Multidimensional Shape Constraints

Maya R. Gupta [*1] **Erez Louidor** [*1] **Olexander Mangylov** [*1] **Nobuyuki Morioka** [*1] **Taman Narayan** [*1] **Sen Zhao** [*1]

Abstract

We propose new multi-input shape constraints across four intuitive categories: complements, diminishers, dominance, and unimodality constraints. We show these shape constraints can be checked and even enforced when training machine-learned models for linear models, generalized additive models, and the nonlinear function class of multi-layer lattice models. Real-world experiments illustrate how the different shape constraints can be used to increase explainability and improve regularization, especially for non-IID train-test distribution shift.

1. Introduction

Shape constraints are a classic way to characterize a function by whether its shape obeys certain properties (see e.g., Barlow et al. (1972); Groeneboom & Jongbloed (2014); Chetverikov et al. (2018)). The most popular shape constraint in machine learning is monotonicity (see, e.g., Archer & Wang (1993); Sill (1998); Howard & Jebara (2007); Minin et al. (2010); Gupta et al. (2016)). For example, suppose one is building a model to predict the price of a house. Then one might expect the price to be a monotonically increasing function with respect to square meters, holding fixed any value of the other inputs such as location.

Shape constraints are useful in machine learning because they aid interpretability: they provide ways to describe the function in terms of everyday relationships between its inputs and outputs that hold everywhere. Shape constraints are also semantically-meaningful regularizers that can improve generalization, especially when the distribution shifts between the train and test data (Canini et al., 2016; You et al., 2017). Shape constraints can also be imposed to ensure some notions of AI safety, societal norms and deontological ethics (Wang & Gupta, 2020).

[*]Equal contribution [1]Google Research, Mountain View, California, USA. Correspondence to: Sen Zhao <senzhao@google.com>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

Shape constraints historically have been defined in terms of how the function responds to changes in a single input, e.g. how house size affects its price, with modelers often applying many such constraints simultaneously on different inputs. Cotter et al. (2019a) extended this idea and proposed two new shape constraints that are defined on *pairs* of features that are complements with one another, such as a measurement and its precision. For example, a model that predicts whether a user will click on a web link might use as features a: the past click-through-rate (CTR) for that link, and b: the number of impressions that a was calculated from. Intuitively, the model output should be more sensitive to the past CTR a if the number of past impressions b used to calculate that CTR was higher, since the model can then trust the CTR to be more accurate. Cotter et al. (2019a) captured this type of feature interaction with two different mathematical formulations, which they termed *Edgeworth* and *trapezoid* shape constraints, and showed these two-feature shape constraints are broadly applicable whenever models include a pair of inputs where input a is a measurement correlated with the label y, and input b estimates how much one can trust input a.

Inspired by that work, here we address the broader question, “What multidimensional shape constraints can be specified that would be useful to machine-learning practitioners?" We identified four useful categories of multidimensional shape constraints: complements, diminishers, dominance, and unimodality. We review and propose new shape constraints spanning these four categories that we think are most valuable to practitioners, pictured in Fig. 1. We show that these shape constraints can be expressed as linear inequality constraints for some function classes including flexible lattice models, and thus can be efficiently checked for trained models. Moreover, we show that machine-learned models from these function classes can be *trained* to respect these shape constraints by minimizing empirical risk with the appropriate linear inequality constraints on the model parameters. We give intuitive examples of usage, and present experimental results on benchmark and real problems illustrating the use and effectiveness of multi-input shape constraints for interpretability and regularization of nonlinear models.

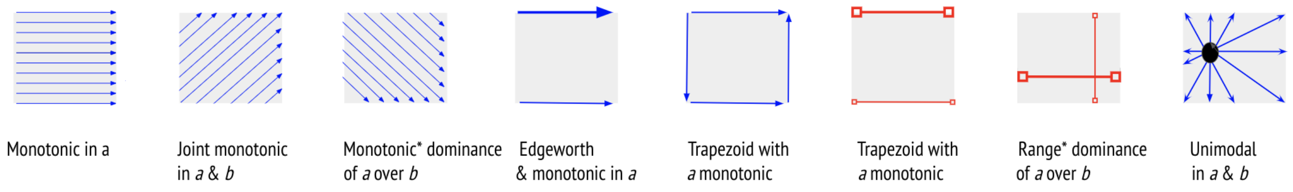
Monotonic in a
Joint monotonic in a & b
Monotonic* dominance of a over b
Edgeworth & monotonic in a
Trapezoid with a monotonic
Trapezoid with a monotonic
Range* dominance of a over b
Unimodal in a & b

Figure 1. Illustration of prior and proposed shape constraints that hold for each a-b slice of the feature space, where a is the horizontal feature and b is the vertical feature. Arrows (in blue) denote the function can only increase in that direction. The * on monotonic dominance and range dominance is to remind readers there are additionally monotonicity constraints in a and b that are not shown; for example, $f(a, b) = 5a + 3b$ is monotonically dominant in a over b, but also increasing in a and b. A bigger arrow denotes a steeper increase than a smaller arrow. The red lines with open-box-ends denote the range of the outputs over the line, with bigger open-boxes denoting a larger output range. As shown, the trapezoid constraint imposes a *twist* shape that can equivalently be expressed in terms of arrows or ranges, where for trapezoid the larger output range must also be a superset of the smaller output ranges.

2. Shape Constraint Properties

We considered many shape constraint definitions during our investigation, but we propose only the ones we think will be useful, based on the following criteria.

Intuitive: Can a data scientist easily understand the meaning of the constraint and apply it appropriately to regularize a model? Can an end user easily understand the meaning of the constraint making it helpful in explaining a model?

Unit-sensitivity: Is the constraint sensitive or dependent on how we measure or count inputs a and b? Unit sensitivity can be a good thing if it enables us to capture more domain knowledge, but can be a bad thing if it makes the constraint too fragile or too difficult to specify.

Composability: Deep models are formed by composing layers. If a shape constraint holds for $f(x)$, what needs to be true about $g : \mathbb{R} \to \mathbb{R}$ for the constraint to hold for the composition $g(f(x))$? For example, for monotonicity it is sufficient (though not necessary) for each layer to be monotonic with respect to an input a for the multi-layer model to be monotonic in a, making it possible to construct monotonic deep models (You et al., 2017).

Verification and Training: To be useful for aiding in the explanation of a machine-learned model, it must be computationally tractable to check if a machine-learned model satisfies the shape constraint. This requires that the shape constraint can be efficiently expressed in terms of the parameters of the model. Further, we would like to be able to *train* models that satisfy shape constraints. Thus, it is helpful if the shape constraint can be expressed as a set of *linear* inequality constraints on the parameters, as is true for monotonicity and diminishing returns constraints for lattice functions (You et al., 2017; Gupta et al., 2018), because then training can be done using empirical risk minimization *subject to* the necessary linear inequalities.

No Distribution Dependence: Shape constraints describe the *shape* of a function and thus their definitions do not depend on data, in contrast to *data-dependent constraints* which are defined in terms of expectations on sample distributions (see e.g., Mann & McCallum (2007); Zafar et al. (2017); Cotter et al. (2019b)).

Consider the goal of a “dominance" shape constraint that feature a is more important than feature b. There are already many feature importance metrics, and most of these can be readily checked given a trained model to interpret if feature A is more important than feature B, but they are not expressible as shape constraints. For example, one could try to capture prior knowledge that the model f should be more sensitive to feature A than feature B by requiring that the variance of $f(A, \bar{B}, X)$ is bigger than the variance of $f(\bar{A}, B, X)$, where $\bar{A}, \bar{B}$ denote the expectations of those inputs, and the variance is taken over the joint distribution $P_{A,B,X}$. That might be a useful constraint, but whether it holds depends on $P_{A,B,X}$, and thus is not a shape constraint. Similarly, measuring feature importance by the effect on validation accuracy if you drop that feature is data-dependent. In addition, for most prior feature importance metrics, it is difficult to train a model that guarantees that a specified feature a is more important than a specified feature b. We restrict our attention to *shape constraints* in this paper.

3. Notation and Preliminaries

For $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, 2, \dots, n\}$. For $\mathbf{x} \in \mathbb{R}^D$ let $\mathbf{x}[d]$ denote the dth entry of $\mathbf{x}$; if bounded we denote its bounds by $[\ell_d, u_d]$, and let $L_d(\mathbf{x})$ and $U_d(\mathbf{x})$ be vectors obtained from $\mathbf{x}$ by replacing its dth entry by ℓ_d or u_d, respectively. Let $\mathbf{e}_d \in [0, 1]^D$ denote the one-hot vector where $\mathbf{e}_d[j] = 1$ if $j = d$ and $\mathbf{e}_d[j] = 0$ for $j \neq d$.

We consider functions $f : \mathbb{R}^D \to \mathbb{R}$ and consider two features indexed by $a, b \in [D]$ as the features involved in the shape constraints. For notational simplicity, some of our shape constraint definitions will assume f is continuous and differentiable. However, all of the shape constraints can be applied to non-differentiable functions by changing derivatives to differences in the standard manner. For example, if f is increasing in input a, it means that $f(x[a]+\epsilon) \geq f(x[a])$ for all $x[a]$ and $\epsilon \geq 0$, and if f is differentiable we write that the slope of f in the direction $x[a]$ is nonnegative: $\partial f / \partial x[a] \geq 0$ for all x.

We will analyze the proposed shape constraints for the following function classes, which have all been previously shown to be particularly amenable to shape constraints. The proposed shape constraints might also be verifiable or trainable for other function classes, such as neural networks and decision trees, but we leave that as an open question.

Linear: $f(\mathbf{x}) = \alpha_0 + \sum_{d=1}^{D} \alpha_d \mathbf{x}[d]$.

Piecewise Linear Function (PLF): Given a set of $K+1$ knot-value pairs $\{(\xi_k, \beta_k)\}_{k=0}^{K} \subseteq \mathbb{R}^2$, where $\xi_0 < \ldots < \xi_K$, we denote by $\mathrm{PLF}(x; \{(\xi_k, \beta_k)\})$ the univariate piecewise linear function interpolating the $K+1$ knot-value pairs. For $k \in [K]$, let γ_k denote the slope of the kth linear segment: $(\beta_k - \beta_{k-1})/(\xi_k - \xi_{k-1})$.

Generalized Additive Model (GAM): $f(\mathbf{x}) = \sum_{d=1}^{D} f_d(\mathbf{x}[d])$, where $f_d : \mathbb{R} \to \mathbb{R}$ for $d \in [D]$. A GAM-PLF is a special case where each $f_d(x)$ is $\mathrm{PLF}(x; \{(\xi_{d,k}, \beta_{d,k})\})$, with slopes $\gamma_{d,k}$.

Lattice: A multidimensional interpolated look-up table, such that the function parameters are the function values sampled on a regular grid (Garcia & Gupta, 2009; Garcia et al., 2012). See Fig. 5 in the Appendix for an example. For $x \in \mathbb{R}^1$, a lattice is simply a PLF with uniform knots. With enough look-up table parameters, one can fit arbitrary bounded continuous functions. The look-up table structure is helpful for imposing shape constraints (Gupta et al., 2016; 2018; Cotter et al., 2019a).

A D-dimensional lattice of size $\mathbf{V} \in \mathbb{N}^D$ consists of a regular D-dimensional grid of look-up table vertices $\mathcal{M}_{\mathbf{V}} = \{0, 1, \ldots, \mathbf{V}[1]-1\} \times \ldots \times \{0, 1, \ldots, \mathbf{V}[D]-1\}$. Thus, $\mathbf{V}[d]$ is the number of vertices in the lattice in the dth dimension, and the grid has $\prod_{d=1}^{D} \mathbf{V}[d]$ vertices. We assume the input vector $\mathbf{x} \in \mathbb{R}^D$ has been bounded (clipped if necessary), shifted and scaled such that each $\mathbf{x}[d]$ lies in $[0, \mathbf{V}[d]-1]$. We define the cell of $\mathbf{x}$ to be the set of its 2^D neighboring grid vertices given by $\mathcal{N}(\mathbf{x}) = \{\lfloor \mathbf{x}[1] \rfloor, \lfloor \mathbf{x}[1] \rfloor + 1\} \times \ldots \times \{\lfloor \mathbf{x}[D] \rfloor, \lfloor \mathbf{x}[D] \rfloor + 1\}$.

For each vertex $\mathbf{v} \in \mathcal{M}_{\mathbf{V}}$, there is a corresponding look-up table parameter $\theta_{\mathbf{v}} \in \mathbb{R}$. The lattice function is produced by *interpolating* the grid's parameters over each cell. While there are many possible interpolation operators, here we consider only the popular multilinear interpolation:

$$f(\mathbf{x}) = \sum_{\mathbf{v} \in \mathcal{N}(\mathbf{x})} \theta_{\mathbf{v}} \Phi_{\mathbf{v}}(\mathbf{x}), \qquad (1)$$

where $\Phi_{\mathbf{v}}(\mathbf{x})$ is the linear interpolation weight on vertex $\mathbf{v}$ given by:

$$\Phi_{\mathbf{v}}(\mathbf{x}) = \prod_{d=1}^{D} \left(1 + (\mathbf{x}[d] - \mathbf{v}[d])(-1)^{I_{\mathbf{v}[d] = \lfloor \mathbf{x}[d] \rfloor}}\right), \qquad (2)$$

and I is the standard indicator function.

Calibrated Lattice: A generalization of a GAM-PLF, where instead of simply summing PLF's, the D PLF's enter a second layer that is a lattice (or ensemble of lattices) that captures nonlinear feature interactions (Gupta et al., 2016). The first layer of PLFs are called *calibrators* and are often *capped* such that $\beta_0 = 0$ and $\beta_K = 1$ to control the domain for the second layer. Ensembles of calibrated lattices perform similarly to random forests (Canini et al., 2016), and calibrator and lattice layers can be cascaded into *deep lattice networks* (You et al., 2017) that perform similarly to DNN's (You et al., 2017; Gupta et al., 2018; Cotter et al., 2019a).

Ensemble: An ensemble assumes the existence of T base models $\{f_t(x)\}$ where each $f_t : \mathbb{R}^D \to \mathbb{R}$. The base models may ignore some of the D inputs, essentially acting on only a subset of the features. The ensemble outputs the sum $f(x) = \sum_{t=1}^{T} f_t(x)$.

4. Dominance Shape Constraints

We propose new shape constraints to capture the prior knowledge or a policy that feature a should be more important than feature b. For example, in time series modeling, we often believe that recent information should be more important than past information at predicting future values. Or if a model is trained to predict CTR for a web link from feature a, the past CTR on that web link, and feature b, the past mean CTR for the whole website, then one might want to constrain the model to be more sensitive to a than b.

One option would be to require the model to always respond more strongly to changes in input a than to changes in input b, that is, for a differentiable model, require $\left|\frac{\partial f(x)}{\partial \mathbf{x}[a]}\right| \geq \left|\frac{\partial f(x)}{\partial \mathbf{x}[b]}\right|$ for all x. This constraint is easy to verify or guarantee for linear models, as it holds if the coefficient on feature a has a larger magnitude than the coefficient on feature b: $|\alpha_a| \geq |\alpha_b|$. For more flexible functions, we can say more if we *also* require the model to be monotonic with respect to both features a and b:

Monotonic Dominance: $\frac{\partial f(x)}{\partial \mathbf{x}[a]} \geq \frac{\partial f(x)}{\partial \mathbf{x}[b]} \geq 0$

This constraint can be expressed as a set of linear inequality constraints for GAM and lattice models too, as detailed in Table 1 with proofs in the Appendix. See Fig. 1 and Fig. 2 for illustrations.

Dominance and monotonic dominance are sensitive to the

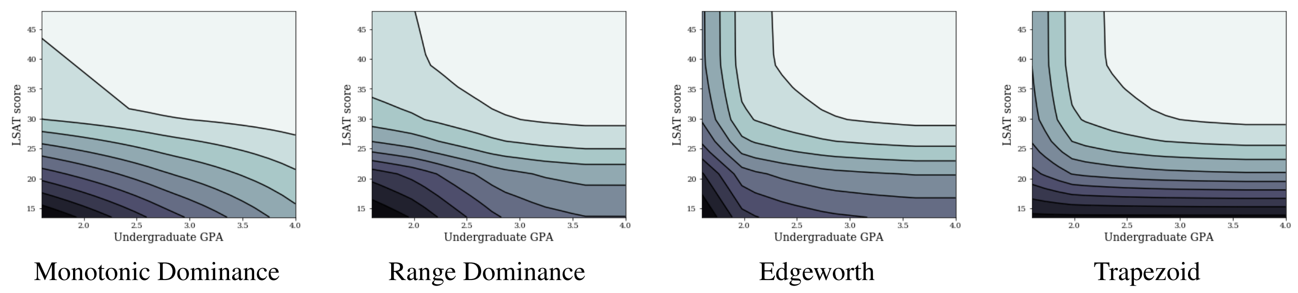

Figure 2. Contour plots illustrating the different functions produced by training a calibrated lattice model with different 2D shape constraints. All were trained on the same dataset to predict the probability that a person passes the bar given their GPA (horizontal axis) and LSAT test score (vertical axis) (Wightman, 1998) (see Appendix for details). A lighter contour color indicates a higher probability. All four plots were constrained to be monotonic in both GPA and LSAT (Wang & Gupta, 2020). **Left:** Trained with scaled monotonic dominance of LSAT over GPA, consistent with folklore that LSAT is the single most important factor (US News, 2018). **Middle Left:** Trained with range dominance of LSAT over GPA. **Middle Right:** Trained with Edgeworth complements constraint. **Far Right:** Trained with the trapezoid constraint that higher GPA means that the model should be more sensitive to the LSAT score.

units in which a and b are defined. For example, suppose one wishes to predict the increase in COVID cases after a large live event given a, the number of people who show up, and b, the number of bathroom stalls available per person at the venue. We expect the cases to be a monotonically increasing function of attendees a, and a monotonically decreasing function of the bathroom density b. One might wish to check or impose a dominance constraint that attendees a is more important than bathroom density b. However, clearly one-to-one is not the right trade-off between attendees-and-bathroom-density, so we also propose a scaled variant:

Scaled Monotonic Dominance: a monotonically dominates b with respect to scale $C \geq 0$ if $\frac{\partial f(x)}{\partial \mathbf{x}[a]} \geq C \frac{\partial f(x)}{\partial \mathbf{x}[b]} \geq 0$.

Given a trained model, we can check if that model satisfies scaled monotonic dominance for any C. Further, if we wish to train a model that satisfies scaled monotonic dominance, we simply need to architect the model with a first-layer that multiplies input a by $1/C$, and then enforce monotonic dominance on the rest of the layers. See Fig. 2 for an illustration.

For GAMs and two-layer calibrated lattice models, satisfying monotonic dominance requires the strong requirement that the slope of the 1-d transform for feature a must be steeper *everywhere* than the slope of the 1-d transform on feature b *anywhere*; that is quite restrictive on the 1-d feature transformations for a and b. For example, the model could not learn to transform a and b into an approximation of $\log a$ and $\log b$ using PLF's. To allow the model to learn flexible feature transformations for a and b, but still capture an intuition that feature a should be more important than feature b, we propose a mathematically more relaxed dominance shape constraint:

Range Dominance: For any input x the range of possible outputs $f(x)$ must be bigger if one varies input a than if one varies input b.

Returning to the example of predicting COVID cases from a live event, bound the domain of a to the number of people allowed at the venue (the capacity) $a_{\max}$, then range dominance of a over b would require that for any choice of $a_0 \in [1, a_{\max}]$ and any bathroom density $b_0 \in [0, 1]$, keeping the bathroom density b fixed at b_0 but evaluating the model for a set to the minimum number of attendees or the maximum number should change the predicted cases f more than keeping the number of attendees a fixed at a_0 and ranging the bathroom density from $b = 0$ to $b = 1$.

See Fig. 1 and Fig. 2 for illustrations. In Table 1 we give the mathematical definition and the sufficient conditions to achieve range dominance for different function classes (proofs in the appendix). Range dominance enables more flexible two-layer modeling than monotonic dominance. Both range dominance and monotonic dominance are not symmetric but are transitive.

5. Complements Shape Constraints

We first review and expand on two notions of complements from Cotter et al. (2019a), and then propose a third shape constraint that also captures complementarity. Table 6 in the Appendix summarizes definitions and properties. Proofs for all statements are in the Appendix.

In economics, *Edgeworth complementarity* says that a feature a and feature b are *complements* if the marginal value of feature a increases for larger values of feature b (Amir, 2005). For example, having more books a in a community's library is more valuable if the community literacy rate b is higher. Cotter et al. (2019a) proposed codifying this as the *Edgeworth shape constraint*. For a

Name	Monotonic Dominance	Range Dominance
Definition	$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[a]} \geq \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[b]} \geq 0$ for all $\mathbf{x}$	$\frac{\partial f(x)}{\partial \mathbf{x}[a]} \geq 0,\ \frac{\partial f(x)}{\partial \mathbf{x}[b]} \geq 0,$ and $f(U_a(\mathbf{x})) - f(L_a(\mathbf{x})) \geq f(U_b(\mathbf{x})) - f(L_b(\mathbf{x}))$
Linear	$\alpha_a \geq \alpha_b \geq 0$	$\alpha_a, \alpha_b \geq 0$ & $\alpha_a(u_a - \ell_a) \geq \alpha_b(u_b - \ell_b)$
GAM-PLF	$\gamma_{a,i} \geq \gamma_{b,j} \geq 0, \forall i \in [K_a], j \in [K_b]$	$\gamma_{a,i} \geq 0, \gamma_{b,j} \geq 0, \forall i \in [K_a], j \in [K_b]$ and $\beta_{a,K_a} - \beta_{a,0} \geq \beta_{b,K_b} - \beta_{b,0}$
Lattice	$\theta_{\mathbf{v}+\mathbf{e}_a+\mathbf{e}_b} \geq \theta_{\mathbf{v}+\mathbf{e}_a}, \theta_{\mathbf{v}+\mathbf{e}_a+\mathbf{e}_b} \geq \theta_{\mathbf{v}+\mathbf{e}_b}$, $\theta_{\mathbf{v}+\mathbf{e}_a} \geq \theta_{\mathbf{v}}, \theta_{\mathbf{v}+\mathbf{e}_b} \geq \theta_{\mathbf{v}}$, $\theta_{\mathbf{v}+\mathbf{e}_b} \leq \frac{\theta_{\mathbf{v}}+\theta_{\mathbf{v}+\mathbf{e}_a+\mathbf{e}_b}}{2} \leq \theta_{\mathbf{v}+\mathbf{e}_a}$ $\forall \mathbf{v} \in \mathcal{M}_{\mathbf{V}}, \mathbf{v}[a] \leq \mathbf{V}[a] - 2, \mathbf{v}[b] \leq \mathbf{V}[b] - 2$	$\theta_{\mathbf{v}+\mathbf{e}_a+\mathbf{e}_b} \geq \theta_{\mathbf{v}+\mathbf{e}_a}, \theta_{\mathbf{v}+\mathbf{e}_a+\mathbf{e}_b} \geq \theta_{\mathbf{v}+\mathbf{e}_b}$, $\theta_{\mathbf{v}+\mathbf{e}_a} \geq \theta_{\mathbf{v}}, \theta_{\mathbf{v}+\mathbf{e}_b} \geq \theta_{\mathbf{v}}$, $\theta_{U_a(\mathbf{v})} - \theta_{L_a(\mathbf{v})} \geq \theta_{U_b(\mathbf{v})} - \theta_{L_b(\mathbf{v})}, \forall \mathbf{v} \in \mathcal{M}_{\mathbf{V}}$
Cal. Lattice	Lattice constraints & GAM-PLF constraints	Lattice constraints & capped increasing calibrators
Composition	$\sum_t f_t(x)$ holds, and $g(f(A, B))$ holds if g is increasing	$\sum_t f_t(x)$ holds, and $g(f(A, B))$ holds if g is increasing and affine

Table 1. Dominance definitions, sufficient conditions, and properties for a dominates b. Proofs in the Appendix.

continuous twice-differentiable function:

Edgeworth: $\frac{\partial}{\partial \mathbf{x}[b]} \left(\frac{\partial f(x)}{\partial \mathbf{x}[a]} \right) \geq 0.$

Note that one can satisfy the Edgeworth shape constraint without f being monotonic in either feature a or b. For example, suppose a model predicts how fun a vacation will be, and that feature a represents the prevalence of mosquitos, and feature b represents packing bug spray. The more bugs there are, the more value there is in packing more bug spray. However, neither bugs nor carrying bug spray is a positive feature for a vacation.

Economists might say that Edgeworth is equivalent to *supermodularity* if it holds for the entire input space (Amir, 2005; Topkis, 1978), though the machine learning literature tends to use the term *supermodular* only for Boolean inputs. There is prior work in learning functions guaranteed to be supermodular (or submodular) on Boolean inputs by learning good weights on submodular component functions (Tschiatschek et al., 2014; Dolhansky & Bilmes, 2016). By Schwarz's Theorem, the Edgeworth shape constraint is symmetric, but it is not transitive.

The major downside to Edgeworth constraints is that they are difficult to guarantee for deep models because monotone transformations of Edgeworth functions are not necessarily still Edgeworth (Amir, 2005; Cotter et al., 2019a). So Cotter et al. (2019a) proposed a second more robust formalization of the notion of complements that they called a *trapezoid* shape constraint. The trapezoid constraint differs from Edgeworth in that its definition is *asymmetric* on a and b, and assumes that f is monotonic with respect to a. Trapezoid requires that for larger values of b, the *range* of possible outputs if you vary a must get bigger, forming a trapezoid of possible outputs (see Fig. 1). Trapezoid is useful when feature a is a measurement correlated with the training label, and b captures the precision or trustworthiness of the measurement a, and thus the higher the precision b the more the model output should vary as a varies. For continuous differentiable functions, trapezoid is equivalent to:

Trapezoid: $\partial f(\mathbf{x})/\partial \mathbf{x}[a] \geq 0$ and $\partial f(L_a(\mathbf{x}))/\partial \mathbf{x}[b] \leq 0$ and $\partial f(U_a(\mathbf{x}))/\partial \mathbf{x}[b] \geq 0$.

Trapezoid is nicer than Edgeworth for deep models because if f is trapezoid on $a - b$, then $g(f)$ is also trapezoid on $a - b$ if g is monotonically increasing (Cotter et al., 2019a).

We propose a third shape constraint that captures a different sense of complementarity: let *joint monotonicity* require the model to be monotonically increasing along the diagonal of every $a - b$ slice of the feature space, as illustrated in Fig. 1. For differentiable functions:

Joint Monotonicity: $\frac{\partial f(x)}{\partial \mathbf{x}[a]} + \frac{\partial f(x)}{\partial \mathbf{x}[b]} \geq 0.$

Joint monotonicity is weaker than requiring the function to be monotonic with respect to each of the constrained features individually. For example, suppose $f = 5ab - 2a + 9b$ models the profit of a hotel given a hotel beds and b hotel guests. Then the profit is monotonically increasing w.r.t. to guests b, but not w.r.t. beds a, but it is *jointly monotonic* along the diagonal of +1 bed for every +1 guest.

The following conditions are sufficient to make a model jointly monotonic on a and b (proofs in the Appendix; see Table 6 in the Appendix to compare and contrast these conditions with Edgeworth and trapezoid conditions):

Linear: $\alpha_a + \alpha_b \geq 0$.

GAM-PLF: $\gamma_{a,i} + \gamma_{b,j} \geq 0 \, \forall i \in [K_a], j \in [K_b]$.

Lattice: $\theta_{\mathbf{v}} \leq \frac{\theta_{\mathbf{v}+\mathbf{e}_a} + \theta_{\mathbf{v}+\mathbf{e}_b}}{2} \leq \theta_{\mathbf{v}+\mathbf{e}_a+\mathbf{e}_b}, \forall \mathbf{v} \in \mathcal{M}_{\mathbf{V}}$, $\mathbf{v}[a] \leq \mathbf{V}[a] - 2, \mathbf{v}[b] \leq \mathbf{V}[b] - 2$.

Calibrated Lattice: The lattice layer must satisfy the lattice conditions, and the calibrators for a and b must be increasing and affine with the same slope.

Composition: If f is jointly monotonic and g is increasing, then $g(f)$ is jointly monotonic. If each f_t is jointly monotonic, then so is an ensemble $\sum_t f_t$.

Mathematically, joint monotonicity is analogous to the monotonic dominance constraint we propose in Section 4, in that both constraints require the function to be monotonic along a diagonal direction of every $a - b$ subspace.

Joint monotonicity is unit-sensitive. For example, consider a model that predicts sale price of a house based on the size of the house (a) and the number of bedrooms (b), as well as other features. For a fixed house size a, having more bedrooms is not necessarily good for the sale price as the bedrooms become too small. But if we increase a by some size for each bedroom we add, we may be confident that *a larger house with more bedrooms will sell for more*. That is, we believe there exists some ray in the $a - b$ space along which the model should be monotonic. In such cases, we propose that a relaxed version of the joint monotonicity constraint may be more appropriate:

Scaled Joint Monotonicity: a is jointly monotonic with b with respect to scale C if $C\frac{\partial f(x)}{\partial \mathbf{x}[b]} + \frac{\partial f(x)}{\partial \mathbf{x}[a]} \geq 0$.

This constraint can be achieved, e.g., with a two-layer model by imposing the joint monotonicity constraints on the second layer, and by allowing the first layer to *only* calibrate a and b, where their calibrators are affine transformations. The slopes of these transformations can additionally be constrained to be sensible.

Two or more of these complements shape constraints may be useful at once. For example, a firm may believe that its profit satisfies scaled joint monotonicity in how many new machines it buys (a) and how much training it pays for (b); and Edgeworth in that the more training they buy, the more value they will get out of their machines.

These complements shape constraints can be extended to address larger sets of complements features, as occurs in such diverse settings as multi-item auctions (Roth, 2002) and manufacturing (Milgrom et al., 1991).

6. Diminisher Shape Constraints

We say that a feature b *diminishes* feature a if the model cares less about feature a if the value for feature b is bigger. Classic examples are substitutable features, like predicting the amount of time a person spends watching videos given the Boolean features a indicating if the person has a subscription to Netflix, and b indicating if the person has a subscription to HBO.

Diminishers also occur in utility models where there is a finite budget. For example, suppose a model predicts the happiness of customers if two products A and B are put on sale for a and b off respectively. Some customers cannot afford to buy both A and B. So the value of either being on sale is of less value given that the other is also on sale.

Diminishers do not need to be substitutes. For example, economists believe many humans value an absolute discount of a less if the sales price b is higher (Husemann-Kopetzky, 2018; Tversky & Kahneman, 1981). Or, in ranking coffee shops, the farther a coffee shop is from you, the less you tend to care about its star rating.

Mathematically, diminishers are simply complements in reverse, so we propose modeling them with analogous shape constraints. For example, *reverse Edgeworth*, which is equivalent to submodularity for Boolean inputs, and which for continuous differentiable functions can be expressed:

Reverse Edgeworth: $\frac{\partial}{\partial \mathbf{x}[b]}\left(\frac{\partial f(x)}{\partial \mathbf{x}[a]}\right) \leq 0$.

7. Unimodality Shape Constraints

The last category of shape constraints we consider are functions where, for any fixed value of the other features, the function is *unimodal* over some subset of features such that there exists a global minimizer and the function is increasing along any ray starting at the global minimizer:

Unimodal: Let $\mathcal{X} \subset \mathbb{R}^D$ be a convex set. If there exists some $x^* \in \mathcal{X}$ such that $f(x^* + \epsilon v) \geq f(x^* + \delta v)$ for any $\epsilon \geq \delta \geq 0$ and $v \in \mathcal{R}^D$ such that $x^* + \epsilon v \in \mathcal{X}$ and $x^* + \delta v \in \mathcal{X}$, then f is unimodal on $\mathcal{X}$.

Examples of unimodal functions are the optimal location on the soccer field for a free kick, and progesterone levels over the menstrual cycle (Dunson, 2005).

Stout (2008) fits *1-dimensional* unimodal functions using

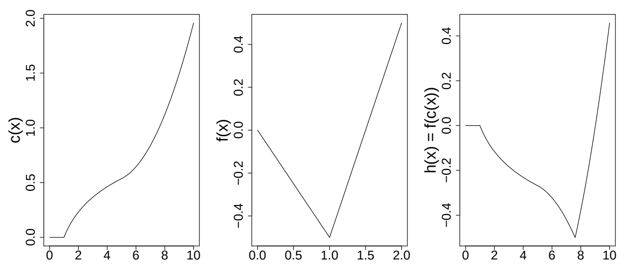

Figure 3. Example two-layer unimodal function. **Left:** The first layer is a 1D monontonically increasing PLF $c(x)$. **Middle:** The second layer is a 1D unimodal PLF $f(x)$ with its minimizer fixed at 1.0. **Right:** The composition $h(x) = f(c(x))$ is unimodal with an arbitrary minimizer (here, roughly 7.8)

two separate isotonic regressions. Similarly, Köllmann et al. (2014) models *1-dimensional* unimodal functions using Bernstein–Schoenberg splines, which can also be trained with linear inequality constraints. Dunson (2005) considered this problem under the name *umbrella-order restriction* for the means of Bayesian models. All of these strategies require either already knowing the minimizer, or require consideration of all possible candidates for the minimizer.

A special case of unimodality is jointly convex functions, which can be achieved by summing jointly convex basis functions (Kim et al., 2004; Magnani & Boyd, 2009). DNN's with ReLU activations can be constrained to be jointly convex over a subset of features (Dugas et al., 2009; Amos et al., 2017). GAMs have also been constrained for convexity (Pya & Wood, 2015; Chen & Samworth, 2016), as have calibrated lattice models (Gupta et al., 2018).

Here we show that one can verify or train *multi-dimensional* unimodal functions, without requiring *convexity*, by using two-layer models.

First, consider the simple 1D case. We propose a two-layer model, where each layer is a PLF. For the second layer PLF, we impose unimodality with a pre-fixed minimizer x^* at the center knot $k = K/2 + 1$ for odd K, then constrain the PLF to be increasing to the right of x^* and decreasing to the left of x^*. Then let the first layer PLF be an arbitrary monotonic 1D function, which nonlinearly stretches the input domain of the second layer, so that the x^* of the second layer can be achieved by any raw input value to the first layer PLF. See Fig. 3 for an example.

For a *multidimensional* lattice function, one can check if it is unimodal by finding its minimal look-up table parameter value x^* and checking if for every look-up table vertex v, the directional derivative in direction $v - x^*$ is non-negative. The training strategy is similar to the 1D case: make the first layer monotonically increasing calibrators to give the model flexibility to place the global minimizer, and make the second layer a lattice layer whose look-up table must have an odd number of knots in each dimension, and constrain the center knot to be the global minimizer x^*. Then, sufficient (but not necessary) constraints on the second layer are that *every* edge in the lattice needs to be increasing in the direction away from the center. Full details, proofs and illustrations can be found in the Appendix.

8. Training With Multi-d Shape Constraints

Let $\{x_i, y_i\}$ be a train set with $i = 1, \dots, n$ training example pairs where $x_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$ for regression problems or $y_i \in [-1, 1]$ for classification problems. Let ℓ be a loss of interest, e.g. squared error or logistic loss. Let θ denote the parameters of any $f \in \mathcal{F}$, where $\mathcal{F}$ is a function class where the desired shape constraints can be expressed as linear inequalities on the parameters θ (as we have shown is the case for the proposed shape constraints for linear, GAM-PLF, lattice, calibrated lattice, ensembles, and $g(f())$ multi-layer models). Collect the linear inequalities corresponding to the set of desired shape constraints into the matrix equation $S^T\theta \geq 0$. Then train by minimizing the empirical risk subject to the linear inequality constraints:

$$\arg\min_{\theta} \sum_{i=1}^{n} l((f(x_i;\theta), y_i) \quad \text{such that } S^T\theta \geq 0. \tag{3}$$

To solve (3), we use projected stochastic gradient descent in TensorFlow, and the TensorFlow Lattice 2.0 library's PLF layers and lattice layers. After each minibatch, we project the model parameters toward the constraint set using ten steps of Dykstra's algorithm, with a longer final projection to ensure constraint satisfaction to within numerical precision.

Open-source code has been pushed to the TensorFlow Lattice 2.0 library and can be downloaded at github.com/tensorflow/lattice. Train time with 2D shape constraints was 10-20% longer than train time without 2D shape constraints (data in the Appendix).

9. Experiments

We present experiments on public and proprietary real-world problems illustrating training with the proposed shape constraints, including the first public experimental evidence with Edgeworth constraints (Cotter et al. (2019a) only presented experiments with trapezoid constraints).

For all experiments, we used the default ADAM stepsize of .001, ran the optimization of (3) until train loss converged, and used squared error as the training loss l in (3). All models that use PLFs use 10 keys $K_d = 10$ for each PLF, fixed before training to the train data quantiles. For each metric, we report a 95% margin of error, computed under a Gaussian assumption as plus-or-minus 1.96 times the estimated standard error.

9.1. Weekly Sales Transactions (Regression)

We compare models that forecast next week's sales based on sales in the past weeks (Kaggle, 2020c). The dataset contains purchases of 811 products with a feature for the normalized transactions for each of the last 52 weeks. We train lattice models using the last $K \in \{2, \ldots, 10\}$ weeks of transactions as features to predict the transactions of the most recent week.

Results in Fig. 4 are averaged over 100 random 80-20 train/test splits. The figure shows that the unconstrained calibrated lattice model overfits to the data as we use a longer transaction history as features, reflected by the fact that the Train MSE improves yet the Test MSE deteriorates. Imposing monotonicity constraints on past history (i.e. any increase in prior sales should only ever increase our prediction) helps alleviate the issue, and imposing dominance constraints that encode our intuition that recent weeks should matter more than distant weeks helps even more.

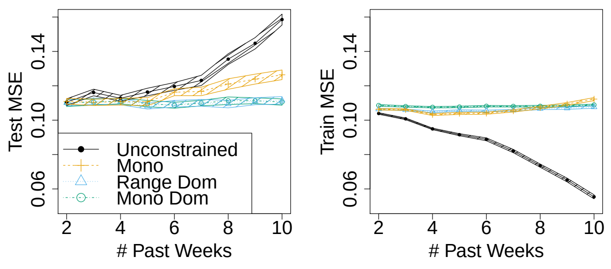

Figure 4. The Test and Train MSE of lattice models with different constraints on the Weekly Sales Transactions data. The x-axis shows what happens if you include more past weeks as features in the model. The shaded area shows the 95% confidence region of the performance.

9.2. Mount Rainier Climbing (Regression)

In this experiment, we compare the performance of different models in predicting the daily success probability of climbing Mount Rainier (Kaggle, 2020a) for 464 days based on five features: temperature, battery voltage, relative humidity, solar radiation and the month of the climbing day. On days people didn't climb (mostly during winter seasons), we labeled as having success probability zero.

The results, shown in Table 2, are averaged over 100 random 80-20 train/test splits. The calibrated lattice model does better than the GAM-PLF, suggesting that feature interactions are important. Preliminary analysis of the marginals of the data suggested to us that climbing success might be unimodal in humidity and month, consistent with a story that average-humidity summer days are the best time to climb Mt. Rainier. In fact, imposing 2D unimodality on humidity-month slices lowered Test MSE.

Table 2. Results of different models on the Mount Rainer Climbing dataset from Kaggle.

Model	Train MSE	Test MSE
GAM-PLF	0.029 ± 0.0004	0.032 ± 0.0018
Calib. Lattice	0.022 ± 0.0004	0.030 ± 0.0019
Above + Unimodal	0.023 ± 0.0004	0.027 ± 0.0018

9.3. Play Store App Installs Dataset (Regression)

The Google Play Store Apps dataset (Kaggle, 2020b) contains various pieces of information about individual apps. We trained a calibrated lattice model on $D = 5$ features: log real-valued number of reviews, average rating, price, download size, and categorical content rating. The label was bucketed number of times the app has been installed: we used the log of the lower bound of each interval as our label, and filtered for apps that have been installed at least once, leaving $N = 10,285$ examples.

We ran two sets of experiments. The first set used a non-IID train/test split based on a 6th piece of information, the app category: we trained on the most common category "Family"(18 percent of samples), and tested on the other app categories (82 percent of samples). The second set of experiments used 80/20 train/test IID random splits, and used the app category as a 6th feature.

We imposed three common sense monotonicity constraints that installs are increasing number of reviews, increasing in average rating, and decreasing in price. We tested an Edgeworth constraint that number of reviews and average rating are complements, with the motivation that lots of reviews are a better predictor of installs if the ratings are good, and high ratings are a more useful and reliable predictor of installs if there are sufficiently many reviews. We also tested range dominance on different feature interactions to better understand the underlying data, and found that the best dominance constraints were that number of reviews dominates price, and price dominates average rating, which are the ones reported in Tables 3 and 4. We also report imposing all the shape constraints of the above rows.

For the non-IID experiments, Table 3 shows that the constrained models never hurt Test MSE, and the dominance constraints improved the Test MSE. Further, one can explain what the constrained models are doing in terms of their shape constraints, which makes them easier to understand and debug. As expected, train MSE's are a little higher for the constrained models.

Table 4 shows the results averaged over 100 random IID

Table 3. Results on the Play Store dataset from Kaggle for 100 retrains of a fixed non-IID train/test split and $D = 5$ features (randomness due to initialization and optimization).

Model	Train MSE	Test MSE
Unconstrained	1.161 ± 0.0003	1.279 ± 0.0028
Mono.	1.198 ± 0.0009	1.276 ± 0.0028
Mono. + Edge.	1.199 ± 0.0009	1.279 ± 0.0032
Mono. + Dom.	1.189 ± 0.0003	1.250 ± 0.0020
All Constraints	1.189 ± 0.0003	1.252 ± 0.0020

80/20 train/test splits with $D = 6$ features. Because the train/test is IID, and because there is more train data, there is less need to regularize, and the Test MSE's are all statistically indistinguishable from the unconstrained model. Still, the constrained models can be explained in terms of their shape constraint properties, which also make these models more predictable and thus easier to debug.

Table 4. Results on the Play Store dataset from Kaggle for 100 different IID 80/20 train/test splits and $D = 6$ features.

Model	Train MSE	Test MSE
Unconstrained	1.232 ± 0.0028	1.234 ± 0.0112
Mono.	1.230 ± 0.0031	1.236 ± 0.0119
Mono. + Edge.	1.233 ± 0.0025	1.229 ± 0.0091
Mono. + Dom.	1.236 ± 0.0029	1.237 ± 0.0097
All Constraints	1.236 ± 0.0028	1.246 ± 0.0093

9.4. User Intent Prediction (Classification)

We predict the user's intent given a query between two classes; the data is Google proprietary. We use a non-IID split: we train on 300K labeled samples from the U.S., and test on 350K from other countries. Of the $D = 19$ features, based on domain expertise, a priori 11 features were constrained to be monotonic, 4 feature pairs were chosen for range dominance, 11 feature pairs were chosen as complements and 3 feature pairs were chosen as diminishers, for which we applied either Edgeworth/reverse Edgeworth or trapezoid/reverse trapezoid constraints. The model was an ensemble of 50 calibrated lattices with each base model seeing 6-10 of the $D = 19$ possible features (Canini et al., 2016). We did not try joint monotonicity or monotonic dominance for this problem as they force the calibrators to be linear, which we knew would reduce the model flexibility too much given the large train set. All models were trained for 100 epochs; longer training and smaller lattices show similar trends but with worse Test MSE - see the Appendix.

While this is a classification problem, we trained with squared-error loss to produce stable prediction scores; Table 5 shows the Train and Test MSE averaged over 5 runs. As expected, Train MSE increases slightly as more constraints are added, but the monotonic models with dominance, Edgeworth, and both dominance and Edgeworth constraints have slightly better Test MSE compared to the monotonic only baseline. In contrast, trapezoid appears to hurt performance by Test MSE a little. The main advantage is that applying these constraints guarantees the model is behaving in reasonable and explainable ways, and its greater predictability makes debuggging model errors easier.

Table 5. Results on User Intent Prediction for 5 retrains of a fixed non-IID train/test split (randonmess due to initialization and optimization).

Model	Train MSE	Test MSE
Mono.	0.666 ± 0.0001	0.752 ± 0.0002
Mono. + Dom.	0.669 ± 0.0001	0.751 ± 0.0002
Mono. + Edge.	0.670 ± 0.0001	0.751 ± 0.0001
Mono. + Edge. + Dom.	0.673 ± 0.0001	0.750 ± 0.0002
Mono. + Trap.	0.686 ± 0.0001	0.764 ± 0.0002
Mono. + Trap. + Dom.	0.695 ± 0.0002	0.766 ± 0.0001

10. Conclusions

We compared and contrasted new and recent definitions for multidimensional shape constraints. Shape constraints play two key roles for interpretability: if applied, we can explain the feature interactions, and we can test different 2D shape constraints and see the effect to understand whether certain interactions fit or fight the data. Experimentally, we showed applying relevant shape constraints can regularize models and improve test metrics, particularly in non-IID settings.

We found that a key differentiator between the definitions is how easy the different constraints are to apply to two-layer models for flexible modeling. The proposed joint monotonicity and monotonic dominance satisfy intellectually for their simplicity: constrain the model to be increasing along a direction in the 2D a-b subspace (see Fig. 1). However, we found these two constraints difficult to use in practice because they overconstrained the calibrator layer. We found Edgeworth to be the most broadly applicable and useful 2D shape constraint for use with ensembles of two-layer calibrated lattices (which have similar flexibility as random forests (Canini et al., 2016)), and easy to explain to non-experts. We saw trapezoid performed worse than Edgeworth, probably due to its greater restrictions on the calibrator layer. We found range dominance to also be useful and easy to apply. The unimodality shape constraints were helpful, and are promising for training functions that one wants to minimize (as in Amos et al. (2017)), but more research is needed.

References

Amir, R. Supermodularity and complementarity in economics: An elementary survey. *Southern Economic Journal*, 71(3):636–660, 2005.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. *ICML*, 2017.

Archer, N. P. and Wang, S. Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. *Decision Sciences*, 24(1):60–75, 1993.

Barlow, R. E., Bartholomew, D. J., and Bremner, J. M. *Statistical inference under order restrictions; the theory and application of isotonic regression*. Wiley, 1972.

Canini, K., Cotter, A., Fard, M. M., Gupta, M. R., and Pfeifer, J. Fast and flexible monotonic functions with ensembles of lattices. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Chen, Y. and Samworth, R. J. Generalized additive and index models with shape constraints. *Journal Royal Statistical Society B*, 2016.

Chetverikov, D., Santos, A., and Shaikh, A. M. The econometrics of shape restrictions. *Annual Review of Economics*, 2018.

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

Cotter, A., Jiang, H., Wang, S., Narayan, T., Gupta, M. R., You, S., and Sridharan, K. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *JMLR*, 2019b.

Dolhansky, B. and Bilmes, J. Deep submodular functions: Definitions and learning. *NeurIPS*, 2016.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., and Garcia, R. Incorporating functional knowledge in neural networks. *JMLR*, 2009.

Dunson, D. A transformation approach for incorporating monotone or unimodal constraints. *Biostatistics*, 2005.

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

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

Groeneboom, P. and Jongbloed, G. *Nonparametric estimation under shape constraints*. Cambridge Press, New York, USA, 2014.

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

Gupta, M. R., Bahri, D., Cotter, A., and Canini, K. Diminishing returns shape constraints for interpretability and regularization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Howard, A. and Jebara, T. Learning monotonic transformations for classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

Husemann-Kopetzky, M. *Handbook on the Psychology of Pricing*. Pricing School Press, New York, USA, 2018.

Kaggle. Kaggle Mount Rainier Weather and Climbing Data, 2020a. URL `https://www.kaggle.com/codersree/mount-rainier-weather-and-climbing-data`.

Kaggle. Kaggle Google Play Store Apps Data, 2020b. URL `https://www.kaggle.com/lava18/google-play-store-apps`.

Kaggle. Kaggle Weekly Sales Transactions, 2020c. URL `https://www.kaggle.com/crawford/weekly-sales-transactions`.

Kim, J., Lee, J., Vandenberghe, L., and Yang, C. Techniques for improving the accuracy of geometric-programming based analog circuit design optimization. *Proc. IEEE International Conference on Computer-aided Design*, 2004.

Köllmann, C., Bornkamp, B., and Ickstadt, K. Unimodal regression using Bernstein–Schoenberg splines and penalties. *Biometrics*, 70(4), 2014.

Magnani, A. and Boyd, S. P. Convex piecewise-linear fitting. *Optimization and Engineering*, 2009.

Mann, G. S. and McCallum, A. Simple, robust, scalable semi-supervised learning with expectation regularization. *ICML*, 2007.

Milgrom, P., Qian, Y., and Roberts, J. Complementarities, momentum, and the evolution of modern manufacturing. *The American Economic Review*, 1991.

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

Pya, N. and Wood, S. N. Shape constrained additive models. *Statistics and Computing*, 2015.

Roth, A. E. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica*, 70(4):1341–1378, 2002.

Sill, J. Monotonic networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 1998.

Stout, Q. Unimodal regression via prefix isotonic regression. *Computational Statistics and Data Analysis*, 2008.

Topkis, D. Minimizing a submodular function on a lattice. *Operations Research*, 78(2):302–321, 1978.

Tschiatschek, S., Iyer, R., Wei, H., and Bilmes, J. Learning mixtures of submodular functions for image collection summarization. *NeurIPS*, 2014.

Tversky, A. and Kahneman, D. The framing of decisions and psychology of choice. *Science*, 1981.

US News. 5 traits that help people get into top law schools, 2018. URL `https://www.usnews.com`.

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

Wightman, L. LSAC national longitudinal bar passage study. *Law School Admission Council*, 1998.

You, S., Canini, K., Ding, D., Pfeifer, J., and Gupta, M. R. Deep lattice networks and partial monotonic functions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Zafar, M. B., Valera, I., Rodriguez, M. G., and Gummadi, K. P. Fairness constraints: Mechanisms for fair classification. In *AIStats*, 2017.