Learning from Uncertain Data: From Possible Worlds to Possible Models

Jiongli Zhu[1] Su Feng[2] Boris Glavic[3] Babak Salimi[1]

[1]University of California, San Diego [2]Nanjing Tech University [3]University of Illinois, Chicago

Abstract

We introduce an efficient method for learning linear models from uncertain data, where uncertainty is represented as a set of possible variations in the data, leading to predictive multiplicity. Our approach leverages abstract interpretation and zonotopes, a type of convex polytope, to compactly represent these dataset variations, enabling the symbolic execution of gradient descent on all possible worlds simultaneously. We develop techniques to ensure that this process converges to a fixed point and derive closed-form solutions for this fixed point. Our method provides sound over-approximations of all possible optimal models and viable prediction ranges. We demonstrate the effectiveness of our approach through theoretical and empirical analysis, highlighting its potential to reason about model and prediction uncertainty due to data quality issues in training data.

1 Introduction

This paper addresses the challenges of learning from uncertain datasets by employing the framework of *possible world semantics*, a well-established concept in AI and database theory [59, 23, 41, 28]. In this approach, uncertainty in a dataset $\boldsymbol{D}$ is conceptualized through a collection of possible datasets $\{\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots\}$, each representing a potential state of the real world, reflecting variations due to missing entries, errors, inconsistencies, and biases. Given this framework and a learning algorithm, our objective is to construct a set of models $\{f_1, f_2, \ldots\}$, where each model f_i is trained on a corresponding potential dataset $\boldsymbol{D}_i$. This method, that we implement in a system called ZORRO (*ZO*notope-based *R*obustness Analysis for *R*egression with *O*ver-approximations), allows for a thorough evaluation of how data uncertainties affect the robustness, reliability, and fairness of models in predictive modeling and statistical inference, particularly in scenarios where the ground truth is unidentifiable, necessitating consideration of all possible dataset variations.

While the framework of possible world semantics is essential for modeling dataset uncertainties, it poses significant challenges due to the potentially infinite number of scenarios each dataset might represent. Exploring every possibility and training a model for each is impractical. The concept of *model multiplicity*, which highlights situations where models with similar accuracy differ in individual predictions, has gained traction [7, 42], yet it primarily focuses on competing models without addressing the full range of dataset variations. Similarly, *dataset multiplicity* introduced in [47] recognizes data variations due to uncertainty but [47] only proposed a solution for linear models with label errors. Our approach expands these ideas by using possible world semantics to systematically manage uncertainty across all features and labels, thus creating a comprehensive framework for evaluating model robustness amid data uncertainties.

To address the challenges of learning from uncertain datasets, we employ the method of *abstract interpretation* [13]. Utilizing zonotopes—a type of convex polytope well-suited for compactly representing high-dimensional data spaces [74, 4]—we over-approximate the set of all possible dataset variations. This framework allows for the simultaneous symbolic execution of gradient descent

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3396

across all possible datasets, compactly over-approximating all possible optimal model parameters as a zonotope. We demonstrate that for linear regression with ℓ_2 regularization (Ridge), our method admits a non-trivial closed-form solution. The zonotope representation of model parameters enables efficient inference, facilitating reasoning about the range of possible predictions or of specific parameters.

Contributions. The key contributions of this research are:

1. We introduce an abstract gradient descent algorithm for learning linear regression models from uncertain data. This method over-approximates data variations using zonotopes and symbolically executes gradient descent on all possible datasets concurrently. We define and prove the existence of a fixed point that soundly over-approximates all potential models.
2. Symbolic execution generates intractable polynomial zonotopes for gradients due to non-linear terms and monomial growth that is exponential in the number of iterations. We use linearization and order reduction to compactly over-approximate these polynomial zonotopes using linear zonotopes at each step, introducing an efficient version of abstract gradient descent.
3. The efficient version, however, does not guarantee a fixed point for arbitrary order reduction techniques. To address this, we develop advanced order reduction techniques that ensure fixed points and provide a non-trivial closed-form solution for these fixed points in ridge regression.
4. We implement our approach in a system called ZORRO and use it to evaluate the impact of data uncertainty on linear regression models. Our empirical results and analytical solutions validate the effectiveness of our approach, demonstrating its efficacy in computing prediction ranges and verifying robustness.

Related Work. Predictive multiplicity has shown that a single dataset can produce multiple optimally fitted models due to variations in training processes [8, 11, 44, 66, 69, 9, 15, 71, 42] or modifications to training parameters such as random seeds, data ordering, and hyperparameters. For predictive multiplicity due to missing data, Khosravi et al.[34] address the issue using a probabilistic method that computes expected predictions for all possible imputations. Our approach can be seen as an extreme case of multiple imputation [62, 61], where we consider all possible data variations rather than just a few plausible scenarios. Meyer et al. [47] recently introduced the concept of dataset multiplicity, using possible world semantics to model how uncertain, biased, or noisy training data can lead to predictive multiplicity. However, their focus is on uncertainty in training labels, and they use interval arithmetic for over-approximation of prediction intervals for linear regression. In contrast, our approach handles arbitrary uncertainty in features and labels during both training and testing using zonotope-based learning for over-approximation of prediction ranges and model parameters. We show that interval arithmetic fails to provide tight prediction ranges even for uncertainty in labels.

Our work is broadly related to robust model learning, which ensures robustness against data quality issues such as attacks [31, 76, 70, 60, 55, 52, 29]. Distributional robustness [6, 53, 65] studies model reliability against varying data distributions, while robust statistics [19, 18] examines model performance under outliers or data errors. Our approach provides exact provable robustness guarantees by exploring the entire range of models under extreme dataset variations, which is crucial for individual-level predictions and reasoning about the robustness of specific parameters.

Our work is also related to robustness certification, which certifies ML models' robustness against data perturbations and uncertainties [27, 68, 50]. These efforts mainly focus on test-time robustness, validating predictions for inputs in the vicinity of a test sample. In contrast, we address training-time robustness, considering the effects of possible datasets on training models. Closest to our approach is the work by Meyer et al. [46] for decision trees and Karlas et al. [33] for nearest neighbor classifiers. We use zonotopes to over-approximate prediction ranges for linear regression, generating robustness certificates. While zonotopes have been used for test-time robustness [49, 20, 22, 51], our work is the first to apply zonotopes for training-time robustness for an iterative learning algorithm.

Approaches for uncertainty quantification (UQ) aim to understand the range of outcomes a model may produce using Bayesian methods, ensembling, conformal prediction, and bootstrapping [54, 43, 35, 17, 16]. UQ focuses on epistemic and aleatoric uncertainty, stemming from insufficient data, noisy data, or uncertainty about the model parameters, and does not account for uncertainty due to systematic data quality issues, such as non-random data errors or missing values [25], which induce a multiplicity of possible datasets. In this case, UQ methods might underestimate the uncertainty as they rely on critical assumptions. Bayesian methods, for instance, require correctly specified priors to accurately model uncertainty, often failing under conditions with unknown or erroneous

priors [73, 67, 72], while conformal prediction (CP) assumes data exchangeability—a condition that breaks down when data errors are systematic [21, 75]. In contrast, our approach addresses this distinct challenge by computing sound over-approximations that guarantee complete coverage of potential predictions across all variations of the dataset. This sound coverage is essential in high-stakes settings such as evaluating the robustness of predictive models for medical use.

2 Notation, Problem Formulation and Background

Denote a training dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ with $\boldsymbol{X} = [\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$ as the matrix of features, and $\boldsymbol{y} = [y_1 \quad \cdots \quad y_n]^T \in \mathbb{R}^n$ as the corresponding ground truth labels. Let $f(\boldsymbol{x}; \boldsymbol{w})$ be a model parameterized by $\boldsymbol{w} \in \mathbb{R}^p$ that maps an input data point to a label. A learning algorithm $\mathcal{A}$ maps a training dataset $\boldsymbol{D}$ to the parameters of the trained model, $\boldsymbol{w}^* = \mathcal{A}(\boldsymbol{D})$. Given a test dataset $\mathbf{X}_{\text{test}} \in \mathbb{R}^{n \times d}$, for any test sample $\boldsymbol{x}$ from $\mathbf{X}_{\text{test}}$, the function f computes a prediction $\hat{y} = f(\boldsymbol{x}; \boldsymbol{w}^*)$.

2.1 Learning Possible Models from Possible Worlds

We use possible world semantics to represent the uncertainty in a dataset $\boldsymbol{D}$.

Definition 2.1 (Possible Datasets)**.** *Given an uncertain dataset $\boldsymbol{D}$, the uncertainty in $\boldsymbol{D}$ can be represented by a set of possible datasets $\boldsymbol{D}^{\odot}$: $\boldsymbol{D}^{\odot} = \{\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots\}$.*

Each dataset $\boldsymbol{D}_i \in \boldsymbol{D}^{\odot}$ is a "possible world", i.e., a hypothetical variation of the dataset $\boldsymbol{D}$ that could potentially exist in the real world based on our knowledge about the uncertainty in $\boldsymbol{D}$.

Example 2.2. *Consider an e-commerce dataset $\boldsymbol{D}$ where some product price is missing, meaning the exact price is unknown. Using possible world semantics, we represent this uncertainty with a set of possible datasets $\boldsymbol{D}^{\odot}$, each containing a possible clean price, which could be obtained from prices of the same items on the market.*

In App. D, we discuss construction methods for common data quality issues. Our goal is to efficiently construct the set of all possible models from uncertain data and understand their behavior in making predictions.

Definition 2.3 (Possible Models and Prediction Range)**.** *Given a set of possible datasets $\boldsymbol{D}^{\odot}$ associated with an uncertain dataset $\boldsymbol{D}$, the possible models, denoted $f^{\odot}$, are obtained by applying the learning algorithm $\mathcal{A}$ to each training dataset $\boldsymbol{D}_i$ within $\boldsymbol{D}^{\odot}$ to obtain the set of all possible optimal model parameters $\boldsymbol{w}^{\odot *}$, i.e.,*

$$\boldsymbol{w}^{\odot *} = \{\boldsymbol{w}_i^* \mid \boldsymbol{w}_i^* = \mathcal{A}(\boldsymbol{D}_i), \boldsymbol{D}_i \in \boldsymbol{D}^{\odot}\}$$

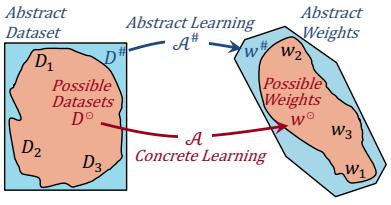
For a test data point $\boldsymbol{x}$, the viable prediction range $V(\boldsymbol{x})$ is defined as the interval between the least upper bound and the greatest lower bound of the outputs produced by all models in $f^{\odot}$, i.e.,

$$V(\boldsymbol{x}) = \left[\inf_{\boldsymbol{w}^* \in \boldsymbol{w}^{\odot *}} f(\boldsymbol{x}, \boldsymbol{w}^*), \sup_{\boldsymbol{w}^* \in \boldsymbol{w}^{\odot *}} f(\boldsymbol{x}, \boldsymbol{w}^*)\right]$$

This prediction range quantifies the minimum and maximum predictions that can be expected for $\boldsymbol{x}$, highlighting the variability in model outputs due to differences in the training data. Our framework supports uncertainty in training and test data. We discuss test data uncertainty in App. F.3.

2.2 Sound Approximation of Possible Models with Abstract Interpretation

The set of all possible datasets associated with uncertain data can be intractable. We use *abstract* interpretation [12] to over-approximate sets of elements of a concrete domain $\mathbb{D}$ (the training data and model weights) with elements from an abstract domain $\mathbb{D}^{\sharp}$. Specifically, we use the abstract domain of zonotopes, a type of convex polytope, to over-approximate the possible datasets $\boldsymbol{D}^{\odot}$ using a zonotope $\boldsymbol{D}^{\sharp}$ that has a compact symbolic representation. Instead of applying the learning algorithm $\mathcal{A}$ to each possible dataset to compute all possible optimal model parameters $\boldsymbol{w}^{\odot *}$, we develop an abstract learning algorithm $\mathcal{A}^{\sharp}$ that operates directly on the abstract domain of zonotopes. Given $\mathbf{D}^{\sharp}$, $\mathcal{A}^{\sharp}$ generates a zonotope $\boldsymbol{w}^{\sharp} = \mathcal{A}^{\sharp}(\mathbf{D}^{\sharp})$ that over-approximates $\boldsymbol{w}^{\odot *}$ (demonstrated in the graph on the right). Intuitively, this represents the symbolic execution of the learning algorithm across all possible datasets simultaneously.



https://doi.org/10.52202/079017-3396

Definition 2.4 (Abstract Domain)**.** *Let $\mathbb{D}$ be a concrete domain. An* abstract domain *for $\mathbb{D}$ is a set $\mathbb{D}^\sharp$ paired with two functions:*

$$\textbf{Abstraction } \alpha : \mathcal{P}(\mathbb{D}) \to \mathbb{D}^\sharp \qquad \textbf{Concretization } \gamma : \mathbb{D}^\sharp \to \mathcal{P}(\mathbb{D})$$

which satisfy the following condition for any subset $S \subseteq \mathbb{D}$: $\gamma(\alpha(S)) \supseteq S$. Two abstract elements d_1 and d_2 are equivalent, written as $d_1 \simeq_\sharp d_2$, if $\gamma(d_1) = \gamma(d_2)$.

Def. 2.4 ensures that the abstract element $\alpha(S)$ associated with a set S through application of the abstraction function α encodes an over-approximation of S. We will use an abstract element $\mathbf{D}^\sharp = \alpha(\boldsymbol{D}^\odot)$ to over-approximate the possible worlds of an uncertain training dataset $\boldsymbol{D}^\odot$. We discuss abstraction functions α for specific types of training data uncertainty in App. D.

Definition 2.5 (Abstract Transformer)**.** *Consider a function $F : \mathbb{D}_1 \to \mathbb{D}_2$ on concrete domains $\mathbb{D}_1$ and $\mathbb{D}_2$. An* abstract transformer *$F^\sharp : {\mathbb{D}_1}^\sharp \to {\mathbb{D}_2}^\sharp$ over-approximates F in the abstract domain:*

$$\forall S \in \mathcal{P}(\mathbb{D}) : \gamma\left(F^\sharp(\alpha(S))\right) \supseteq F(S)$$

An abstract transformer is exact *(does not loose precision) if $\forall d \in \mathbb{D}^\sharp : \gamma\left(F^\sharp(d)\right) = F(\gamma(d))$.*

Importantly, (exact) abstract transformers compose (see App. B.2, Prop. B.1) and, thus, we can construct an abstract transformer for complex functions from simpler parts.

To over-approximate the set of possible model parameters $\boldsymbol{w}^{\odot *}$, we will develop an abstract transformer $\mathcal{A}^\sharp$ for the learning algorithm $\mathcal{A}$ to get $\gamma\left(\mathcal{A}^\sharp(\mathbf{D}^\sharp)\right) \supseteq \boldsymbol{w}^{\odot *}$.

Symbolic Abstract Domains and Zonotopes. We consider a symbolic abstract domain Ψ of vectors and matrices (marked with $\cdot^\sharp$) with elements that are polynomials ψ over variables $\mathcal{E} = \{\epsilon_i\}$. The concretization of a polynomial ψ is the result of evaluating ψ on all assignments $e : \mathcal{E} \to [-1, 1]$, encoded as vectors $[-1, 1]^{|\mathcal{E}|}$: $\gamma(\psi) = \{\psi(e) \mid e \in [-1, 1]^{|\mathcal{E}|}\}$. We lift concretization to vectors and matrices through point-wise application. Such an object $\boldsymbol{z}^\sharp$ is typically referred to as a *polynomial zonotope* or *zonotope* if all symbolic expressions are linear (see App. C.2). The concretization of $\boldsymbol{z}^\sharp$ is: $\gamma\left(\boldsymbol{z}^\sharp\right) = \left\{\boldsymbol{z}^\sharp(e) \mid e \in [-1, 1]^{|\mathcal{E}|}\right\}$.

3 Exact Abstract Transformers for Learning Linear Models

Given an uncertain training dataset $\boldsymbol{D}^\odot$, we aim to over-approximate the set of possible optimal linear models $\boldsymbol{w}^{\odot *} = \{\boldsymbol{w}_1^*, \boldsymbol{w}_2^*, \ldots\}$, where $\boldsymbol{w}_i^* \in \mathbb{R}^p$ represents the optimal parameters of a linear model trained on $\boldsymbol{D}_i \in \boldsymbol{D}^\odot$. These optimal parameters are the fixed point of the sequence $\{\boldsymbol{w}_i^k\}_{k=0}^\infty$ generated by: $\boldsymbol{w}_i^{k+1} = \Phi(\boldsymbol{w}_i^k)$ where the operator $\Phi : \mathbb{R}^p \to \mathbb{R}^p$ captures one step of gradient descent, i.e., $\Phi(\boldsymbol{w}) = \boldsymbol{w} - \eta \nabla L(\boldsymbol{w})$, for a learning rate η and a loss function $L(\boldsymbol{w})$ [57].

In the abstract domain, we use the zonotope representations $\mathbf{D}^\sharp$ and $\boldsymbol{w}^\sharp$ to abstract the possible datasets $\boldsymbol{D}^\odot$ and the set of possible model weights $\boldsymbol{w}^{\odot *}$. While it is theoretically possible to compute symbolic expressions for the standard closed form solution for linear regression, this can result in large expressions that contain fractions with polynomial numerators and denominators and, thus, computing prediction intervals based on such expressions is computationally infeasible (see App. L). Instead, we over-approximate the optimal parameters using an abstract operator $\Phi^\sharp_{exact} : \boldsymbol{w}^\sharp \to \boldsymbol{w}^\sharp$ that generates a sequence of abstract elements $\{\boldsymbol{w}^{\sharp\, k}\}_{k=0}^\infty$:

$$\boldsymbol{w}^{\sharp\, k+1} = \Phi^\sharp_{exact}(\boldsymbol{w}^{\sharp\, k}),$$

where the abstract operator $\Phi^\sharp_{exact}$ given by $\Phi^\sharp_{exact}(\boldsymbol{w}^\sharp) = \boldsymbol{w}^\sharp - \eta \nabla L(\boldsymbol{w}^\sharp)$ captures one gradient descent step in the abstract domain. Specifically, for any loss function L whose gradient ∇L consists of linear or polynomial expressions, such as the mean squared error (MSE) loss, $\Phi^\sharp_{exact}$ is an exact abstract transformer. This follows from the existence of exact abstract transformers for addition and multiplication over polynomial zonotopes [37] and the fact that abstract transformers compose (see App. E, Prop. B.1 and Prop. E.1).

Proposition 3.1. *The abstract gradient descent operator $\Phi^\sharp_{exact}$ is an exact abstract transformer for the concrete gradient descent operator Φ. Formally, for any abstract $\boldsymbol{w}^\sharp$,*

$$\gamma\left(\Phi^\sharp_{exact}(\boldsymbol{w}^\sharp)\right) = \Phi(\gamma\left(\boldsymbol{w}^\sharp\right)),$$

Unfortunately, the sequence $\{\boldsymbol{w}^{\sharp\, k}\}_{k=0}^{\infty}$ does not have a fixed point as the highest-order symbolic terms in $\boldsymbol{w}^{\sharp\, k}$ are multiplied with symbolic terms in the gradient. Thus, $\boldsymbol{w}^{\sharp\, k+1} = \Phi^{\sharp}_{exact}(\boldsymbol{w}^{\sharp\, k})$ has terms of higher orders than any term in $\boldsymbol{w}^{\sharp\, k}$. To identify when the abstract model weights represent all fixed points in the concrete domain, we define abstract fixed points using concretization.

Definition 3.2 (Fixed Point of Abstract Gradient Descent)**.** *An abstract model weight $\boldsymbol{w}^{\sharp\,*}$ is a fixed point for an abstract transformer $F^{\sharp}$ for the gradient descent operator Φ if,*

$$\gamma\left(\boldsymbol{w}^{\sharp\,*}, \mathbf{D}^{\sharp}\right) = \gamma\left(F^{\sharp}(\boldsymbol{w}^{\sharp\,*}), \mathbf{D}^{\sharp}\right).$$

Here, $\gamma(\cdot,\cdot)$ denotes the joint concretization of $\boldsymbol{w}^{\sharp\,*}$ and $\mathbf{D}^{\sharp}$ (see App. C.2 for more details).

Any abstract fixed point $\boldsymbol{w}^{\sharp\,*}$ over-approximates all possible optimal model weights $\boldsymbol{w}^{\odot *}$.

Proposition 3.3. *Consider an abstract transformer $F^{\sharp}$ for the gradient descent operator Φ. Let $\boldsymbol{w}^{\sharp\,*}$ be a fixed point according to Def. 3.2. Then it holds that $\gamma\left(\boldsymbol{w}^{\sharp\,*}\right) \supseteq \boldsymbol{w}^{\odot *}$.*

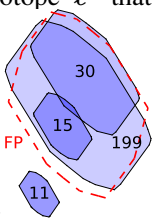
As we demonstrate in App. F.3, $\boldsymbol{w}^{\sharp\,*}$ can also be used to compute prediction ranges. In general, an abstract fixed point is not guaranteed to exist for every abstract transformer for Φ. While $\Phi^{\sharp}_{exact}$ as defined above has a fixed point (see Prop. F.1 in the appendix), two significant barriers remain. First, the representation size of the polynomial zonotope $\boldsymbol{w}^{\sharp\, i}$ for abstract model parameters grows exponentially in i, the number of the steps of abstract gradient descent. Additionally, testing for convergence is challenging, as checking for containment is already NP-hard for linear zonotopes [39].

4 Efficient Sound Approximation for Learning Linear Models

In this section, we present an efficient abstract gradient descent method for ridge regression that guarantees a fixed point and addresses the exponential growth of generated abstract model parameters. The core idea is to use the linearization and order reduction techniques introduced in the following to deal with intractable polynomial zonotopes. We develop a specific order reduction technique that admits a fixed point and enables us to find a closed-form solution for the fixed point.

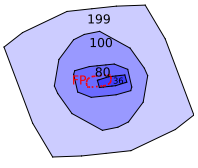
4.1 Gradient Descent With Order Reduction

A *linearization* operator $\mathbf{L}$ maps a polynomial zonotope $\boldsymbol{z}^{\sharp}$ to a linear zonotope $\boldsymbol{\ell}^{\sharp}$ that over-approximates $\boldsymbol{z}^{\sharp}$: $\gamma\left(\mathbf{L}(\boldsymbol{z}^{\sharp})\right) \supseteq \gamma\left(\boldsymbol{z}^{\sharp}\right)$. An *order reduction* operator $\mathbf{R}$ takes a linear zonotope $\boldsymbol{\ell}^{\sharp}$ as input and returns another linear zonotope $\boldsymbol{\ell}^{\sharp}$ of reduced order (representation size) such that: $\gamma\left(\mathbf{R}(\boldsymbol{\ell}^{\sharp})\right) \supseteq \gamma\left(\boldsymbol{\ell}^{\sharp}\right)$. Specific linearization and order reduction operators are discussed in App. G. The graph on the right shows the progression of the zonotope of parameters toward a fixed point from a random initialization by applying 200 iterations of the following abstract gradient descent operator $\Phi^{\sharp}$. The number inside each zonotope indicates its corresponding iteration. After 200 iterations the resulting zonotope is close to a fixed point shown in red and computed using techniques we introduce in the following. This operator first linearizes the gradient zonotope using $\mathbf{L}$, subtracts it from the current abstract model parameters $\boldsymbol{w}^{\sharp}$, and then reduces the order of the resulting zonotope using $\mathbf{R}$:



$$\Phi^{\sharp}(\boldsymbol{w}^{\sharp}) = \mathbf{R}\Big(\boldsymbol{w}^{\sharp} - \mathbf{L}\big(\eta \nabla L(\boldsymbol{w}^{\sharp})\big)\Big), \tag{1}$$

This operator is an abstract transformer for the gradient descent operator (Prop. H.1). While each step can be computed efficiently by bounding the order of the resulting zonotope, $\Phi^{\sharp}$ may not always converge to a fixed point. The graph on the left illustrates a real example where, despite the existence of a zonotope containing all optimal possible parameters (shown again in red), abstract gradient descent with linearization and order reduction keeps diverging, generating larger and larger zonotopes due to over-approximation error. In the following, we develop an order reduction operator that ensures abstract gradient descent with linearization converges to a fixed point for linear regression with ℓ_2 regularization. Additionally, we derive a closed-form solution for this fixed point. The red dotted zonotope in the graphs represents the fixed point generated by our method.



https://doi.org/10.52202/079017-3396

Decomposition of Gradients. A core idea in our approach is to decompose the abstract dataset into real and symbolic parts and use an order reduction operator that allows for the decomposition of the abstract gradient descent operator into components processing the real and symbolic parts in a specific evaluation order. Specifically, we observe that the components of the abstract training dataset $\mathbf{D}^\sharp = (\boldsymbol{X}^\sharp, \boldsymbol{y}^\sharp)$ can be decomposed into a sum of a real part (without error symbols) and a symbolic part (containing only symbolic terms). Furthermore, the abstract model weight $\boldsymbol{w}^\sharp$ produced by $\Phi^\sharp$ can be decomposed as shown below. $\boldsymbol{w}_R$ is the real part of the zonotope and does not contain any error symbols, $\boldsymbol{w}_D^\sharp$ contains only symbolic terms with error symbols that also occur in $\mathbf{D}^\sharp$, and $\boldsymbol{w}_N^\sharp$ contains only symbolic terms introduced by linearization and order reduction, hence does not contain any error symbols from $\mathbf{D}^\sharp$ since these methods introduce fresh symbols.

$$\boldsymbol{X}^\sharp = \boldsymbol{X}_R + \boldsymbol{X}_S^\sharp, \qquad \boldsymbol{y}^\sharp = \boldsymbol{y}_R + \boldsymbol{y}_S^\sharp \qquad \boldsymbol{w}^\sharp = \boldsymbol{w}_R + \boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp$$

The key observation that enables our approach is that one step of abstract gradient descent can now be decomposed into three components:

$$\Phi^\sharp(\boldsymbol{w}^\sharp) = \Phi_R(\boldsymbol{w}_R) + {\Phi^\sharp}_D(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp) + {\Phi^\sharp}_N(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp, \boldsymbol{w}_N^\sharp) \tag{2}$$

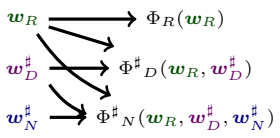
where

$$\Phi_R(\boldsymbol{w}_R) = \boldsymbol{w}_R - \eta \cdot \left(\frac{2}{n}({\boldsymbol{X}_R}^T \boldsymbol{X}_R \boldsymbol{w}_R - {\boldsymbol{X}_R}^T \boldsymbol{y}_R) + 2\lambda \boldsymbol{w}_R\right)$$

$${\Phi^\sharp}_D(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp) = \boldsymbol{w}_D^\sharp - \eta \cdot \left((2\lambda I + \frac{2}{n}{\boldsymbol{X}_R}^T \boldsymbol{X}_R)\boldsymbol{w}_D^\sharp + \frac{2}{n}({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)\boldsymbol{w}_R - \frac{2}{n}\boldsymbol{X}_S^\sharp \boldsymbol{y}_R - \frac{2}{n}{\boldsymbol{X}_R}^T \boldsymbol{y}_S^\sharp\right)$$

$$\begin{aligned}{\Phi^\sharp}_N(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp, \boldsymbol{w}_N^\sharp) = \mathbf{R}\Bigg(\boldsymbol{w}_N^\sharp - \eta \cdot \Bigg((2\lambda I + \frac{2}{n}{\boldsymbol{X}_R}^T \boldsymbol{X}_R)\boldsymbol{w}_N^\sharp \\ + \mathbf{L}\Big(\frac{2}{n}\Big(({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)(\boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp) + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_S^\sharp(\boldsymbol{w}_R + \boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp) - {\boldsymbol{X}_S^\sharp}^T \boldsymbol{y}_S^\sharp\Big)\Big)\Bigg)\Bigg)\end{aligned}$$

$\Phi_R(\boldsymbol{w}_R)$, the *real part updater*, only relies on and updates $\boldsymbol{w}_R$, the real part of the abstract weight, and coincides with the abstract gradient operator Φ on the real part of abstract weights. ${\Phi^\sharp}_D(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp)$, the *symbolic data-dependent updater*, which is a function of $\boldsymbol{w}_D^\sharp$ itself and $\boldsymbol{w}_R$, updates $\boldsymbol{w}_D^\sharp$ based on some linear terms and consists of symbols that come from the abstract dataset $\mathbf{D}^\sharp$ and does not include any non-linear terms; hence applying this does not generate higher-order terms, so the output is a linear zonotope, and no linearization is needed. Additionally, no order reduction is needed since the number of error symbols remains constant. ${\Phi^\sharp}_N(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp, \boldsymbol{w}_N^\sharp)$, the *symbolic data-independent updater*, is a function of $\boldsymbol{w}_N^\sharp$ itself, $\boldsymbol{w}_D^\sharp$, and $\boldsymbol{w}_R$. It consists of the non-linear terms in the gradient, hence performs linearization to over-approximate higher-order terms and performs order reduction to reduce the number of error symbols, which would otherwise grow exponentially with the number of iterations. Therefore, it generates the updated $\boldsymbol{w}_N^\sharp$ as a linear zonotope that does not share any symbol with $\mathbf{D}^\sharp$, i.e., only consists of fresh symbols introduced by linearization and order reduction. See Appendix I for formal statements and proofs for this section.



Proposition 4.1. *Any abstract model weights* $\boldsymbol{w}^{\sharp *} = \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$ *is a fixed point for the abstract gradient descent operator* $\Phi^\sharp$ *if the following conditions are satisfied:*

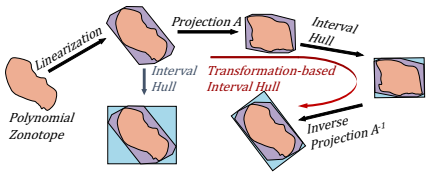
$$\boldsymbol{w}_R^* = \Phi_R(\boldsymbol{w}_R^*), \qquad \boldsymbol{w}_D^{\sharp *} = {\Phi^\sharp}_D(\boldsymbol{w}_R^*, \boldsymbol{w}_D^{\sharp *}) \qquad \boldsymbol{w}_N^{\sharp *} \simeq_\sharp {\Phi^\sharp}_N(\boldsymbol{w}_R^*, \boldsymbol{w}_D^{\sharp *} \boldsymbol{w}_N^{\sharp *}) \tag{3}$$

Algorithm 1: Abstract Learning

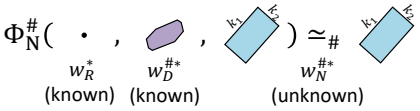
Input: abstract dataset $\mathbf{D}^\sharp$, learning rate η, regularization coeff. λ, transformation $\boldsymbol{A}$
$\boldsymbol{w}_R^* \leftarrow$ `closedFormReal`$(\mathbf{D}^\sharp, \lambda)$
$\boldsymbol{w}_D^{\sharp *} \leftarrow$ `closedFormSymb`$(\mathbf{D}^\sharp, \lambda, \boldsymbol{w}_R^*)$
$\Xi \leftarrow$ `genNonDataEq`$(\mathbf{D}^\sharp, \lambda, \eta, \boldsymbol{w}_R^*, \boldsymbol{w}_D^{\sharp *})$
$\boldsymbol{w}_N^{\sharp *} \leftarrow$ `solveFixedPointEq`(Ξ, λ, η)
$\boldsymbol{w}^{\sharp *} \leftarrow \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$
return $\boldsymbol{w}^{\sharp *}$

We summarize the fixed point construction process in Alg. 1. The algorithm takes as input an abstract dataset $\mathbf{D}^\sharp$, a learning rate η, a regularization coefficient λ, and a transformation matrix $\boldsymbol{A}$. The construction order is: first $\boldsymbol{w}_R^*$, then $\boldsymbol{w}_D^{\sharp *}$, and finally $\boldsymbol{w}_N^{\sharp *}$, each step building on the previous results. Closed-form solutions for $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp *}$ exist, as detailed in Lem. I.4. The fixed point of Φ_R matches the fixed point of gradient descent on the real part of the abstract data, independent of other parts. Given $\boldsymbol{w}_R^*$, $\boldsymbol{w}_D^{\sharp *}$ is obtained

by solving a system of linear equations from Eq. (3), ensuring zonotope containment by equalizing the coefficients of the same error symbols, which is a sufficient condition for zonotope equivalence. This involves solving a system of m equations, where m is the number of error symbols in $\mathbf{D}^\sharp$.



To construct $\boldsymbol{w}_N^{\sharp *}$ in Eq. (3), which involves zonotope equivalence, we encounter a challenge because the RHS and LHS have different error symbols, making symbolic enforcement of equivalence inapplicable. We exploit the properties of the order reduction operator to create a system of linear equations whose solution yields a fixed point for ${\Phi^\sharp}_N$. Recall that this component uses linearization to handle non-linear terms and applies order reduction to efficiently manage the order of the resulting linear zonotope. As shown on the right side, the order reduction operator in ${\Phi^\sharp}_N$ utilizes a transformation matrix $\boldsymbol{A}$ to project the higher-order linear zonotope into a new space and then over-approximates it using an interval hull—a zonotope that is a box enclosing the input higher-order zonotope in the projected space—and finally projects this box back into the original space using $\boldsymbol{A}^{-1}$. The core idea of transformation-based order reduction [38] is that interval hulls provide a better over-approximation in the projected space, where the shape of the input zonotope is closer to a box than in the original space, as shown in the graph on the right.



At a fixed point, the input and output interval hulls of ${\Phi^\sharp}_N$ should be equivalent in the projected space. This equivalence can be effectively enforced by checking for equal edge lengths along each dimension, which is sufficient for two interval hulls to be equivalent. This translates to a system of k equations, where k is the number of model parameters, hence can be done efficiently. We show that this system of equations is guaranteed to have a solution. We are now ready to state our main technical result.

Theorem 4.2 (Correctness of Alg. 1)**.** *Given a set of possible training datasets $\boldsymbol{D}^\odot$ associated with uncertain data $\boldsymbol{D}$, and an appropriate abstraction function α in the zonotope abstract domain, and given a regularization coefficient λ, Alg. 1, when provided with $\mathbf{D}^\sharp = \alpha(\boldsymbol{D}^\odot)$ as input, computes abstract model parameters $\boldsymbol{w}^{\sharp *}$ such that:*

$$\gamma\left(\boldsymbol{w}^{\sharp *}\right) \supseteq \boldsymbol{w}^{\odot *},$$

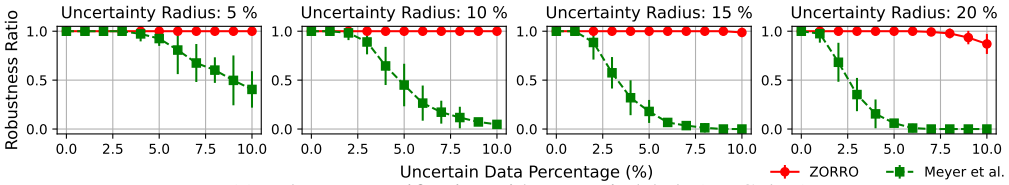
where $\boldsymbol{w}^{\odot *}$ denotes the set of all optimal model parameters corresponding to $\boldsymbol{D}^\odot$ for linear regression with ℓ_2 regularization with regularization coefficient λ.

5 Experiments

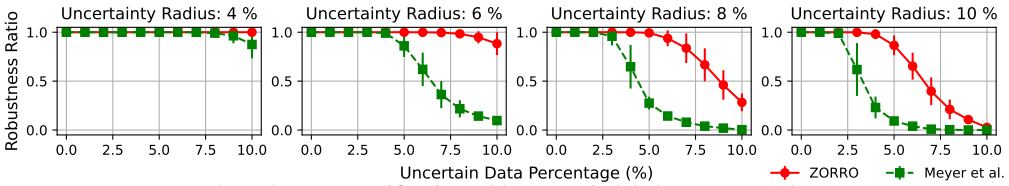
We implement ZORRO using SymPy [45], a Python library for symbolic computations and evaluate the system on two key applications: (1) computing prediction ranges and robustness certification for linear models trained on uncertain data, and (2) robustness of model weights for causal inference using linear models as a case study. We also measured the performance of ZORRO under varying conditions, including varying the degree of training data uncertainty. All our experiments are performed on a single machine with an Apple M1 chip, 8 cores, and 16 GB RAM. Experiments are repeated 5 times with different random seeds, and we report the mean (error bars denote 3σ). The code is shared at https://github.com/lodino/Zorro.

Datasets, Baselines, and Metrics. For robustness verification we use regression tasks: for *MPG* [58] (392 instances) we predict fuel consumption based on car features (cylinders, horsepower, weight); for *Insurance* [30] (1338 instances) we predict medical insurance charges based on demographics (age, gender, BMI), habits (smoking), and geographical features. We use a 80:20 train-test split and inject random errors to the training data varying (i) the *Uncertain Data Percentage*, the percentage of instances that have uncertain features / labels, and (ii) the *Uncertainty Radius*, the difference between the minimum and maximum possible value of an uncertain feature expressed as a fraction of the feature's domain. There is no direct baseline capturing prediction ranges for learning linear regression models from uncertain data. Most existing works in robustness certification focuses on test-time robustness (cf. Sec. 1). The exception is [47], which only supports uncertainty in labels using interval arithmetic. We compare ZORRO against this approach for robustness certification, referred to as MEYER in the following, only for training label uncertainty. For robustness verification, a prediction is *robust* if the size of the prediction interval is smaller than a given threshold. We use

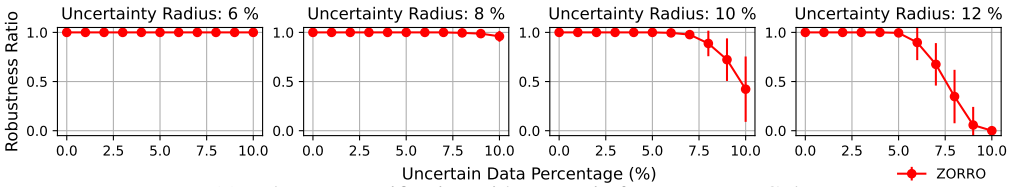
https://doi.org/10.52202/079017-3396



(a) Robustness verification with uncertain labels (MPG data).



(b) Robustness verification with uncertain labels (Insurance data).



(c) Robustness verification with uncertain features on MPG data.

Figure 1: Robustness verification on using intervals (MEYER [47]) and zonotopes (ZORRO).

the *robustness ratio* which is the fraction of the test data receiving robust predictions as a metric in all robustness verification experiments. The robustness threshold is set to 5% of the label range for the *MPG* data, and 0.8% of the label range for the *Insurance* data. Additionally, we assess the *worst-case test loss* using certain test data and uncertain model weights trained from uncertain training data.

5.1 Robustness Verification

Prediction Robustness (Uncertain Labels). Fig. 1(a)(b) compares ZORRO with the baseline MEYER [47] on a setting where only training labels suffer from uncertainty. We vary the uncertainty radius and uncertain data percentage. As both systems provide sound over-approximations of prediction ranges, they may underestimate a model's robustness. As shown, ZORRO consistently certifies significantly higher robustness ratios than MEYER. This is due to the fact that MEYER uses interval-arithmetic which ignores the correlation between model weights in different dimensions and between the training labels and the weights, leading to overly conservative prediction ranges. Specifically, for higher uncertainty radius values, MEYER fails to certify robustness for most of the data while ZORRO still can certify robustness for 100% of the instances.

Prediction Robustness (Uncertain Features). We also evaluate the impact of training feature uncertainty (not supported by MEYER). Specifically, we introduce uncertainty into the vehicle weight column for the MPG dataset. As shown in Fig. 1(c), uncertainty in the features results in relatively less robust predictions compared to uncertain labels for a similar uncertainty radius (Fig. 1(a)). This is primarily because uncertain features result in more high-order terms in the closed form solution than uncertain labels which in turn leads to larger over-approximation errors during linearization.

In all experiments the standard deviation of the robustness ratio, calculated by repeating experiments with different random seeds, is large when the average robustness ratio is close to 0.5. This is because when the uncertain data percentage is low, the model will be robust no matter which training instances are selected to be uncertain. Likewise, when the uncertain data percentage is high, then most predictions will be uncertain no matter which training data points are uncertain.

Parameter Robustness. Next, we apply ZORRO for robustness certification of parameters in linear regression models, crucial for statistical estimation and causal analysis. We compare the ground truth coefficients for a treatment variable with the results obtained by ZORRO and through KNN imputation on a dataset with injected missing data. Fig. 2 shows the treatment variable coefficient and

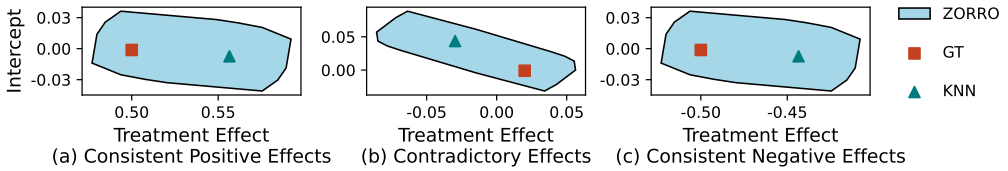


Figure 2: Applying ZORRO to causal inference. The intercept (y-axis) is the model's bias term, the treatment effect (x-axis) is the coefficient for the treatment variable.

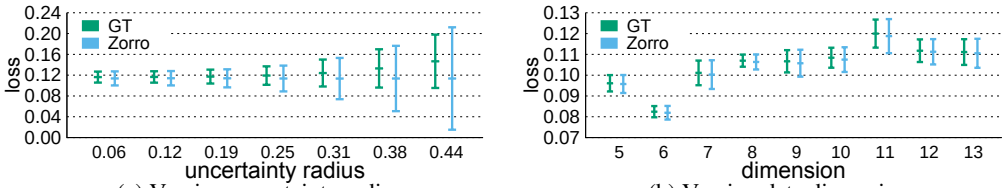


Figure 3: Range of the loss, through enumeration of all possible worlds (GT) and ZORRO.

the regression model's intercept. The intercept captures the baseline level of the outcome variable when all predictors are zero, highlighting how baseline values can shift under uncertainty. While the model trained after KNN imputation sometimes correctly identifies the directionality of the treatment effect, this is not always the case as shown in Fig. 2(b). This highlights the needs for techniques like ZORRO which guarantee that the true treatment effect is within certain bounds.

5.2 Solution Quality with Varying Uncertainty and Hyperparameters

We evaluate ZORRO's effectiveness by testing the tightness of the over-approximation and the accuracy of possible models. Specifically, we examine how the over-approximation quality and worst-case loss are influenced by the level of data uncertainty and the regularization coefficient.

Varying Data Uncertainty. To evaluate how specific characteristics of the data affect the effectiveness of ZORRO, we injected errors into real datasets, varying uncertain data percentage and uncertainty range. We compare ZORRO with the ground truth range of the loss computed by enumerating all possible worlds (GT). The results shown in Fig. 3(a) demonstrate that ZORRO tightly over-approximates the ground truth loss range, especially for smaller uncertainty radius values. As uncertainty increases, the over-approximation gap widens due to the increased coefficient of higher-order terms in the gradient, which are linearized, leading to higher linearization errors. As shown in Fig. 3(b), the tightness of ZORRO's over-approximation is not affected by the dimension of the data.

Effect of Regularization. We investigate the impact of the regularization coefficient on the robustness of predictions and the worst-case loss of possible models. Following a similar approach to Sec. 5.1, we introduce uncertainty in both features and labels in the MPG dataset and use zonotopes with varying levels of uncertainty to over-approximate the training data uncertainty. The results, shown in Fig. 4, indicate that a higher regularization coefficient leads to more robust predictions, as regularization tends to "compress" all possible model weights towards the origin. Interestingly, the worst-case loss shows that $\lambda = 0$ is not optimal across all scenarios, especially when the fraction uncertain instances is high. Instead, a small, positive λ (e.g., 0.02 or 0.025) generally yields the best worst-case losses. Combining these results, the optimal regularization coefficient should enhance robustness (i.e., a higher robustness ratio) while maintaining an acceptable worst-case loss. Therefore, the regularization coefficient should be tuned based on a validation dataset to achieve a small range of accurate possible models.

6 Conclusions, limitations and broader impacts

We introduce an approach for propagating uncertainty through model training and inference for linear models. Given an abstract uncertain training dataset that over-approximates the possible worlds of a training dataset, we develop abstract interpretation techniques to over-approximate the set of possible models and inference results for this set of models using zonotopes. This is challenging, as we need to compute fixed points of gradient descent in the abstract domain. Our main technical

https://doi.org/10.52202/079017-3396

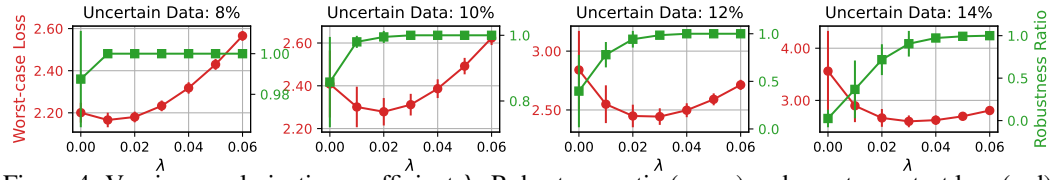


Figure 4: Varying regularization coefficient λ: Robustness ratio (green) and worst-case test loss (red).

contribution is the development of closed-form solutions for such fixed points that can be solved efficiently. Our techniques efficiently over-approximate models and inference for several use cases, including robustness verification, uncertainty management in causal reasoning, and improving the interpretability and reliability of predictions and inferences. This framework can be particularly valuable in critical applications where data quality and robustness are paramount. While we propose an effective method for abstract learning of linear models, extending our approach to more complex models is challenging. Non-linear models, such as neural networks, would require more advanced linearization and order reduction techniques, as well as parallelization, to manage the increased complexity of the involved symbolic operations. In addition, adapting our method to a broader classes of models through efficiently approximating the fixed points remains a challenging and promising future direction.

7 Acknowledgment

This research was supported by NSF awards IIS-2340124, IIS-2420691, and IIS-2420577, as well as by NIH grant U54HG012510. The views, opinions, and findings presented are those of the authors and do not necessarily represent those of the NSF or NIH.

References

[1] Amr Alanwar, Anne Koch, Frank Allgöwer, and Karl Henrik Johansson. Data-driven reachability analysis using matrix zonotopes. In *Learning for Dynamics and Control*, pages 163–175. PMLR, 2021.

[2] Matthias Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010.

[3] Matthias Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 173–182, 2013.

[4] Matthias Althoff and Bruce H Krogh. Zonotope bundles for the efficient computation of reachable sets. In *2011 50th IEEE conference on decision and control and European control conference*, pages 6814–6821. IEEE, 2011.

[5] Stanley Bak, Sergiy Bogomolov, Brandon Hencey, Niklas Kochdumper, Ethan Lew, and Kostiantyn Potomkin. Reachability of koopman linearized systems using random fourier feature observables and polynomial zonotope refinement. In *International Conference on Computer Aided Verification*, pages 490–510. Springer, 2022.

[6] Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Advanced Risk & Portfolio Management® Research Paper Series*, 2011.

[7] Emily Black, Manish Raghavan, and Solon Barocas. Model multiplicity: Opportunities, concerns, and solutions. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 850–863, 2022.

[8] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021.

[9] Leo Breiman. Heuristics of instability and stabilization in model selection. *The annals of statistics*, 24(6):2350–2383, 1996.

[10] Christophe Combastel. Functional sets with typed symbols: Mixed zonotopes and polynotopes for hybrid nonlinear reachability and filtering. *Automatica*, 143:110457, 2022.

[11] A Feder Cooper, Yucheng Lu, Jessica Forde, and Christopher M De Sa. Hyperparameter optimization is deceiving us, and how to stop it. *Advances in Neural Information Processing Systems*, 34:3081–3095, 2021.

[12] P. Cousot. Abstract interpretation. *CSUR*, 28(2):324–328, 1996.

[13] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252, 1977.

[14] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 84–96, 1978.

[15] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.

[16] Fabricio Olivetti de Franca and Gabriel Kronberger. Prediction intervals and confidence regions for symbolic regression models based on likelihood profiles, 2022.

[17] Nicolas Dewolf, Bernard De Baets, and Willem Waegeman. Valid prediction intervals for regression problems. *Artificial Intelligence Review*, 56(1):577–613, April 2022.

[18] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robustness meets algorithms. *Commun. ACM*, 64(5):107–115, apr 2021.

[19] Ilias Diakonikolas and Daniel M. Kane. Recent advances in algorithmic high-dimensional robust statistics. *CoRR*, abs/1911.05911, 2019.

[20] Tianyu Du, Shouling Ji, Lujia Shen, Yao Zhang, Jinfeng Li, Jie Shi, Chengfang Fang, Jianwei Yin, Raheem Beyah, and Ting Wang. Cert-rnn: Towards certifying the robustness of recurrent neural networks. *CCS*, 21(2021):15–19, 2021.

[21] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. The limits of distribution-free conditional predictive inference. *Information and Inference: A Journal of the IMA*, 10(2):455–482, 2021.

[22] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2018.

[23] Matthew L Ginsberg. Counterfactuals. *Artificial intelligence*, 30(1):35–79, 1986.

[24] Eric Goubault, Tristan Le Gall, and Sylvie Putot. An accurate join for zonotopes, preserving affine input/output relations. In Jan Midtgaard and Matthew Might, editors, *Proceedings of International Workshop on Numerical and Symbolic Abstract Domains*, volume 287, pages 65–76, 2012.

[25] Cornelia Gruber, Patrick Oliver Schenk, Malte Schierholz, Frauke Kreuter, and Göran Kauermann. Sources of Uncertainty in Machine Learning – A Statisticians' View. *arXiv*, 2023.

[26] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

[27] Chengqiang Huang, Zheng Hu, Xiaowei Huang, and Ke Pei. Statistical certification of acceptable robustness for neural networks. In *Artificial Neural Networks and Machine Learning*, pages 79–90. Springer, 2021.

[28] Tomasz Imieliński and Witold Lipski Jr. Incomplete information in relational databases. *Journal of the ACM (JACM)*, 31(4):761–791, 1984.

[29] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)*, pages 19–35. IEEE, 2018.

[30] Arun Jangir and Willian Oliveira. Healthcare insurance, 2023. This dataset is licensed under CC0: Public Domain.

 https://doi.org/10.52202/079017-3396

[31] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7961–7969, 2021.

[32] Matt Jordan, Jonathan Hayase, Alex Dimakis, and Sewoong Oh. zonotope domains for lagrangian neural network verification. In *NIPS*, 2022.

[33] Bojan Karlas, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *Proc. VLDB Endow.*, 14(3):255–267, 2020.

[34] Pasha Khosravi, Yitao Liang, YooJung Choi, and Guy Van den Broeck. What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features. *arXiv*, 2019.

[35] Byol Kim, Chen Xu, and Rina Foygel Barber. Predictive inference is free with the jackknife+-after-bootstrap. In *International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2020.

[36] Niklas Kochdumper and Matthias Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 66(9):4043–4058, 2020.

[37] Niklas Kochdumper and Matthias Althoff. Constrained polynomial zonotopes. *Acta Informatica*, 60(3):279–316, 2023.

[38] Anna-Kathrin Kopetzki, Bastian Schürmann, and Matthias Althoff. Methods for order reduction of zonotopes. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5626–5633. IEEE, 2017.

[39] Adrian Kulmburg and Matthias Althoff. On the co-np-completeness of the zonotope containment problem. *Eur. J. Control*, 62:84–91, 2021.

[40] Sungyoon Lee, Hoki Kim, and Jaewook Lee. Graddiv: Adversarial robustness of randomized neural networks via gradient diversity regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[41] David Lewis. Counterfactuals and comparative possibility. In *IFS: Conditionals, Belief, Decision, Chance and Time*, pages 57–85. Springer, 1973.

[42] Charles Marx, Flavio Calmon, and Berk Ustun. Predictive multiplicity in classification. In *International Conference on Machine Learning*, pages 6765–6774. PMLR, 2020.

[43] Peter McCullagh, Vladimir Vovk, Ilia Nouretdinov, Dmitry Devetyarov, and Alex Gammerman. Conditional prediction intervals for linear regression. In *International Conference on Machine Learning and Applications*, pages 131–138, 2009.

[44] Johannes Mehrer, Courtney J Spoerer, Nikolaus Kriegeskorte, and Tim C Kietzmann. Individual differences among deep neural network models. *Nature communications*, 11(1):5725, 2020.

[45] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[46] Anna P. Meyer, Aws Albarghouthi, and Loris D'Antoni. Certifying data-bias robustness in linear regression. *CoRR*, abs/2206.03575, 2022.

[47] Anna P Meyer, Aws Albarghouthi, and Loris D'Antoni. The dataset multiplicity problem: How unreliable data impacts predictions. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 193–204, 2023.

[48] Antoine Miné. The octagon abstract domain. *Higher-order and symbolic computation*, 19:31–100, 2006.

[49] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pages 3578–3586. PMLR, 2018.

[50] Matthew Mirman, Alexander Hägele, Pavol Bielik, Timon Gehr, and Martin Vechev. Robustness certification with generative models. In *ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 1141–1154, 2021.

[51] Mark Niklas Müller, Marc Fischer, Robin Staab, and Martin Vechev. Abstract interpretation of fixpoint iterators with applications to neural networks. *Proceedings of the ACM on Programming Languages*, 7(PLDI):786–810, 2023.

[52] Nicolas Müller, Daniel Kowatsch, and Konstantin Böttinger. Data poisoning attacks on regression learning and corresponding defenses. In *Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 80–89. IEEE, 2020.

[53] Hongseok Namkoong and John C Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[54] David J. Olive. Prediction intervals for regression models. *Computational Statistics & Data Analysis*, 51(6):3115–3122, 2007.

[55] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label sanitization against label flipping poisoning attacks. In *ECML PKDD Workshops*, pages 5–15. Springer, 2019.

[56] Robert J Plemmons. M-matrix characterizations. i-nonsingular m-matrices. *Linear Algebra and its applications*, 18(2):175–188, 1977.

[57] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[58] R. Quinlan. Auto MPG. UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5859H. This dataset is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

[59] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[60] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

[61] Donald B Rubin. Multiple imputations in sample surveys-a phenomenological bayesian approach to nonresponse. In *Proceedings of the survey research methods section of the American Statistical Association*, volume 1, pages 20–34. American Statistical Association Alexandria, VA, USA, 1978.

[62] Donald B Rubin. Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489, 1996.

[63] Anian Ruoss, Mislav Balunovic, Marc Fischer, and Martin Vechev. Learning certified individually fair representations. In *Advances in Neural Information Processing Systems*, pages 7584–7596, 2020.

[64] Christian Schilling, Marcelo Forets, and Sebastián Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8169–8177, 2022.

[65] Soroosh Shafieezadeh-Abadeh, Peyman Mohajerin Esfahani, and Daniel Kuhn. Distributionally robust logistic regression. In *Neural Information Processing Systems*, 2015.

[66] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross J Anderson. Manipulating sgd with data ordering attacks. *Advances in Neural Information Processing Systems*, 34:18021–18032, 2021.

[67] Daniele Silvestro and Tobias Andermann. Prior choice affects ability of bayesian neural networks to identify unknowns. *arXiv preprint arXiv:2005.04987*, 2020.

[68] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.

[69] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[70] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *NeurIPS*, 2017.

[71] Damien Teney, Maxime Peyrard, and Ehsan Abbasnejad. Predicting is not understanding: Recognizing and addressing underspecification in machine learning. In *European Conference on Computer Vision*, pages 458–476. Springer, 2022.

 https://doi.org/10.52202/079017-3396

[72] Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56, 2022.

[73] Rens van de Schoot, Sarah Depaoli, Ruth King, Bianca Kramer, Kaspar Märtens, Mahlet G Tadesse, Marina Vannucci, Andrew Gelman, Duco Veen, Joukje Willemsen, et al. Bayesian statistics and modelling. *Nature Reviews Methods Primers*, 1(1):1, 2021.

[74] Marianne Southall Winslett. *Reasoning about action using a possible models approach*. Number 1427-1432. Department of Computer Science, University of Illinois at Urbana-Champaign, 1988.

[75] Margaux Zaffran, Aymeric Dieuleveut, Julie Josse, and Yaniv Romano. Conformal prediction with missing values. In *International Conference on Machine Learning*, pages 40578–40604. PMLR, 2023.

[76] Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. Training set debugging using trusted items. In *AAAI*, 2018.

A Linear Regression and Ridge Regression

In this section, we review standard loss functions for linear regression, specifically mean squared error (MSE) and the loss function used in ridge regression. can be trained via either the closed-form solution or gradient descent. Specifically, the closed-form solution requires computing the inversion of the covariance matrix, while the gradient descent only involves matrix addition and multiplication. Since the linear regression model is convex, gradient descent is guaranteed to converge to the global optimum.

Suppose we have a training data $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ with n i.i.d. samples , where the feature matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$, and the labels $\boldsymbol{y} = [y_1, \cdots, y_n]^T \in \mathbb{R}^n$. Given an input $\boldsymbol{x}$ and the model weight $\boldsymbol{w}$, the prediction of the linear regression model $\hat{\boldsymbol{y}} = \boldsymbol{w}^T \boldsymbol{x}$ (the bias term can be integrated into $\boldsymbol{w}$, corresponding to an added column in $\boldsymbol{X}$ with constant 1's).

Loss Functions The mean squared error (MSE) loss on $\boldsymbol{D}$ is defined as shown below.

$$L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i' - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x_i} - y_i)^2 = \frac{1}{n} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}). \tag{4}$$

In practice, regularization terms, e.g., based on the l_p-norm of the model parameters, are often added to the original MSE loss to prevent overfitting by penalizing large weights. Using l_2-regularization with a regularization coefficient λ which determines the strength of regularization is often called *ridge regression*. The loss function for ridge regression is:

$$L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) = \frac{1}{n} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + \lambda \cdot \boldsymbol{w}^T \boldsymbol{w}. \tag{5}$$

Gradient Descent for Linear Regression Due to the convexity of linear models, the locally optimal point, which can be obtained by gradient descent with an appropriate learning rate η, is globally optimal. In gradient descent, the model weights $\boldsymbol{w}$ are iteratively updated (with some learning rate η) towards the reverse direction of the gradient $\frac{\partial L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w})}{\partial \boldsymbol{w}}$. Thus, for ridge regression:

$$\frac{\partial L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{2}{n} (\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y}) + 2\lambda \boldsymbol{w}. \tag{6}$$

Thus, one step of gradient descent Φ is:

$$\boldsymbol{w}^{i+1} = \boldsymbol{w}^i - \eta \frac{2}{n} (\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y}) + 2\lambda \boldsymbol{w} \tag{7}$$

Closed-form Solution for Linear Regression The convex nature of linear models ensures that the optimal weight $\boldsymbol{w} = \boldsymbol{w}^*$, which minimizes the loss $L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w})$, can be computed by establishing $\frac{\partial L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w})}{\partial \boldsymbol{w}} = 0$:

$$\begin{aligned} & \frac{\partial L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}^*)}{\partial \boldsymbol{w}^*} = \frac{2}{n} (\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w}^* - \boldsymbol{X}^T \boldsymbol{y}) + 2\lambda \boldsymbol{w}^* = 0 \\ \Rightarrow \quad & \boldsymbol{X}^T \boldsymbol{y} = (\boldsymbol{X}^T \boldsymbol{X} + \lambda n I) \boldsymbol{w}^* \\ \Rightarrow \quad & \boldsymbol{w}^* = (\boldsymbol{X}^T \boldsymbol{X} + \lambda n I)^{-1} \boldsymbol{X}^T \boldsymbol{y} \end{aligned} \tag{8}$$

B Background on Abstract Interpretation

Abstract interpretation [13] is a technique for over-approximating the results of computations over a set of inputs. This is achieved by associating sets of elements of a concrete domain $\mathbb{D}$ with elements from an abstract domain $\mathbb{D}^\sharp$. In this context, various abstract domains are employed: interval domains represent variables as ranges of possible values [13], octagon domains allow for constraints between

https://doi.org/10.52202/079017-3396

pairs of variables within a specific bound [48], and polynomial zonotopes which represent convex polytopes using polynomial constraints [14]. Abstract interpretation, while originally designed for static program analysis such as strictness analysis, has also found applications in wide range of other domains including reachability analysis [1, 5, 2, 10, 36, 4], robustness verification for neural networks [32, 64], learning robust models by providing bounds on the loss for a set of inputs [51, 63, 49, 22], and many others.

We now state some important, but well-known, facts about abstract transformers that we utilize in our derivations.

B.1 Abstract Transformers

In Def. 2.5, we presented the standard definition of abstract transformers as functions over abstract domains that over-approximate the application of functions in the concrete domains to sets of elements. To clarify the connection between abstract transformers and possible world semantics observe that both take a concrete function $F : \mathbb{D} \to \mathbb{D}'$ and lift it to sets of inputs $S \in \mathcal{P}(\mathbb{D})$ through point-wise applications:

$$F(S) = \{F(e) \mid e \in S\}$$

Thus, abstract interpretation is a natural fit for over-approximating PWS by over-approximating sets of possible worlds using abstract elements and then over-approximates computations with PWS for a function F using abstract transformers for such a function.

B.2 Abstract Transformers Compose

Importantly, abstract transformers compose. This enables us to decompose a complex computation into simpler operations and build an abstract transformer for the computation by composing abstract transformations for these simpler operations.

Proposition B.1 (Abstract Transformers Compose)**.** *Consider (exact) abstract transformers $f^\sharp$ and $g^\sharp$ for functions f and g, then $g^\sharp \circ f^\sharp$ is an (exact) abstract transformer for $g \circ f$.*

C Symbolic and Standard Representation of Zonotopes

We now provide a more detailed account of the correspondence between the symbolic and standard geometric representation of zonotopes and polynomial zonotopes and discuss why matrices and sets of symbolic matrices as used in our abstract domain can equivalently be thought of as (polynomial) zonotopes.

C.1 Symbolic vs. Geometric Representation

We defined zonotopes $\boldsymbol{z}^\sharp$ as vectors Ψ^d where Ψ denotes polynomials over variables (called error symbols) $\mathcal{E}$. The concretization of such a symbolic representation of a zonotope is the set of vectors in $\mathbb{R}^d$ that can be derived from $\boldsymbol{z}^\sharp$ by assigning values from $[-1, 1]$ to each variable $\epsilon \in \mathcal{E}$. We encode such variable assignments as vectors $[-1, 1]^{|\,\mathcal{E}\,|}$.

Definition C.1 (Polynomial zonotopes - Symbolic Representation)**.** *A d-dimensional polynomial zonotope is a vector $\boldsymbol{z}^\sharp \in \Psi^d$ [10]. Let $m = |\,\mathcal{E}\,|$. The concretization of $\boldsymbol{z}^\sharp$ is defined as:*

$$\gamma\left(\boldsymbol{z}^\sharp\right) = \left\{\boldsymbol{z}^\sharp(e) \mid e \in [-1, 1]^{|\,\mathcal{E}\,|}\right\}$$

A common measure of the representation size of a zonotope is its order. The order of a d-dimensional (polynomial) $\boldsymbol{z}^\sharp$ in symbolic representation is the total number of distinct monomials in $\boldsymbol{z}^\sharp$ divided by d:

$$\text{ORD}(\boldsymbol{z}^\sharp) = \frac{\#\mathcal{M}(\boldsymbol{z}^\sharp)}{d}$$

where $\#\mathcal{M}(\psi)$ denotes the cardinality of the set of distinct monomials in polynomial ψ.

A more common way to represent d-dimensional zonotopes is by fixing a set $\mathcal{S}$ of monomials over $\mathcal{E}$ and writing the zonotope as a central point $\boldsymbol{c} \in \mathbb{R}^d$ and a sum of generator vectors $\boldsymbol{g_i} \in \mathbb{R}^d$ multiplied with the monomials from $\mathcal{S}$. The generator vector $\boldsymbol{g_i}$ assigns a coefficient to monomial $\mathcal{S}[i]$ for each of the d dimensions.

Definition C.2 (Polynomial zonotopes - Geometric Representation)**.** *The geometric representation of a d-dimensional polynomial zonotope is a sum of a center point $\boldsymbol{c} \in \mathbb{R}^d$ and the monomials over error symbols in $\mathcal{E}$ from a set $\mathcal{S}$ multiplied by coefficients encoded in set of generator vectors $\boldsymbol{g_i} \in \mathbb{R}^d$:*

$$\boldsymbol{z}^\sharp = \boldsymbol{c} + \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{g}_i \mathcal{S}[i]$$

For the geometric representation, the order of a d-dimensional (polynomial) $\boldsymbol{z}^\sharp$ over monomials $\mathcal{S}$ is defined as:

$$\text{ORD}(\boldsymbol{z}^\sharp) = \frac{|\mathcal{S}|}{d}$$

Note that the order of the symbolic and geometric representation of a zonotope are the same. As an example consider the 4-dimensional polynomial zonotope $\boldsymbol{z}^\sharp$ shown in symbolic representation (left) and standard representation (right).

$$\begin{bmatrix} 1 + 1{\epsilon_2}^3 + 1{\epsilon_1}^2\epsilon_2\epsilon_3 \\ 4 + 2{\epsilon_2}^3 + 2{\epsilon_1}^2\epsilon_2\epsilon_3 \\ 0 + 2{\epsilon_2}^3 + 3{\epsilon_1}^2\epsilon_2\epsilon_3 \\ 2 + 1{\epsilon_2}^3 + 4{\epsilon_1}^2\epsilon_2\epsilon_3 \end{bmatrix} \qquad \underbrace{\begin{bmatrix} 1 \\ 4 \\ 0 \\ 2 \end{bmatrix}}_{\boldsymbol{c}} + \underbrace{\begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}}_{\boldsymbol{g_1}} \cdot \underbrace{\epsilon_2^3}_{\mathcal{S}[1]} + \underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}}_{\boldsymbol{g_2}} \cdot \underbrace{{\epsilon_1}^2\epsilon_2\epsilon_3}_{\mathcal{S}[2]}$$

C.2 Symbolic Matrices and Sets of Symbolic Matrices

We use matrices over symbolic expressions and sets of heterogeneous abstract matrices to represent the state of a computation and allow matrices from such a set to share variables to encode relationships between the elements of such matrices. The semantics we associate with such a set is that of a *joint concretization*. For $S^\sharp = \{{\boldsymbol{M}_i}^\sharp\}$, we define:

$$\gamma\left(S^\sharp\right) = \{{\boldsymbol{M}_i}^\sharp(e) \mid {\boldsymbol{M}_i}^\sharp \in S^\sharp \wedge e \in [-1,1]^n\}$$

zonotopes encode convex sets of points in $\mathbb{R}^d$. We can think of a symbolic matrix $M^\sharp \in \Psi^{n\times m}$ as a $n \cdot m$-dimensional polynomial zonotope. Similarly, a set of symbolic matrices $S^\sharp = \{{\boldsymbol{M}_i}^\sharp \in \mathbb{R}^{n_i \times m_i}\}$ can be thought of as a l-dimensional polynomial zonotope $\mathbb{R}^l$ where $l = \sum_i n_i \cdot m_i$. If every symbolic expression in a matrix or set of matrices is a linear combination of error symbols (an affine form), then such objects can equivalently be represented as zonotopes.

For instance, below on the left we show a zonotope matrix $\boldsymbol{W}^\sharp$ (all expressions are linear) and two possible worlds in its concretization (for assignments $[-1,-1]$ and $[0,0.5]$).

$$\boldsymbol{W}^\sharp = \begin{bmatrix} \epsilon_1 + 3 & \epsilon_2 \\ 15 & 22\epsilon_1 \end{bmatrix} \qquad \boldsymbol{W}^\sharp\left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 2 & -1 \\ 15 & -22 \end{bmatrix} \qquad \boldsymbol{W}^\sharp\left(\begin{bmatrix} 0 \\ 0.5 \end{bmatrix}\right) = \begin{bmatrix} 3 & 0.5 \\ 15 & 11 \end{bmatrix}$$

C.3 Abstract Training Data and Model Weights

Using possible world semantics to compute all possible model weights $\boldsymbol{w}^{\odot *}$ given a set of possible training datasets $\boldsymbol{D}^\odot$, there will be a natural correspondence between a model weight $\boldsymbol{w} \in \boldsymbol{w}^{\odot *}$ and the dataset $\boldsymbol{D} \in \boldsymbol{D}^\odot$ from which is was derived. When training in the abstract domain such correlations can be preserved by sharing error symbols between the abstract training data $\mathbf{D}^\sharp$ and corresponding model weights $\boldsymbol{w}^\sharp$. When such sharing occurs, then it is critical to reason about the joint concretization when determining whether an abstract fixed point as been achieved. Testing equivalence (equal concretization) of $\boldsymbol{w}^\sharp$ alone can lead to false positives, as ${\boldsymbol{w}^\sharp}_1 \simeq_\sharp {\boldsymbol{w}^\sharp}_2$ does not in general imply $(\mathbf{D}^\sharp, {\boldsymbol{w}^\sharp}_1) \simeq_\sharp (\mathbf{D}^\sharp, {\boldsymbol{w}^\sharp}_2)$. This is due to the fact that a particular concrete model weight $\boldsymbol{w} \in \gamma\left({\boldsymbol{w}^\sharp}_1\right) = \gamma\left({\boldsymbol{w}^\sharp}_2\right)$ may be associated with different datasets in $(\mathbf{D}^\sharp, {\boldsymbol{w}^\sharp}_1)$ and $\mathbf{D}^\sharp, {\boldsymbol{w}^\sharp}_2)$ because of shared error symbols between $\mathbf{D}^\sharp$ and the abstract model weights. Thus,

https://doi.org/10.52202/079017-3396

concretization equivalence between model weights only does not imply equivalent results after application of a gradient decent step.

To further illustrate this consider, two pairs of abstract model weights $\boldsymbol{w}^\sharp{}_1$ and $\boldsymbol{w}^\sharp{}_2$.

$$\left(\boldsymbol{w}^\sharp{}_1 = \begin{bmatrix} 2+2\epsilon_1 \\ 3+\epsilon_2 \end{bmatrix}, \mathbf{D}^\sharp = \begin{bmatrix} \epsilon_1 \\ 5 \end{bmatrix}\right)$$
$$\left(\boldsymbol{w}^\sharp{}_2 = \begin{bmatrix} 2+2\epsilon_3 \\ 3+\epsilon_2 \end{bmatrix}, \mathbf{D}^\sharp = \begin{bmatrix} \epsilon_1 \\ 5 \end{bmatrix}\right)$$

The only difference between $\boldsymbol{w}^\sharp{}_1$ and $\boldsymbol{w}^\sharp{}_2$ is that $\boldsymbol{w}^\sharp{}_1$ shares a variable (ϵ_1) with $\mathbf{D}^\sharp$ while $\boldsymbol{w}^\sharp{}_2$ does not. Observe that $\gamma\left(\boldsymbol{w}^\sharp{}_1\right) = \gamma\left(\boldsymbol{w}^\sharp{}_2\right)$. However, applying a step of gradient decent to $\boldsymbol{w}^\sharp{}_1$ and $\boldsymbol{w}^\sharp{}_2$ may lead to abstract models that are not equivalent. For sake of this example, consider a hypothetical gradient operator $\Phi^\sharp_{dummy}$ that subtracts the data from the current model weight.

$$\left(\Phi^\sharp_{dummy}(\boldsymbol{w}^\sharp{}_1) = \begin{bmatrix} 2+\epsilon_1 \\ -2+\epsilon_2 \end{bmatrix}, \mathbf{D}^\sharp = \begin{bmatrix} \epsilon_1 \\ 5 \end{bmatrix}\right)$$
$$\left(\Phi^\sharp_{dummy}(\boldsymbol{w}^\sharp{}_2) = \begin{bmatrix} 2+2\epsilon_3-\epsilon_1 \\ -2+\epsilon_2 \end{bmatrix}, \mathbf{D}^\sharp = \begin{bmatrix} \epsilon_1 \\ 5 \end{bmatrix}\right)$$

Note that $\Phi^\sharp_{dummy}(\boldsymbol{w}^\sharp{}_1) \not\simeq_\sharp \Phi^\sharp_{dummy}(\boldsymbol{w}^\sharp{}_2)$. Thus, even if two abstract model weights have equal concretization this does not guarantee that a fixed point has been reached when also taking the data into account. This is important as otherwise it is not possible that the abstract model weights contain all possible optimal model weights $\boldsymbol{w}^{\odot *}$. Instead, we need to consider the *joint concretization* of model weight and data, s.t. if $\gamma\left(S^\sharp{}_1\right) = \gamma\left(S^\sharp{}_2\right)$ then concretization equivalence is guaranteed to hold for all subsequent iterations.

D Examples Of Training Data Uncertainty and Abstraction

In this section, we discuss several causes of training (and test) data uncertainty, how to encode them as possible worlds, and abstraction functions for approximating such uncertainty in our symbolic model.

D.1 Measurement Uncertainty

Sensors typically have some measurement uncertainty. If the measurement error ϵ is know, e.g., provided by the instrument manufacturer or estimated through repeated measurements and calibration, then for a set of sensor readings used in training $\boldsymbol{D} = \{(\boldsymbol{x}_i, y_i)\}$, then each possible world in $\boldsymbol{D}^\odot$ is derived from $\boldsymbol{D}$ by replacing values $\boldsymbol{x}_i[j]$ with values in $\boldsymbol{x}_i[j] \pm \epsilon$.

D.2 Missing Values and Imputation

Consider a training dataset $\boldsymbol{D}$ where some of the features are missing for some of the datapoints in $\boldsymbol{D}$. If each feature $\mathcal{X}_i$'s domain is an interval $[l_i, u_i]$ and assuming that missing values can be represented as independent random variables, then the set of possible worlds of the training data $\boldsymbol{D}^\odot$ are all training datasets that can be derived from $\boldsymbol{D}$ by replacing each missing value in a feature $\mathcal{X}_i$ with a value from $[l_i, u_i]$.

We can use an abstraction function $\alpha_{missing}$ that represents each missing value as an interval $[l_i, u_i]$. In the symbolic representation, this is encoded as the central point of the interval of an error symbol ϵ with a coefficient half of the interval's length. We associate a separate error symbol with each missing value. For $\boldsymbol{D} \in \mathbb{R}^{n \times m}$, we define $\alpha_{missing}$ for $i \in [1, n]$ and $j \in [1, m]$ as:

$$\alpha_{missing}(\boldsymbol{D}_{ij}) = \begin{cases} \frac{u_j + l_j}{2} + \frac{u_j - l_j}{2}\epsilon_{i,j} & \textbf{if } \boldsymbol{D}_{ij} = \perp \\ \boldsymbol{D}_{ij} & \textbf{otherwise} \end{cases}$$

For instance, consider the training dataset $\boldsymbol{D} \in \mathbb{R}^{3\times 2}$ shown below where $\perp$ denotes a missing value.

$$\boldsymbol{D} = \begin{pmatrix} 3 & \perp \\ 1 & 6 \\ \perp & 9 \end{pmatrix}$$

Assuming that both features have a domain $[0, 10]$, we get:

$$\alpha_{missing}(\boldsymbol{D}) = \begin{pmatrix} 3 & 5 + 5\epsilon_{1,2} \\ 1 & 6 \\ 5 + 5\epsilon_{3,1} & 9 \end{pmatrix}$$

If the independence assumption on missing values holds, then $\gamma\left(\alpha_{missing}(\boldsymbol{D})\right) = \boldsymbol{D}^{\odot}$. If the independence assumption does not hold, then $\boldsymbol{D}^{\odot}$ would be a subset of the worlds described above and $\alpha_{missing}(\boldsymbol{D})$ is a still a valid abstraction function, albeit an over-approximating one:

$$\gamma\left(\alpha_{missing}(\boldsymbol{D})\right) \supseteq \boldsymbol{D}^{\odot}$$

Imputation If we make the stronger assumption the unknown ground truth value corresponding to a missing value is in a set of estimations $\{a_1, \ldots, a_m\}$ returned by a set of imputation methods, then we can use $[min(\{a_i\}), max(\{a_i\})]$ instead of $[l_i, u_i]$.

Training Label Uncertainty The abstract transformer $\alpha_{missing}$ can also be used if training data labels are missing.

E Abstract Transformers for Zonotopes

In this section we introduce (exact) abstract transformers for (polynomial) zonotopes that have been introduced in related work and discuss their computation complexity and the space requirements for the output zonotope $\boldsymbol{z}^{\sharp}$ in terms of its order $\text{ORD}(\boldsymbol{z}^{\sharp})$ (see Sec. 2.2). Kochdumper et al. [37, Table 1] shows an overview of which operations are exact for linear and polynomial zonotopes (and other set representations). Relevant to for our purpose is that exact transformers exist for polynomial zonotopes for all operations used in the learning algorithms for linear models we consider.

Proposition E.1 (Exact Transformers for Polynomial Zonotopes)**.** *There exist exact abstract transformers for scalar addition and multiplication as well as for matrix addition and multiplication for polynomial zonotopes [3]. There exist exact transformers for scalar addition and matrix addition for zonotopes. Abstract transformers for multiplication and matrix multiplication for zonotopes exist, but are not exact.*

We present the details of these operations in the following.

E.1 Arithmetic Operations

Addition. Scalar and matrix addition are exact in both zonotopes and polynomial zonotopes. Given two matrix (polynomial) zonotopes $\boldsymbol{V}^{\sharp}$ and $\boldsymbol{W}^{\sharp}$ in $\Psi^{n\times m}$, their addition $\boldsymbol{Z}^{\sharp} = \boldsymbol{V}^{\sharp} + \boldsymbol{W}^{\sharp}$ is defined by adding entries. For each $i \in [1, n]$ and $j \in [1, m]$:

$$\boldsymbol{Z}^{\sharp}{}_{ij} = \boldsymbol{V}^{\sharp}{}_{ij} + \boldsymbol{W}^{\sharp}{}_{ij}$$

The order of $\boldsymbol{Z}^{\sharp}$ is the sum of the orders of $\boldsymbol{V}^{\sharp}$ and $\boldsymbol{W}^{\sharp}$:

$$\text{ORD}(\boldsymbol{Z}^{\sharp}) = \text{ORD}(\boldsymbol{V}^{\sharp}) + \text{ORD}(\boldsymbol{W}^{\sharp})$$

Scalar multiplication. Multiplying a (polynomial) zonotope matrix $\boldsymbol{W}^{\sharp}$ with a scalar c is exact using the abstract transformer $\cdot_{poly}$ defined below. We simply multiply each entry in the matrix by c.For each $i \in [1, n]$ and $j \in [1, m]$:

$$(c \cdot_{poly} \boldsymbol{W}^{\sharp})_{ij} = c \cdot \boldsymbol{W}^{\sharp}{}_{ij}$$

https://doi.org/10.52202/079017-3396

Multiplication with a scalar abstract value $d \in \Psi$ is exact for polynomial zonotope matrices, but increases the order of the input zonotope:

$$\text{ORD}(d \cdot \boldsymbol{W}) = \text{ORD}(d) \cdot \text{ORD}(\boldsymbol{W})$$

Here we define the order of a scalar abstract value $d \in \Psi$ to be the number of monomials in the representation of d. For a linear zonotope $\boldsymbol{W}$, $d \cdot \boldsymbol{W}^\sharp{}_{ij}$ is in general not linear as it contains higher-order terms. Thus, matrix multiplication for linear zonotopes requires application of linearization (see App. G):

$$d \cdot_{Lin} \boldsymbol{W} = \mathbf{L}(d \cdot_{poly} \boldsymbol{W})$$

Matrix multiplication. Matrix multiplication is defined using scalar multiplication and addition. As discussed above, addition is exact for both linear and polynomial zonotopes. However, scalar multiplication of symbolic expressions is only exact for polynomial zonotopes, but requires linearization and, thus, over-approximation, for linear zonotopes. That is, matrix multiplication is exact for polynomial zonotopes only. Consider two matrices $\boldsymbol{V}^\sharp \in \Psi^{n \times m}$ and $\boldsymbol{V}^\sharp \in \Psi^{m \times k}$, then matrix multiplication is defined as usual, but using abstract transformers for scalar addition and multiplication. Each symbolic entry in the matrix $\boldsymbol{V}^\sharp \cdot \boldsymbol{W}^\sharp$ is a sum of m elements, each the multiplication of one entry of $\boldsymbol{V}^\sharp$ with one entry of $\boldsymbol{W}^\sharp$. Thus,

$$\text{ORD}(\boldsymbol{V}^\sharp \cdot \boldsymbol{W}^\sharp) = m \cdot \text{ORD}(\boldsymbol{V}^\sharp) \cdot \text{ORD}(\boldsymbol{W}^\sharp)$$

For linear zonotopes, we have to again apply linearization to make sure that the output is a linear zonotope.

F Abstract Transformers for Gradient Descent

F.1 Abstract Fixed Points Over-Approximate Possible Fixed Points

Proof of Prop. 3.3. Initially, we will assume that $\boldsymbol{w}^{\sharp\, j}$ is computed through repeated application of $F^\sharp$. Let n be the smallest number such that $\boldsymbol{w}^{\sharp\, *} = \boldsymbol{w}^{\sharp\, n} = F^\sharp(\boldsymbol{w}^{\sharp\, n-1}, \mathbf{D}^\sharp)$ for iteration with abstract transformer $F^\sharp$. As $F^\sharp$ is an abstract transformer for Φ and abstract transformers compose (Prop. B.1), we know that for every $\boldsymbol{D}_i \in \boldsymbol{D}^\odot$ and $n \in \mathbb{N}$, we have $(\boldsymbol{w}_i^n, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$. To prove the claim it is sufficient to show that for every such $\boldsymbol{D}_i$ we have $(\boldsymbol{w}_i^*, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$. WLOG consider some $\boldsymbol{D}_i \in \boldsymbol{D}^\odot$ and $(\boldsymbol{w}_i^n, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$. Let $m \in \mathbb{N}$ be the smallest number such that $\boldsymbol{w}_i^* = \boldsymbol{w}_j^m$. If $m \leq n$, then based on the fact that $F^\sharp$ is an abstract transformer (over-approximates Φ), the result holds as $(\boldsymbol{w}_i^n, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$. Now consider the case where $m > n$. We will show through induction that for all $j \in [n+1, m]$, $(\boldsymbol{w}_i^j, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$ and, thus $(\boldsymbol{w}_i^*, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$.

Induction start. For $j = n$ the result trivially holds based on the definition of abstract fixed points.

Induction step. Assume that $(\boldsymbol{w}_i^j, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$ for $j \in [n, m-1]$, we have to show that this implies that $(\boldsymbol{w}_i^{j+1}, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$. By definition, we have

$$\boldsymbol{w}_i^{j+1} = \Phi(\boldsymbol{w}_i^j)$$

As $F^\sharp$ is an abstract transformer for Φ, we have, $(\boldsymbol{w}_i^{j+1}, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n+1}, \mathbf{D}^\sharp\right)$. Now based on the fact that $\boldsymbol{w}^{\sharp\, n}$ is an abstract fixed point according to Def. 3.2 and, thus, $\gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right) \supseteq \gamma\left(\boldsymbol{w}^{\sharp\, n+1}, \mathbf{D}^\sharp\right)$, it follows that $(\boldsymbol{w}_i^{j+1}, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right)$.

So far we have demonstrated that a fixed point $\boldsymbol{w}^{\sharp\, *}$ that appears in the iteration sequence $\{\boldsymbol{w}^{\sharp\, j}\}_{j=0}^\infty$ contains all optimal model weights $\boldsymbol{w}^{\odot *}$. We now prove the stronger result that as long as $\boldsymbol{w}^{\sharp\, *}$ fulfills the condition of Def. 3.2, no matter it is the result of an iteration sequence using $F^\sharp$ or not, its concretization encloses $\boldsymbol{w}^{\odot *}$. Consider one $\boldsymbol{D}_i \in \gamma\left(\mathbf{D}^\sharp\right)$ and as above let $\boldsymbol{w}_i^*$ denote its optimal model weight. We will demonstrate that $(\boldsymbol{w}_i^*, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, *}, \mathbf{D}^\sharp\right)$. First note that given that gradient descent for linear models is convex, for any initial model weight $\boldsymbol{w}_i^0$, the sequence $\{\boldsymbol{w}_i^j\}_{j=0}^\infty$ converges to $\boldsymbol{w}_i^*$. Specifically, let $\boldsymbol{w}$ denote the model weight such that $(\boldsymbol{w}, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp\, *}, \mathbf{D}^\sharp\right)$

and let n_i denote the smallest integer such that $\boldsymbol{w}_i^{n_i} = \boldsymbol{w}_i^*$ for the sequence generated starting from $\boldsymbol{w}_i^0 = \boldsymbol{w}$. However, now we can apply the same proof by induction shown above to demonstrate that $(\boldsymbol{w}_i^{n_i}, \boldsymbol{D}_i) \in \gamma\left(\boldsymbol{w}^{\sharp *}, \mathbf{D}^\sharp\right)$. This concludes the proof. □

F.2 Exact Abstract Transformer for Gradient Descent

In this section we show that the abstract transformer for gradient descent introduced in sec. 3 has a fixed point.

Proposition F.1. *The abstract gradient descent operator $\Phi^\sharp$ has a fixed point $\boldsymbol{w}^{\sharp *}$.*

Proof of Prop. F.1. The existence of a fixed point is implied by the fact that $\Phi^\sharp$ is an exact abstract transformer of the concrete gradient descent operator Φ. Consider $\gamma\left(\boldsymbol{w}^{\sharp\, 0}, \mathbf{D}^\sharp\right) = \{(\boldsymbol{w}_i^0, \boldsymbol{D}_i)\}$. Let n_i be the smallest integer such that $\boldsymbol{w}_i^{n_i} = \boldsymbol{w}_i^*$ and let $n = \max_i n_i$, i.e., at iteration n, the concrete model weights have converged in every possible world in the concretization of $(\boldsymbol{w}^{\sharp\, 0}, \mathbf{D}^\sharp)$. As $\Phi^\sharp$ is an exact abstract transformer, we can show by induction that $\gamma\left(\boldsymbol{w}^{\sharp\, j}, \mathbf{D}^\sharp\right) = \{(\boldsymbol{w}_i^j, \boldsymbol{D}_i)\}$ for any j. As all computations in the concretization have converged at n, we know that $\boldsymbol{w}_i^{n+1} = \Phi(\boldsymbol{w}_i^n) = \boldsymbol{w}_i^n$ and, thus, $\{(\boldsymbol{w}_i^n, \boldsymbol{D}_i)\} = \{(\boldsymbol{w}_i^{n+1}, \boldsymbol{D}_i)\}$. As $\Phi^\sharp$ is exact, we get the desired result: $\gamma\left(\boldsymbol{w}^{\sharp\, n}, \mathbf{D}^\sharp\right) = \{(\boldsymbol{w}_i^n, \boldsymbol{D}_i)\} = \{(\boldsymbol{w}_i^{n+1}, \boldsymbol{D}_i)\} = \gamma\left(\boldsymbol{w}^{\sharp\, n+1}, \mathbf{D}^\sharp\right)$. □

F.3 Prediction with Abstract Model Weight Fixed Points

We now discuss how to use the abstract model weights $\boldsymbol{w}^{\sharp *}$ returned by our abstract transformer for learning linear models during inference to over-approximate the possible set of predictions for a test data point. We start by discussing test data that is not uncertain and then extend the discussion to the case where the test data is also uncertain.

Deterministic Test Data For now let us assume that the test data is not uncertain. The following corollary then enables us to use abstract gradient descent for inference and for over-approximating the prediction ranges.

Corollary F.2. *Let $f_{\boldsymbol{w}}(\boldsymbol{x})$ denote the linear model for parameters $\boldsymbol{w}$. Given an incomplete training dataset $\boldsymbol{D}$ associated with a set of possible worlds $\boldsymbol{D}^\odot$, the prediction range $V(\boldsymbol{x})$ for a test data point $\boldsymbol{x}$ can be over-approximated by:*

$$V^\sharp(\boldsymbol{x}) = \left[\min_{\boldsymbol{w} \in \gamma(\boldsymbol{w}^{\sharp *})} f_{\boldsymbol{w}}(\boldsymbol{x}), \max_{\boldsymbol{w} \in \gamma(\boldsymbol{w}^{\sharp *})} f_{\boldsymbol{w}}(\boldsymbol{x})\right],$$

where $\boldsymbol{w}^{\sharp *}$ is the fixed point of the abstract gradient descent operator $\Phi^\sharp$ applied to $\mathbf{D}^\sharp = \alpha(\boldsymbol{D}^\odot)$, the abstract representation of $\boldsymbol{D}^\odot$ in the zonotope domain.

The prediction returned by a linear model with parameters $\boldsymbol{w}$ for a data point $\boldsymbol{x}$ is $\boldsymbol{w}^T\boldsymbol{x}$. As $\boldsymbol{x}$ does not contain any symbolic term this is the sum of linear terms multiplied by constants, i.e., the result is a 1-dimensional linear zonotope which is a linear expression of the form:

$$c_0 + \sum_{i=1}^{m} c_i \epsilon_i$$

where $c_i \in R$ and $\epsilon_i \in \mathcal{E}$. The minimum and maximum value of a 1-d linear zonotope can be determined efficiently as shown below:

$$\left[c_0 - \sum_{i=1}^{m} |c_i|, c_0 + \sum_{i=1}^{m} |c_i|\right]$$

Uncertain Test Data In Sec. 2 we modeled uncertainty in the training data as sets of possible worlds $\boldsymbol{D}^\odot$. As mentioned in that section, our techniques also supports uncertain test data, i.e., both the training and test data may be uncertain.

https://doi.org/10.52202/079017-3396

In the most general case, the training and test data may be correlated.[1] We model this as a set of possible worlds $(\boldsymbol{D}^{\odot}, \mathbf{X}_{\text{test}}{}^{\odot})$ where each world is a pair of a training dataset $\boldsymbol{D}_i$ and a test dataset $\mathbf{X}_{\text{test}\,i}$:

$$(\boldsymbol{D}, \mathbf{X}_{\text{test}})^{\odot} = \{(\boldsymbol{D}_1, \mathbf{X}_{\text{test}\,1}), \ldots, (\boldsymbol{D}_m, \mathbf{X}_{\text{test}\,m})\}$$

If training and test data are independent, then we can specify their worlds separately ($\boldsymbol{D}^{\odot}$ and $\mathbf{X}_{\text{test}}{}^{\odot}$) and assume the worlds of $(\boldsymbol{D}, \mathbf{X}_{\text{test}})^{\odot}$ to be their cross product. Uncertainty propagation for inference then requires us to compute the set of possible predictions:

$$\boldsymbol{y}^{\odot} = \{f(\mathbf{X}_{\text{test}\,i}) \mid \exists (\boldsymbol{D}_i, \mathbf{X}_{\text{test}\,i}) \in (\boldsymbol{D}, \mathbf{X}_{\text{test}})^{\odot} : f_i = \mathcal{A}(\boldsymbol{D}_i)\}$$

For inference in the abstract domain we first have to select an appropriate abstraction function for the test data, e.g., using the same abstraction function α we use for the training data. The only difference to the case discussed above is that a test data point is now also an abstract element $\boldsymbol{x}^{\sharp}$ and the prediction $\boldsymbol{w}^{\sharp}\boldsymbol{x}^{\sharp}$ is a polynomial zonotope as it contains higher order terms that are the result of multiplying linear terms. To efficiently determine the minimum and maximum we can, e.g., employ linearization to map the polynomial zonotope into a linear one and apply the solution described above for finding the minimum and maximum of a linear zonotope.

G Linearization and Order Reduction Techniques

G.1 Linearization

The purpose of linearization is to over-approximate an input polynomial zonotope with a linear zonotope.

Definition G.1 (Linearization Operator $\mathbf{L}$)**.** *A linearization operator* $\mathbf{L}$ *maps a polynomial zonotope* $\boldsymbol{z}^{\sharp}$ *to a linear zonotope* $\boldsymbol{\ell}^{\sharp}$*. It replaces high-order polynomial terms with new error symbols, ensuring that all expressions are linear while maintaining an over-approximation:*

$$\gamma\left(\mathbf{L}(\boldsymbol{z}^{\sharp})\right) \supseteq \gamma\left(\boldsymbol{z}^{\sharp}\right).$$

During inference, computing the prediction intervals using the linear zonotope representation can be done efficiently with linear programming. However, more importantly, in our construction of abstract fixed points we use a specific order reduction technique that requires prior linearization in each gradient descent step to enforce the existence of a fixed point. For a d-dimensional polynomial zonotope $\boldsymbol{z}^{\sharp}$ with a set of monomials $\mathcal{S}$:

$$\boldsymbol{z}^{\sharp} = \boldsymbol{c} + \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{g}_i \mathcal{S}[i],$$

we have its linearization:

$$\mathbf{L}(\boldsymbol{z}^{\sharp}) = \boldsymbol{c} + \underbrace{\sum_{i \in \sigma_l} \boldsymbol{g}_i \mathcal{S}[i]}_{\textbf{Linear monomials}} + \underbrace{\sum_{i \notin \sigma_l} \boldsymbol{g}_i \epsilon_i'}_{\textbf{Replaced with linear monomials}}$$

where σ_l denotes the set of indices of all linear terms in $\boldsymbol{z}^{\sharp}$. Here, the third part over-approximates each high-order term in $\boldsymbol{z}^{\sharp}$ by replacing it with a new error symbol.

G.2 Order Reduction Operators

Order reduction operators are used to reduce the representation size of a zonotope. That is, an order reduction operator takes as input a linear zonotope $\boldsymbol{\ell}^{\sharp}$ and return a linear zonotope of smaller order

[1] Note that this does not necessarily imply a violation of the i.i.d. assumption. For instance, consider a dataset with a textual feature race that is first translated into a categorical feature that is then one-hot encoded into multiple binary attributes. If we are uncertain about the meaning of a particular value of the original attribute, then this leads to a correlation between the uncertainty of both datasets as the interpretation of this value affects all data points with that particular value in the race feature before preprocessing.

(smaller representation size) that over-approximates $\ell^\sharp$. For linear zonotopes this means that order reduction operators reduce the number of distinct error symbols that occur in a zonotope by merging error symbols.

Definition G.2 (Order Reduction Operator $\mathbf{R}$)**.** *An order reduction operator* $\mathbf{R}$ *takes a linear zonotope* $\ell^\sharp$ *as input and returns another linear zonotope of reduced order:*

$$\gamma\left(\mathbf{R}(\ell^\sharp)\right) \supseteq \gamma\left(\ell^\sharp\right) \qquad\qquad \text{ORD}(\mathbf{R}(\ell^\sharp)) < \text{ORD}(\ell^\sharp)$$

We now present details about two commonly adopted order reduction techniques: *Interval Hull* (*IH*) and *transformation-based Interval Hull* (*TIH*) in App. G. In this section, we use the geometric representation of zonotopes (see Def. C.2). Note that for a linear zonotope, each $\mathcal{S}[i]$ consists of a single error term ϵ_i. Thus, we can write such a zonotope $\ell^\sharp$ as:

$$\ell^\sharp = \boldsymbol{c} + \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{g}_i \epsilon_i$$

IH merges a selected subset δ_s of the error symbols of a zonotope and their corresponding generator vectors. For a d-dimensional zonotope $\ell^\sharp$:

$$\mathbf{R}_{IH}(\delta_s, \ell^\sharp) = \mathbf{R}_{IH}\left(\delta_s, \boldsymbol{c} + \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{g}_i \epsilon_i\right) = \boldsymbol{c} + \underbrace{\sum_{i \notin \delta_s} \boldsymbol{g}_i \epsilon_i}_{\textbf{Retained error symbols}} + \underbrace{\begin{bmatrix} \left(\sum_{i \in \delta_s} |\boldsymbol{g}_i[1]|\right) \epsilon'_1 \\ \vdots \\ \left(\sum_{i \in \delta_s} |\boldsymbol{g}_i[d]|\right) \epsilon'_d \end{bmatrix}}_{\textbf{Over-approximated with } d\textbf{-dimensional box}}$$

The selected terms δ_s are merged into a d-dimensional box described by d new error symbols $\{\epsilon'_1, \cdots, \epsilon'_d\}$. We will drop δ_s if all error symbols of the input zonotope are selected.

The error symbols getting merged (δ_s) are often determined based on some heuristic, e.g., symbols with lowest coefficients. TIH $\mathbf{R}_{TIH}$ first projects the zonotope to another space using an invertible linear transformation matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, then applies IH, and finally projects the resulting zonotope back with $\boldsymbol{A}^{-1}$. IH is a special case of TIH where $\boldsymbol{A} = I$.

$$\mathbf{R}_{TIH}(\delta_s, \boldsymbol{A}, \boldsymbol{z}^\sharp) = \boldsymbol{A}^{-1} \mathbf{R}_{IH}(\boldsymbol{A}\boldsymbol{z}^\sharp).$$

Intuitively, the purpose of the linear transformation $\boldsymbol{A}$ is to project the zonotope to a space where its shape is closer to a box, to reduce the loss of precision brought by order reduction [2, 38]. One of the most widely used TIH is PCA-based order reduction, whose transformation is obtained from the PCA of the set of all generator vectors of the input zonotope [38, 51].

Order Reduction for PTIME Zonotope Training With linearization, the number of error symbols in the model weights zonotope $\boldsymbol{w}^\sharp$ still grows exponentially with rate $\mathcal{O}(p^2)$, leading to exponential time complexity for gradient descent. We can overcome this challenge through *order reduction*, which enforces the maximum number of terms in the symbolic expressions of model weights zonotopes [38]. Note that for a linear zonotope, each $\mathcal{S}[i]$ consists of a single error term ϵ_i. Thus, we can write such a zonotope $\ell^\sharp$ as:

$$\ell^\sharp = \boldsymbol{c} + \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{g}_i \epsilon_i$$

Order reduction $\mathbf{R}$ reduces the order (representation size) of a linear zonotope $\ell^\sharp$. This is achieved by *merging* error symbols in $\mathcal{S}$, while ensuring that the result over-approximation the input zonotope, i.e.,

$$\gamma\left(\mathbf{R}(\ell^\sharp)\right) \supseteq \gamma\left(\ell^\sharp\right)$$

https://doi.org/10.52202/079017-3396

Two commonly adopted order reduction approaches [38] are *Interval Hull* (*IH*), denoted as $\mathbf{R}_{IH}$, and *Transformation-based Interval Hull* (*TIH*), denoted as $\mathbf{R}_{TIH}$ [2, 38, 64]. Specifically, IH merges a set of error symbols $\delta_s \subseteq \{\epsilon_i\}$ and their corresponding generator vectors (here $\delta_k = \{\epsilon_i\} - \delta_s$):

$$\mathbf{R}_{IH}(\boldsymbol{z}^\sharp) = \mathbf{R}_{IH}\left(\boldsymbol{c} + \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{g}_i \mathcal{S}[i]\right) = \boldsymbol{c} + \sum_{i\in\delta_k} \boldsymbol{g}_i \mathcal{S}[i] + diag(\sum_{i\in\delta_s} |\boldsymbol{g}_i[1]|, \cdots, \sum_{i\in\delta_s} |\boldsymbol{g}_i[d]|) \begin{bmatrix} \epsilon'_1 \\ \vdots \\ \epsilon'_d \end{bmatrix}$$

The selected terms δ_s are merged into a d-dimensional box described by d new error symbols $\{\epsilon'_1, \cdots, \epsilon'_d\}$. The error symbols getting merged (δ_s) are often determined based on some heuristic [38], e.g., the symbols with lowest coefficients. Similar to IH, TIH also merges error terms using IH, but in some projected space. TIH first projects the zonotope to another space using an invertible linear transformation matrix $A \in \mathbb{R}^{d\times d}$, then conducts IH, and finally projects the resulting zonotope back into the original space with A^{-1}:

$$\mathbf{R}_{TIH}(\boldsymbol{z}^\sharp) = A^{-1}\mathbf{R}_{IH}(A\boldsymbol{z}^\sharp).$$

Intuitively, the linear transformation A aims to project the zonotope to a space where its shape is closer to a box, to reduce the loss of precision brought by order reduction [2, 38]. One of the most widely used TIH is PCA-based order reduction, whose transformation is obtained from the PCA of all generator vectors [38, 51].

H Efficient Abstract Gradient Descent with Order Reduction

As mentioned in sec. 4.1, to address the tractability issues with abstract gradient descent, we employ two key techniques: linearization and order reduction. We employ linearization at each step of gradient descent to ensure that the resulting abstract representation of model parameters remains a linear zonotope.

Given a linearization operator $\mathbf{L}$ and order reduction operator $\mathbf{R}$, we construct an abstract gradient descent operator, $\Phi^\sharp$:

$$\Phi^\sharp(\boldsymbol{w}^\sharp) = \mathbf{R}\Big(\boldsymbol{w}^\sharp - \mathbf{L}\big(\eta\nabla L(\boldsymbol{w}^\sharp)\big)\Big), \tag{9}$$

which ensures that the abstract representation size remains bounded while providing efficient over-approximation. This operator is an abstract transformer for the concrete gradient descent operator.

Proposition H.1. *For any linearization* $\mathbf{L}$ *and order reduction* $\mathbf{R}$*, the abstract gradient descent* $\Phi^\sharp$ *is an abstract transformer for the concrete gradient descent operator* Φ*. Formally, for any abstract* $\boldsymbol{w}^\sharp$*,*

$$\gamma\left(\Phi^\sharp(\boldsymbol{w}^\sharp)\right) \supseteq \Phi(\gamma\left(\boldsymbol{w}^\sharp\right)),$$

.

Proof. Given that abstract transformers compose (Prop. B.1), we can decompose Φ into separate steps and construct an abstract transformer for Φ by composing abstract transformers for the individual steps. We can inject identity functions anywhere into the computation without changing its result. Let $ident$ represent an identify function of an appropriate type and $F^\sharp$ be the abstract operator resulting from injecting identity functions as shown below, then:

$$F^\sharp(\boldsymbol{w}^\sharp) = ident\Big(\boldsymbol{w}^\sharp - ident\big(\eta\nabla L(\boldsymbol{w}^\sharp)\big)\Big) = \boldsymbol{w}^\sharp - \big(\eta\nabla L(\boldsymbol{w}^\sharp)\big) = \Phi^\sharp_{exact}(\boldsymbol{w}^\sharp)$$

Since $\Phi^\sharp_{exact}$ is an abstract transformer for Φ, so is $F^\sharp$. Now observe that both $\mathbf{L}$ and $\mathbf{R}$ are abstract transformers for $ident$ as

$$\gamma\left(\mathbf{L}(\boldsymbol{w}^\sharp)\right) \supseteq \gamma\left(\boldsymbol{w}^\sharp\right) = ident(\gamma\left(\boldsymbol{w}^\sharp\right))$$

Thus, $\Phi^\sharp$ is an abstract transformer for Φ. □

Note that Prop. 3.3 then implies that a fixed point $\boldsymbol{w}^{\sharp *}$ for $\Phi^\sharp$is an over-approximation of all possible model weights $\boldsymbol{w}^{\odot *}$:

$$\gamma\left(\boldsymbol{w}^{\sharp *}\right) \supseteq \boldsymbol{w}^{\odot *}$$

H.1 Abstract Gradient Descent With Order Reduction And Fixed Points

While $\Phi^\sharp$ ensures that every step of gradient descent can be computed efficiently as we bound the order of the resulting zonotope in each step, this operator typically does not have a fixed point and even if it does, we still would have to solve an NP-hard problem [39] to detected that we have converged. The reason for the lack of a fixed point is that both linearization and order reduction results in over-approximation and the over-approximation error may grow in each iteration. Recall that we did show a real example of where this operator diverges in Sec. 4.1.

I An Efficient Approximate Abstract Transformer for Ridge Regression

In this section and the next section we present the proof of our main technical result: Thm. 4.2. Recall that Thm. 4.2 states that Alg. 1 computes abstract model parameters $\boldsymbol{w}^{\sharp *}$ that are a fixed point for the abstract transformer $\Phi^\sharp$using a closed form solution. Because $\Phi^\sharp$was shown to be an abstract transformer for gradient decent this then implies that $\boldsymbol{w}^{\sharp *}$ over-approximates all possible model weights $\boldsymbol{w}^{\odot *}$. We start by presenting additional details of the decomposition we employ to force a fixed points, formally prove that the condition on $\boldsymbol{w}^{\sharp *}$ from Prop. 4.1 is a sufficient for $\boldsymbol{w}^{\sharp *}$ being a fixed point for $\Phi^\sharp$, and then develop a closed form solution that requires solving a system of linear equations. For $\boldsymbol{w}_N^\sharp$, the parts of the abstract model weights that exclusively contains symbols that do not appear in the abstract training dataset $\mathbf{D}^\sharp$, the equation system only has a solution if the regularization coefficient λ is larger then or equal to a constant β that depends on $\mathbf{D}^\sharp$. Based on our experience and extensive experimental evaluation, we typically have $\beta = 0$, i.e., the closed form solution exists for any regularization coefficient λ. Nonetheless, we present a technique for achieving any desired $\beta \geq 0$ by splitting zonotopes into smaller parts with lower β, computing fixed points for each split individually, and merging the final result. These techniques will be presented in app. J.

I.1 Decomposing Fixed Point Equations For Abstract Gradient Descent

We now present additional details about our decomposition of the gradient from Sec. 4.1 for linear regression with ℓ_2 regularization (ridge regression). The loss function L for ridge regression an ℓ_2 penalty, is given by:

$$L(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) = \frac{1}{n}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + \lambda \cdot \boldsymbol{w}^T\boldsymbol{w},$$

where λ is the regularization coefficient (see App. A for details).

Recall that we observed that an abstract model weight zonotope $\boldsymbol{w}^\sharp$ can be decomposed into parts that can be dealt with separately in the sense that we will show that a sufficient condition for achieving a fixed point $\boldsymbol{w}^{\sharp *}$ is that each component has a fixed point. Given an abstract training dataset $\mathbf{D}^\sharp$, we use $\boldsymbol{X}^\sharp$ and $\boldsymbol{y}^\sharp$ to denote its feature matrix and labels. We decompose them into real (concrete) and symbolic components:

$$\boldsymbol{X}^\sharp = \boldsymbol{X}_R + \boldsymbol{X}_S^\sharp$$
$$\boldsymbol{y}^\sharp = \boldsymbol{y}_R + \boldsymbol{y}_S^\sharp$$

where $\boldsymbol{X}_R \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y}_R \in \mathbb{R}^n$ represent the real centers of zonotopes $\boldsymbol{X}^\sharp$ and $\boldsymbol{y}^\sharp$, while $\boldsymbol{X}_S^\sharp \in \Lambda^{n \times d}$ and $\boldsymbol{y}_S^\sharp \in \Lambda^n$ contain the symbolic terms.

Similarly, we decompose the abstract model weights at iteration i into real and symbolic components:

$$\boldsymbol{w}^{\sharp\, i} = \boldsymbol{w}_R^i + \boldsymbol{w}_S^{\sharp\, i},$$

where $\boldsymbol{w}_R^i \in \mathbb{R}^d$ represents the real center, and $\boldsymbol{w}_S^{\sharp\, i} \in \Lambda^d$ contains the symbolic terms. The symbolic terms are further decomposed into those containing data symbols ($\boldsymbol{w}_D^{\sharp\, i}$), i.e., symbols that are shared with $\mathbf{D}^\sharp$, and those introduced via linearization and order reduction in linear abstract gradient descent ($\boldsymbol{w}_N^{\sharp\, i}$), i.e., that do not appear in $\mathbf{D}^\sharp$.

$$\boldsymbol{w}_S^{\sharp\, i} = \boldsymbol{w}_D^{\sharp\, i} + \boldsymbol{w}_N^{\sharp\, i}.$$

https://doi.org/10.52202/079017-3396

Accordingly, we decompose the abstract gradient presented in Eq. (9) into several distinct components: real numbers ($\mathcal{G}_R$), linear symbolic expressions that share symbols with $\mathbf{D}^\sharp$ ($\mathcal{G}_L^D$), linear symbolic expressions that do not share symbols with $\mathbf{D}^\sharp$ ($\mathcal{G}_L^N$), and high-order symbolic expressions ($\mathcal{G}_H$). Using the loss function for ridge regression (see Eq. (5) in App. A):

$$\begin{aligned}\Phi^\sharp(\boldsymbol{w}^\sharp) &= \mathbf{R}\Big(\boldsymbol{w}^\sharp - \mathbf{L}(\eta\nabla L(\boldsymbol{w}^\sharp))\Big) = \mathbf{R}\Big(\boldsymbol{w}^\sharp - \eta\mathbf{L}\Big(\frac{2}{n}({\boldsymbol{X}^\sharp}^T\boldsymbol{X}^\sharp\boldsymbol{w}^\sharp - {\boldsymbol{X}^\sharp}^T\boldsymbol{y}^\sharp) + 2\lambda\boldsymbol{w}^\sharp\Big)\Big)\\ &= \mathbf{R}\Big(\boldsymbol{w}_R + \boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp - \eta\mathbf{L}\Big(\frac{2}{n}\Big((\boldsymbol{X}_R + \boldsymbol{X}_S^\sharp)^T(\boldsymbol{X}_R + \boldsymbol{X}_S^\sharp)(\boldsymbol{w}_R + \boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp)\\ &\qquad - (\boldsymbol{X}_R + \boldsymbol{X}_S^\sharp)^T(\boldsymbol{y}_R + \boldsymbol{y}_S^\sharp)\Big) + 2\lambda(\boldsymbol{w}_R + \boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp)\Big)\Big) \qquad (10)\\ &= \mathbf{R}\Big(\boldsymbol{w}_R - \eta\cdot\mathcal{G}_R + \boldsymbol{w}_D^\sharp - \eta\cdot\mathcal{G}_L^D + \boldsymbol{w}_N^\sharp - \eta\cdot\mathcal{G}_L^N - \eta\cdot\mathbf{L}(\mathcal{G}_H)\Big) \qquad (11)\end{aligned}$$

where

$$\mathcal{G}_R = \frac{2}{n}({\boldsymbol{X}_R}^T\boldsymbol{X}_R\boldsymbol{w}_R - {\boldsymbol{X}_R}^T\boldsymbol{y}_R) + 2\lambda\boldsymbol{w}_R \qquad (12)$$

$$\mathcal{G}_L^D = (2\lambda I + \frac{2}{n}{\boldsymbol{X}_R}^T\boldsymbol{X}_R)\boldsymbol{w}_D^\sharp + \frac{2}{n}({\boldsymbol{X}_R}^T\boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T\boldsymbol{X}_R)\boldsymbol{w}_R - \frac{2}{n}\boldsymbol{X}_S^\sharp\boldsymbol{y}_R - \frac{2}{n}{\boldsymbol{X}_R}^T\boldsymbol{y}_S^\sharp \qquad (13)$$

$$\mathcal{G}_L^N = (2\lambda I + \frac{2}{n}{\boldsymbol{X}_R}^T\boldsymbol{X}_R)\boldsymbol{w}_N^\sharp \qquad (14)$$

$$\mathcal{G}_H = \frac{2}{n}\Big(({\boldsymbol{X}_R}^T\boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T\boldsymbol{X}_R)(\boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp) + {\boldsymbol{X}_S^\sharp}^T\boldsymbol{X}_S^\sharp(\boldsymbol{w}_R + \boldsymbol{w}_D^\sharp + \boldsymbol{w}_N^\sharp) - {\boldsymbol{X}_S^\sharp}^T\boldsymbol{y}_S^\sharp\Big) \qquad (15)$$

In Eq. (10) we substitute definitions which are further expanded in Eq. (11) using new notation for rearranged parts of the gradient (Eq. (12) to (15)). Furthermore, in this step we also use the fact that linearization only affects higher order terms and $\mathcal{G}_H$ is the only component of the gradient with non-linear terms. Now consider the relationship of Eq. (12) to (15) to

Enforcing Fixed Points With Parameterized Order Reduction. We now introduce an order reduction operator $\mathbf{R}_{\boldsymbol{A}}$ that is a specific type of TIH (see App. G) which only merges non-data symbols and is parameterized by its transformation matrix $\boldsymbol{A}$, identify a sufficient condition for achieving an abstract fixed point for gradient descent with this order reduction operator, and demonstrate that the fixed point exists for any coefficient λ of l_2-regularization that is larger than a data-dependent constant β. Furthermore, we show how such a fixed point $\boldsymbol{w}^{\sharp *}$ can be computed using closed form solutions for $\boldsymbol{w}_R$, $\boldsymbol{w}_D^\sharp$ and $\boldsymbol{w}_N^\sharp$.

Recall from App. G that the TIH order reduction operator $\mathbf{R}_{TIH}(\delta_s, \boldsymbol{A})$ is parameterized by δ_s (the set of error symbols to merge) and a transformation matrix $\boldsymbol{A}$. Consider the input to order reduction as shown in Eq. (10).We use $\mathcal{E}_D$ to denote the error symbols that this input shares with the data (that occur in $\mathbf{D}^\sharp$) and $\mathcal{E}_N$ to denote those that only occur in the input to order reduction. Then we define $\mathbf{R}_{\boldsymbol{A}} = \mathbf{R}_{TIH}(\mathcal{E}_N, \boldsymbol{A})$, i.e., we only merge symbols that are not shared with the data to ensure that the correspondence between model weights and possible worlds is preserved by letting $\boldsymbol{w}^\sharp$ share symbols in linear expressions with $\mathbf{D}^\sharp$. The linear part that does not share symbols with $\mathbf{D}^\sharp$ has $O(p^2q)$ monomials (where $p = |\,\mathcal{E}_D\,|$ and q is the number of error symbols in $\boldsymbol{w}^\sharp$). Now expanding the definition of abstract gradient descent with order reduction using $\mathbf{R}_{\boldsymbol{A}}$ as the order reduction operator (for some given transformation matrix $\boldsymbol{A}$), we get:

$$\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^\sharp) = \mathbf{R}_{\boldsymbol{A}}(\boldsymbol{w}_R - \eta\mathcal{G}_R + \boldsymbol{w}_D^\sharp - \eta\mathcal{G}_L^D + \boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))) \qquad (16)$$

$$= \boldsymbol{w}_R - \eta\mathcal{G}_R + \boldsymbol{w}_D^\sharp - \eta\mathcal{G}_L^D + \mathbf{R}_{\boldsymbol{A}}(\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))) \qquad (17)$$

$$= \underbrace{\boldsymbol{w}_R - \eta\mathcal{G}_R}_{\Phi_R(\boldsymbol{w}_R)} + \underbrace{\boldsymbol{w}_D^\sharp - \eta\mathcal{G}_L^D}_{\Phi^\sharp{}_D(\boldsymbol{w}_R,\boldsymbol{w}_D^\sharp)} + \underbrace{\boldsymbol{A}^{-1}\big(\mathbf{R}_{IH}(\boldsymbol{A}(\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H)))))\big)}_{\Phi^\sharp{}_N(\boldsymbol{w}_R,\boldsymbol{w}_D^\sharp,\boldsymbol{w}_N^\sharp)} \qquad (18)$$

Eq. (18) shows the components of $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^\sharp)$ and related them to the notation $\Phi_R(\boldsymbol{w}_R)$, $\Phi^\sharp{}_D(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp)$, and $\Phi^\sharp{}_N(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp, \boldsymbol{w}_N^\sharp)$ we introduced in Eq. (2). Specifically, $\Phi_R(\boldsymbol{w}_R) = (\boldsymbol{w}_R - \eta\mathcal{G}_R)$ does not contain any error symbols and, thus, corresponds to the real number part of the updated model weight. Similarly, $\Phi^\sharp{}_D(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp) = (\boldsymbol{w}_D^\sharp - \eta\mathcal{G}_L^D)$ only contains linear symbolic expression with symbols that are from $\mathbf{D}^\sharp$. The remaining part, $\Phi^\sharp{}_N(\boldsymbol{w}_R, \boldsymbol{w}_D^\sharp, \boldsymbol{w}_N^\sharp) =$

$\boldsymbol{A}\big(\mathbf{R}_{IH}(\boldsymbol{A}^{-1}(\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))))\big)$ does not share symbols with $\mathbf{D}^\sharp$, as it consists of new symbols generated from order reduction.

I.2 Constructing Fixed Points For Abstract Gradient Descent

Having defined $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}$ which was denoted by $\Phi^\sharp_{(\mathbf{L},\mathbf{R})}$ in app. H, we are now ready to state the first part of our main technical result: under some mild assumptions on λ being larger than some data-dependent constant β, $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}$ has a fixed point and, importantly, this fixed point can be computed efficiently. In practice, we often have $\beta = 0$. However, if this is not the case, we demonstrate later that at the cost of reduced computational efficiency we can reduce β by splitting $\mathbf{D}^\sharp$ into several zonotopes and solving the problem independently for each zonotope as we will details in App. J.

Theorem I.1 (Existence of Abstract Fixed Points)**.** *Consider an abstract training dataset $\mathbf{D}^\sharp$. There exists a constant β specific to $\mathbf{D}^\sharp$ such that $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}$ has a fixed point for any $\lambda \geq \beta$.*

In the remainder of this section, we prove Thm. I.1 by demonstrating how to construct such a fixed point efficiently. We start by identifying a sufficient condition for $\boldsymbol{w}^{\sharp *}$ to be a fixed point, then demonstrate that the problem of finding a $\boldsymbol{w}^{\sharp *}$ that fulfills this sufficient condition can be decomposed based on our decomposition $\boldsymbol{w}^{\sharp *} = \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$ presented earlier. Specifically, certain components are independent of other components suggesting an evaluation order where we find a fixed point for one component treating the previously computed fixed points for other components as constants. Then we proceed to prove closed form solutions for $\boldsymbol{w}_R^*$ and for $\boldsymbol{w}_D^{\sharp *}$ given $\boldsymbol{w}_R^*$. Finally, given $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp *}$ we construct a system of equations whose solution gives a fixed point for $\boldsymbol{w}_N^{\sharp *}$ and demonstrate that this system of equations has a closed form solution for any $\lambda \geq \beta$.

A Sufficient Condition for Abstract Fixed Points. Recall from Def. 3.2 that the fixed point $\boldsymbol{w}^{\sharp *}$ must satisfy $\gamma\left(\boldsymbol{w}^{\sharp *}, \mathbf{D}^\sharp\right) \supseteq \gamma\left(\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^{\sharp *}), \mathbf{D}^\sharp\right)$. We now present a sufficient condition for this to hold. Intuitively this condition requires the **invariance** of different components of $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^\sharp)$ when doing joint concretization with $\mathbf{D}^\sharp$.

Proposition I.2. *If an abstract model weight $\boldsymbol{w}^{\sharp *} = \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$ satisfies the following three conditions, then it is an abstract fixed point of the abstract gradient descent operator $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}$:*

$$\boldsymbol{w}_R^* = \boldsymbol{w}_R^* - \eta\mathcal{G}_R \tag{19}$$

$$\boldsymbol{w}_D^{\sharp *} = \boldsymbol{w}_D^{\sharp *} - \eta\mathcal{G}_L^D \tag{20}$$

$$\boldsymbol{w}_N^{\sharp *} \simeq_\sharp \mathbf{R}_{\boldsymbol{A}}\left(\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))\right). \tag{21}$$

Proof. Consider the definition of an abstract fixed point $\boldsymbol{w}^{\sharp *}$ (Def. 3.2 applied to $\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}$):

$$\gamma\left(\boldsymbol{w}^{\sharp *}, \mathbf{D}^\sharp\right) \supseteq \gamma\left(\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^{\sharp *}), \mathbf{D}^\sharp\right)$$

This can certainly be achieved if

$$\boldsymbol{w}^{\sharp *}, \mathbf{D}^\sharp \simeq_\sharp \Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^{\sharp *}), \mathbf{D}^\sharp$$

This requires equal joint concretization of the fixed point the pairs $(\boldsymbol{w}^{\sharp *}, \mathbf{D}^\sharp)$ and $(\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^{\sharp *}), \mathbf{D}^\sharp)$. To understand the need for joint concretization here note that certain model weights $\boldsymbol{w}$paired with certain datasets $\boldsymbol{D}$ in the joint concretization. If we would only consider equivalence of the abstract model weight, two zonotopes $\boldsymbol{w}^\sharp$ and $\boldsymbol{w}^{\sharp\prime}$ may be equivalent $\boldsymbol{w}^\sharp \simeq_\sharp \boldsymbol{w}^{\sharp\prime}$, but pair a particular model weight $\boldsymbol{w} \in \gamma\left(\boldsymbol{w}^\sharp\right) = \gamma\left(\boldsymbol{w}^{\sharp\prime}\right)$ with different datasets $\boldsymbol{D}_1 \in \gamma\left(\mathbf{D}^\sharp\right)$ and $\boldsymbol{D}_2 \in \gamma\left(\mathbf{D}^\sharp\right)$. Thus, it is possible that $\boldsymbol{w}^\sharp \simeq_\sharp \Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^\sharp)$ holds but $\gamma\left(\boldsymbol{w}^\sharp\right) = \gamma\left(\boldsymbol{w}^{\sharp\prime}\right)$ does not contain all model weights from $\boldsymbol{w}^{\odot *}$.

https://doi.org/10.52202/079017-3396

First observe that requiring equality of abstract elements is a stricter condition than equivalence wrt. $\simeq_\sharp$ as

$$\boldsymbol{w}^\sharp{}_1 = \boldsymbol{w}^\sharp{}_2 \Rightarrow \boldsymbol{w}^\sharp{}_1 \simeq_\sharp \boldsymbol{w}^\sharp{}_2$$

Also note that $\boldsymbol{w}_D^\sharp$ is the only component of $\boldsymbol{w}^{\sharp\,*}$ that contains symbols. Thus, requiring equality in Eq. (20) is sufficient for ensuring equivalent join concretization with $\mathbf{D}^\sharp$ as long as Eq. (19) and Eq. (21) also hold. Furthermore, Eq. (19) and Eq. (20) imply:

$$\mathcal{G}_R = 0 \qquad\qquad \mathcal{G}_L^D = 0$$

Expanding $\boldsymbol{w}^{\sharp\,*} = \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp\,*} + \boldsymbol{w}_N^{\sharp\,*}$ and substituting into Eq. (18) we get

$$\begin{aligned}\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^{\sharp\,*}) &= \boldsymbol{w}_R^* - \eta\mathcal{G}_R + \boldsymbol{w}_D^{\sharp\,*} - \eta\mathcal{G}_L^D + \mathbf{R}_{\boldsymbol{A}}\big(\boldsymbol{w}_N^{\sharp\,*} - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))\big)\\ &= \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp\,*} + \mathbf{R}_{\boldsymbol{A}}\big(\boldsymbol{w}_N^{\sharp\,*} - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))\big)\end{aligned}$$

Thus, using Eq. (21) and the fact that $\boldsymbol{w}_N^\sharp$ does not share any symbols with $\mathbf{D}^\sharp$, we get the desired result:

$$(19) \wedge (20) \wedge \boldsymbol{w}_N^{\sharp\,*} \simeq_\sharp \mathbf{R}_{\boldsymbol{A}}\left(\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))\right) \Rightarrow (\boldsymbol{w}^{\sharp\,*}, \mathbf{D}^\sharp) \simeq_\sharp (\Phi^\sharp_{(\mathbf{L},\mathbf{R}_{\boldsymbol{A}})}(\boldsymbol{w}^{\sharp\,*}), \mathbf{D}^\sharp)$$

□

Independence of Fixed Point Components. Recall that we observed that some components of the gradient depend only on some components of $\boldsymbol{w}^{\sharp\,*}$. Specifically, $\mathcal{G}_R$ only depends on $\boldsymbol{w}_R^*$ and $\mathcal{G}_L^D$ only depends on $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp\,*}$. This implies that we can find a fixed point by first computing $\boldsymbol{w}_R^*$, then finding $\boldsymbol{w}_D^{\sharp\,*}$ treating $\boldsymbol{w}_R^*$ as a constant, and finally determine $\boldsymbol{w}_N^{\sharp\,*}$ treating both $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp\,*}$ as a constant.

Lemma I.3 (Independence of Fixed Point Components)**.** *The following independence relationships hold:*

(i) *$\mathcal{G}_R$ does not depend on $\boldsymbol{w}_D^{\sharp\,*}$ nor $\boldsymbol{w}_N^{\sharp\,*}$* ***(ii)*** *$\mathcal{G}_L^D$ does not depend on $\boldsymbol{w}_N^{\sharp\,*}$*

Proof. The definition of $\mathcal{G}_R$ is shown in Eq. (12). The only component of $\boldsymbol{w}^{\sharp\,*}$ that appears in the definition of $\mathcal{G}_R$ is $\boldsymbol{w}_R^*$. Thus, the equations defining $\boldsymbol{w}_R^*$ only depend on $\boldsymbol{w}_R^*$ and are independent of $\boldsymbol{w}_D^{\sharp\,*}$ and $\boldsymbol{w}_N^{\sharp\,*}$. Using a similar argument we can show that $\mathcal{G}_L^D$ does not depend on $\boldsymbol{w}_N^{\sharp\,*}$. □

Closed Form Solutions for Real And Linear Data Components of Abstract Model Weights. Before explaining how to determine $\boldsymbol{w}_N^{\sharp\,*}$, utilizing the lemma above we derive closed form solutions for $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp\,*}$.

Lemma I.4 (Closed Form Solutions for $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp\,*}$)**.** *Given $\mathbf{D}^\sharp$, $\boldsymbol{w}_R^*$ as defined below fulfills Eq. (19)*

$$\boldsymbol{w}_R^* = (\boldsymbol{X}_R{}^T\boldsymbol{X}_R + \lambda nI)^{-1}\boldsymbol{X}_R{}^T\boldsymbol{y}_R \tag{22}$$

Given $\mathbf{D}^\sharp$ and $\boldsymbol{w}_R^*$, $\boldsymbol{w}_D^{\sharp\,*}$ as defined below fulfills Eq. (20).

$$\boldsymbol{w}_D^{\sharp\,*} = (\boldsymbol{X}_R{}^T\boldsymbol{X}_R + \lambda nI)^{-1}\Big(\boldsymbol{X}_S^\sharp\boldsymbol{y}_R + \boldsymbol{X}_R{}^T\boldsymbol{y}_S^\sharp - (\boldsymbol{X}_R{}^T\boldsymbol{X}_S^\sharp + \boldsymbol{X}_S^{\sharp\,T}\boldsymbol{X}_R)\boldsymbol{w}_R^*\Big) \tag{23}$$

Proof. Fixed Point for $\boldsymbol{w}_R^*$. Substituting the definition of $\mathcal{G}_R$ from Eq. (12) into Eq. (21):

$$\begin{aligned}&\boldsymbol{w}_R^* = \boldsymbol{w}_R^* - \eta\cdot\Big(\frac{2}{n}(\boldsymbol{X}_R{}^T\boldsymbol{X}_R\boldsymbol{w}_R^* - \boldsymbol{X}_R{}^T\boldsymbol{y}_R) + 2\lambda\boldsymbol{w}_R^*\Big)\\ \Leftrightarrow&\frac{2\eta}{n}\cdot\Big(\boldsymbol{X}_R{}^T\boldsymbol{X}_R\boldsymbol{w}_R^* - \boldsymbol{X}_R{}^T\boldsymbol{y}_R + \lambda n\boldsymbol{w}_R^*\Big) = 0\\ \Leftrightarrow&\boldsymbol{X}_R{}^T\boldsymbol{X}_R\boldsymbol{w}_R^* - \boldsymbol{X}_R{}^T\boldsymbol{y}_R + \lambda n\boldsymbol{w}_R^* = 0 \qquad (\eta > 0)\\ \Leftrightarrow&(\boldsymbol{X}_R{}^T\boldsymbol{X}_R + \lambda nI)\boldsymbol{w}_R^* = \boldsymbol{X}_R{}^T\boldsymbol{y}_R\\ \Leftrightarrow&\boldsymbol{w}_R^* = (\boldsymbol{X}_R{}^T\boldsymbol{X}_R + \lambda nI)^{-1}\boldsymbol{X}_R{}^T\boldsymbol{y}_R \qquad (24)\end{aligned}$$

The last step relies on the fact that ${\boldsymbol{X}_R}^T \boldsymbol{X}_R + \lambda n I$ is invertible which is guaranteed for $\lambda > 0$ [26]. Note that, as expected since the fixed point equation for $\boldsymbol{w}_R^*$ does not contain any symbolic expressions, the closed form for $\boldsymbol{w}_R^*$ is equal to the well-known closed form for ridge regression.

Fixed Point for $\boldsymbol{w}_D^{\sharp *}$. Note that given the independence of $\boldsymbol{w}_R^*$ from $\boldsymbol{w}_D^{\sharp *}$, we can assume that $\boldsymbol{w}_R^*$ has been determined using the closed form solution shown above. Thus, we can treat $\boldsymbol{w}_R^*$ as a constant in the following derivation. We start by substituting the definition or Eq. (13) into Eq. (20).

$$\begin{aligned}
\boldsymbol{w}_D^{\sharp *} &= \boldsymbol{w}_D^{\sharp *} - \eta\Big((2\lambda I + \frac{2}{n}{\boldsymbol{X}_R}^T \boldsymbol{X}_R)\boldsymbol{w}_D^{\sharp *} + \frac{2}{n}({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)\boldsymbol{w}_R^* \\
&\qquad - \frac{2}{n}(\boldsymbol{X}_S^\sharp \boldsymbol{y}_R + {\boldsymbol{X}_R}^T \boldsymbol{y}_S^\sharp)\Big) \\
\Leftrightarrow 0 &= \frac{2\eta}{n}\Big((\boldsymbol{X}_R{}^T \boldsymbol{X}_R + \lambda n I)\boldsymbol{w}_D^{\sharp *} + (\boldsymbol{X}_R{}^T \boldsymbol{X}_S^\sharp + \boldsymbol{X}_S^\sharp{}^T \boldsymbol{X}_R)\boldsymbol{w}_R^* - \boldsymbol{X}_S^\sharp \boldsymbol{y}_R - \boldsymbol{X}_R{}^T \boldsymbol{y}_S^\sharp\Big) \\
\Leftrightarrow 0 &= (\lambda n I + {\boldsymbol{X}_R}^T \boldsymbol{X}_R)\boldsymbol{w}_D^{\sharp *} + ({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)\boldsymbol{w}_R^* - \boldsymbol{X}_S^\sharp \boldsymbol{y}_R - {\boldsymbol{X}_R}^T \boldsymbol{y}_S^\sharp \qquad (\eta > 0) \\
\Leftrightarrow (\lambda n I + {\boldsymbol{X}_R}^T \boldsymbol{X}_R)\boldsymbol{w}_D^{\sharp *} &= -\Big(({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)\boldsymbol{w}_R^* - \boldsymbol{X}_S^\sharp \boldsymbol{y}_R - {\boldsymbol{X}_R}^T \boldsymbol{y}_S^\sharp\Big) \\
\Leftrightarrow \boldsymbol{w}_D^{\sharp *} &= ({\boldsymbol{X}_R}^T \boldsymbol{X}_R + \lambda n I)^{-1}\Big(\boldsymbol{X}_S^\sharp \boldsymbol{y}_R + {\boldsymbol{X}_R}^T \boldsymbol{y}_S^\sharp - ({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)\boldsymbol{w}_R^*\Big) \qquad (25)
\end{aligned}$$

Note that matrix inversion is applied to $({\boldsymbol{X}_R}^T \boldsymbol{X}_R + \lambda n I)$ which does not contain any symbolic terms and as discussed above this matrix is guaranteed to be invertible. □

Analysis of the Fixed Point Equations for $\boldsymbol{w}_N^{\sharp *}$. Using the closed form solutions established in Lem. I.4, we can compute $\boldsymbol{w}_R^*$ and $\boldsymbol{w}_D^{\sharp *}$ fulfilling Eq. (19) and Eq. (20). In the remainder of this subsection we show how to compute a solution $\boldsymbol{w}_N^{\sharp *}$ to Eq. (21) for a given $\lambda \geq \beta$ (regularization coefficient), i.e., we prove Thm. I.1 that postulated the existence of such fixed points $\boldsymbol{w}^{\sharp *}$ constructively.

To find a fixed point $\boldsymbol{w}_N^{\sharp *}$, we are searching for an abstract model weight that fulfills Eq. (21):

$$\boldsymbol{w}_N^{\sharp *} \simeq_\sharp \mathbf{R}_{\boldsymbol{A}}\left(\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))\right) \qquad (26)$$

This implies that $\boldsymbol{w}_N^{\sharp *}$ needs to be equivalent to the result of $\mathbf{R}_{\boldsymbol{A}}$ on some input zonotope $\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H)$. Recall the definition of $\mathbf{R}_{\boldsymbol{A}}$ for a linear input zonotope $\boldsymbol{\ell}^\sharp$:

$$\mathbf{R}_{\boldsymbol{A}}(\boldsymbol{\ell}^\sharp) = \boldsymbol{A}^{-1}\mathbf{R}_{IH}(\boldsymbol{A}\boldsymbol{\ell}^\sharp) \qquad (27)$$

Note that $\mathbf{R}_{\boldsymbol{A}}$ applies $\mathbf{R}_{IH}$ to all error symbols in its input. Thus, $\mathbf{R}_{IH}(\cdot)$ is a d-dimensional box in the projected space into which the input $\boldsymbol{\ell}^\sharp$ is mapped into by $\boldsymbol{A}$. Furthermore, as $\boldsymbol{w}_N^\sharp - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H)$ does only contain symbolic terms, the interval hull (the box computed by $\mathbf{R}_{IH}$) is centered at $\mathbf{0}$.

Lemma I.5. *Any solution $\boldsymbol{w}_N^\sharp$ of Eq. (26) is equivalent to the image of a box $\boldsymbol{b}^\sharp$ centered at $\mathbf{0}$ produced by $\boldsymbol{A}^{-1}$, i.e.,*

$$\boldsymbol{w}_N^\sharp \simeq_\sharp \boldsymbol{A}^{-1}\boldsymbol{b}^\sharp$$

Proof. IH does not change the center of a zonotope and $\boldsymbol{A}$ is a linear map. Now observe that all terms in Eq. (14) and (15) are symbolic (contain error symbols). Thus, the RHS of Eq. (26) that is the input to order reduction is a zonotope with center $\mathbf{0}$. □

A linear zonotope that is a box in d-dimensional space with center $\mathbf{0}$ is, up to equivalence, uniquely determined by the diameter of the box in each dimension. That is, if we use $k_i \geq 0$ to denote the diameter of the box in the i^{th} dimension and $\boldsymbol{k} \in \mathbb{R}^d$ to denote the vector of these elements, then we can write any such zonotope $\boldsymbol{b}^\sharp$ as:

$$\boldsymbol{b}^\sharp = \begin{bmatrix} k_1\epsilon_1 \\ \vdots \\ k_d\epsilon_d \end{bmatrix} = \begin{bmatrix} k_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_d \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_d \end{bmatrix} = diag\{\boldsymbol{k}\} \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_d \end{bmatrix}$$

https://doi.org/10.52202/079017-3396

Combining Eq. (26) with Eq. (27), we know that $\boldsymbol{w}_N^{\sharp}$ is a zonotope that is equivalent to the image of $\boldsymbol{A}^{-1}$ on a box $\boldsymbol{b}^{\sharp}$ parameterized by $\boldsymbol{k}$. We will only consider solutions of this form:[2]

$$\boldsymbol{w}_N^{\sharp *} = \boldsymbol{A}^{-1} diag\{\boldsymbol{k}\} \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_d \end{bmatrix} = \sum_{i=1}^{d} k_i \boldsymbol{a}_i \epsilon_i \tag{28}$$

where $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_d$ are the column vectors of $\boldsymbol{A}^{-1}$. As the naming of error symbols is irrelevant, finding a solution to Eq. (26) now amounts to finding a vector $\boldsymbol{k}$ as the error symbols are now treated as fixed.

Now we can substitute the formula for $\boldsymbol{w}_N^{\sharp *}$ from Eq. (28) into the RHS of Eq. (26) as shown below. Recall from app. G.2 that IH merges different error symbols by adding up the absolute values of their coefficients. For the i^{th} dimension, each k_j will appear with a certain coefficient. For some terms, the symbolic expression for the i^{th} dimension may contain error symbols that do not have any k_l in their coefficients. Thus, without considering for now the precise values for each coefficient we know that the box produced by IH in the RHS is of the form shown below where $\boldsymbol{C} \in \mathbb{R}^{d \times d}$ is a matrix with non-negative entries $c_{i,j} \geq 0$ (at position (i, j)), and $\boldsymbol{c}_0 \in \mathbb{R}^d$ is a vector with non-negative elements $c_{i,0} \geq 0$ (at dimension i). Intuitively, $c_{i,j}$ is the coefficient of k_j in the i^{th} dimension and $c_{i,0}$ is the sum of the coefficients of error symbols that do not have any k_j in their coefficients (e.g., the fresh symbols introduced by linearization). Note since IH does calculate coefficients of the error symbols in the output as a sum of absolute values, we know that all entries of $\boldsymbol{C}$ and $\boldsymbol{c}_0$ are positive. Furthermore, recall that we assumed that $k_j \geq 0$.

$$\begin{aligned} \mathbf{R}_{\boldsymbol{A}}(\boldsymbol{w}_N^{\sharp} - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))) &= \boldsymbol{A}^{-1}\mathbf{R}_{IH}(\boldsymbol{A}(\boldsymbol{w}_N^{\sharp} - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H)))) \\ &= \boldsymbol{A}^{-1} \begin{bmatrix} (c_{1,0} + \sum_{j=1}^{d} c_{1,j} k_j)\epsilon'_1 \\ \vdots \\ (c_{d,0} + \sum_{j=1}^{d} c_{d,j} k_j)\epsilon'_d \end{bmatrix} = \boldsymbol{A}^{-1} diag\{\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{k}\} \begin{bmatrix} \epsilon'_1 \\ \vdots \\ \epsilon'_d \end{bmatrix}. \end{aligned} \tag{29}$$

Substituting the LHS and RHS of Eq. (21) with Eq. (28) and (29), we get:

$$\boldsymbol{A}^{-1} diag\{\boldsymbol{k}\} \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_d \end{bmatrix} \simeq_{\sharp} \boldsymbol{A}^{-1} diag\{\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{k}\} \begin{bmatrix} \epsilon'_1 \\ \vdots \\ \epsilon'_d \end{bmatrix}$$

$$\Leftrightarrow diag\{\boldsymbol{k}\} \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_d \end{bmatrix} \simeq_{\sharp} diag\{\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{k}\} \begin{bmatrix} \epsilon'_1 \\ \vdots \\ \epsilon'_d \end{bmatrix} \tag{30}$$

$$\Leftrightarrow \boldsymbol{k} = \boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{k} \tag{31}$$

$$\Leftrightarrow (I - \boldsymbol{C})\boldsymbol{k} = \boldsymbol{c}_0 \quad s.t.\ \boldsymbol{k} \succcurlyeq \boldsymbol{0} \tag{32}$$

$$\Leftrightarrow (1 - c_{i,i})k_i - \sum_{\substack{j=1 \\ j \neq i}}^{d} (c_{i,j} k_j) = c_{i,0} \quad s.t.\ k_i \geq 0\ \forall i \in [1, d] \tag{33}$$

Intuitively, the equivalence of boxes from Eq. (30) is equivalent to the equality of their diameters which yields Eq. (31). Therefore, any $\boldsymbol{k} \in \mathbb{R}^d$ satisfying Eq. (31) gives us a fixed point $\boldsymbol{w}_N^{\sharp *}$. In Eq. (32) we make explicit our assumption that $k_i \geq 0$ and finally in Eq. (33) we write Eq. (32) as a system of linear equations.

[2]This does not restrict the possible concretization of any such zonotope we can achieve. As $\boldsymbol{A}$ is invertible, $\boldsymbol{A}^{-1}$ has full rank and thus, its column vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_d$ form a basis of a vector space. Given $\boldsymbol{A}^{-1}$, the concretization of a zonotope of this form is uniquely determined by $\boldsymbol{k}$ (recall that we assume that all k_i are positive).

Existence of a Closed Form Solution for the Fixed Point Equations for $\boldsymbol{w}_N^{\sharp *}$ What remains to be shown to prove Thm. I.1 is that Eq. (33) is guaranteed to have a solution. For that we will investigate the structure of $\boldsymbol{C}$ and $\boldsymbol{c}_0$. Recall that the matrix $\boldsymbol{C}$ and $\boldsymbol{c}_0$ are obtained from IH order reduction on input (see Eq. (29))

$$\boldsymbol{A}(\boldsymbol{w}_N^{\sharp} - \eta(\mathcal{G}_L^N + \mathbf{L}(\mathcal{G}_H))).$$

$\boldsymbol{w}_N^{\sharp *}$ and $\mathcal{G}_L^N$ share the same set of symbols, while they do not share any error symbols with $\mathbf{L}(\mathcal{G}_H)$, as linearization always generates new error symbols. This implies that the terms of $\boldsymbol{A}(\boldsymbol{w}_N^{\sharp} - \eta\mathcal{G}_L^N)$ and $-\eta\boldsymbol{A}\mathbf{L}(\mathcal{G}_H)$ will not cancel out. In other words, they contribute separately to the diameter $\boldsymbol{k}$ of the merged box produced by IH. Therefore, we will look into their contributions separately.

For $\boldsymbol{A}(\boldsymbol{w}_N^{\sharp} - \eta\mathcal{G}_L^N)$, we substitute the definitions of $\mathcal{G}_L^N$ and $\boldsymbol{w}_N^{\sharp *}$ to get:

$$\begin{aligned}
\boldsymbol{A}(\boldsymbol{w}_N^{\sharp} - \eta\mathcal{G}_L^N) &= \boldsymbol{A}\Big((1-2\eta\lambda)I - \frac{2\eta}{n}{\boldsymbol{X}_R}^T\boldsymbol{X}_R\Big)\boldsymbol{w}_N^{\sharp} && (34)\\
&= \boldsymbol{A}\left((1-2\eta\lambda)I - \frac{2\eta}{n}{\boldsymbol{X}_R}^T\boldsymbol{X}_R\right)\boldsymbol{A}^{-1}\begin{bmatrix} k_1\epsilon_1 \\ \vdots \\ k_d\epsilon_d \end{bmatrix} && (\text{expand } \boldsymbol{w}_N^{\sharp})\\
&= \boldsymbol{A}\left(\boldsymbol{A}(1-2\eta\lambda)\boldsymbol{A}^{-1} - \frac{2\eta}{n}\boldsymbol{A}{\boldsymbol{X}_R}^T\boldsymbol{X}_R\boldsymbol{A}^{-1}\right)\boldsymbol{A}^{-1}\begin{bmatrix} k_1\epsilon_1 \\ \vdots \\ k_d\epsilon_d \end{bmatrix}\\
& && (I = \boldsymbol{A}\boldsymbol{A}^{-1} \text{ and } {\boldsymbol{X}_R}^T\boldsymbol{X}_R = \boldsymbol{A}{\boldsymbol{X}_R}^T\boldsymbol{X}_R\boldsymbol{A}^{-1})\\
&= \left(\boldsymbol{A}(1-2\eta\lambda)\boldsymbol{A}^{-1} - \frac{2\eta}{n}\boldsymbol{A}{\boldsymbol{X}_R}^T\boldsymbol{X}_R\boldsymbol{A}^{-1}\right)\begin{bmatrix} k_1\epsilon_1 \\ \vdots \\ k_d\epsilon_d \end{bmatrix} && (35)
\end{aligned}$$

This contributes $\left(|1-2\eta\lambda - \frac{2\eta}{n}q_{i,i}|k_i + \frac{2\eta}{n}\sum_{\substack{j=1\\ j\neq i}}^{d}|q_{i,j}|k_j\right)$ to the diameter of the merged box along dimension i, where $q_{i,j}$ is the element in $\boldsymbol{A}{\boldsymbol{X}_R}^T\boldsymbol{X}_R\boldsymbol{A}^{-1}$ at position (i,j). By choosing a small learning rate η to let $1-2\eta\lambda - \frac{2\eta}{n}q_{i,i} \geq 0$ the contribution is equal to $\left((1-2\eta\lambda - \frac{2\eta}{n}q_{i,i})k_i + \frac{2\eta}{n}\sum_{\substack{j=1\\ j\neq i}}^{d}|q_{i,j}|k_j\right)$. From Eq. (34) it is obvious that this component has a shared term $\boldsymbol{w}_N^{\sharp *}$ containing $\boldsymbol{k}$ and as $\boldsymbol{w}_N^{\sharp *}$ is centered at $\mathbf{0}$, we know that this component has only symbolic terms. This in turn implies that this component only contributes to the coefficient matrix $\boldsymbol{C}$ and does not contribute to the constant part $\boldsymbol{c}_0$ of the system of linear equations.

Now consider the other component

$$\begin{aligned}
&-\eta\boldsymbol{A}\mathbf{L}(\mathcal{G}_H) && (36)\\
&= -\eta\boldsymbol{A}\mathbf{L}\left(\frac{2}{n}\Big((\boldsymbol{X}_R{}^T\boldsymbol{X}_S^{\sharp} + \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{X}_R)(\boldsymbol{w}_D^{\sharp} + \boldsymbol{w}_N^{\sharp}) + \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{X}_S^{\sharp}(\boldsymbol{w}_R + \boldsymbol{w}_D^{\sharp} + \boldsymbol{w}_N^{\sharp}) - \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{y}_S^{\sharp}\Big)\right)\\
&= -\eta\boldsymbol{A}\mathbf{L}\left(\frac{2}{n}\Big(\underbrace{(\boldsymbol{X}_R{}^T\boldsymbol{X}_S^{\sharp} + \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{X}_R + \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{X}_S^{\sharp})\boldsymbol{w}_N^{\sharp}}_{\text{contributes to } \boldsymbol{C}}\right.\right.\\
&\qquad\left.\left. + \underbrace{(\boldsymbol{X}_R{}^T\boldsymbol{X}_S^{\sharp} + \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{X}_R)\boldsymbol{w}_D^{\sharp} + \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{X}_S^{\sharp}(\boldsymbol{w}_R + \boldsymbol{w}_D^{\sharp}) - \boldsymbol{X}_S^{\sharp}{}^T\boldsymbol{y}_S^{\sharp}}_{\text{contributes to } \boldsymbol{c}_0}\Big)\right) && (37)
\end{aligned}$$

Note that the first part of the equation is the only part that contains $\boldsymbol{w}_N^{\sharp}$ which in turn contains the variables $\boldsymbol{k}$ for which we want to solve. As every element of this part of the equation contains some k_i it only contributes to $\boldsymbol{C}$. In contrast the second part of the equation does not contain any variable k_i and, thus, is the only part of both components that contribute to $\boldsymbol{c}_0$. Let us use $\frac{2\eta}{n}\boldsymbol{C'}\boldsymbol{k}$ to denote the matrix of coefficients that is the sum of absolute values of coefficients of

 https://doi.org/10.52202/079017-3396

$\frac{2}{n}({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_S^\sharp)\boldsymbol{w}_N^\sharp$ that will be projected by $\boldsymbol{A}$ stemming from the first part of the equation in the expanded version of $\mathbf{L}(\mathcal{G}_H)$. We use $c'_{i,j} \geq 0$ to denote the element of $\boldsymbol{C}'$ at (i, j). As mentioned before this is the only part of this component that contains terms containing unknowns k_i. We use $\boldsymbol{c}'_0$ to denote the contribution from the second part of Eq. (37) towards the output of linearization $\mathbf{L}$. Then based on the fact that the second part of this component is the only part of both components that contributes to $\boldsymbol{c}_0$, we know that the whole contribution of the component from Eq. (36) towards $\boldsymbol{C} + \boldsymbol{c}_0$ is:

$$\eta \boldsymbol{c}'_0 + \frac{2\eta}{n}\boldsymbol{C}'\boldsymbol{k} \tag{38}$$

In summary, the system of linear equations from Eq. (33) can be written as:

$$\Big(2\eta\lambda + \frac{2\eta}{n}q_{i,i} - \frac{2\eta}{n}c'_{i,i}\Big)k_i - \frac{2\eta}{n}\sum_{\substack{j=1\\ j\neq i}}^{d}(|q_{i,j}| + c'_{i,j})k_j = \eta c_{i,0} \quad s.t.\ k_i \geq 0 \quad \forall i \in [1, d]$$

$$\Leftrightarrow (\lambda n + q_{i,i} - c'_{i,i})k_i - \sum_{\substack{j=1\\ j\neq i}}^{d}(|q_{i,j}| + c'_{i,j})k_j = \frac{n}{2}c_{i,0} \quad s.t.\ k_i \geq 0 \quad \forall i \in [1, d]. \tag{39}$$

In the expanded system of linear equations, the diagonal elements of the coefficient matrix are $(\lambda n + q_{i,i} - c'_{i,i})$, while the off-diagonal elements are $-(|q_{i,j}| + c'_{i,j})$.

Lemma I.6. *If for all $i \in [1, d]$,*

$$(\lambda n + q_{i,i} - c'_{i,i}) \geq \sum_{\substack{j=1\\ j\neq i}}^{d}(|q_{i,j}| + c'_{i,j}),$$

then the system of linear equations from Eq. (39) has a solution.

Proof. When this condition holds, the coefficient matrix of this system of linear equations is diagonally dominant. Also considering all diagonal elements are non-negative ($(\lambda n + q_{i,i} - c'_{i,i}) \geq \sum_{\substack{j=1\\ j\neq i}}^{d}(|q_{i,j}| + c'_{i,j}) \geq 0$) and all off-diagonal elements are non-positive, the coefficient matrix is an M-matrix[3]. For any M-matrix $\boldsymbol{M}$, the inverse $\boldsymbol{M}^{-1}$ exists and is has all non-negative entries. Furthermore, for any system of linear equations represented as matrix equation $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{x}$ are the variables of the equation system, $\boldsymbol{M}^{-1}\boldsymbol{b}$ is a solution. As our matrix $I - \boldsymbol{C}$ is an M-matrix, we know that its inverse exists and has only positive entries and we can compute a solution $\boldsymbol{k}$ as:

$$\boldsymbol{k} = (I - \boldsymbol{C})^{-1}\boldsymbol{c}_0$$

Given that all elements $c_{i,0}$ are positive and that $(I - \boldsymbol{C})^{-1}$ is positive, the solution $\boldsymbol{k}$ for Eq. (39) is also positive. □

The correctness of Thm. I.1 follows immediately based on the sufficient condition for the existence of solution stated in Lem. I.6, .

Corollary I.7 (Proof of Thm. I.1)**.** *Given an input abstract training dataset $\mathbf{D}^\sharp$, let $\beta = \frac{1}{n}\max_i\big(\sum_{\substack{j=1\\ j\neq i}}^{d}(|q_{i,j}| + c'_{i,j}) + c'_{i,i} - q_{i,i}\big)$ For any regularization coefficient $\lambda \geq \beta$, we can compute an abstract fixed point $\boldsymbol{w}^{\sharp *}$ for the model weights according to Def. 3.2.*

Proof. To find an abstract fixed point of the model weights

$$\boldsymbol{w}^{\sharp *} = \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$$

[3]There are over 40 sufficient conditions for a matrix to be an M-matrix [56]. The one we use here requires that the matrix fulfills two conditions: (1) all off-diagonal elements are non-positive, and (2) each diagonal element is no less than the sum of absolute values of the off-diagonal elements at the same row (also known as diagonally dominant).

that fulfills the sufficient condition for an abstract fixed point from Prop. I.2, we first use the closed form solutions from Lem. I.4 to compute $\boldsymbol{w}_R^*$ and then using $\boldsymbol{w}_R^*$ compute $\boldsymbol{w}_D^{\sharp *}$. These are guaranteed to fulfill the first two conditions of Prop. I.2. As the regularization coefficient $\lambda \geq \beta = \frac{1}{n} \max_i \left(\sum_{\substack{j=1 \\ j \neq i}}^{d} (|q_{i,j}| + c'_{i,j}) + c'_{i,i} - q_{i,i} \right)$, the condition from Lem. I.6 holds. Thus, the linear system of equations for a $\boldsymbol{w}_N^{\sharp *}$ that fulfills the last condition of Prop. I.2 has a solution that we can compute using any techniques for solving a system of linear equation. It follows that $\boldsymbol{w}^{\sharp *} = \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$ fulfills all conditions of Prop. I.2 and is an abstract fixed point according to Def. 3.2. □

J Weakening the Requirements on Regularization

The existence of a fixed point $\boldsymbol{w}^{\sharp *}$ as postulated in Thm. I.1 only holds for sufficiently large regularization $\lambda \geq \beta$ where β depends on $\mathbf{D}^{\sharp}$. The majority of real world examples without multicolinearity in the real part of the training data that we have investigated have $\beta = 0$. However, we still provide a technique to reduce β if the need should arise at the cost of increasing the runtime of our approach. Specifically, we will make use of *splitting* [2] which divides an input zonotope into multiple zonotopes of smaller expand. We apply splitting to $\mathbf{D}^{\sharp}$, then compute the fixed point for each zonotope in the result of splitting, and finally combine these fixed point using a *join* operation for zonotopes that over-approximates the union of the concretizations of two zonotopes (allows us to merge multiple zonotope such that the merged zonotope has a concretization that over-approximates the concretizations of all of the inputs). We will demonstrate that this approach enables us to reduce β to any value ≥ 0. Intuitively, this works because we can reduce the extend of the input zonotope for the fixed point computation to the extend necessary to ensure $\beta = 0$. Albeit in worst-case this can require a splitting operator that generates a number of zonotopes that is exponential in the dimension.

J.1 Splitting and Join for zonotopes

Intuitively, a splitting operator divides an input zonotope into two or more zonotopes such that the union of their concretization is the same as the concretization of the input zonotope.

Definition J.1 (Split Operator)**.** *An operator* $\mathbf{S}$ *that maps a zonotope* $\boldsymbol{\ell}^{\sharp}$ *to a set of zonotopes* $\{\boldsymbol{\ell}_1^{\sharp}, \cdots, \boldsymbol{\ell}_m^{\sharp}\}$ *is for any input zonotope* $\boldsymbol{\ell}^{\sharp}$ *we have:*

$$\mathbf{S}(\boldsymbol{\ell}^{\sharp}) = \{\boldsymbol{\ell}_1^{\sharp}, \cdots, \boldsymbol{\ell}_m^{\sharp}\} \quad s.t. \bigcup_i \gamma\left(\boldsymbol{\ell}_i^{\sharp}\right) = \gamma\left(\boldsymbol{\ell}^{\sharp}\right).$$

As an example of a split operator consider the binary split (2-split) at dimension i denoted by $\mathbf{S}_{2,i}$ that splits the input zonotope into two parts by scaling the i^{th} generator $\boldsymbol{g}_i$ by $\frac{1}{2}$ and shifting the center of the zonotope by $\frac{1}{2}\boldsymbol{g}_i$ ($-\frac{1}{2}\boldsymbol{g}_i$):

$$\begin{aligned}\mathbf{S}_{2,i}(\boldsymbol{\ell}^{\sharp}) = \left\{\boldsymbol{\ell}_1^{\sharp}, \boldsymbol{\ell}_2^{\sharp}\right\} &= \Big\{\boldsymbol{c} + \underbrace{\sum_j \boldsymbol{g}_j \epsilon_j}_{\substack{\epsilon_i \in [-1,0] \\ \text{other } \epsilon_j \in [-1,1]}} , \boldsymbol{c} + \underbrace{\sum_j \boldsymbol{g}_j \epsilon_j}_{\substack{\epsilon_i \in [0,1] \\ \text{other } \epsilon_j \in [-1,1]}} \Big\} \\ &= \left\{\boldsymbol{c} - \frac{1}{2}\boldsymbol{g}_i + \frac{1}{2}\boldsymbol{g}_i\epsilon_i + \sum_{j \neq i} \boldsymbol{g}_j \epsilon_j, \quad \boldsymbol{c} + \frac{1}{2}\boldsymbol{g}_i + \frac{1}{2}\boldsymbol{g}_i\epsilon_i + \sum_{j \neq i} \boldsymbol{g}_j \epsilon_j \right\} \end{aligned} \tag{40}$$

As $\gamma\left(\boldsymbol{\ell}_1^{\sharp}\right) \cup \gamma\left(\boldsymbol{\ell}_2^{\sharp}\right) = \gamma\left(\boldsymbol{\ell}^{\sharp}\right)$, $\mathbf{S}_{2,i}(\cdot)$ is a split operator according to Def. J.1. We can generalize 2-split to an (m,i)-split $\mathbf{S}_{m,i}$ that divides the zonotope evenly into m parts along the i^{th} dimension. We will call the composition of (m,i)-splits across all dimensions as a μ-split. Given $\mu = \frac{1}{m}$ for $m \in \mathbb{N}$ as input, the effect of the μ-split $\mathbf{S}_{\mu}$ is the generators of each zonotope in the result of splitting are the generators of the input zonotope scaled by μ. Thus, μ-splitting allows us to downscale the generators of a zonotope. For a d-dimensional zonotope and $\mu = \frac{1}{m}$, $\mathbf{S}_{\mu}$ returns d^m zonotopes.

Definition J.2 (μ-split)**.** *For d-dimensional zonotopes and $\mu = \frac{1}{m}$ for $m \in \mathbb{N}$, the μ-split $\mathbf{S}_{\mu}$ is defined as:*

$$\mathbf{S}_{m,d} \circ \ldots \circ \mathbf{S}_{m,1}$$

https://doi.org/10.52202/079017-3396

where the (m, i)-split $\mathbf{S}_{m,i}$ is defined as shown below and the application of a split operator to a set of zonotopes is defined as applying the split operator to every element in the set.

$$\mathbf{S}_{m,i}(\boldsymbol{\ell}^\sharp) = \{\boldsymbol{\ell}^\sharp{}_i\}_{j=1}^m$$

for

$$\boldsymbol{\ell}^\sharp{}_j = \boldsymbol{c} + \frac{-m+2j-1}{m}\boldsymbol{g}_i + \frac{1}{m}\boldsymbol{g}_i\epsilon_i + \sum_{j\neq i}\boldsymbol{g}_j\epsilon_j$$

For example, $\mathbf{S}_2(1+\epsilon_1) = \{0.5 + 0.5\epsilon_1, 1.5 + 0.5\epsilon_1\}$. It is easy to see that $\mathbf{S}_\mu$ is indeed a split according to Def. J.1.

Proposition J.3 ($\mathbf{S}_\mu$ is a Split Operator)**.** *$\mathbf{S}_\mu$ is a split operator for any $\mu = \frac{1}{m}$ were $m \in \mathbb{N}$.*

Next we introduce join operators for zonotopes and then demonstrate that for any abstract transformer $F^\sharp$ for a function F, we can construct another abstract transformer for F by (i) splitting the input zonotope using some split operator $\mathbf{S}$, (ii) applying $F^\sharp$ separately on each zonotope returned by the split, and (iii) merge the results using a join that we introduce next.[4]

Definition J.4 (Join)**.** *An operator $\sqcup$ is a join for zonotopes if it over-approximates union of concretization. That is, for any set $S^\sharp$ of zonotopes:*

$$\gamma\left(\bigsqcup S^\sharp\right) \supseteq \bigcup_{\boldsymbol{z}^\sharp \in S^\sharp} \gamma\left(\boldsymbol{z}^\sharp\right).$$

Several join operators have been proposed for zonotopes, e.g., [24]. Next we establish that new abstract transformers can be constructed by wrapping existing transformers with split and join. This ensures that our approach of combining our abstract fixed point calculation with splitting yields valid fixed points.

Lemma J.5 (Abstract Transformers Are Sound on Splits)**.** *Consider a zonotope $\boldsymbol{z}^\sharp$ and let $F^\sharp$ be an abstract transformer for a function F. Furthermore, let $\sqcup$ be a join operator, $\mathbf{S}$ be a split operator, and let $\mathbf{S}(\boldsymbol{\ell}^\sharp) = \{\boldsymbol{\ell}^\sharp{}_1, \ldots, \boldsymbol{\ell}^\sharp{}_m\}$. Then, ${F_{\mathbf{S},\sqcup}}^\sharp = \sqcup \circ F^\sharp \circ \mathbf{S}$ is an abstract transformer for F.*

Proof. We have to show that

$$\gamma\left({F_{\mathbf{S},\sqcup}}^\sharp(\boldsymbol{\ell}^\sharp)\right) \supseteq F(\gamma\left(\boldsymbol{\ell}^\sharp\right)).$$

We have

$$\begin{aligned}
& F(\gamma\left(\boldsymbol{\ell}^\sharp\right)) \\
=& F(\bigcup_{\boldsymbol{\ell}^{\sharp\prime} \in \mathbf{S}(\boldsymbol{\ell}^\sharp)} \gamma\left(\boldsymbol{\ell}^{\sharp\prime}\right)) && \text{(Def. J.1 for } \mathbf{S}\text{)} \\
=& F(\bigcup_{i=1}^m \gamma\left(\boldsymbol{\ell}^\sharp{}_i\right)) \\
=& \bigcup_{i=1}^m F(\gamma\left(\boldsymbol{\ell}^\sharp{}_i\right)) && \text{(function application on sets of concrete elements is point-wise application)} \\
\subseteq& \bigcup_{i=1}^m \gamma\left(F^\sharp(\boldsymbol{\ell}^\sharp{}_i)\right) && (F^\sharp \text{ is an abstract transformer}) \\
\subseteq& \gamma\left(\bigsqcup_{i=1}^m F^\sharp(\boldsymbol{\ell}^\sharp{}_i)\right) && (\gamma\left(\boldsymbol{z}_1^\sharp \sqcup \boldsymbol{z}_2^\sharp\right) \supseteq \gamma\left(\boldsymbol{z}_1^\sharp\right) \cup \gamma\left(\boldsymbol{z}_2^\sharp\right)) \\
=& \gamma\left({F_{\mathbf{S},\sqcup}}^\sharp(\boldsymbol{\ell}^\sharp)\right)
\end{aligned}$$

□

[4] Technically, join is normally defined as an over-approximation of the least upper bound of two abstract elements wrt. a partial order of the abstract domain. We order abstract elements based on set inclusion of their concretizations here and only define join regarding to this partial order.

J.2 Weakening Regularization Requirements

Equipped with the μ-split and a join operator, we are ready to analyze how the introduction of a μ-split can be used to achieve an arbitrarily small β. Given the input data $\mathbf{D}^\sharp$, the learning rate $\eta \geq \beta$, the regularization coefficient λ, the transformation matrix $\boldsymbol{A}$, we use $\Gamma(\mathbf{D}^\sharp, \eta, \lambda, \boldsymbol{A})$ to denote the function that computes the fixed point ${\boldsymbol{w}_S^\sharp}^*$ for these inputs using the process outlined in Corollary I.7. From Lem. J.5 follows that $\Gamma_{\mathbf{S}_\mu, \sqcup}$ is an abstract transformer for gradient descent and, thus, according to Prop. 3.3 over-approximates ${\boldsymbol{w}}^{\odot *}$

Next, we show that given an abstract training dataset $\mathbf{D}^\sharp$ and a desired regularization coefficient λ_{target}, we can find a value of μ such that the abstract fixed point construction with μ-splitting computes a fixed point. Recall that Γ can construct a fixed point if $\lambda \geq \beta$ where β is a constant that depends on $\mathbf{D}^\sharp$. We will show that by choosing μ carefully, we can achieve $\beta \leq \lambda_{target}$ for each zonotope in the result of the split and, thus, Γ will return a valid fixed point for each zonotope in the split result.

Lemma J.6. *For any abstract training dataset $\mathbf{D}^\sharp$ and desired regularization coefficient λ, there exists $m \in \mathbb{N}$ such that for $\mu = \frac{1}{m}$, $\Gamma_{\mathbf{S}_\mu, \sqcup}$ returns a fixed point ${\boldsymbol{w}^\sharp}^*$.*

Proof. Consider an abstract training dataset $\mathbf{D}^\sharp = (\boldsymbol{X}^\sharp, \boldsymbol{y}^\sharp)$ with $\boldsymbol{X}^\sharp = \boldsymbol{X}_R + \boldsymbol{X}_S^\sharp$. WLOG consider a single abstract training dataset ${\mathbf{D}^\sharp}_i$ in the result of μ-split on $\mathbf{D}^\sharp$:

$$ {\mathbf{D}^\sharp}_i \in \mathbf{S}_\mu(\mathbf{D}^\sharp) $$

We will show that using the regularization coefficient λ, we can find μ such that the precondition $\lambda \geq \beta$ for Γ to compute a fixed point holds. Then applying Lem. J.5 we get the desired result.

First observe that since ${\mathbf{D}^\sharp}_i$ is in the result of μ-splitting, we know that it has the following shape where the symbolic component $\boldsymbol{X}_S^\sharp$ is scaled by μ and the real component $\boldsymbol{X}_{Ri}$ of ${\mathbf{D}^\sharp}_i$ typically differs from $\boldsymbol{X}^\sharp$ as μ-splitting changes the center of the zonotope.

$$ {\boldsymbol{X}^\sharp}_i = \boldsymbol{X}_{Ri} + \mu \boldsymbol{X}_S^\sharp $$

Recall that $\frac{2\eta}{n}\boldsymbol{C'k}$ denotes the matrix of coefficients that is the sum of absolute values of coefficients of $\frac{2}{n}\boldsymbol{A}({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_S^\sharp)\boldsymbol{w}_N^\sharp$. Plugging in the definition of ${\boldsymbol{X}^\sharp}_i$ from above, we get a matrix $\frac{2\eta}{n}\boldsymbol{C'k}$ that is the sum of absolute values of coefficients of

$$ \frac{2}{n}\boldsymbol{A}({\boldsymbol{X}_{Ri}}^T \mu \boldsymbol{X}_S^\sharp + \mu {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_{Ri} + \mu {\boldsymbol{X}_S^\sharp}^T \mu \boldsymbol{X}_S^\sharp)\boldsymbol{w}_N^\sharp \tag{41} $$

$$ = \frac{2}{n}\boldsymbol{A}(\mu {\boldsymbol{X}_{Ri}}^T \boldsymbol{X}_S^\sharp + \mu {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_{Ri} + \mu^2 {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_S^\sharp)\boldsymbol{A}^{-1} \begin{bmatrix} k_1\epsilon_1 \\ \vdots \\ k_d\epsilon_d \end{bmatrix} \tag{42} $$

$$ = \Big(\underbrace{\frac{2\mu}{n}\boldsymbol{A}({\boldsymbol{X}_{Ri}}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_{Ri})\boldsymbol{A}^{-1}}_{\text{contributes to } c^1_{i,j}} + \underbrace{\frac{2\mu^2}{n}\boldsymbol{A}({\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_S^\sharp)\boldsymbol{A}^{-1}}_{\text{contributes to } c^2_{i,j}} \Big) \begin{bmatrix} k_1\epsilon_1 \\ \vdots \\ k_d\epsilon_d \end{bmatrix}. \tag{43} $$

Recall that $c'_{i,j}$ denotes entry of $\boldsymbol{C'}$ at row i and column j. Here, ${\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_S^\sharp$ has higher order than $({\boldsymbol{X}_R}^T \boldsymbol{X}_S^\sharp + {\boldsymbol{X}_S^\sharp}^T \boldsymbol{X}_R)$, thus their terms will not cancel out. Therefore, we can decompose the contribution to $c'_{i,j}$ into the above two parts, denoted as

$$ c'_{i,j} = c^1_{i,j} + c^2_{i,j}. $$

Given scalar μ, let

$$ c'_\mu = \max_{u,v} c'_{u,v} $$

i.e., c'_μ denotes the largest entry $c'_{i,j}$ in ${\mathbf{D}^\sharp}_i$.

https://doi.org/10.52202/079017-3396

Let i and j such that $c'_\mu = c'_{i,j}$, we have:

$$c'_\mu = \mu c^1_{i,j} + \mu^2 c^2_{i,j} \leq \mu c^1_{i,j} + \mu c^2_{i,j} = \mu c'_{max},$$

where c'_{max} is a constant that is equal to the maximum $c'_{i,j}$ before scaling $\boldsymbol{X}^\sharp_S$ through μ-splitting.

Next, we prove the lemma for a specific transformation $\boldsymbol{A} = V^T$ during order reduction derived through singular value decomposition (SVD). Using SVD, we can decompose the covariance matrix $\boldsymbol{X}_{Ri}{}^T \boldsymbol{X}_{Ri}$ of the real number part of features $\boldsymbol{X}_{Ri}$ as shown below

$$\boldsymbol{X}_{Ri}{}^T \boldsymbol{X}_{Ri} = V \Sigma V^T.$$

For $\boldsymbol{A} = V^T$, and using the fact that V is a rotation in SVD which implies $V^{-1} = V^T$, we get:

$$\begin{aligned}
&\boldsymbol{A}\boldsymbol{X}_{Ri}{}^T \boldsymbol{X}_{Ri} \boldsymbol{A}^{-1} \\
=&\boldsymbol{A}(V \Sigma V^T)\boldsymbol{A}^{-1} \\
=&V^T(V \Sigma V^T)V^{T^{-1}} \\
=&(V^{-1}V)\Sigma(V^{-1}V^{-1^{-1}}) \\
=&\Sigma
\end{aligned}$$

Thus, the matrix $\boldsymbol{A}\boldsymbol{X}_{Ri}{}^T \boldsymbol{X}_{Ri}\boldsymbol{A}^{-1} = \Sigma$ is a diagonal matrix, meaning that all off-diagonal elements of it ($|q_{i,j}|$ for all $(i \neq j)$ in Corollary I.7) are zero. In addition, the diagonal element $\Sigma[i,i]$ ($q_{i,i}$ in Corollary I.7) is the i^{th} eigenvalue of the covariance matrix $\boldsymbol{X}_{Ri}{}^T \boldsymbol{X}_{Ri}$, which must be positive when assuming no multicolinearity.

To sum up, β from Corollary I.7 can be re-written as:

$$\beta = \frac{1}{n} \max_i \left(c'_{i,i} + \sum_{\substack{j=1 \\ j \neq i}}^{d} (|q_{i,j}| + c'_{i,j}) - q_{i,i}\right) \tag{44}$$

$$\leq \frac{1}{n} \max_i \left(c'_\mu + \sum_{\substack{j=1 \\ j \neq i}}^{d} c'_\mu - q_{i,i}\right) \tag{45}$$

$$\leq \frac{1}{n} \max_i \left(\mu c'_{max} + \sum_{\substack{j=1 \\ j \neq i}}^{d} \mu c'_{max} - q_{i,i}\right) \tag{46}$$

$$= \frac{1}{n} \max_i \left(d\mu c'_{max} - q_{i,i}\right) = \frac{1}{n} d\mu c'_{max} - \frac{1}{n} \min_i(q_{i,i}) \tag{47}$$

where $\frac{1}{n}\min_i(q_{i,i})$ is a positive constant. Therefore, when setting $\mu = \frac{\min_i(q_{i,i})}{dc'_{max}} > 0$, we have $\beta \leq \frac{1}{n} d\mu c'_{max} - \frac{1}{n}\min_i(q_{i,i}) = 0$. □

K An Approximate Abstract Transformer for Ridge Regression

Alg. 2 is the detailed version of Alg. 1 we presented in the main paper. Like Alg. 1, Alg. 2 takes as input an abstract dataset $\mathbf{D}^\sharp$ that over-approximates $\boldsymbol{D}^\odot$, a learning rate η, a regularization coefficient λ, and a transformation matrix $\boldsymbol{A}$ used for order reduction (e.g., the SVD-based transformation discussed in App. J.2). In Line 1, we first use function `constructFPEquations` to compute fixed points for $\boldsymbol{w}^*_R$ and $\boldsymbol{w}^\sharp_D$ using the closed form solution from Lem. I.4 and construct the equation system Ξ for $\boldsymbol{w}^{\sharp *}_N$ (Eq. (39)). Furthermore, this function also computes the threshold β on the regularization coefficient for this system of equations to have a solution. If β is smaller or equal to the desired regularization coefficient λ (Line 10), then we solve the linear equations Ξ (Line 11) and then returns the fixpoint $\boldsymbol{w}^{\sharp *}$. Otherwise, we determine a sufficiently small splitting factor μ that ensures that $\beta \leq \lambda$ for each $\boldsymbol{w}^\sharp{}_i$ in the result of the split (Line 3), compute the fixed point for each such $\boldsymbol{w}^\sharp{}_i$ (Lines 6 and 8), and merge these fixed points using join to compute the final result (Line 9).

Algorithm 2: Constructing a Fixed Point for Abstract Gradient Descent

Input: abstract dataset $\mathbf{D}^\sharp = (\boldsymbol{X}^\sharp, \boldsymbol{y}^\sharp)$, learning rate η, Regularization coefficient λ, transformation matrix $\boldsymbol{A}$

Output: abstract fixed point $\boldsymbol{w}^{\sharp *}$

$\boldsymbol{w}_R^*, \boldsymbol{w}_D^{\sharp *}, \Xi, \beta \leftarrow$ `constructFPequations`$(\mathbf{D}^\sharp, \lambda, \eta)$
if $\beta > \lambda$ **then** `// Is splitting necessary?`
  $\mu \leftarrow$ `determineNumSplits`(Ξ, λ)
  $\{\mathbf{D}^\sharp{}_i\}_{i=1}^m \leftarrow \mathbf{S}_\mu(\mathbf{D}^\sharp)$ `// Apply` μ`-splitting`
  for $\mathbf{D}^\sharp{}_i \in \{\mathbf{D}^\sharp{}_i\}_{i=1}^m$ **do** `// Construct FP for each` $\mathbf{D}^\sharp{}_i$ `in the split`
    $\boldsymbol{w}_R^*, \boldsymbol{w}_D^{\sharp *}, \Xi, \beta \leftarrow$ `constructFPequations`$(\mathbf{D}^\sharp{}_i, \lambda, \eta)$
    $\boldsymbol{w}_N^{\sharp *} \leftarrow$ `solveFixedPointEquations`(Ξ, λ, η)
    $\boldsymbol{w}^\sharp{}_i \leftarrow \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$
  return $\bigsqcup_{i=1}^m \boldsymbol{w}^\sharp{}_i$ `// Merge FPs using Join`
else `// No splitting required`
  $\boldsymbol{w}_N^{\sharp *} \leftarrow$ `solveFixedPointEquations`(Ξ, λ, η)
  $\boldsymbol{w}^{\sharp *} \leftarrow \boldsymbol{w}_R^* + \boldsymbol{w}_D^{\sharp *} + \boldsymbol{w}_N^{\sharp *}$
return $\boldsymbol{w}^{\sharp *}$

def `constructFPEquations`$(\mathbf{D}^\sharp, \lambda, \eta)$**:**
  $\boldsymbol{w}_R^* \leftarrow$ `evalClosedFormReal`$(\mathbf{D}^\sharp, \lambda)$ `// Eq. (22)`
  $\boldsymbol{w}_D^{\sharp *} \leftarrow$ `evalClosedFormSymbolic`$(\mathbf{D}^\sharp, \lambda, \boldsymbol{w}_R^*)$ `// Eq. (23)`
  $\Xi \leftarrow$ `constructNonDataFixedPointEquations`$(\mathbf{D}^\sharp, \lambda, \eta, \boldsymbol{w}_R^*, \boldsymbol{w}_D^{\sharp *})$ `// Eq. (39)`
  $\beta \leftarrow$ `determineMinRegularization`(Ξ)
  return $\boldsymbol{w}_R^*$, $\boldsymbol{w}_D^{\sharp *}$, Ξ, β

L Infeasibility of Symbolically Evaluating the Closed-form Solution for Ridge Regression

In this section, we demonstrate why evaluating the closed-form solution for linear regression with MSE is not feasible. It produces symbolic expressions with many variables and large symbolic terms (their representation size) that include the fractions with denominators and enumerators that are polynomial expressions. Apart from the size of these expressions, computing viable prediction intervals from such a symbolic representation of model weights is computationally hard. We use MSE loss for simplicity, but the same arguments also apply to ridge regression.

We illustrate the issues using a randomly-generated toy dataset with 10 samples and 3 uncertain data cells in the third column (corresponding to 3 error symbols $\epsilon_0, \epsilon_1, \epsilon_2$)[5]:

$$\boldsymbol{X}^\sharp = \begin{bmatrix} 1.0 & 1.6 & 1.3\epsilon_0 + 1.4 \\ 1.0 & 0.8 & 1.5 \\ 1.0 & -1.7 & -1.9 \\ 1.0 & -0.57 & 1.1 \\ 1.0 & -0.39 & 0.36 \\ 1.0 & 0.035 & 1.2 \\ 1.0 & -0.34 & -0.73 \\ 1.0 & 0.038 & 0.3\epsilon_1 + 0.44 \\ 1.0 & 1.5 & 1.2\epsilon_2 + 1.3 \\ 1.0 & -0.98 & -0.66 \end{bmatrix} \quad \boldsymbol{y}^\sharp = \begin{bmatrix} 4.6 \\ -1.5 \\ 0.39 \\ -5.7 \\ -3.0 \\ -3.6 \\ -0.44 \\ 0.46 \\ 2.2 \\ -3.0 \end{bmatrix}$$

[5]The first column of $\boldsymbol{X}^\sharp$ consisting of 1's corresponds to the bias term or the intercept term in the weights.

https://doi.org/10.52202/079017-3396

Then, symbolically evaluating the closed-form solution for linear regression with MSE loss, we obtain the symbolic expression representing all possible model weights:

$$\boldsymbol{w}^{\sharp *} = (\boldsymbol{X}^{\sharp T}\boldsymbol{X}^{\sharp})^{-1}\boldsymbol{X}^{\sharp T}\boldsymbol{y}^{\sharp}$$
$$= \begin{bmatrix} \frac{1.5\cdot10^6\epsilon_0^2+8.7\cdot10^4\epsilon_0\epsilon_1-4.5\cdot10^5\epsilon_0\epsilon_2-1.2\cdot10^6\epsilon_0+9.2\cdot10^4\epsilon_1^2+4.0\cdot10^3\epsilon_1\epsilon_2-2.4\cdot10^5\epsilon_1+1.0\cdot10^6\epsilon_2^2-2.1\cdot10^6\epsilon_2-1.8\cdot10^6}{-1.1\cdot10^6\epsilon_0^2+8.4\cdot10^4\epsilon_0\epsilon_1+1.1\cdot10^6\epsilon_0\epsilon_2+1.1\cdot10^6\epsilon_0-8.3\cdot10^4\epsilon_1^2+7.7\cdot10^4\epsilon_1\epsilon_2+6.1\cdot10^3\epsilon_1-9.9\cdot10^5\epsilon_2^2+9.5\cdot10^5\epsilon_2-4.0\cdot10^6} \\ \frac{-9.5\cdot10^4\epsilon_0^2+2.3\cdot10^4\epsilon_0\epsilon_1+2.6\cdot10^5\epsilon_0\epsilon_2+2.3\cdot10^5\epsilon_0-1.4\cdot10^4\epsilon_1^2+2.0\cdot10^4\epsilon_1\epsilon_2+3.6\cdot10^4\epsilon_1-1.4\cdot10^5\epsilon_2^2-6.0\cdot10^2\epsilon_2-1.8\cdot10^6}{-1.1\cdot10^5\epsilon_0^2+8.4\cdot10^3\epsilon_0\epsilon_1+1.1\cdot10^5\epsilon_0\epsilon_2+1.1\cdot10^5\epsilon_0-8.3\cdot10^3\epsilon_1^2+7.7\cdot10^3\epsilon_1\epsilon_2+6.1\cdot10^2\epsilon_1-9.9\cdot10^4\epsilon_2^2+9.5\cdot10^4\epsilon_2-4.0\cdot10^5} \\ \frac{-3.7\cdot10^3\epsilon_0-4.1\cdot10^2\epsilon_1-9.2\cdot10^2\epsilon_2+1.3\cdot10^4}{-1.1\cdot10^3\epsilon_0^2+84.0\epsilon_0\epsilon_1+1.1\cdot10^3\epsilon_0\epsilon_2+1.1\cdot10^3\epsilon_0-83.0\epsilon_1^2+77.0\epsilon_1\epsilon_2+6.1\epsilon_1-9.9\cdot10^2\epsilon_2^2+9.5\cdot10^2\epsilon_2-4.0\cdot10^3} \end{bmatrix}.$$

Next, we consider one test sample $\boldsymbol{x_t} = [1, -1, 1]$ and apply this closed-form model weight expression to infer its prediction range $V^{\sharp}(\boldsymbol{x_t})$ (cf. Corollary F.2). With *merely* 3 uncertain data points in the training data, the prediction of this test data point

$$\begin{aligned}\hat{\boldsymbol{y}}_{\boldsymbol{t}} =& \boldsymbol{w}^{\sharp *}\boldsymbol{x_t} \\ =& -\frac{3.7\cdot10^3\epsilon_0+4.1\cdot10^2\epsilon_1+9.2\cdot10^2\epsilon_2-1.3\cdot10^4}{-1.1\cdot10^3\epsilon_0^2+84.0\epsilon_0\epsilon_1+1.1\cdot10^3\epsilon_0\epsilon_2+1.1\cdot10^3\epsilon_0-83.0\epsilon_1^2+77.0\epsilon_1\epsilon_2+6.1\epsilon_1-9.9\cdot10^2\epsilon_2^2+9.5\cdot10^2\epsilon_2-4.0\cdot10^3} \\ &+\frac{9.5\cdot10^4\epsilon_0^2-2.3\cdot10^4\epsilon_0\epsilon_1-2.6\cdot10^5\epsilon_0\epsilon_2-2.3\cdot10^5\epsilon_0+1.4\cdot10^4\epsilon_1^2-2.0\cdot10^4\epsilon_1\epsilon_2-3.6\cdot10^4\epsilon_1+1.4\cdot10^5\epsilon_2^2+6.0\cdot10^2\epsilon_2+1.8\cdot10^6}{-1.1\cdot10^5\epsilon_0^2+8.4\cdot10^3\epsilon_0\epsilon_1+1.1\cdot10^5\epsilon_0\epsilon_2+1.1\cdot10^5\epsilon_0-8.3\cdot10^3\epsilon_1^2+7.7\cdot10^3\epsilon_1\epsilon_2+6.1\cdot10^2\epsilon_1-9.9\cdot10^4\epsilon_2^2+9.5\cdot10^4\epsilon_2-4.0\cdot10^5} \\ &+\frac{1.5\cdot10^6\epsilon_0^2+8.7\cdot10^4\epsilon_0\epsilon_1-4.5\cdot10^5\epsilon_0\epsilon_2-1.2\cdot10^6\epsilon_0+9.2\cdot10^4\epsilon_1^2+4.0\cdot10^3\epsilon_1\epsilon_2-2.4\cdot10^5\epsilon_1+1.0\cdot10^6\epsilon_2^2-2.1\cdot10^6\epsilon_2-1.8\cdot10^6}{-1.1\cdot10^6\epsilon_0^2+8.4\cdot10^4\epsilon_0\epsilon_1+1.1\cdot10^6\epsilon_0\epsilon_2+1.1\cdot10^6\epsilon_0-8.3\cdot10^4\epsilon_1^2+7.7\cdot10^4\epsilon_1\epsilon_2+6.1\cdot10^3\epsilon_1-9.9\cdot10^5\epsilon_2^2+9.5\cdot10^5\epsilon_2-4.0\cdot10^6}\end{aligned} \tag{48}$$

already consists of fractions of complex polynomial expressions. Finding the viable prediction range for this data point, i.e. the minimum and maximum possible prediction for the point across all models in the concretization of the symbolic expression, using this expression is infeasible, as it is harder than finding an extrema for multivariate polynomials under linear constraints[6], which is known to be NP-hard. In practice, expressions will be significantly larger as they will involve more error symbols. Assume each column has p uncertain cells, and q is the number of uncertain labels, the number of distinct monomials is $O(p^d d^{2d} q)$, where d is the number of dimensions, and the d in exponents mainly comes from the matrix inversion when computing the determinant.

In summary, the symbolic expressions obtained from symbolically evaluating the closed-form solution for linear regression are not suitable for representing the space of possible model weights.

M Additional Experiments

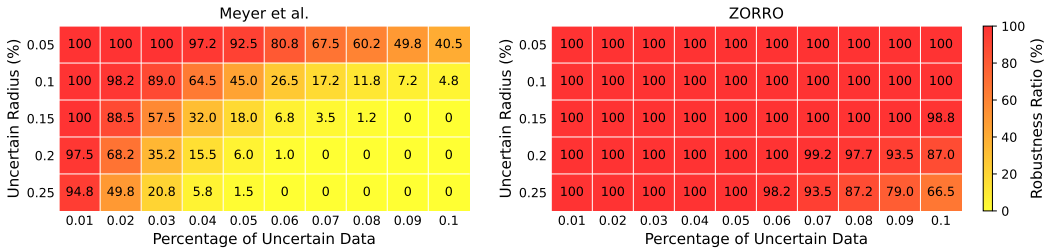
M.1 Robustness Verification Additional Results

For a better comparison with MEYER [47], we use heatmaps to visualize the robustness ratios averaged over 5 repeated experiments with different random seeds. In App. M.1, ZORRO exhibits much less yellow regions (representing higher uncertainty) compared to MEYER. Especially when the uncertainty is high (bottom-right part), there are many cases that ZORRO returns high robustness ratio but MEYER does not. Recall that both approach gives sound over-approaximation of prediction robustness, thus the robustness ratios returned by both are lower bounds of the ground truth prediction robustness ratio. Therefore, the aforementioned cases correspond to highly robust ground truth (no less than the result of ZORRO), where ZORRO gives tight over-approximations but MEYER does not.

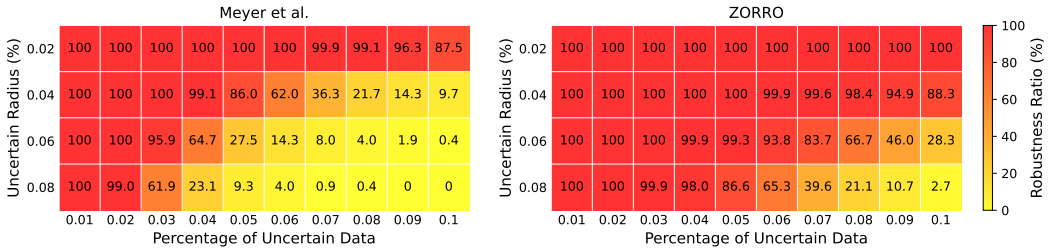
M.2 Micro Benchmark

We use range of loss as the metric to measure the quality (tightness) of uncertain training result. range of loss for samples are calculated by max loss - min loss across all sampled results. Loss range for symbolic representations are calculated my measuring the range of the interval concretization of the loss function result. Notice that samplings return a subset of all possible result which has no soundness guarantees, as a result, produces an under-approximation of the loss range. The symbolic fixed point approach produces an over-approximated loss range. We take 1,000 (1k) and 10,000 (10k) samples from all possible worlds as under approximations to the concrete range when number of possible worlds are to large to compute the ground truth range.

[6]The expression is a sum of fractions in which both numerator and denominators are multivariate polynomial expressions. The range $[-1, 1]$ for each error symbol can be formulated as linear constraints over the input space.

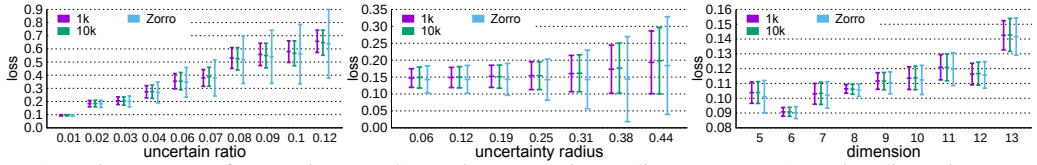


(a) Mpg



(b) Insurance

Figure 5: Robustness verification under label errors using intervals (MEYER) or zonotopes (ZORRO).



(a) varying amount of uncertainty (b) varying uncertainty radius (c) varying dimension

Figure 6: Fixed point versus samplings

Varying ratio of tuples contains uncertain data Figure 6a shows the loss range results while updating the ratio of uncertain rows from 0.01 to 0.12. ZORRO has a tight range considering all samples are under approximations of ground truth range especially in . As uncertainty increases, the over-approximation gap widens due to the increased coefficient of higher order terms in the gradient, which are linearized, leading to higher linearization errors.

Increasing uncertainty radius Figure 6b shows the result by increasing the radius of uncertain data (imputation results) by multiplier of 0.06 to 0.44 where 0.06 means the range for each uncertain data is increased by 6%. Similar to amount of uncertainty,

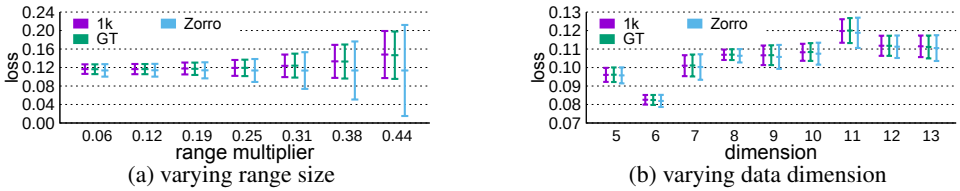
Varying dimensions Figure 6c Shows the result by change number of dimensions to the training data. Result indicates tightness of ZORRO 's over-approximation is not affected by the dimension of the data.

Figure 7 added sampling result to Figure 3 shows additionally that sampling result is an under approximation of the ground truth result.

M.3 Additional Experiments on Varying Regularization Coefficients

Fig. 8 shows the effect of varying regularization coefficients on the worst-case loss and prediction robustness ratio. To avoid much overlapping in the plots, we used one standard deviation as the error bar. Similar to the conclusions in Sec. 5.2, $\lambda = 0$ is often not the optimal regularization coefficient in terms of accuracy or robustness. In fact, a small, positive λ could result in higher accuracy and better robustness, and this optimal λ varies across different settings. In general, higher data uncertainty requires higher λ, which coincides with the intuition of using regularization in traditional settings of linear regressions, which aims to mitigate the effect of data noises or errors.

https://doi.org/10.52202/079017-3396



(a) varying range size (b) varying data dimension

Figure 7: The range of losses obtained by enumerating all possible worlds (GT), sampling 1000 possible worlds (1k), and ZORRO.

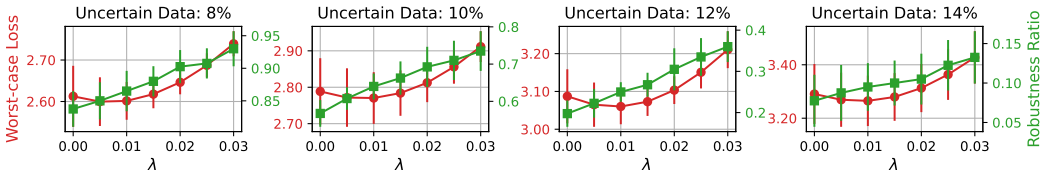


Figure 8: Results with uncertain training labels.

Figure 9: Robustness ratio (red y-axis) and worst-case test loss (green y-axis) vs. regularization coefficient λ (x-axis), with varying percentages of uncertain labels.

M.4 Comparing with Bayesian Regression

We ran empirical evaluations to demonstrate how data quality issues pose challenges for Bayesian linear regression (implemented with torchbnn [40]), making them inapplicable to our setting. Using the setting from the third plot in Fig 1c, where the uncertain data percentage is set to 10%, we tested Bayesian linear regression on different possible worlds using two methods: impute-and-predict and sampling from possible worlds. The results show that the prediction intervals generated by Bayesian methods do not cover the ground truth prediction, i.e., the prediction by the model trained on the ground truth training data. In contrast, our approach guarantees 100% coverage across all cases.

Uncertain	kNN Imputation (k=5)		kNN Imputation (k=10)		Multiple Imputation		Zorro	
Radius (%)	Coverage (%)	Avg. Intv.	Coverage (%)	Avg. Intv.	Coverage (%)	Avg. Intv.	Coverage (%)	Avg. Intv.
2.5	48.8	0.157	42.5	0.153	46.3	0.176	100	0.183
5	48.8	0.181	40	0.153	37.5	0.176	100	0.506
7.5	38.8	0.157	37.5	0.154	35	0.176	100	1.107
10	36.3	0.157	31.3	0.154	32.5	0.176	100	2.387

Table 1: Bayesian approach on imputed data.

Uncertain	Possible World 1		Possible World 2		Possible World 3		Zorro	
Radius (%)	Coverage (%)	Avg. Intv.	Coverage (%)	Avg. Intv.	Coverage (%)	Avg. Intv.	Coverage (%)	Avg. Intv.
2.5	38.75	0.151	40	0.154	55	0.176	100	0.183
5	30	0.158	40	0.154	37.5	0.176	100	0.506
7.5	42.5	0.161	38.8	0.155	31.3	0.155	100	1.107
10	57.5	0.351	31.3	0.169	27.5	0.163	100	2.387

Table 2: Bayesian approach on sampled possible worlds.

M.5 Varying Robustness Threshold

Different practical applications may differ in how much uncertainty they are willing to tolerate, thus leading to varying choices of robustness thresholds. To account for this in the experiments, we selected both a low threshold (0.5% for the insurance data) and a high threshold (5% for the MPG data). As presented in Fig 10, we also explored other thresholds, which did not impact the trends significantly: Zorro can consistently certify a larger fraction of the test data points than the baseline due to its use of the more expressive zonotope domain.

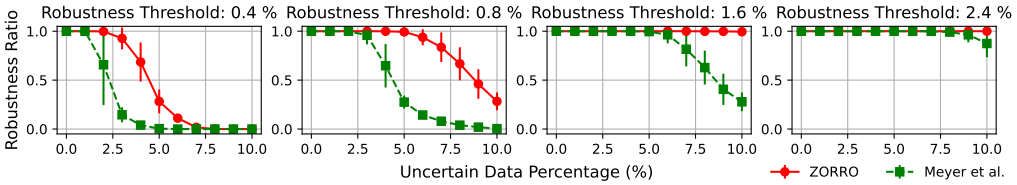


Figure 10: Varying robustness threshold from 0.4% to 2.4% with fixed uncertainty radius 6%

M.6 Varying Uncertain Feature

The variability of trained models will depend on the correlation that features with uncertainty have with the label, which then also affects our over-approximation of this set. As a rule of thumb, model variability will increase with the correlation between the uncertain feature and the labels, leading to less robust predictions. We conducted an additional experiment using the MPG dataset, focusing on the feature "acceleration," which has relatively low correlation with the label. Unlike the feature "weight" (Fig. 1c), where the robustness drops when the uncertainty radius is 10%, the robustness for "acceleration" starts to drop only when the uncertainty radius is increased to 16%. The result shown in Fig 11 indicates that features with lower predictive power have less impact on the robustness of a model compared to features that are highly correlated with the label.

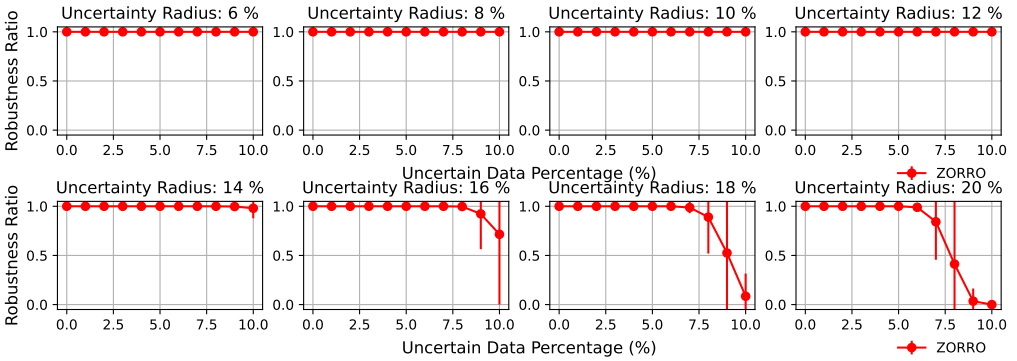
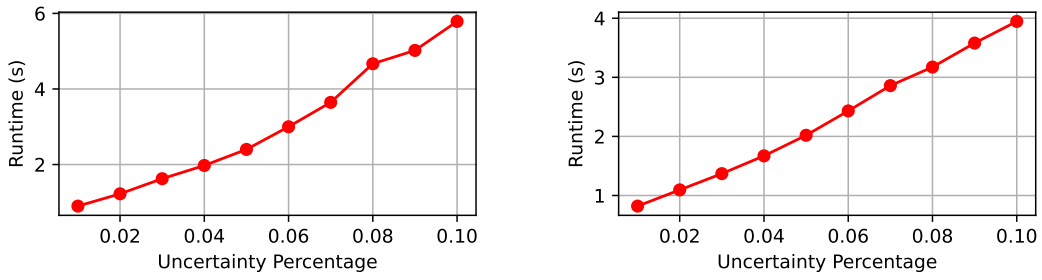


Figure 11: Robustness verification with errors in feature "acceleration".

M.7 Runtime of ZORRO

We evaluated the runtimes for computing the closed-form solution on the MPG dataset, varying the numbers of uncertain data points, in Fig 12. With the same number of uncertainty data points, uncertain features lead to more complex computations of covariance matrices, and result in higher runtimes compared to label uncertainty, where the covariance matrix remains real-valued.



(a) Uncertain feature

(b) Uncertain label

Figure 12: ZORRO runtimes on MPG data.

https://doi.org/10.52202/079017-3396

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We state our contributions and scope in Sec. 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our approach in Sec. 5 and app. I and J.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide intuitions in the main body of the paper (Sec. 3 and 4). We include the complete proofs in the appendix, and refer to them in the main body of the paper (App. F and H to J)

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Descriptions in the main body of the paper (Sec. 3 to 5), and the formulas and technical details in App. I and J are sufficient for reproducing our approach and experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

https://doi.org/10.52202/079017-3396

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We share the anonymize Github repository containing the data and code in the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe the experiment settings in detail in Sec. 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We repeat experiments with different random seeds and show the mean and standard deviation in the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the sufficient information on the computer resources in Sec. 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and make sure the paper conforms with it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of our technique in Sec. 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

https://doi.org/10.52202/079017-3396

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper has no such risk.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit all existing assests used in the paper, and include the license information in the reference.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide documentations for the released anonymized code (Sec. 5).

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

https://doi.org/10.52202/079017-3396

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.