer stands out as it balances the predictive performance and shape behavior well.

Model	$\mathcal{L}_k$ (Used Cars)	$\mathcal{L}_k$ (Ancillary)
DNN	0.012 ± 0.001	0.061 ± 0.009
Lattice	0	0
PenDer	0.012	0.010 ± 0.003

Table 3: Lipschitz metric for Used Cars and Airline datasets.

Airline Ancillary

For the real-world ancillary dataset, we aim to predict the probability of ancillary purchase using 57.7K train, 14.4K validation, and 18.2K test samples. To stay accountable to customer expectations, conform to business regulations, and incorporate the price elasticity of demand, the airline wants to impose *diminishing returns* in the predicted willingness to pay with respect to offered price. From Table 4, we see that DNN does not learn this desired shape which can reduce trust in the airline's pricing. While both Lattice and PenDer conform to the enforced shape, Lattice has a 2% drop in test

Model	Test Acc. (%)	$\mathcal{M}_k$	$\mathcal{C}_k$
DNN	66.55 ± 0.15	0.51 ± 0.13	0.16 ± 0.04
Lattice	64.28 ± 0.28	1	1
PenDer	66.39 ± 0.33	1	0.98 ± 0.02

Table 4: Performance of methods: *diminishing returns* constraint with ancillary price.

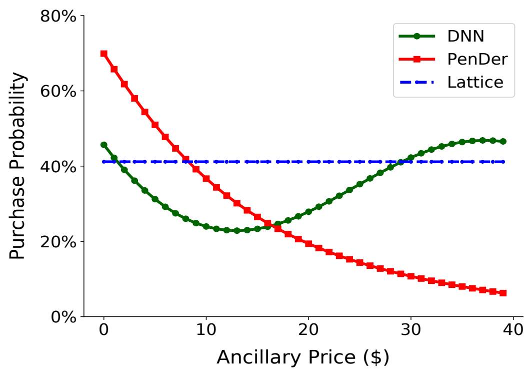

Figure 4: Comparison of models for an arbitrary customer in the Airline ancillary dataset. PenDer learns the enforced *diminishing returns* shape well, unlike Lattice.

accuracy, significant enough to reduce revenues compared to the DNN. PenDer achieves test accuracy similar to the DNN while maintaining the shape, which demonstrates the ability of regularization-driven PenDer to balance both the objectives in the search space. Considering a customer from the test set in Fig. 4, we hypothetically vary the price from \$0 to \$40 to understand model sensitivity. From the $\mathcal{L}_k$ value in Table 3 and Fig. 4, we find that Lattice is nearly insensitive towards price. This is undesirable as the customer's willingness to purchase should definitely change with increase in price. On the other hand, PenDer overcomes this limitation of Lattice by being reasonably sensitive to ancillary price.

Conclusion

In this work, we presented PenDer, a new approach for *a priori* incorporating common shapes desired by the system designer into neural networks – with the objective of increasing interpretability, fairness and trust. Along with accuracy, "conformance to shape" and "sensitivity of shape" are two important properties that are desired during *a posteriori* analysis of such systems. We show that state-of-the-art Lattice models do not satisfy all these requirements together.

PenDer penalizes derivatives of the objective function to enforce a desired shape. We devise a general framework that handles derivative-based constraints, and solves the constrained problem using an Augmented Lagrangian Method. We focus on common shapes such as monotonicity and convexity, defined using the 1st and 2nd order derivatives, in our experiments. To measure model sensitivity, we introduce a Lipschitz metric. We compare the performance of PenDer, state-of-the-art Lattice and deep neural network along three metrics: test set predictions, conformance to shape, and associated sensitivity. Our experiments on real-world datasets illustrate the advantage of PenDer to balance all objectives, and that Lattice often leads to insensitive shape behavior.

Our work demonstrates that while most neural network systems are black-boxes, they need not remain so. We can design efficient approaches to help neural networks conform to user- or designer- desired behavior, balance multiple metrics well, and increase trust and accountability.

Acknowledgments

We sincerely acknowledge our airline partners for their continuing support. The academic partners are also thankful to deepair (www.deepair.io) for funding this research. In particular, we would like to thank Kartik Yellepeddi for making this research possible through the collaborative environment at deepair.

Ethical Impact

Shape constraints serve the purpose of enforcing prior domain knowledge in systems for conformance to social ethics and meeting users' expectations. Reliable shape behavior makes models more interpretable and trustworthy, and less discriminatory, by introducing transparency. Shape behavior is desired in several real-world settings, from school admissions to pricing, as discussed in the main text.

Our approach for formulating and solving for shape constraints, PenDer, is regularization-driven which makes it simple to understand and easy to implement into existing systems. We leverage a variety of evaluation metrics, particularly sensitivity, to ensure that system is analyzed appropriately after the model is trained. In addition, incorporating shape behavior makes the model resilient to adversarial behavior as test-time attacks have lower chances of success on a model which preserves the order, by design.

Although our method has been designed for fair use, we acknowledge that malicious system designers can potentially use shape constraints with an intent to retain bias or unfairness in their system. Such misuse can often only be detected through an *a posteriori* analysis of the system decisions via a thorough audit-type investigation.

References

Aikenhead, M. 1996. The Uses and Abuses of Neural Networks in Law. *Santa Clara High Technology Law Journal* 12: 31.

Archer, N. P.; and Wang, S. 1993. Application of the Back Propagation Neural Network Algorithm with Monotonicity Constraints for Two-Group Classification Problems. *Decision Sciences* 24(1): 60–75.

Brunet Marks, A.; and Moss, S. A. 2016. What Predicts Law Student Success? A Longitudinal Study Correlating Law Student Applicant Data and Law School Outcomes. *Journal of Empirical Legal Studies* 13(2): 205–265.

Chiarazzo, V.; Caggiani, L.; Marinelli, M.; and Ottomanelli, M. 2014. A Neural Network based Model for Real Estate Price Estimation Considering Environmental Quality of Property Location. *Transportation Research Procedia* 3: 810–817.

Cotter, A.; Gupta, M.; Jiang, H.; Louidor, E.; Muller, J.; Narayan, T.; Wang, S.; and Zhu, T. 2019. Shape Constraints for Set Functions. volume 97 of *Proceedings of Machine Learning Research*, 1388–1396. Long Beach, California, USA: PMLR.

Daniels, H.; and Kamp, B. 1999. Application of MLP Networks to Bond Rating and House Pricing. *Neural Computing & Applications* 8(3): 226–234.

Daniels, H.; and Velikova, M. 2010. Monotone and Partially Monotone Neural Networks. *IEEE Transactions on Neural Networks* 21(6): 906–917.

Doshi-Velez, F.; and Kim, B. 2017. Towards A Rigorous Science of Interpretable Machine Learning. URL https://arxiv.org/abs/1702.08608.

Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C.; and Garcia, R. 2000. Incorporating Second-Order Functional Knowledge for Better Option Pricing. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, 451–457. Cambridge, USA: MIT Press.

Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C.; and Garcia, R. 2009. Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research* 10(Jun): 1239–1262.

Einav, L.; Jenkins, M.; and Levin, J. 2013. The impact of credit scoring on consumer lending. *The RAND Journal of Economics* 44(2): 249–274.

Fard, M. M.; Canini, K.; Cotter, A.; Pfeifer, J.; and Gupta, M. 2016. Fast and flexible monotonic functions with ensembles of lattices. In *Advances in Neural Information Processing Systems*, 2919–2927.

Feelders, A. 2000. Prior Knowledge in Economic Applications of Data Mining. In Zighed, D. A.; Komorowski, J.; and Żytkow, J., eds., *Principles of Data Mining and Knowledge Discovery*, 395–400. Springer Berlin Heidelberg.

Gamarnik, D. 1998. Efficient Learning of Monotone Concepts via Quadratic Optimization. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 24–26.

Ghosh, D. 2007. Incorporating monotonicity into the evaluation of a biomarker. *Biostatistics* 8(2): 402–413.

Groeneboom, P.; and Jongbloed, G. 2014. *Nonparametric Estimation under Shape Constraints: Estimators, Algorithms and Asymptotics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Gupta, M.; Bahri, D.; Cotter, A.; and Canini, K. 2018. Diminishing Returns Shape Constraints for Interpretability and Regularization. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 6834–6844. Curran Associates, Inc.

Gupta, M.; Cotter, A.; Pfeifer, J.; Voevodski, K.; Canini, K.; Mangylov, A.; Moczydlowski, W.; and van Esbroeck, A. 2016. Monotonic Calibrated Interpolated Look-Up Tables. *Journal of Machine Learning Research* 17(109): 1–47.

Guresen, E.; Kayakutlu, G.; and Daim, T. U. 2011. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications* 38(8): 10389–10397.

Hall, P.; and Xue, J.-H. 2014. On selecting interacting features from high-dimensional data. *Computational Statistics & Data Analysis* 71: 694–708.

Harrison, D.; and Rubinfeld, D. L. 1978. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management* 5(1): 81–102.

Hestenes, M. R. 1969. Multiplier and gradient methods. *Journal of Optimization Theory and Applications* 4(5): 303–320.

Johnson, A.; and Jiang, D. R. 2018. Shape Constraints in Economics and Operations Research. *Statistical Science* 33: 527–546.

Kay, H.; and Ungar, L. H. 2000. Estimating monotonic functions and their bounds. *AIChE Journal* 46(12): 2426–2434.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lang, B. 2005. Monotonic Multi-layer Perceptron Networks as Universal Approximators. In Duch, W.; Kacprzyk, J.; Oja, E.; and Zadrożny, S., eds., *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, 31–37. Springer Berlin Heidelberg.

Leka, O. 2019. Used cars database — Kaggle. URL https://www.kaggle.com/orgesleka/used-cars-database. (Accessed August 24, 2020).

Lipton, Z. C. 2016. The Mythos of Model Interpretability. *CoRR* abs/1606.03490.

Mankiw, N. 2011. *Principles of Economics, 5th edition*. South-Western Cengage Learning. URL http://mankiw.swlearning.com/. The Introductory-Level Textbook.

Nelson, D.; Pereira, A. M.; and de Oliveira, R. A. 2017. Stock market's price movement prediction with LSTM neural networks. *2017 International Joint Conference on Neural Networks (IJCNN)* 1419–1426.

Rankin, R. 2020. MyLSN. URL https://mylsn.info/. (Accessed August 24, 2020).

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'16, 1135–1144. New York, NY, USA: Association for Computing Machinery.

Royston, P. 2000. A useful monotonic non-linear model with applications in medicine and epidemiology. *Statistics in Medicine* 19(15): 2053–2066.

Sahin, M. F.; Eftekhari, A.; Alacaoglu, A.; Latorre, F.; and Cevher, V. 2019. An Inexact Augmented Lagrangian Framework for Nonconvex Optimization with Nonlinear Constraints. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; dÁlché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 13965–13977. Curran Associates, Inc.

Sberbank. 2017. Sberbank Russian Housing Market — Kaggle. URL https://www.kaggle.com/c/sberbank-russian-housing-market. (Accessed July 30, 2020).

Shahid, N.; Rappon, T.; and Berta, W. 2019. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLOS ONE* 14(2): 1–22.

Shukla, N.; Kolbeinsson, A.; Otwell, K.; Marla, L.; and Yellepeddi, K. 2019. Dynamic Pricing for Airline Ancillaries with Customer Context. In *Proceedings of the 25th ACM SIGKDD Conference on ACM Knowledge Discovery and Data Mining*. ACM.

Sill, J. 1998. Monotonic Networks. In Jordan, M. I.; Kearns, M. J.; and Solla, S. A., eds., *Advances in Neural Information Processing Systems 10*, 661–667. MIT Press.

Sill, J.; and Abu-Mostafa, Y. S. 1997. Monotonicity hints. In *Advances in Neural Information Processing Systems*, 634–640.

Strang, G. 1972. Approximation in the finite element method. *Numerische Mathematik* 19(1): 81–98.

Wang, S.; and Gupta, M. 2020. Deontological Ethics By Monotonicity Shape Constraints. volume 108 of *Proceedings of Machine Learning Research*, 2043–2054. PMLR.

Wilmott, P.; Howson, S.; Howison, S.; Dewynne, J.; et al. 1995. *The mathematics of financial derivatives: a student introduction*. Cambridge university press.

Xu, Y. 2019. Iteration complexity of inexact augmented Lagrangian methods for constrained convex programming. *Mathematical Programming* .

You, S.; Ding, D.; Canini, K.; Pfeifer, J.; and Gupta, M. 2017. Deep Lattice Networks and Partial Monotonic Functions. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 2981–2989. Curran Associates, Inc.

You, S.; Fard, M. M.; Zhang, X.; and Kim, R. 2019. TensorFlow Lattice. https://github.com/tensorflow/lattice. (Accessed May 15, 2020).

Zhong, V. W.; Van Horn, L.; Cornelis, M. C.; Wilkins, J. T.; Ning, H.; Carnethon, M. R.; Greenland, P.; Mentz, R. J.; Tucker, K. L.; Zhao, L.; Norwood, A. F.; Lloyd-Jones, D. M.; and Allen, N. B. 2019. Associations of Dietary Cholesterol or Egg Consumption With Incident Cardiovascular Disease and Mortality. *JAMA* 321(11): 1081–1095.