Optimal Cross-Validation for Sparse Linear Regression

Ryan Cory-Wright
Department of Analytics, Marketing and Operations, Imperial Business School, London, UK
ORCID: 0000-0002-4485-0619
r.cory-wright@imperial.ac.uk

Andrés Gómez
Department of Industrial and Systems Engineering, Viterbi School of Engineering, University of Southern California, CA
ORCID: 0000-0003-3668-0653
gomezand@usc.edu

Given a high-dimensional covariate matrix and a response vector, ridge-regularized sparse linear regression selects a subset of features that explains the relationship between covariates and the response in an interpretable manner. To choose hyperparameters that control the sparsity level and amount of regularization, practitioners commonly use k-fold cross-validation. However, cross-validation substantially increases the computational cost of sparse regression as it requires solving many mixed-integer optimization problems (MIOs) for each hyperparameter combination. To address this computational burden, we derive computationally tractable relaxations of the k-fold cross-validation loss, facilitating hyperparameter selection while solving 50–80% fewer MIOs in practice. Our computational results demonstrate, across eleven real-world UCI datasets, that exact MIO-based cross-validation can be competitive with mature software packages such as glmnet and L0Learn —particularly when the sample-to-feature ratio is small.

Key words: Cross-validation; perspective formulation; sparse regression; bilevel convex relaxation

1. Introduction

Over the past fifteen years, Moore's law has spurred an explosion of high-dimensional datasets for scientific discovery across multiple fields (McAfee et al. 2012). These datasets often consist of a design matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ of explanatory variables and an output vector $\boldsymbol{y} \in \mathbb{R}^n$ of response variables. Accordingly, practitioners often aim to explain the response variables linearly via the equation $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Using this equation, the vector of regression coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ is inferred by minimizing the least squares (LS) error of the residuals $\boldsymbol{\epsilon}$.

Despite its computational efficiency, LS regression exhibits two practical limitations. First, when $p \gg n$, there is not enough data to accurately infer $\boldsymbol{\beta}$ via LS, and LS regression generates estimators which perform poorly out-of-sample due to the curse of dimensionality (Bühlmann and Van De Geer 2011, Gamarnik and Zadik 2022). Second, LS regression generically selects every feature, including irrelevant ones. This is a significant challenge when regression coefficients are used for high-stakes decision-making tasks and non-zero coefficients guide decisions.

To tackle the challenges of dimensionality and false discovery, sparse learning has emerged as a popular methodology for explaining the relationship between inputs $\boldsymbol{X}$ and outputs $\boldsymbol{y}$. A popular

sparse learning model is ridge-regularized sparse regression, which admits the formulation (Bertsimas and Van Parys 2020, Xie and Deng 2020, Hastie et al. 2020, Atamtürk and Gómez 2020, Kenney et al. 2021, Hazimeh et al. 2022, Liu et al. 2023)

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \quad \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \leq \tau, \tag{1}$$

where $\tau \in \{1,\ldots,p\}$ and $\gamma > 0$ are hyperparameters that respectively control the sparsity of $\boldsymbol{\beta}$ and the amount of ℓ_2^2 regularization (cf. Xu et al. 2008, Bertsimas and Copenhaver 2018), and we assume that $\boldsymbol{X}$ and $\boldsymbol{y}$ have undergone standard preprocessing so that $\boldsymbol{y}$ is a zero-mean vector and $\boldsymbol{X}$ has zero-mean, unit-variance columns, meaning γ penalizes each feature equally.

Problem (1) is NP-hard (Natarajan 1995) and computationally challenging, and early mixed-integer formulations could not scale to problems with thousands of features (Hastie et al. 2020). In a more positive direction, by developing and exploiting tight conic relaxations of appropriate substructures of (1), e.g., the perspective relaxation (Ceria and Soares 1999, Stubbs and Mehrotra 1999, Günlük and Linderoth 2010), more recent mixed-integer optimization methods, such as branch-and-bound (Hazimeh et al. 2022), can scale to larger instances with thousands of features. We refer to Bertsimas et al. (2021), Atamtürk and Gómez (2025) for reviews of perspective and related relaxations.

To be sure, the aforementioned works solve (1) rapidly. Unfortunately, they do not address arguably the most significant difficulty in performing sparse regression. The hyperparameters (τ,γ) are not known to the decision-maker ahead of time, as is often assumed in the literature for convenience. Rather, they must be selected by the decision-maker, which is potentially much more challenging than solving (1) for a single value of (τ,γ) (Hansen et al. 1992). Indeed, selecting (τ,γ) typically involves minimizing a validation metric over a grid of values, which is computationally expensive (Larochelle et al. 2007).

Perhaps the most popular validation procedure is hold-out (Hastie et al. 2009), where one omits a portion of the data when training the model and then evaluates performance on this hold-out set as a proxy for the model's test set performance. However, hold-out validation is sometimes called a high-variance approach (Hastie et al. 2009), because the validation errors can vary significantly depending on the hold-out set selected.

To reduce the variance in this procedure, a number of authors have proposed what we call *the cross-validation paradigm*. Early iterations of this paradigm, as reviewed by Stone (1978), suggest solving Problem (1) a total of n times, each time leaving out a single data point $i \in \{1,\ldots,n\}$, and estimating out-of-sample performance via the average prediction error of each estimator on its left-out observation. This approach is known as leave-one-out cross-validation (LOOCV).

A popular variant of LOOCV, known as k-fold cross-validation, involves removing subsets of n/k data points at a time and breaking the data into k folds in total, which significantly reduces

the computational burden of cross-validation while having less variance than a hold-out approach (Burman 1989, Arlot and Celisse 2010). However, even k-fold cross-validation may be prohibitive in the case of MIOs such as (1). Indeed, as identified by Hastie et al. (2020), with a time limit of 3 minutes per MIO, using 10-fold cross-validation to choose between subset sizes $\tau = 1, \ldots, 50$ in an instance of Problem (1) with $p = 100$ and $n = 500$ requires 25 hours of computational time.

For sparse regression, given a partition $\mathcal{N}_1, \ldots, \mathcal{N}_k$ of $[n]$, performing k-fold cross-validation corresponds to selecting hyperparameters γ, τ which minimize the function

$$h(\gamma, \tau) = \frac{1}{n} \sum_{j=1}^{k} \sum_{i \in \mathcal{N}_j} (y_i - \boldsymbol{x_i}^\top \boldsymbol{\beta}^{(\mathcal{N}_j)}(\gamma, \tau))^2 \tag{2}$$

where $\boldsymbol{\beta}^{(\mathcal{N}_j)}(\gamma, \tau)$ denotes an optimal solution to the following lower-level problem for any $\mathcal{N}_j$:

$$\boldsymbol{\beta}^{(\mathcal{N}_j)}(\gamma, \tau) \in \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \ \frac{\gamma}{2} \|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{y}^{(\mathcal{N}_j)} - \boldsymbol{X}^{(\mathcal{N}_j)} \boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \leq \tau, \tag{3}$$

$\gamma > 0$ is a hyperparameter, τ is a sparsity budget, $\boldsymbol{X}^{(\mathcal{N}_j)}, \boldsymbol{y}^{(\mathcal{N}_j)}$ denote the dataset with the data in $\mathcal{N}_j$ removed, and we take $\boldsymbol{\beta}^{(\mathcal{N}_j)}(\gamma, \tau)$ to be unique for a given τ, γ for convenience[1]. In words, $h(\gamma, \tau)$ denotes the average prediction error on each left-out fold for a sparse regressor with hyperparameters (γ, τ) trained on the remaining folds.

We remark that if all sets $\mathcal{N}_j$ are taken to be singletons and $k = n$, minimizing h corresponds to LOOCV. Moreover, if $k = 2$ and the term with $j = 2$ is removed from h, optimizing h reduces to minimizing the hold-out error. After selecting (γ, τ), practitioners usually train a final model on the entire dataset, by solving Problem (1) with the selected hyperparameter combination.

Our Approach: We propose techniques for obtaining strong bounds on validation metrics in polynomial time and leverage these bounds to design algorithms for minimizing the cross-validation error in Sections 2 and 3. By performing a perturbation analysis of perspective relaxations of sparse regression problems, we construct convex relaxations of the k-fold cross-validation error, which allows us to minimize it without explicitly solving MIOs at each data fold and for each hyperparameter combination. This results in a branch-and-bound algorithm for hyperparameter selection that is substantially more efficient than state-of-the-art methods such as grid search. As an aside, we remark that as cross-validation is more general than hold-out validation, our convex relaxations can be generalized immediately to the hold-out case.

In numerical experiments (Section 4), we assess the impact of our contributions. We observe on real UCI datasets that our branch-and-bound scheme reduces the number of MIOs that need to be solved by an average of 50%–80%. Further, we leverage our branch-and-bound scheme to design a cyclic alternating minimization scheme that iteratively minimizes τ and γ. We observe that on real UCI datasets, our scheme reduces the five-fold cross-validation error on underdetermined datasets by an average of 20% compared to MCP and by a few percent compared to glmnet and L0Learn,

although it is outperformed by these methods by 2%–40% on more overdetermined datasets. Our experiments demonstrate that the techniques developed in Sections 2 and 3 could be integrated within existing statistical software packages, when training an ML model involves solving a MIO.

1.1. Literature Review

Our work falls at the intersection of three areas of the optimization literature: (i) hyperparameter selection techniques for optimizing the performance of a machine learning model by selecting hyperparameters that perform well on a validation set, (ii) bilevel approaches that reformulate and solve hyperparameter selection problems as bilevel problems, and (iii) perspective reformulation techniques for mixed-integer problems with logical constraints, as discussed above. To put our contributions into context, we now review the two remaining areas of the literature.

Hyperparameter Selection Techniques for Machine Learning Problems: A wide variety of hyperparameter selection techniques have been proposed for machine learning problems such as sparse regression, including grid search (Larochelle et al. 2007) as reviewed in Section 1, and random search (cf. Bergstra and Bengio 2012). In random search, we let $\mathcal{L}$ be a random sample from a space of valid hyperparameters, e.g., a uniform distribution over $[10^{-4}, 10^{4}] \times [p]$ for sparse regression. Remarkably, in settings with many hyperparameters, random search usually outperforms grid search for a given budget on the number of training problems that can be solved, because validation functions often have a lower effective dimension than the number of hyperparameters present in the model (Bergstra and Bengio 2012). However, grid search remains competitive for problems with a small number of hyperparameters, such as sparse regression.

The modern era of hyperparameter selection strategies was ushered in by the increasing prominence of deep learning methods in applications from voice recognition to drug discovery (see LeCun et al. 2015, for a review). The volume of data available and the number of hyperparameters to be selected have challenged the aforementioned methods and led to new techniques, including evolutionary strategies, Bayesian optimization techniques (Frazier 2018) and bandit methods (Falkner et al. 2018). However, in sparse regression problems where we aim to optimize two hyperparameters, these methods are effectively equivalent to grid or random search. Further, none of these approaches provide locally optimal hyperparameter combinations with respect to the LOOCV error, which suggests there is room for improvement upon the state-of-the-art in sparse regression.

We point out that current approaches for hyperparameter selection are similar to existing methods for multi-objective mixed-integer optimization. While there has been recent progress in improving multi-objective algorithms for mixed-integer linear programs (Lokman and Köksalan 2013, Stidsen et al. 2014), a direct application of these methods might be unnecessarily expensive. Indeed, these approaches seek to compute the efficient frontier (Boland et al. 2015a,b), i.e., solving

problems for all possible values of the regularization parameter. In contrast, we are interested in only the combination of parameters that optimize a well-defined metric (e.g., the cross-validation error).

Bilevel Optimization for Hyperparameter Selection: In a complementary direction, several authors have proposed selecting hyperparameters via bilevel optimization (see Beck and Schmidt 2021, for a general theory), since Bennett et al. (2006) recognized that cross-validation is a special case of bilevel optimization. Therefore, in principle, we could minimize the cross-validation error in sparse regression by invoking bilevel techniques. Unfortunately, this approach seems intractable in both theory and practice (Ben-Ayed and Blair 1990, Hansen et al. 1992). Indeed, standard bilevel approaches such as dualizing the lower-level problem are challenging to apply in our context because our lower-level problems are non-convex and cannot easily be dualized.

Although bilevel hyperparameter optimization is slow in its original implementation, several authors have proposed making it more tractable by combining it with efficient modeling paradigms to obtain locally optimal sets of hyperparameters. Among others, Sinha et al. (2020) recommend taking a gradient-based approximation of the lower-level problem and thereby reducing the bilevel problem to a single-level problem, Okuno et al. (2021) advocate selecting hyperparameters by solving the KKT conditions of a bilevel problem, and Ye et al. (2022) propose solving bilevel hyperparameter problems via difference-of-convex methods to obtain a stationary point.

Specializing our review to regression, three works aim to optimize the performance of regression models on a validation metric. First, Takano and Miyashiro (2020) propose optimizing the k-fold validation loss, assuming all folds share the same support. Unfortunately, although their assumption improves their method's tractability, it may lead to subpar statistical performance because using the same set of non-zero regressors for all folds shares information between the folds. Second, Stephenson et al. (2021) propose first-order methods for minimizing the leave-one-out error in ridge regression problems (without sparsity constraints). However, it is unclear how to generalize their approach to settings with sparsity constraints. Finally, perhaps closest to our work, Kenney et al. (2021) propose a bisection algorithm for selecting the optimal sparsity parameter in a sparse regression problem by approximately minimizing the k-fold cross-validation error. It is, however, worth noting that this approach is not guaranteed to converge to an optimal sparsity parameter with respect to the k-fold error, because it does not develop lower bounds on the k-fold error.

1.2. Structure

The rest of the paper is laid out as follows:

- In Section 2, we observe that validation metrics are potentially expensive to evaluate, because they involve solving up to $k+1$ MIOs (in the k-fold case), and accordingly develop tractable lower and upper bounds that can be computed without solving any MIOs.

- In Section 3, we propose an efficient alternating minimization scheme for identifying locally optimal hyperparameters with respect to the validation error. Specifically, in Section 3.1, we develop an efficient scheme for minimizing the cross-validation error with respect to τ, and in Section 3.2, we propose a scheme for optimizing with respect to γ.
- In Section 4, we benchmark our proposed approaches on real UCI datasets. The proposed approach leads to a 50-80% reduction in the number of MIOs solved compared to standard grid search techniques. Moreover, it performs comparably to the glmnet, MCP and L0Learn software packages in terms of solution quality, especially in relatively underdetermined settings, thus demonstrating that the ideas in Sections 2–3 could be integrated within existing software packages.

Notation

We let non-boldface characters such as b denote scalars, lowercase bold-faced characters such as $\boldsymbol{x}$ denote vectors, uppercase bold-faced characters such as $\boldsymbol{A}$ denote matrices, and calligraphic uppercase characters such as $\mathcal{Z}$ denote sets. We let $[n]$ denote the running set of indices $\{1,\dots,n\}$, and $\|\boldsymbol{x}\|_0 := |\{j : x_j \neq 0\}|$ denote the ℓ_0 pseudo-norm, i.e., the number of non-zero entries in $\boldsymbol{x}$. Finally, we let $\boldsymbol{e}$ denote the vector of ones, and $\boldsymbol{0}$ denote the vector of all zeros.

Furthermore, we consistently use notation commonly found in the supervised learning literature. We consider a setting where we observe covariates $\boldsymbol{X} := (\boldsymbol{x}_1{}^\top;\dots;\boldsymbol{x}_n{}^\top) \in \mathbb{R}^{n\times p}$ and response data $\boldsymbol{y} := (y_1,\dots y_n) \in \mathbb{R}^n$. With this notation, the ith row of $\boldsymbol{X}$ is denoted by $\boldsymbol{x}_i^\top \in \mathbb{R}^p$. We say that $(\boldsymbol{X},\boldsymbol{y})$ is a training set, and let $\boldsymbol{\beta}$ denote a regressor fitted on this training set. In cross-validation, we are also interested in the behavior of $\boldsymbol{\beta}$ after leaving out portions of the training set. We let $(\boldsymbol{X}^{(i)},\boldsymbol{y}^{(i)})$ denote the training set with the ith data point left out, and denote by $\boldsymbol{\beta}^{(i)}$ the regressor obtained after leaving out the ith point. Similarly, given a partition $\mathcal{N}_1,\dots,\mathcal{N}_k$ of $[n]$ and $j \in [k]$, we let $(\boldsymbol{X}^{(\mathcal{N}_j)},\boldsymbol{y}^{(\mathcal{N}_j)})$ denote the training set with the jth fold left out, and $\boldsymbol{\beta}^{(\mathcal{N}_j)}$ be the associated regressor.

2. Convex Relaxations of k-fold Cross-Validation Error

In this section, we develop tractable upper and lower approximations of the k-fold cross-validation error of a sparse regression model, which can be evaluated at a given (γ,τ) without solving any MIOs. From a theoretical perspective, one of our main contributions is that, given $\boldsymbol{x} \in \mathbb{R}^p$, we show how to construct bounds $\underline{\xi},\overline{\xi}$ such that $\underline{\xi} \leq \boldsymbol{x}^\top \boldsymbol{\beta}^{(\mathcal{N}_j)} \leq \overline{\xi}$, which we can use to infer out-of-sample predictions. In particular, we leverage this insight to bound from above and below the function:

$$h(\gamma,\tau) = 1/n \sum_{j=1}^{k} h_j(\gamma,\tau) = 1/n \sum_{j=1}^{k} \sum_{i\in\mathcal{N}_j} \left(y_i - \boldsymbol{x_i}^\top \boldsymbol{\beta}^{(\mathcal{N}_j)}(\gamma,\tau)\right)^2, \tag{4}$$

i.e., the k-fold cross-validation error. Note that (4) is a restatement of (2).

2.1. Bounds on the Prediction Spread

Given any $0 < \epsilon \leq \gamma$, it is well-known that Problem (1) admits the conic quadratic relaxation:

$$\zeta_{\text{persp}} = \min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{z}\in[0,1]^p} \quad \|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \frac{\gamma-\epsilon}{2}\sum_{i=1}^p \frac{\beta_i^2}{z_i} + \frac{\epsilon}{2}\sum_{i=1}^p \beta_i^2 \quad \text{s.t.} \quad \sum_{i=1}^p z_i \leq \tau, \tag{5}$$

which is also known as the perspective relaxation (Ceria and Soares 1999, Xie and Deng 2020). In the formulation, ϵ is presumed to be a small number, and the associated regularization term is added to ensure strong convexity. Note that here and throughout the rest of the paper, there is an implicit dependence of ζ_{persp} on ϵ. If integrality constraints $\boldsymbol{z} \in \{0,1\}^p$ are added to (5), then the resulting mixed-integer optimization problem (MIO) is a reformulation of (1), where the logical constraints $\beta_i = 0$ if $z_i = 0$ $\forall i \in [p]$ are implicitly imposed via the domain of the perspective function β_i^2/z_i. Moreover, the optimal objective ζ_{persp} of (5) often provides tight lower bounds on the objective value of (1) (Pilanci et al. 2015, Bertsimas and Van Parys 2020, Askari et al. 2022), and the optimal solution $\boldsymbol{\beta}^\star_{\text{persp}}$ is often a good estimator in its own right. As we establish in our main results, the perspective relaxation can also be used to obtain accurate approximations of and bounds on the k-fold cross-validation error.

Our first main result (Theorem 1) reveals that any optimal solution of (1) lies in an ellipsoid centered at its continuous (perspective) relaxation, and whose radius depends on the duality gap:

THEOREM 1. *Given any bound*

$$\bar{u} \geq \min_{\boldsymbol{\beta}\in\mathbb{R}^p} \|\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{y}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 \text{ s.t. } \|\boldsymbol{\beta}\|_0 \leq \tau, \tag{6}$$

and any $0 \leq \epsilon < \gamma$ the inequality

$$(\boldsymbol{\beta}^\star_{persp} - \boldsymbol{\beta}^\star_{MIO})^\top \left(\boldsymbol{X}^\top\boldsymbol{X} + \frac{\epsilon}{2}\mathbb{I}\right)(\boldsymbol{\beta}^\star_{persp} - \boldsymbol{\beta}^\star_{MIO}) \leq (\bar{u} - \zeta_{persp}) \tag{7}$$

holds, where $\boldsymbol{\beta}^\star_{MIO}$ is an optimal solution of (6) *and $\boldsymbol{\beta}^\star_{persp}$ is optimal to* (5).

We note that Problem (6) is a restatement of Problem (1).

Proof of Theorem 1 Let

$$f(\boldsymbol{\beta}) := \min_{\boldsymbol{z}\in[0,1]^p:\boldsymbol{e}^\top\boldsymbol{z}\leq\tau} \quad \underbrace{\|\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{y}\|_2^2 + \frac{\epsilon}{2}\|\boldsymbol{\beta}\|_2^2}_{=q(\boldsymbol{\beta})} + \underbrace{\frac{\gamma-\epsilon}{2}\sum_{i\in[p]}\frac{\beta_i^2}{z_i}}_{=r(\boldsymbol{\beta})} \tag{8}$$

denote the objective value of the perspective relaxation at a given $\boldsymbol{\beta}$, which can be decomposed in a quadratic part $q(\boldsymbol{\beta})$ and a convex nonlinear part $r(\boldsymbol{\beta})$. We find that for any $\boldsymbol{\beta} \in \mathbb{R}^p$,

$$q(\boldsymbol{\beta}) = q(\boldsymbol{\beta}^\star_{\text{persp}}) + \nabla q(\boldsymbol{\beta}^\star_{\text{persp}})^\top(\boldsymbol{\beta}-\boldsymbol{\beta}^\star_{\text{persp}}) + (\boldsymbol{\beta}-\boldsymbol{\beta}^\star_{\text{persp}})^\top\left(\boldsymbol{X}^\top\boldsymbol{X}+\frac{\epsilon}{2}\mathbb{I}\right)(\boldsymbol{\beta}-\boldsymbol{\beta}^\star_{\text{persp}}), \text{ and}$$
$$r(\boldsymbol{\beta}) \geq r(\boldsymbol{\beta}^\star_{\text{persp}}) + \boldsymbol{s}^\top(\boldsymbol{\beta}-\boldsymbol{\beta}^\star_{\text{persp}})$$

for $\boldsymbol{s} \in \partial r(\boldsymbol{\beta}^\star_{\text{persp}})$ (the subdifferential of r). Moreover, since $\boldsymbol{\beta}^\star_{\text{persp}}$ is a minimizer of $f = q + r$, there exists $\bar{\boldsymbol{s}} \in \partial r(\boldsymbol{\beta}^\star_{\text{persp}})$ such that $\bar{\boldsymbol{s}} + \nabla q(\boldsymbol{\beta}^\star_{\text{persp}}) = \boldsymbol{0}$. Adding the two inequalities, we find that

$$(\boldsymbol{\beta} - \boldsymbol{\beta}^\star_{\text{persp}})^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \frac{\epsilon}{2}\mathbb{I}\right)(\boldsymbol{\beta} - \boldsymbol{\beta}^\star_{\text{persp}}) \leq f(\boldsymbol{\beta}) - f(\boldsymbol{\beta}^\star_{\text{persp}}).$$

Finally, setting $\boldsymbol{\beta} = \boldsymbol{\beta}^\star_{MIO}$ and using that $f(\boldsymbol{\beta}^\star_{MIO}) \leq \bar{u}$, we obtain the desired result. □

Using Theorem 1, we can compute bounds on $h_j(\gamma, \tau)$ in (4) by solving problems of the form

$$\min / \max_{\boldsymbol{\beta} \in \mathbb{R}^p} \quad \sum_{i \in \mathcal{N}_j} \left(y_i - \boldsymbol{x_i}^\top \boldsymbol{\beta}\right)^2 \tag{9a}$$

$$\text{s.t.} \quad (\boldsymbol{\beta}^{(\mathcal{N}_j)}_{\text{persp}} - \boldsymbol{\beta})^\top \left(\boldsymbol{X}^{(\mathcal{N}_j)^\top} \boldsymbol{X}^{(\mathcal{N}_j)} + \frac{\epsilon}{2}\mathbb{I}\right)(\boldsymbol{\beta}^{(\mathcal{N}_j)}_{\text{persp}} - \boldsymbol{\beta}) \leq (\bar{u}^{(\mathcal{N}_j)} - \zeta^{(\mathcal{N}_j)}_{\text{persp}}), \tag{9b}$$

where $\boldsymbol{\beta}^{(\mathcal{N}_j)}_{\text{persp}}$ and $\zeta^{(\mathcal{N}_j)}_{\text{persp}}$ are the optimal solution and objective value of the perspective relaxation with fold $\mathcal{N}_j$ removed, and $\bar{u}^{(\mathcal{N}_j)}$ is an associated upper bound. Bounds for the function $h(\gamma, \tau)$ then immediately follow by simply adding the bounds associated with $h_j(\gamma, \tau)$ for all $j \in [k]$.

Remark 1 (Computability of the bounds). Observe that a lower bound on the k-fold error can easily be computed by solving a convex quadratically constrained quadratic problem, while an upper bound can be computed by noticing that the maximization problem (9) is a trust region problem in $\boldsymbol{\beta}$, which can be reformulated as a semidefinite problem (Hazan and Koren 2016). One could further tighten these bounds by imposing a sparsity constraint on $\boldsymbol{\beta}$, but this may not be practically tractable.

2.2. Closed-form Bounds on the Prediction Spread

While solving the perspective relaxation (5) is necessary to solve the MIO (6) via branch-and-bound (in particular, the perspective relaxation is the root node in a branch-and-bound scheme (Mazumder et al. 2023)), the additional two optimization problems (9) are not. Moreover, solving trust-region problems can be expensive in large-scale problems. Accordingly, in this section, we present alternative bounds that may be weaker, but can be obtained in closed form. In numerical experiments (Section 4), these closed-form bounds typically reduce the number of MIOs that need to be solved by 50%–80% when compared to grid search.

Theorem 2. *Given any vector $\boldsymbol{x} \in \mathbb{R}^p$ and any bound*

$$\bar{u} \geq \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 \text{ s.t. } \|\boldsymbol{\beta}\|_0 \leq \tau, \tag{10}$$

the inequalities

$$\boldsymbol{x}^\top \boldsymbol{\beta}^\star_{persp} - \sqrt{(\bar{u} - \zeta_{persp})\, \boldsymbol{x}^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \frac{\epsilon}{2}\mathbb{I}\right)^{-1} \boldsymbol{x}} \leq \boldsymbol{x}^\top \boldsymbol{\beta}^\star_{MIO} \leq \boldsymbol{x}^\top \boldsymbol{\beta}^\star_{persp} + \sqrt{(\bar{u} - \zeta_{persp})\, \boldsymbol{x}^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \frac{\epsilon}{2}\mathbb{I}\right)^{-1} \boldsymbol{x}}$$

hold, where $\boldsymbol{\beta}^\star_{MIO}$ is an optimal solution of (10) *and $\boldsymbol{\beta}^\star_{persp}$ is optimal to* (5).

Proof of Theorem 2 From Theorem 1, we have the inequality

$$(\boldsymbol{\beta}^\star_{\text{persp}} - \boldsymbol{\beta}^\star_{\text{MIO}})^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \frac{\epsilon}{2}\mathbb{I}\right)(\boldsymbol{\beta}^\star_{\text{persp}} - \boldsymbol{\beta}^\star_{\text{MIO}}) \leq (\bar{u} - \zeta_{\text{persp}}). \tag{11}$$

By the Schur Complement Lemma (see, e.g., Boyd et al. 1994), this is equivalent to

$$(\bar{u}-\zeta_{\text{persp}})\left(\boldsymbol{X}^\top\boldsymbol{X}+\frac{\epsilon}{2}\mathbb{I}\right)^{-1}\succeq(\boldsymbol{\beta}^\star_{\text{persp}}-\boldsymbol{\beta}^\star_{\text{MIO}})(\boldsymbol{\beta}^\star_{\text{persp}}-\boldsymbol{\beta}^\star_{\text{MIO}})^\top.$$

Next, we can left/right multiply this expression by an arbitrary matrix $\boldsymbol{W}\in\mathbb{R}^{m\times p}$. This gives:

$$(\bar{u}-\zeta_{\text{persp}})\boldsymbol{W}\left(\boldsymbol{X}^\top\boldsymbol{X}+\frac{\epsilon}{2}\mathbb{I}\right)^{-1}\boldsymbol{W}^\top\succeq(\boldsymbol{W}\boldsymbol{\beta}^\star_{\text{persp}}-\boldsymbol{W}\boldsymbol{\beta}^\star_{\text{MIO}})(\boldsymbol{W}\boldsymbol{\beta}^\star_{\text{persp}}-\boldsymbol{W}\boldsymbol{\beta}^\star_{\text{MIO}})^\top.$$

In particular, setting $\boldsymbol{W}=\boldsymbol{x}^\top$ for a vector $\boldsymbol{x}\in\mathbb{R}^p$ gives the inequality

$$(\bar{u}-\zeta_{\text{persp}})\boldsymbol{x}^\top\left(\boldsymbol{X}^\top\boldsymbol{X}+\frac{\epsilon}{2}\mathbb{I}\right)^{-1}\boldsymbol{x}\geq(\boldsymbol{x}^\top(\boldsymbol{\beta}^\star_{\text{persp}}-\boldsymbol{\beta}^\star_{\text{MIO}}))^2,$$

which we rearrange to obtain the result. □

COROLLARY 1. *For any $\boldsymbol{W}\in\mathbb{R}^{m\times p}$ we have that*

$$(\bar{u}-\zeta_{persp})\text{tr}\left(\boldsymbol{W}\left(\boldsymbol{X}^\top\boldsymbol{X}+\frac{\epsilon}{2}\mathbb{I}\right)^{-1}\boldsymbol{W}^\top\right)\geq\|\boldsymbol{W}(\boldsymbol{\beta}^\star_{persp}-\boldsymbol{\beta}^\star_{MIO})\|_2^2.$$

Applying Theorem 2 to the problem

$$\bar{u}^{(\mathcal{N}_j)}\geq\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\ \|\boldsymbol{X}^{(\mathcal{N}_j)}\boldsymbol{\beta}-\boldsymbol{y}^{(\mathcal{N}_j)}\|_2^2+\frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2\ \text{s.t.}\ \|\boldsymbol{\beta}\|_0\leq\tau,\tag{12}$$

where $\boldsymbol{x}=\boldsymbol{x_i}$ was chosen as the reference vector, we have the bounds

$$\underline{\xi}_{i,j}:=\boldsymbol{x_i}^\top\boldsymbol{\beta}^{(\mathcal{N}_j)\star}_{persp}-\sqrt{\boldsymbol{x_i}^\top\left(\boldsymbol{X}^{(\mathcal{N}_j)^\top}\boldsymbol{X}^{(\mathcal{N}_j)}+\frac{\epsilon}{2}\mathbb{I}\right)^{-1}\boldsymbol{x_i}\left(\bar{u}^{(\mathcal{N}_j)}-\zeta^{(\mathcal{N}_j)}\right)},\tag{13}$$

$$\bar{\xi}_{i,j}:=\boldsymbol{x_i}^\top\boldsymbol{\beta}^{(\mathcal{N}_j)\star}_{persp}+\sqrt{\boldsymbol{x_i}^\top\left(\boldsymbol{X}^{(\mathcal{N}_j)^\top}\boldsymbol{X}^{(\mathcal{N}_j)}+\frac{\epsilon}{2}\mathbb{I}\right)^{-1}\boldsymbol{x_i}\left(\bar{u}^{(\mathcal{N}_j)}-\zeta^{(\mathcal{N}_j)}\right)}\tag{14}$$

where $\underline{\xi}\leq\boldsymbol{x_i}^\top\boldsymbol{\beta}^{(\mathcal{N}_j)\star}_{MIO}\leq\bar{\xi}$, where $\boldsymbol{\beta}^{(\mathcal{N}_j)\star}_{MIO}$ is the optimal solution of (12). We can then compute bounds on the ith prediction error associated with fold j, namely

$$\nu_{i,j}=(\boldsymbol{x_i}^\top\boldsymbol{\beta}^{(\mathcal{N}_j)\star}_{MIO}-y_i)^2,$$

as formalized in Corollary 2.

COROLLARY 2. *We have the following bounds on the ith prediction error associated with fold j*

$$\max\left((y_i-\underline{\xi}_{i,j})^2,(y_i-\bar{\xi}_{i,j})^2\right)\geq\nu_{i,j}(\gamma,\tau)\geq\begin{cases}(y_i-\underline{\xi}_{i,j})^2 & \text{if } y_i<\underline{\xi}_{i,j}\\ 0 & \text{if } y_i\in[\underline{\xi}_{i,j},\bar{\xi}_{i,j}],\\ (\bar{\xi}_{i,j}-y_i)^2 & \text{if } y_i>\bar{\xi}_{i,j}.\end{cases}\tag{15}$$

Moreover, since $h(\gamma,\tau)=\frac{1}{n}\sum_{j=1}^k\sum_{i\in\mathcal{N}_j}\nu_{i,j}(\gamma,\tau)$, we can compute lower and upper bounds on the k-th fold cross-validation error by adding the individual bounds. Observe that the bounds computed by summing disaggregated bounds could be substantially worse than those obtained by letting $\boldsymbol{W}$ be a matrix with all omitted columns in the jth fold of $\boldsymbol{X}$ in the proof of Theorem 2. Nonetheless, the approach outlined here may be the only feasible one in large-scale instances, as it is obtained directly from the perspective relaxation without solving additional optimization problems, whereas an aggregated approach would involve solving an auxiliary semidefinite optimization problem.

Despite the loss in quality, we show in our computational sections that (combined with the methods discussed in §3), the disaggregated bounds are sufficient to lead to a 50%–80% reduction in the number of MIOs solved with respect to grid search.

We conclude this subsection with three remarks.

Remark 2 (Choice of the strong convexity parameter). While it may appear that large values ϵ result in tighter bounds in Theorem 2, we point out that large values also negatively affect the quality of the lower bound ζ_{persp}. Note that if $\boldsymbol{X}^\top \boldsymbol{X}$ is invertible, then we can also set $\epsilon = 0$ – the approach we use if $n > p$.

Remark 3 (Relaxation Tightness). If the perspective relaxation is tight, as occurs when n is sufficiently large under certain assumptions on the data generation process (Pilanci et al. 2015, Reeves et al. 2019) then $\underline{\xi} = \bar{\xi} = \boldsymbol{x_i}^\top \boldsymbol{\beta}^\star_{persp}$, and Corollary 2's bounds on the cross-validation error are tight by definition. Otherwise, as pointed out in Remark 4, (15)'s bound quality depends on the tightness of the relaxation and on how close the features $\boldsymbol{x}_i$ are to the rest of the data.

Remark 4 (Intuition). Theorem 2 states that $\boldsymbol{x}^\top \boldsymbol{\beta}^\star_{MIO} \approx \boldsymbol{x}^\top \boldsymbol{\beta}^\star_{persp}$, where the approximation error is determined by two components. The quantity $\sqrt{\bar{u} - \zeta_{\text{persp}}}$ is related to the strength of the perspective relaxation, with a stronger relaxation resulting in a better approximation. The quantity $\sqrt{\boldsymbol{x}^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \frac{\epsilon}{2}\mathbb{I}\right)^{-1} \boldsymbol{x}}$ is related to the likelihood that $\boldsymbol{x}$ is generated from the same distribution as the rows of $\boldsymbol{X}$, with larger likelihoods resulting in better approximations. Indeed, if $n > p$, each column of $\boldsymbol{X}$ has 0 mean but has not been standardized, and each row of $\boldsymbol{X}$ is generated iid from a multivariate Gaussian distribution, then $\frac{n(n-1)}{n+1} \boldsymbol{x}^\top \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{x} \sim T^2(p, n-1)$ is Hotelling's two-sample T-square test statistic (Hotelling 1931), used to test whether $\boldsymbol{x}$ is generated from the same Gaussian distribution. Note that if $\boldsymbol{x}$ is drawn from the same distribution as the rows of $\boldsymbol{X}$ (as may be the case in cross-validation), then $\mathbb{E}\left[\boldsymbol{x}^\top \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{x}\right] = \frac{p(n+1)}{n(n-p-2)}$ for $n > p+2$.

2.3. Further Improvements for Lower Bounds

Corollary 2 implies we obtain valid upper and lower bounds on the k-fold cross-validation loss h at a given hyperparameter combination γ, τ after solving k perspective relaxations and computing n terms of the form

$$\sqrt{\boldsymbol{x_i}^\top \left(\boldsymbol{X}^{(\mathcal{N}_j)^\top} \boldsymbol{X}^{(\mathcal{N}_j)} + \frac{\epsilon}{2}\mathbb{I}\right)^{-1} \boldsymbol{x_i}}.$$

A drawback of Corollary 2 is that if $\boldsymbol{x_i}^\top \boldsymbol{\beta}^\star_{persp} \approx y_i$ for each $i \in \mathcal{N}_j$, i.e., the prediction of the perspective relaxation (without the jth fold) is close to the response associated with point i, then Corollary 2's lower bound is 0. A similar situation can happen with the stronger bounds for $h_j(\gamma, \tau)$ obtained from Theorem 1 and Problem (9). We now propose a different bound on $h_j(\gamma, \tau)$, which is sometimes effective in this circumstance.

First, define the function $F(\gamma,\tau)$ to be the in-sample training error without removing any folds and with parameters (γ,τ),

$$F(\gamma,\tau) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x_i}^\top \boldsymbol{\beta}(\gamma,\tau))^2 \quad \text{s.t.} \quad \boldsymbol{\beta}(\gamma,\tau) \in \underset{\boldsymbol{\beta}\in\mathbb{R}^p:\ \|\boldsymbol{\beta}\|_0\leq\tau}{\arg\min}\ \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{X\beta} - \boldsymbol{y}\|_2^2,$$

and let $F_j(\gamma,\tau) := \sum_{i\in\mathcal{N}_j}\left(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta}(\gamma,\tau)\right)^2$ denote the training error associated with the jth fold, with $1/n\sum_{j=1}^{k}F_j(\gamma,\tau) = F(\gamma,\tau)$. Observe that evaluating h involves solving k MIOs, while evaluating F requires solving one.

PROPOSITION 1. *For any $\gamma>0$, any $\tau\in[p]$ and any $j\in[k]$, $F_j(\gamma,\tau)\leq h_j(\gamma,\tau)$. Moreover, we have that $F(\gamma,\tau)\leq h(\gamma,\tau)$.*

Proof of Proposition 1 Given $j\in[k]$, consider the following two optimization problems

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p:\|\boldsymbol{\beta}\|_0\leq\tau}\sum_{i=1}^{n}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta})^2 + \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 \tag{16}$$

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p:\|\boldsymbol{\beta}\|_0\leq\tau}\sum_{i\notin\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta})^2 + \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2, \tag{17}$$

let $\boldsymbol{\beta^\star}$ be an optimal solution of (16), and let $\boldsymbol{\beta^j}$ be an optimal solution of (17). Since

$$\sum_{i\notin\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta^j})^2 + \frac{\gamma}{2}\|\boldsymbol{\beta^j}\|_2^2 \leq \sum_{i\notin\mathcal{N}_j}(y_i - \boldsymbol{x}^\top\boldsymbol{\beta^\star})^2 + \frac{\gamma}{2}\|\boldsymbol{\beta^\star}\|_2^2, \quad \text{and}$$

$$\sum_{i\notin\mathcal{N}_j}(y_i - \boldsymbol{x}^\top\boldsymbol{\beta^j})^2 + \sum_{i\in\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta^j})^2 + \frac{\gamma}{2}\|\boldsymbol{\beta^j}\|_2^2 \geq \sum_{i\notin\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta^\star})^2 + \sum_{i\in\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta^\star})^2 + \frac{\gamma}{2}\|\boldsymbol{\beta^\star}\|_2^2,$$

we conclude that $\sum_{i\in\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta^\star})^2 \leq \sum_{i\in\mathcal{N}_j}(y_i - \boldsymbol{x_i}^\top\boldsymbol{\beta^j})^2$. The result immediately follows. □

Next, we develop a stronger bound on the k-fold error, by observing that our original proof technique relies on interpreting the optimal solution when training on the entire dataset as a feasible solution when leaving out the jth fold, and that this feasible solution can be improved to obtain a tighter lower bound. Therefore, given any $\boldsymbol{z}\in\{0,1\}^p$, let us define the function:

$$f^{(\mathcal{N}_j)}(\boldsymbol{z}) := \min_{\boldsymbol{\beta}\in\mathbb{R}^p} \quad \frac{\gamma}{2}\sum_{j\in[p]}\beta_j^2 + \|\boldsymbol{X}^{(\mathcal{N}_j)}\boldsymbol{\beta} - \boldsymbol{y}^{(\mathcal{N}_j)}\|_2^2 \quad \text{s.t.} \quad \beta_j = 0 \text{ if } z_j = 0\ \forall j\in[p],$$

to be the optimal training loss (including regularization) when we leave out the jth fold and have the binary support vector $\boldsymbol{z}$. Then, fixing γ,τ and letting $u^\star$ denote the optimal objective value of (16), i.e., the optimal training loss on the entire dataset (including regularization) and $\boldsymbol{\beta}^{(\mathcal{N}_j)}(\boldsymbol{z})$ denote an optimal choice of $\boldsymbol{\beta}$ for this $\boldsymbol{z}$, we have the following result:

PROPOSITION 2. *For any τ-sparse binary vector $\boldsymbol{z}$, the following inequality holds:*

$$u^\star \leq f^{(\mathcal{N}_j)}(\boldsymbol{z}) + \sum_{i\in\mathcal{N}_j}\left(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}^{(\mathcal{N}_j)}(\boldsymbol{z})\right)^2 \tag{18}$$

Proof of Proposition 2 The right-hand side of this inequality corresponds to the objective value of a feasible solution to (16), while $u^\star$ is the optimal objective value of (16). □

COROLLARY 3. *Let $\boldsymbol{z}$ denote a τ-sparse binary vector. Then, we have the following bound on the jth partial cross-validation error:*

$$h_j(\gamma,\tau) \geq u^\star - f^{(\mathcal{N}_j)}(\boldsymbol{z}). \tag{19}$$

Proof of Corollary 3 The right-hand side of this bound is maximized by setting $\boldsymbol{z}$ to be a binary vector which minimizes $f^{(\mathcal{N}_j)}(\boldsymbol{z})$, and therefore this bound is valid for any $\boldsymbol{z}$. □

We close this section with two remarks:

REMARK 5 (BOUND QUALITY). Observe that bound (19) is at least as strong as $F_j(\gamma,\tau)$ with $\boldsymbol{z}$ encoding an optimal choice of support in (16). Indeed, if $\boldsymbol{\beta}^{(\mathcal{N}_j)}(\boldsymbol{z})$ solves (16), then both bounds agree and equal $h_j(\gamma,\tau)$ but otherwise (19) is strictly stronger. Moreover, since $F_j(\gamma,\tau)$ is typically nonzero, then the bound (19) is positive as well and can improve upon the lower bound in (15). Finally, it is easy to construct examples in which the lower bound in (15) is stronger than (19) and vice versa, so neither lower bound dominates the other; see Section EC.2.

REMARK 6 (COMPUTATIONAL EFFICIENCY). Computing lower bound (19) for each $j \in [k]$ requires solving at least one MIO, corresponding to (16), which is a substantial improvement over the k MIOs required to compute h but may still be an expensive computation. However, using any lower bound on $u^\star$, for example, corresponding to the optimal solution of a perspective relaxation, gives valid lower bounds. Therefore, in practice, we suggest using a heuristic instead to bound h_j from below, e.g., rounding a perspective relaxation as suggested by Xie and Deng (2020), Bertsimas et al. (2021), building upon the work of Pilanci et al. (2015).

3. Optimizing the Cross-Validation Loss

In this section, we present an efficient alternating minimization scheme that identifies (approximately) optimal hyperparameters (γ,τ) with respect to the k-fold cross-validation error as previously defined in (2), (4):

$$h(\gamma,\tau) := \frac{1}{n}\sum_{j\in[k]} h_j(\gamma,\tau) \tag{20}$$

by iteratively minimizing τ and γ. Specifically, with initialization τ_0,γ_0, we repeatedly solve the following two optimization problems:

$$\tau_t \in \arg\min_{\tau\in[p]} \quad h(\gamma_t,\tau), \tag{21}$$

$$\gamma_{t+1} \in \arg\min_{\gamma>0} \quad h(\gamma,\tau_t), \tag{22}$$

until we either detect a cycle, converge to a locally optimal solution to (20), or exceed a user-imposed limit on the number of iterations for this alternating minimization procedure. To develop this scheme, in Section 3.1 we propose an efficient technique for solving Problem (21) (Algorithm 1), and in Section 3.2 we propose an efficient technique for (approximately) solving Problem (22).

Our overall scheme alternates between solving the two minimization problems. Accordingly, our scheme could also be used to identify an optimal choice of γ if τ is already known, e.g., in a context where regulatory constraints specify the number of features that may be included in a model.

Our overall approach is motivated by three key observations. First, we design a method that obtains local, rather than global, minima, because h is a highly non-convex function and even evaluating h requires solving k MIOs, which suggests that global minima of h may not be attainable in a practical amount of time at scale. Second, we use alternating minimization to seek local minima because if either τ or γ is fixed, it is possible to efficiently optimize the remaining hyperparameter with respect to h by leveraging the convex relaxations developed in the previous section. Third, we should expect our alternating minimization scheme to perform well in practice, because similar schemes are highly effective in other machine learning contexts, e.g., solving certain matrix completion problems in polynomial time (Mazumder et al. 2011, Cifuentes and Moitra 2022).

3.1. Parametric Optimization of k-fold Error With Respect to Sparsity

Consider the following optimization problem, where γ is fixed here and throughout this subsection:

$$\begin{aligned} \min_{\tau\in[p]} \quad & h(\gamma,\tau) := \frac{1}{n}\sum_{j\in[k]}\sum_{i\in\mathcal{N}_j}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}^{(\mathcal{N}_j)})^2, \\ \text{s.t.} \quad & \boldsymbol{\beta}^{(\mathcal{N}_j)} \in \underset{\boldsymbol{\beta}\in\mathbb{R}^p:\ \|\boldsymbol{\beta}\|_0\leq\tau}{\arg\min} \ \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{X}^{(\mathcal{N}_j)}\boldsymbol{\beta} - \boldsymbol{y}^{(\mathcal{N}_j)}\|_2^2 \quad \forall j\in[k]. \end{aligned} \tag{23}$$

Note that solving Problem (23) corresponds to performing one iteration of the alternating minimization scheme described at the start of the section, namely minimizing h with respect to τ with γ fixed.

Problem (23) can be solved by complete enumeration, i.e., for each $\tau\in[p]$, we compute an optimal $\beta^{(\mathcal{N}_j)}$ for each $j\in[k]$ by solving an MIO. This involves solving $(k+1)p$ MIOs, which is extremely expensive at scale. We now propose a technique for minimizing h without solving all these MIOs:

Algorithm 1 has two main phases, each implemented as a loop over sparsity budgets τ and then over folds j. In the first phase, we construct valid lower and upper bounds on $h_j(\gamma,\tau)$ for each $j\in[k]$ and each τ without solving any MIOs. We begin by solving, for each potential sparsity budget $\tau\in[p]$, the perspective relaxation with all data points included. Call this relaxation's objective value $\bar{v}_\tau$. We then solve each perspective relaxation that arises after omitting one data fold $\mathcal{N}_j : j\in[k]$, with objective values $v_{\tau,j}$ and solutions $\boldsymbol{\beta}_{\tau,j}$. Next, we compute lower and upper bounds on the k-fold error $h_j(\gamma,\tau)$ using the methods derived in Section 2, which are summarized in the routine `compute_bounds` described in Algorithm 2. By solving $\mathcal{O}(pk)$ relaxations (and no MIOs), we have upper and lower estimates on the k-fold error that are often accurate in practice, as described by Theorem 2.

After completing the first loop in Algorithm 1, one may already terminate the algorithm. Indeed, according to our numerical experiments in Section 4, this already provides high-quality solutions. Alternatively, one may proceed with the second phase of Algorithm 1 and solve (21) to optimality, at the expense of solving (a potentially large number of) MIOs.

In the second phase, Algorithm 1 identifies the cardinality $\tau^\star$ with the best lower bound (and thus, in an optimistic scenario, the best potential value). Then, it identifies the index $j^* \in [k]$ such that partition $\mathcal{N}_{j^\star}$ has the largest uncertainty around the k-fold estimate $h_{j^\star}(\gamma,\tau^\star)$, and solves an MIO to compute the exact partial k-fold error. This process is repeated until (23) is solved within a certain prescribed optimality tolerance ϵ, or a suitable termination condition (e.g., a limit on computational time) is met. Note that if an MIO were solved for all τ and all folds of the data, then we could immediately solve (23), and thus Algorithm 1 terminates in pk iterations of the second phase in the worst case.

To solve each MIO in Algorithm 1, we invoke a Generalized Benders Decomposition scheme (Geoffrion 1972), which was specialized to sparse regression problems by Bertsimas and Van Parys (2020), enhanced with some ideas from the optimization literature summarized in the works Bertsimas et al. (2020), Hazimeh and Mazumder (2020). For the sake of conciseness, we defer these implementation details to Appendix EC.1.

Algorithm 1 in Action: Figure 1 depicts visually the lower and upper bounds on h from Algorithm 2 (left) and after running Algorithm 1 to completion (right) on a synthetic sparse regression instance generated in the fashion described by Bertsimas et al. (2020) and restated in Section EC.1.2 for completeness, with $k = n,\ n = 200, p = 20,\ \gamma = 1/\sqrt{n},\ \tau_{\text{true}} = 10,\ \rho = 0.7,$ $\nu = 1$, where $\tau \in \{2, \ldots, 19\}$, we have the tolerance parameters $r = kp$ and $\epsilon = 10^{-2}$, and using the outer-approximation method of Bertsimas and Van Parys (2020) as our solver for each MIO with a time limit of 60s. We observe that Algorithm 1 solved 1694 MIOs to identify the optimal τ, which is a 53% improvement over complete enumeration. Interestingly, when $\tau =$ 19, the perspective relaxation is tight after omitting any fold of the data and we have tight bounds on the LOOCV error without solving any MIOs. In Section 4.1, we test Algorithm 1 on real datasets and find that it reduces the number of MIOs that need to be solved by 50-80% with respect to complete enumeration. For more information on how the bounds evolve over time, we provide a GIF with one frame each time a MIO is solved at the link `https://drive.google.com/file/d/1EZdNwlV9sEEnludGGM7v2nGpB7tzZvz4/view?usp=sharing`.

3.2. Parametric Optimization of k-fold Error With Respect to γ

In this section, we propose a technique for approximately minimizing the k-fold error with respect to the regularization hyperparameter γ. Note that this corresponds to performing half an iteration

Algorithm 1: Computing optimal sparsity parameter for k-fold error

Data: γ: ℓ_2^2 regularization parameter; $\epsilon > 0$: desired optimality tolerance; r: budget on number of MIOs

Result: Cardinality with best estimated k-fold error

for $\tau \in [p]$ **do**
  $\bar{v}_\tau \leftarrow \min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{z}\in[0,1]^p} \|\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{y}\|_2^2 + \frac{\gamma}{2}\sum_{i=1}^p \beta_i^2/z_i$ s.t. $\boldsymbol{e}^\top \boldsymbol{z} \leq \tau$
  for $j \in [k]$ **do**
    $v_{\tau,j} \leftarrow \min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{z}\in[0,1]^p} \|\boldsymbol{X}^{(\mathcal{N}_j)}\boldsymbol{\beta}-\boldsymbol{y}^{(\mathcal{N}_j)}\|_2^2 + \frac{\gamma}{2}\sum_{i=1}^p \beta_i^2/z_i$ s.t. $\boldsymbol{e}^\top \boldsymbol{z} \leq \tau$
    $\boldsymbol{\beta}_{\tau,j}, \boldsymbol{z}_{\tau,j} \in \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{z}\in[0,1]^p} \|\boldsymbol{X}^{(\mathcal{N}_j)}\boldsymbol{\beta}-\boldsymbol{y}^{(\mathcal{N}_j)}\|_2^2 + \frac{\gamma}{2}\sum_{i=1}^p \beta_i^2/z_i$ s.t. $\boldsymbol{e}^\top \boldsymbol{z} \leq \tau$
    $\hat{\boldsymbol{z}}_{\tau,j} \leftarrow \texttt{round}(\boldsymbol{z}_{\tau,j})$
    $u_{\tau,j} \leftarrow \min_{\boldsymbol{\beta}\in\mathbb{R}^p} \|\boldsymbol{X}^{(\mathcal{N}_j)}\boldsymbol{\beta}-\boldsymbol{y}^{(\mathcal{N}_j)}\|_2^2 + \frac{\gamma}{2}\sum_{i=1}^p \beta_i^2$ s.t. $\beta_i = 0$ if $\hat{z}_i = 0$ $\forall i \in [p]$
    $\zeta_j^L(\tau), \zeta_j^U(\tau) \leftarrow \texttt{compute\_bounds}(\mathcal{N}_j, \boldsymbol{\beta}_{\tau,j}, \bar{v}_\tau, v_{\tau,j}, u_{\tau,j})$
$LB \leftarrow \min_{\tau\in[p]} \sum_{j\in[k]} \zeta_j^L(\tau)$; $UB \leftarrow \min_{\tau\in[p]} \sum_{j\in[k]} \zeta_j^U(\tau)$; `// Bounds on k-fold`
$num\_mip \leftarrow 0$
repeat
  $\bar{\tau} \leftarrow \arg\min_{\tau\in[p]} \sum_{j\in[k]} \zeta_j^L(\tau)$; `// Cardinality with best bound`
  $j^\star \leftarrow \arg\max_{j\in[k]} \{\zeta_j^U(\bar{\tau}) - \zeta_j^L(\bar{\tau})\}$; `// Fold with largest k-fold uncertainty`
  $\boldsymbol{\beta}^\star \leftarrow \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{z}\in\{0,1\}^p} \|\boldsymbol{X}^{(\mathcal{N}_j^*)}\boldsymbol{\beta}-\boldsymbol{y}^{(\mathcal{N}_j^*)}\|_2^2 + \frac{\gamma}{2}\sum_{i=1}^p \beta_i^2/z_i$ s.t. $\boldsymbol{e}^\top \boldsymbol{z} \leq \bar{\tau}$; `// Solve the training problem with fold` $\mathcal{N}_{j^\star}$ `left out.`
  $h_{j^\star}(\gamma, \bar{\tau}) \leftarrow \sum_{i\in\mathcal{N}_{j^\star}} (y_i - x_i^\top \beta^\star)^2$; `// Evaluate valid loss on held-out fold` $\mathcal{N}_{j^\star}$.
  $\zeta_{j^\star}^L(\bar{\tau}) \leftarrow h_{j^\star}(\gamma, \bar{\tau})$, $\zeta_{j^\star}^U(\bar{\tau}) \leftarrow h_{j^\star}(\gamma, \bar{\tau})$
  $LB \leftarrow \min_{\tau\in[p]} \sum_{j\in[k]} \zeta_j^L(\tau)$
  $UB \leftarrow \min_{\tau\in[p]} \sum_{j\in[k]} \zeta_j^U(\tau)$
  $num\_mip \leftarrow num\_mip + 1$
until $(UB - LB)/UB \leq \epsilon$ *or* $num\_mip > r$;
return $\arg\min_{\tau\in[p]} \sum_{j\in[k]} h_j(\gamma, \tau)$; `// Cardinality with best error`

of the alternating minimization scheme described at the start of the section, namely approximately minimizing h with respect to γ with τ fixed.

We begin with two observations from the literature. First, as observed by Stephenson et al. (2021), the LOOCV error $h(\gamma, \tau)$ is often quasi-convex with respect to γ when $\tau = p$. Second, Bertsimas et al. (2021) and Bertsimas and Cory-Wright (2022) report that, for sparsity-constrained problems, the optimal support does not often change as we vary γ. Combining these observations suggests that, after optimizing τ with γ fixed, a good strategy for minimizing h with respect to γ is to fix the optimal support $\boldsymbol{z}^{(\mathcal{N}_j)}$ with respect to each fold j and invoke a root-finding method to find a γ which locally minimizes h.

Algorithm 2: `compute_bounds`$(\mathcal{N}_j, \boldsymbol{\beta}, \bar{v}, v, u)$

Data: $\mathcal{N}_j$: fold left out; $\boldsymbol{\beta}$: optimal solution of perspective relaxation with $\mathcal{N}_j$ left out; $\bar{v}$: lower bound of the objective value of MIO with all data; v: optimal objective value of perspective relaxation with $\mathcal{N}_j$ left out; u: upper bound of the objective value of MIO with $\mathcal{N}_j$ left out

Result: Lower and upper bounds on the k-fold error attributable to fold j

for $i \in \mathcal{N}_j$ **do**

$$\underline{\xi_i} \leftarrow \boldsymbol{x_i}^\top \boldsymbol{\beta} - \sqrt{\boldsymbol{x_i}^\top \left(\boldsymbol{X}^{(\mathcal{N}_j)^\top} \boldsymbol{X}^{(\mathcal{N}_j)} + \frac{\epsilon}{2}\mathbb{I}\right)^{-1} \boldsymbol{x_i}\,(u-v)}$$

$$\overline{\xi_i} \leftarrow \boldsymbol{x_i}^\top \boldsymbol{\beta} + \sqrt{\boldsymbol{x_i}^\top \left(\boldsymbol{X}^{(\mathcal{N}_j)^\top} \boldsymbol{X}^{(\mathcal{N}_j)} + \frac{\epsilon}{2}\mathbb{I}\right)^{-1} \boldsymbol{x_i}\,(u-v)}$$

$\zeta_i{}^L \leftarrow 0$, $\zeta_i^U \leftarrow \max\{(y_i - \underline{\xi})^2, (\overline{\xi} - y_i)^2\}$

if $\underline{\xi_i} > y_i$ **then**

$\zeta_i{}^L \leftarrow \max\{\zeta_i{}^L, (\underline{\xi} - y_i)^2\}$

if $\overline{\xi} < y_i$ **then**

$\zeta_i{}^L \leftarrow \max\{\zeta_i{}^L, (y_i - \overline{\xi})^2\}$

return $\left(\max(\bar{v} - u, \sum_{i\in\mathcal{N}_j} \zeta_i^L), \sum_{i\in\mathcal{N}_j} \zeta_i^U\right)$

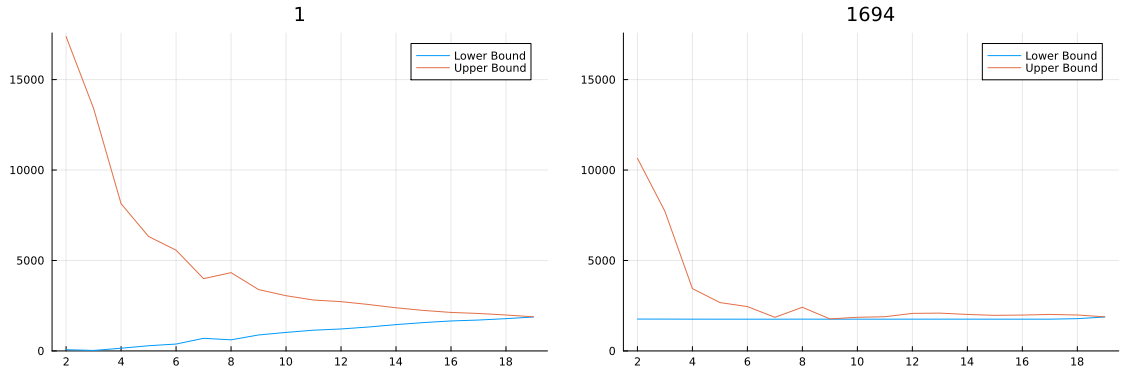

Figure 1 **Comparison of initial bounds on LOOCV (k-fold with $k=n$) from Algorithm 2 (left) and bounds after running Algorithm 1 (right) for a synthetic sparse regression instance where $p=20, n=200, \tau_{\text{true}}=10$, for varying τ; see Section EC.1.2 for a full description of our synthetic data generation process. The black number in the top middle depicts the iteration number of the method.**

Accordingly, we now use the fact that γ and $\boldsymbol{z}^{(\mathcal{N}_j)}$ fully determine $\boldsymbol{\beta}^{(\mathcal{N}_j)}$ to rewrite

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \quad \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 \quad \text{s.t.} \quad \beta_i = 0 \text{ if } \hat{z}_i = 0,$$

$$\text{as} \qquad \boldsymbol{\beta}^\star = \left(\frac{\gamma}{2}\mathbb{I} + \boldsymbol{X_{\hat{z}}}^\top \boldsymbol{X_{\hat{z}}}\right)^{-1} \boldsymbol{X_{\hat{z}}}^\top \boldsymbol{y},$$

where $\boldsymbol{X_{\hat{z}}}$ denotes a matrix with the columns of $\boldsymbol{x}$ such that $z_i = 1$.

Therefore, we fix each $\boldsymbol{z}^{(\mathcal{N}_j)}$ and substitute the resulting expressions for each $\boldsymbol{\beta}^{(\mathcal{N}_j)}$ into the k-fold error. This substitution yields the following univariate optimization problem, which can be

solved via standard root-finding methods to approximately minimize the k-fold loss via a local approximation:

$$\min_{\gamma>0} \sum_{j\in[k]} \sum_{i\in\mathcal{N}_j} \left(y_i - \boldsymbol{x}_i^\top \text{Diag}(\boldsymbol{z}^{(\mathcal{N}_j)}) \left(\frac{\gamma}{2}\mathbb{I} + \boldsymbol{X}^{(\mathcal{N}_j)^\top} \text{Diag}(\boldsymbol{z}^{(\mathcal{N}_j)}) \boldsymbol{X}^{(\mathcal{N}_j)^\top} \right)^{-1} \text{Diag}(\boldsymbol{z}^{(\mathcal{N}_j)}) \boldsymbol{X}^{(\mathcal{N}_j)^\top} \boldsymbol{y}^{(\mathcal{N}_j)} \right)^2. \tag{24}$$

Details on minimizing γ using `Julia` are provided in EC.1.1.

4. Numerical Experiments

We now present numerical experiments testing our proposed methods. First, in Section 4.1, we isolate the algorithmic impact of Algorithm 1 when optimizing the k-fold cross-validation error with respect to the sparsity parameter τ. Then, in Section 4.2, we evaluate the statistical performance of the entire alternating minimization pipeline proposed in Section 3 (jointly minimizing γ and τ) against widely used sparse regression software packages.

4.1. Exact k-fold Optimization

We first assess whether Algorithm 1 significantly reduces the number of MIOs that need to be solved to minimize the k-fold CV error with respect to τ, compared to grid search. We set either $k = n$ or $k = 10$, corresponding to leave-one-out and 10-fold cross-validation problems (23) respectively.

We compare the performance of two approaches. First, a standard grid search approach (`Grid`), where we solve the inner MIO in (23) for all combinations of cardinality $\tau \in [p]$ and all folds of the data $j \in [k]$, and select the hyperparameter combination which minimizes the objective. To ensure the quality of the resulting solution, we solve all MIOs to optimality (without any time limit). Second, we consider using Algorithm 1 with parameter $r = \infty$ (thus solving MIOs to optimality until the desired optimality gap ϵ for problem (23) is proven). We test regularization parameter $\gamma \in \{0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00\}$ in Algorithm 1, and solve all MIOs via their perspective reformulations, namely

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{z}\in\{0,1\}^p} \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \frac{\gamma}{2}\sum_{j=1}^p \frac{\beta_j^2}{z_j} \quad \text{s.t.} \quad \sum_{j=1}^p z_j \leq \tau,$$

using Mosek 10.0. Since the approach `Grid` involves solving $\mathcal{O}(kp)$ MIOs (without a time limit), we are limited to testing these approaches on small datasets, and accordingly use the Diabetes, Housing, Servo, and AutoMPG datasets for this experiment, as described by Gómez and Prokopyev (2021). Moreover, we remark that the specific solution times and the number of nodes expanded by each method are not crucial, as those could vary substantially if relaxations other than the perspective are used, different solvers or solution approaches are used, or if advanced techniques are implemented (but both methods would be affected in the same way). Thus, we focus our analysis on relative performance.

Figures 2 and 3 summarize the percentage reduction of the number of MIOs and the number of branch-and-bound nodes achieved by Algorithm 1 over `Grid`, computed as

$$\text{Reduction in MIOs} = \frac{\#\ \text{MIO}_{\texttt{Grid}} - \#\ \text{MIO}_{\text{Alg.1}}}{\#\ \text{MIO}_{\texttt{Grid}}},\ \text{Reduction in nodes} = \frac{\#\ \text{nodes}_{\texttt{Grid}} - \#\ \text{nodes}_{\text{Alg.1}}}{\#\ \text{nodes}_{\texttt{Grid}}},$$

where $\#\ \text{MIO}_{\text{Alg.1}}$ and $\#\ \text{nodes}_{\text{Alg.1}}$ indicate the number of MIOs or branch-and-bound nodes used by Algorithm 1. Tables 1 and 2 present the detailed computational results.

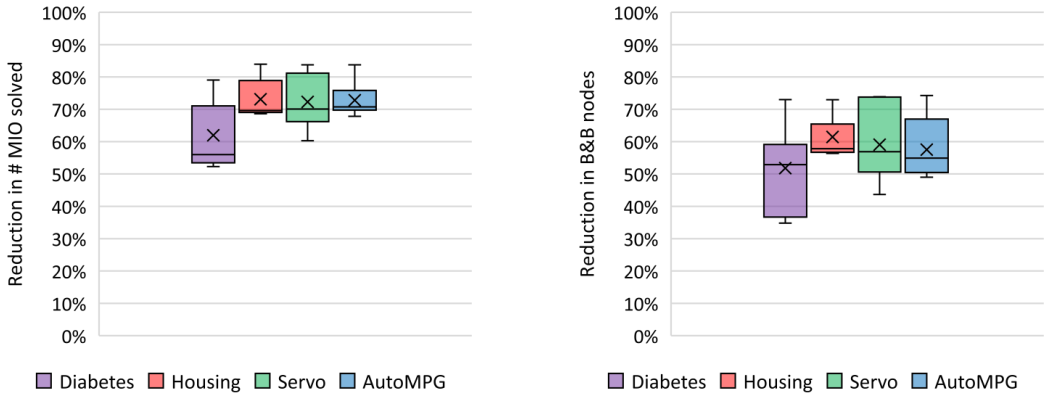

Figure 2 **Reduction in the number of MIOs solved (left) and the total number of branch-and-bound nodes (right) when using Algorithm 1 for leave-one-out cross-validation, when compared with `Grid` (i.e., independently solving $\mathcal{O}(pn)$ MIOs) in four real datasets. The distributions shown in the figure correspond to solving the same instance with different values of γ. All MIOs are solved to optimality, without imposing any time limits.**

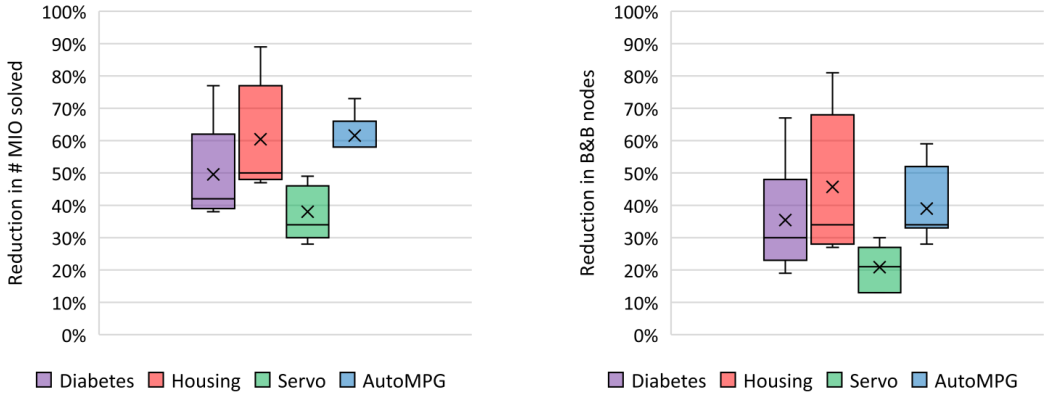

Figure 3 **Distribution of the reduction in the number of MIOs solved (left) and the total number of branch-and-bound nodes (right) when using Algorithm 1 for 10-fold cross-validation, when compared with `Grid` (i.e., independently solving $\mathcal{O}(pk)$ MIOs) in four real datasets (across different regularization parameters). The distributions shown in the figure correspond to solving the same instance with different values of γ. All MIOs are solved to optimality, without imposing any time limits.**

Table 1 Comparison between using Algorithm 1 and solving $\mathcal{O}(pn)$ MIOs independently (`Grid`) for leave-one-out cross-validation in four real datasets, for different values of regularization γ. Times reported are in minutes and correspond to the time to solve all required mixed-integer optimization problems to optimality. No time limits are imposed on the MIOs. Algorithm 1 consistently reduces the number of calls to the MIO solver by 50–80%.

Dataset	p	n	γ	Grid			Algorithm 1			Improvement		
				Time	# MIO	Nodes	Time	# MIO	Nodes	**Time**	**# MIO**	**Nodes**
			0.01	68	3,978	126,085	37	1,714	59,406	**45%**	**56%**	**53%**
			0.02	52	3,978	82,523	37	1,768	52,264	**30%**	**56%**	**37%**
			0.05	42	3,978	42,411	29	1,898	27,652	**29%**	**52%**	**35%**
Diabetes	11	442	0.10	39	3,978	31,116	26	1,852	16,202	**34%**	**53%**	**48%**
			0.20	35	3,978	22,165	20	1,332	9,278	**42%**	**67%**	**58%**
			0.50	32	3,978	11,889	16	1,152	4,852	**50%**	**71%**	**59%**
			1.00	34	3,978	9,278	14	833	2,501	**58%**	**79%**	**73%**
			0.01	247	6,072	512,723	102	1,906	217,918	**59%**	**69%**	**57%**
			0.02	187	6,072	324,238	65	1,843	141,493	**65%**	**70%**	**56%**
			0.05	166	6,072	216,116	92	1,879	93,543	**45%**	**69%**	**57%**
Housing	13	506	0.10	40	6,072	96,387	19	1,880	40,664	**51%**	**69%**	**58%**
			0.20	82	6,072	68,581	36	1,661	25,171	**55%**	**73%**	**63%**
			0.50	90	6,072	60,067	34	1,281	20,761	**62%**	**79%**	**65%**
			1.00	107	6,072	49,770	24	976	13,460	**77%**	**84%**	**73%**
			0.01	466	3,006	1,669,537	276	1,194	940,831	**41%**	**60%**	**44%**
			0.02	110	3,006	811,432	53	1,016	400,817	**52%**	**66%**	**51%**
			0.05	44	3,006	324,877	25	986	160,369	**77%**	**84%**	**73%**
Servo	19	167	0.10	23	3,006	162,223	9	686	58,326	**59%**	**77%**	**64%**
			0.20	15	3,006	76,739	8	900	33,098	**48%**	**70%**	**57%**
			0.50	10	3,006	40,197	4	566	10,496	**56%**	**81%**	**74%**
			1.00	8	3,006	25,683	4	488	6,738	**52%**	**84%**	**74%**
			0.01	1,100	9,408	6,772,986	590	3,131	3,532,057	**46%**	**67%**	**48%**
			0.02	1,356	9,408	3,900,417	450	2,846	1,888,766	**67%**	**70%**	**52%**
			0.05	519	9,408	2,286,681	227	2,808	1,133,175	**56%**	**70%**	**50%**
AutoMPG	25	392	0.10	355	9,408	1,548,369	145	2,751	687,187	**59%**	**71%**	**56%**
			0.20	143	9,408	629,020	65	2,686	283,755	**54%**	**71%**	**55%**
			0.50	66	9,408	176,950	28	2,272	58,464	**58%**	**76%**	**67%**
			1.00	68	9,408	116,982	38	1,528	30,120	**43%**	**84%**	**74%**

We observe that across these four datasets, Algorithm 1 reduces the number of MIOs that need to be solved by an average of 70% for leave-one-out cross-validation and by 52% for 10-fold cross-validation. The overall number of branch-and-bound nodes is reduced by an average of 57% for leave-one-out cross-validation and 35% for 10-fold cross-validation (the reduction in computational times is similar to the reduction of nodes). Note that MIOs with strong continuous relaxations are less likely to be solved to optimality by Algorithm 1. In general, these MIOs are also easier to solve to optimality, thus resulting in smaller improvements in the number of nodes and solution times than those suggested by the number of MIOs solved. Nonetheless, this discrepancy is relatively small, indicating that the proposed approach is still effective in avoiding solving several non-trivial MIOs.

Table 2 Comparison between using Algorithm 1 and solving $\mathcal{O}(pk)$ MIOs independently (`Grid`) for 10-fold cross validation in four real datasets, for different values of regularization γ. Times reported are in minutes, and correspond to the time to solve all required mixed-integer optimization problems to optimality. No time limits are imposed on the MIOs.

Dataset	p	n	γ	`Grid`			Algorithm 1			**Improvement**		
				Time	# MIO	Nodes	Time	# MIO	Nodes	**Time**	**# MIO**	**Nodes**
			0.01	3	396	11,666	2	242	8,224	**14%**	**39%**	**30%**
			0.02	2	396	8,371	2	235	6,785	**12%**	**41%**	**19%**
			0.05	2	396	4,436	2	228	3,430	**10%**	**42%**	**23%**
Diabetes	11	442	0.10	2	396	3,185	2	247	2,277	**10%**	**38%**	**29%**
			0.20	1	396	2,268	1	206	1,536	**8%**	**48%**	**32%**
			0.50	1	396	1,233	1	149	643	**26%**	**62%**	**48%**
			1.00	1	396	872	1	93	287	**42%**	**77%**	**67%**
			0.01	25	600	48,069	19	321	35,227	**25%**	**47%**	**27%**
			0.02	19	600	34,915	14	310	25,090	**28%**	**48%**	**28%**
			0.05	14	600	21,350	10	303	14,933	**29%**	**50%**	**30%**
Housing	13	506	0.10	10	600	11,012	7	300	7,308	**31%**	**50%**	**34%**
			0.20	9	600	7,406	5	230	3,524	**46%**	**62%**	**52%**
			0.50	9	600	6,168	3	141	1,977	**62%**	**77%**	**68%**
			1.00	8	600	4,993	2	66	930	**77%**	**89%**	**81%**
			0.01	15	288	148,168	12	191	128,592	**16%**	**34%**	**13%**
			0.02	8	288	77,457	7	190	67,416	**10%**	**34%**	**13%**
			0.05	3	288	29,056	3	157	23,653	**16%**	**45%**	**19%**
Servo	19	167	0.10	2	288	15,951	2	146	12,562	**16%**	**49%**	**21%**
			0.20	1	288	8,117	1	155	6,275	**12%**	**46%**	**23%**
			0.50	1	288	4,028	1	201	2,922	**3%**	**30%**	**27%**
			1.00	1	288	2,541	1	206	1,768	**1%**	**28%**	**30%**
			0.01	111	936	691,816	76	389	460,187	**31%**	**58%**	**33%**
			0.02	68	936	401,905	44	374	264,179	**35%**	**60%**	**34%**
			0.05	42	936	225,318	30	396	161,639	**28%**	**58%**	**28%**
AutoMPG	25	392	0.10	30	936	149,243	20	389	98,261	**35%**	**58%**	**34%**
			0.20	14	936	61,534	10	389	41,323	**32%**	**58%**	**33%**
			0.50	7	936	17,865	4	318	8,550	**43%**	**66%**	**52%**
			1.00	6	936	10,848	3	251	4,480	**48%**	**73%**	**59%**

We observe that solution times for both methods decrease on a given dataset as γ increases (as expected, since the perspective reformulation is stronger). Interestingly, while the improvements of Algorithm 1 over `Grid` (in terms of time, MIOs solved, and nodes) are more pronounced in regimes with large regularization γ, this effect on γ is slight: Algorithm 1 consistently results in improvements over 40% (and often more) even for the smallest values of γ tested. These results indicate that the relaxations of the bilevel optimization (23) derived in §2 are sufficiently strong to avoid solving most of the MIOs that traditional methods such as `Grid` would solve, without sacrificing solution quality. The proposed methods are especially beneficial for settings where k is large, that is, in the settings that would require more MIOs and are more computationally expensive using standard approaches.

The resulting approach still requires solving several MIOs, but, as we show throughout the rest of this section, approximating each MIO with its perspective relaxation yields similarly high-quality statistical estimators at a fraction of the computational cost.

4.2. Statistical Results With Real Data

In this section, we investigate the performance of our alternating minimization procedure as described in Section 3 (called MIO throughout the section) on a suite of eleven real datasets from the UCI repository. Specifically, we use the following experimental setup for the MIO method:

- We use a grid of ten different values of γ log-uniformly distributed on $[10^{-4}, 10^4]$, with $\gamma_0 = \frac{1}{\sqrt{n}}$.
- We set $\tau_{\min} = 2$ and $\tau_{\max}$ to be the largest integer such that $\tau_{\max} \log \tau_{\max} \leq \min(n, p)$ when optimizing τ in this experiment[2] as in Gamarnik and Zadik (2022).
- We impose a limit of at most 10 iterations of alternating minimization on minimizing the k-fold cross-validation error with respect to γ and with respect to τ (terminating early if $(\gamma_t, \tau_t) = (\gamma_{t-1}, \tau_{t-1})$).
- For tractability, for the overdetermined (underdetermined) instances, we impose a time limit of 120s (600s) for each MIO solved when cross-validating τ, and a time limit of 3600s (7200s) for fitting the final MIO once all hyperparameters are fixed. When we exceed a time limit, we use the best regression model found by the MIO solver at that time.

We refer to this implementation of our alternating minimization approach as "MIO" (short for mixed-integer optimization).

We compare against the following state-of-the-art methods, using built-in functions to approximately minimize the cross-validation loss with respect to the method's hyperparameters via grid search, and subsequently fit a regression model on the entire dataset with these cross-validated parameters (see also Bertsimas et al. (2020) for a detailed discussion of these approaches):

- The `ElasticNet` method in the ubiquitous `glmnet` package, with grid search on their parameter $\alpha \in \{0, 0.1, 0.2, \ldots, 1\}$
- The Minimax Concave Penalty (MCP) as implemented in the `R` package `ncvreg`, using the `cv.ncvreg` function with default parameters to minimize the five-fold cross-validation error.
- The `L0Learn.cvfit` method implemented in the `L0Learn` `R` package (cf. Hazimeh and Mazumder 2020), with five folds, a grid of 10 different values of γ and default parameters otherwise.

Note that we use default parameters for glmnet, MCP, and L0Learn to reflect typical practitioner usage. In particular, we do not impose an explicit cardinality budget for ElasticNet or MCP because neither method provides a parameter option to do so. Similarly, we do not explicitly impose a cardinality constraint in `L0Learn`, because `L0Learn` penalizes rather than constrains cardinality.

We compare performance in terms of the Mean Squared Error, namely

$$MSE(\boldsymbol{\beta}) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2,$$

which can either be taken over the validation set (CV)–that is, the objective (2) we attempt to minimize–or over an unseen test set (MSE), acting as a proxy for generalization error.

To measure the validation and test set errors, we repeat the following procedure five times and report the average result: we randomly shuffle the data into 80% training/validation and 20% test data, perform five-fold cross-validation on the 80% training/validation data, fit a model with the cross-validated (τ, γ) on the combined 80% train/validation data, and evaluate the model's test-set performance on the remaining 20% test data. We also report the average value of τ, the cross-validated sparsity, for each method. Note that the use of a "2" after the dataset name indicates that a dataset includes second-order interactions, in order to increase the computational difficulty of processing the dataset.

We observe in Tables 3–4 that MIO performs comparably to widely used methods for most datasets, especially when they are relatively underdetermined. In particular, across the five most underdetermined datasets considered, it yields a five-fold cross-validation error 21.2% lower than MCP, 7.3% lower than glmnet, and 1.5% lower than L0Learn. However, it admittedly yields a higher average cross-validation error on the more overdetermined datasets (by 14.6%, 37.6% and 27.4% respectively). In terms of test-set errors, for the five most underdetermined datasets, MIO performs 6.9% better than MCP, 2.1% better than glmnet, and 4.2% better than L0Learn on average. However, it performs 1.9%, 18.2%, and 19.1% worse on the more overdetermined datasets in terms of test-set error, respectively. The worst performance of MIO on the most overdetermined datasets could be partly due to deriving regressors that are 37% sparser than MCP, 62% sparser than glmnet, and 47% sparser than L0Learn on the most overdetermined datasets. All in all, our results show that exact MIO-based cross-validation performs best on relatively underdetermined problems, whereas for highly overdetermined datasets, standard packages such as glmnet or L0Learn remain preferable from both statistical and computational perspectives.

Finally, we note that the average runtime across all datasets was 47510s for MIO (median: 69.90s, range: 0.42s–541511s), 0.180s for MCP, 1.154s for glmnet, and 0.841s for L0Learn, respectively. This shows that while our scheme is significantly slower than state-of-the-art regression packages with sophisticated codebases, our cross-validation scheme does return reasonable results on real-world datasets. Thus, the ideas in this paper could potentially be integrated within these software packages. For instance, one could apply the ideas in this paper to enhance the L0BnB software package Hazimeh et al. (2022) by incorporating a cross-validation scheme.

Table 3 Average (± one standard deviation) performance of five-fold versions of the methods across a suite of real-world datasets where the ground truth is unknown (and may not be sparse), sorted by how overdetermined the dataset is (n/p), and separated into the underdetermined and overdetermined cases, where $\tau := \|\boldsymbol{\beta}\|_0$ denotes the sparsity of the regression model β. In underdetermined settings, MIO often yields competitive or lower out-of-sample MSEs than MCP, glmnet, and L0Learn, whereas in more overdetermined settings, glmnet or L0Learn tend to yield lower out-of-sample MSEs.

Dataset	n	p	MIO			MCP		
			τ	CV	MSE	τ	CV	MSE
Wine	6497	11	6 ± 0	0.567 ± 0.003	0.545 ± 0.021	10.8 ± 0.447	0.543 ± 0.005	0.543 ± 0.020
AutoMPG	392	25	11 ± 0	10.192 ± 0.722	9.595 ± 2.901	16.6 ± 1.817	9.079 ± 0.486	9.043 ± 1.970
Hitters	263	19	8.6 ± 0.894	0.077 ± 0.006	0.079 ± 0.02	12.8 ± 4.604	0.08 ± 0.008	0.081 ± 0.024
Prostate	97	8	3.8 ± 1.304	0.53 ± 0.035	0.609 ± 0.159	5.8 ± 2.168	0.572 ± 0.051	0.574 ± 0.153
Servo	167	19	7.6 ± 2.608	0.785 ± 0.106	0.769 ± 0.173	14.000 ± 1.000	0.752 ± 0.062	0.723 ± 0.212
Housing2	506	91	22.6 ± 3.05	19.182 ± 1.57	16.776 ± 7.548	34.2 ± 5.718	16.311 ± 2.325	16.879 ± 3.449
Toxicity	38	9	3.6 ± 1.342	0.036 ± 0.009	0.05 ± 0.037	2.6 ± 0.894	0.049 ± 0.01	0.057 ± 0.049
Steam	25	8	3.4 ± 1.14	0.463 ± 0.084	0.490 ± 0.300	2.6 ± 0.894	0.629 ± 0.103	0.532 ± 0.200
Alcohol2	44	21	3.6 ± 1.517	0.225 ± 0.032	0.256 ± 0.094	1.8 ± 0.447	0.238 ± 0.035	0.258 ± 0.047
TopGear	242	373	18.2 ± 10.134	0.046 ± 0.006	0.049 ± 0.005	7.4 ± 1.517	0.061 ± 0.006	0.061 ± 0.018
Bardet	120	200	18.8 ± 5.891	0.007 ± 0	0.009 ± 0.003	5.6 ± 1.14	0.009 ± 0.002	0.009 ± 0.002

Table 4 Average (± one standard deviation) performance of five-fold versions of the methods across a suite of real-world datasets where the ground truth is unknown (and may not be sparse), sorted by how overdetermined the dataset is (n/p), and separated into the underdetermined and overdetermined cases (cont).

Dataset	n	p	glmnet			L0Learn		
			τ	CV	MSE	τ	CV	MSE
Wine	6497	11	11 ± 0	0.542 ± 0.005	0.543 ± 0.02	10.6 ± 0.548	0.542 ± 0.005	0.543 ± 0.02
AutoMPG	392	25	22.6 ± 1.949	8.627 ± 0.493	9.201 ± 2.436	15.8 ± 4.266	9.099 ± 0.616	9.061 ± 2.351
Hitters	263	19	14.4 ± 4.561	0.077 ± 0.006	0.082 ± 0.023	10.6 ± 5.55	0.075 ± 0.006	0.08 ± 0.021
Prostate	97	8	6.8 ± 1.095	0.507 ± 0.051	0.581 ± 0.154	5.6 ± 2.408	0.508 ± 0.048	0.569 ± 0.183
Servo	167	19	16 ± 1.414	0.693 ± 0.051	0.726 ± 0.211	10.2 ± 3.493	0.696 ± 0.079	0.746 ± 0.221
Housing2	506	91	86 ± 2.915	12.317 ± 0.282	12.867 ± 2.133	58.6 ± 9.099	13.669 ± 0.997	12.818 ± 2.009
Toxicity	38	9	6 ± 0.707	0.041 ± 0.009	0.047 ± 0.029	3 ± 1.871	0.037 ± 0.012	0.062 ± 0.049
Steam	25	8	4.4 ± 0.894	0.492 ± 0.14	0.507 ± 0.200	2.2 ± 0.447	0.475 ± 0.179	0.499 ± 0.103
Alcohol2	44	21	8.8 ± 4.087	0.253 ± 0.05	0.263 ± 0.058	7.6 ± 6.269	0.220 ± 0.021	0.272 ± 0.072
TopGear	242	373	39.4 ± 21.686	0.045 ± 0.004	0.047 ± 0.014	28.8 ± 37.626	0.050 ± 0.010	0.049 ± 0.015
Bardet	120	200	29.8 ± 6.907	0.007 ± 0.001	0.008 ± 0.003	30.4 ± 13.446	0.007 ± 0.000	0.009 ± 0.004

5. Conclusion

In this paper, we propose a new optimization-based approach for selecting hyperparameters via cross-validation in ridge-regularized sparse regression problems, by leveraging perspective relaxations and bounds on the cross-validation error. The proposed approach substantially decreases the number of MIOs and branch-and-bound nodes explored to optimize the cross-validation loss. Overall, these results suggest that perspective relaxations can help to make exact MIO-driven cross-validation practically viable, and could be incorporated into existing sparse regression software when training models already require solving MIOs. As future work, it could be interesting

to explore how strong convex relaxations can help accelerate MIO-driven cross-validation in other contexts, e.g., when designing optimal decision trees or neural networks.

Acknowledgments: Andrés Gómez is supported in part by grant FA9550-24-1-0086 from the Air Force Office of Scientific Research. Ryan Cory-Wright gratefully acknowledges the MIT-IBM Research Lab for hosting him while part of this work was conducted. We are grateful to Jean Pauphilet for valuable discussions on an earlier draft of this manuscript.

Endnotes

1. This assumption seems plausible, as the training objective is strongly convex for a fixed binary support vector, and therefore for each binary support vector there is indeed a unique solution. One could relax this assumption by defining $h(\gamma, \tau)$ to be the minimum cross-validation error over all training-optimal solutions $\boldsymbol{\beta}^{(i)}$, as is commonly done in the bilevel optimization literature, giving what is called an optimistic formulation of a bilevel problem (see Beck and Schmidt 2021, for a review). However, this would make the cross-validation error less tractable.
2. We previously tried setting $\tau_{\max} = p$. We found that this yielded the same optimal hyperparameters, but increased the total runtime of the method substantially.

References

Arlot S, Celisse A (2010) A survey of cross-validation procedures for model selection. *Statistics Surveys* 4:40–79.

Askari A, d'Aspremont A, Ghaoui LE (2022) Approximation bounds for sparse programs. *SIAM Journal on Mathematics of Data Science* 4(2):514–530.

Atamtürk A, Gómez A (2020) Safe screening rules for l0-regression from perspective relaxations. *ICML*, 421–430.

Atamtürk A, Gómez A (2025) Rank-one convexification for sparse regression. *Journal of Machine Learning Research* 26(35):1–50.

Beck Y, Schmidt M (2021) A gentle and incomplete introduction to bilevel optimization. *Optimization Online* URL `https://optimization-online.org/2021/06/8450/`.

Ben-Ayed O, Blair CE (1990) Computational difficulties of bilevel linear programming. *Operations Research* 38(3):556–560.

Bennett KP, Hu J, Ji X, Kunapuli G, Pang JS (2006) Model selection via bilevel optimization. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 1922–1929 (IEEE).

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13(2).

Bertsimas D, Copenhaver MS (2018) Characterization of the equivalence of robustification and regularization in linear and matrix regression. *European Journal of Operational Research* 270(3):931–942.

Bertsimas D, Cory-Wright R (2022) A scalable algorithm for sparse portfolio selection. *INFORMS Journal on Computing* 34(3):1489–1511.

Bertsimas D, Cory-Wright R, Pauphilet J (2021) A unified approach to mixed-integer optimization problems with logical constraints. *SIAM Journal on Optimization* 31(3):2340–2367.

Bertsimas D, Pauphilet J, Van Parys B (2020) Sparse regression: Scalable algorithms and empirical performance. *Statistical Science* 35(4):555–578.

Bertsimas D, Van Parys B (2020) Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics* 48(1):300–323.

Boland N, Charkhgard H, Savelsbergh M (2015a) A criterion space search algorithm for biobjective integer programming: The balanced box method. *INFORMS Journal on Computing* 27(4):735–754.

Boland N, Charkhgard H, Savelsbergh M (2015b) A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. *INFORMS Journal on Computing* 27(4):597–618.

Boyd S, El Ghaoui L, Feron E, Balakrishnan V (1994) *Linear matrix inequalities in system and control theory* (SIAM).

Bühlmann P, Van De Geer S (2011) *Statistics for high-dimensional data: methods, theory and applications* (Springer Science & Business Media).

Burman P (1989) A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika* 76(3):503–514.

Ceria S, Soares J (1999) Convex programming for disjunctive convex optimization. *Mathematical Programming* 86:595–614.

Cifuentes D, Moitra A (2022) Polynomial time guarantees for the burer-monteiro method. *Advances in Neural Information Processing Systems* 35:23923–23935.

Falkner S, Klein A, Hutter F (2018) Bohb: Robust and efficient hyperparameter optimization at scale. *International Conference on Machine Learning*, 1437–1446 (PMLR).

Frazier PI (2018) A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811* .

Gamarnik D, Zadik I (2022) Sparse high-dimensional linear regression. estimating squared error and a phase transition. *The Annals of Statistics* 50(2):880–903.

Geoffrion AM (1972) Generalized Benders decomposition. *Journal of Optimization Theory and Applications* 10(4):237–260.

Gómez A, Prokopyev OA (2021) A mixed-integer fractional optimization approach to best subset selection. *INFORMS Journal on Computing* 33(2):551–565.

Günlük O, Linderoth J (2010) Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical Programming* 124(1):183–205.

Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing* 13(5):1194–1217.

Hastie T, Tibshirani R, Friedman JH, Friedman JH (2009) *The elements of statistical learning: data mining, inference, and prediction*, volume 2 (Springer).

Hastie T, Tibshirani R, Tibshirani R (2020) Best subset, forward stepwise or Lasso? analysis and recommendations based on extensive comparisons. *Statistical Science* 35(4):579–592.

Hazan E, Koren T (2016) A linear-time algorithm for trust region problems. *Mathematical Programming* 158(1-2):363–381.

Hazimeh H, Mazumder R (2020) Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Operations Research* 68(5):1517–1537.

Hazimeh H, Mazumder R, Saab A (2022) Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Mathematical Programming* 196(1):347–388.

Hotelling H (1931) The generalization of student's ratio. *The Annals of Mathematical Statistics* 2(3):360–378.

Kenney A, Chiaromonte F, Felici G (2021) MIP-boost: Efficient and effective l 0 feature selection for linear regression. *Journal of Computational and Graphical Statistics* 30(3):566–577.

Larochelle H, Erhan D, Courville A, Bergstra J, Bengio Y (2007) An empirical evaluation of deep architectures on problems with many factors of variation. *ICML*, 473–480.

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444.

Liu J, Rosen S, Zhong C, Rudin C (2023) Okridge: Scalable optimal k-sparse ridge regression. *Advances in Neural Information Processing Systems* 36:41076–41258.

Lokman B, Köksalan M (2013) Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization* 57:347–365.

Mazumder R, Friedman JH, Hastie T (2011) Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association* 106(495):1125–1138.

Mazumder R, Radchenko P, Dedieu A (2023) Subset selection with shrinkage: Sparse linear modeling when the snr is low. *Operations Research* 71(1):129–147.

McAfee A, Brynjolfsson E, Davenport TH, Patil D, Barton D (2012) Big data: The management revolution. *Harvard Business Review* 90(10):60–68.

Natarajan BK (1995) Sparse approximate solutions to linear systems. *SIAM Journal on Computing* 24(2):227–234.

Okuno T, Takeda A, Kawana A, Watanabe M (2021) On lp-hyperparameter learning via bilevel nonsmooth optimization. *The Journal of Machine Learning Research* 22:245–1.

Pilanci M, Wainwright MJ, El Ghaoui L (2015) Sparse learning via boolean relaxations. *Mathematical Programming* 151(1):63–87.

Reeves G, Xu J, Zadik I (2019) The all-or-nothing phenomenon in sparse linear regression. *Conference on Learning Theory*, 2652–2663 (PMLR).

Sinha A, Khandait T, Mohanty R (2020) A gradient-based bilevel optimization approach for tuning hyperparameters in machine learning. *arXiv preprint arXiv:2007.11022* .

Stephenson W, Frangella Z, Udell M, Broderick T (2021) Can we globally optimize cross-validation loss? quasiconvexity in ridge regression. *Advances in Neural Information Processing Systems* 34.

Stidsen T, Andersen KA, Dammann B (2014) A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science* 60(4):1009–1032.

Stone M (1978) Cross-validation: A review. *Statistics: A Journal of Theoretical and Applied Statistics* 9(1):127–139.

Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming* 86:515–532.

Takano Y, Miyashiro R (2020) Best subset selection via cross-validation criterion. *Top* 28(2):475–488.

Xie W, Deng X (2020) Scalable algorithms for the sparse ridge regression. *SIAM Journal on Optimization* 30(4):3359–3386.

Xu H, Caramanis C, Mannor S (2008) Robust regression and lasso. *Advances in Neural Information Processing Systems* 21.

Ye JJ, Yuan X, Zeng S, Zhang J (2022) Difference of convex algorithms for bilevel programs with applications in hyperparameter selection. *Mathematical Programming* 1–34.

Supplementary Material

EC.1. Implementation Details

To solve each MIO in Algorithm 1, we invoke a Generalized Benders Decomposition scheme (Geoffrion 1972), which was specialized to sparse regression problems by Bertsimas and Van Parys (2020). For any fixed γ, τ, the method proceeds by minimizing a piecewise linear approximation of

$$f(\boldsymbol{z}, \gamma) := \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \ \frac{\gamma}{2} \sum_{j \in [p]} \frac{\beta_j^2}{z_j} + \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2, \tag{EC.1}$$

with respect to the support $\boldsymbol{z}$, until it converges to an optimal solution or encounters a time limit.

We now discuss two enhancements that improve this method's performance in practice.

Warm-Starts: as noted by Bertsimas et al. (2021), a greedily rounded solution to the Boolean relaxation constitutes an excellent warm-start for a Generalized Benders Decomposition scheme. Therefore, when computing the lower and upper bounds on $h_j(\gamma, \tau)$ for each τ by solving a perspective relaxation, we save the greedily rounded solution to the relaxation in memory, and provide the relevant rounding as a high-quality warm-start before solving the corresponding MIO.

Screening Rules: as observed by Atamtürk and Gómez (2020), if we have an upper bound on the optimal value of $f(\boldsymbol{z}, \gamma)$, say $\bar{f}$, an optimal solution to the Boolean relaxation of minimizing (EC.1) over $\boldsymbol{z} \in [0,1]^p$, say $(\boldsymbol{\beta}, \boldsymbol{z})$, and a lower bound on the optimal value of $h(\boldsymbol{z}, \gamma)$ from the Boolean relaxation, say $\underline{f}$, then, letting $\beta_{[\tau]}$ be the τth largest value of β in absolute magnitude, we have the following screening rules:

- If $\beta_i^2 \leq \beta_{[\tau+1]}^2$ and $\underline{f} - \frac{1}{2\gamma}(\beta_i^2 - \beta_{[\tau]}^2) > \bar{f}$ then $z_i = 0$.
- If $\beta_i^2 \geq \beta_{[\tau]}^2$ and $\underline{f} + \frac{1}{2\gamma}(\beta_i^2 - \beta_{[\tau+1]}^2) > \bar{f}$ then $z_i = 1$.

Accordingly, to reduce the dimensionality of our problems, we solve a perspective relaxation for each fold of the data with $\tau = \tau_{\max}$ as a preprocessing step, and screen out the features where $z_i = 0$ at $\tau = \tau_{\max}$ (for this fold of the data) before running Generalized Benders Decomposition.

As reported by Atamtürk and Gómez (2020), screening rules often reduce the number of decision variables in an MIO by 20%–97%, with the most significant benefits when γ is relatively large, and the duality gap between an MIO and its perspective relaxation is relatively small.

EC.1.1. Implementation Details for Section 3.2

In our numerical experiments, we find local minimizers of our approximation of h by invoking the `ForwardDiff` function in `Julia` to automatically differentiate our approximation of h, and subsequently identify local minima via the `Order0` method in the `Roots.jl` package, which is designed to be a robust root-finding method. To avoid convergence to a low-quality local minimum, we run the search algorithm initialized at the previous iterate γ_{t-1} and ten points log-uniformly

distributed in $[10^{-4}, 10^4]$, and set γ_t to be a local minimum with the smallest estimated error. Moreover, to ensure numerical robustness, we require that γ_t remains within the bounds $[10^{-4}, 10^4]$ and project γ_t onto this interval if it exceeds these bounds (this almost never occurs in practice, because the data is preprocessed to be standardized). This approach is highly efficient in practice, particularly when the optimal support does not vary significantly with γ.

EC.1.2. Synthetic Data Generation Process

For our synthetic experiments, we follow the experimental setup in Bertsimas et al. (2020). Given a fixed number of features p, number of data points n, true sparsity $1 \leq \tau_{\text{true}} \leq p$, autocorrelation parameter $0 \leq \rho \leq 1$ and signal to noise parameter ν:

1. The rows of the model matrix are generated iid from a p-dimensional multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, where $\Sigma_{ij} = \rho^{|i-j|}$ for all $i, j \in [p]$.
2. A "ground-truth" vector $\boldsymbol{\beta}_{\text{true}}$ is sampled with exactly τ_{true} non-zero coefficients. The position of the non-zero entries is randomly chosen from a uniform distribution, and the value of the non-zero entries is either 1 or -1 with equal probability.
3. The response vector is generated as $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}_{\text{true}} + \boldsymbol{\varepsilon}$, where each ε_i is generated iid from a scaled normal distribution such that $\sqrt{\nu} = \|\boldsymbol{X}\boldsymbol{\beta}_{\text{true}}\|_2 / \|\boldsymbol{\varepsilon}\|_2$.
4. We standardize $\boldsymbol{X}, \boldsymbol{y}$ to normalize and center them.

EC.2. Non-dominance of Relaxations (15) and (19)

In this section, we construct examples demonstrating that the lower bounds in (15) and (19) are non-dominated by each other.

Problem (15) can be stronger than Problem (19): Consider a setting with

$$n = 3,\ p = 1,\ X = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}, k = 3$$

and hyperparameters $\gamma = 1,\ \tau = 1$.

Then, since the sparsity constraint is not binding (for either the problem of training over the entire dataset or with one fold left out), our lower-level problems all reduce to ridge regression in one dimension, and our perspective relaxations are all tight. In particular, the full-data training problem admits an optimal solution $\beta^\star = 40/29$ with an optimal objective value of 2.414. On the other hand, on leaving out the third observation and training on the first two folds, we obtain $\beta^\star = 10/11$ with training objective 0.455, with a fold-3 contribution to the LOOCV error of 5.165. In particular, the lower bound from (15) is 625/121, which is larger than the lower bound from (19), namely 625/319.

Problem (15) **can be weaker than Problem** (19)**:** Consider a setting with

$$n=3,\ p=2,\ X=\begin{pmatrix}1 & 1\\ -2 & 0\\ 2 & 1\end{pmatrix},\quad y=\begin{pmatrix}0\\ 3\\ 3\end{pmatrix}, k=3$$

and hyperparameters $\gamma=1,\ \tau=1$, where we focus on leaving out the first fold. Then, the optimal full-data solution is $\beta^\star=(0,6/5)$ with objective value $72/5$. On the other hand, if we leave out the first observation then the optimal solution becomes $\beta^\star=(0,2)$ with objective value 12, and optimal value of the perspective relaxation of 10.714. Accordingly, the lower bound on the LOOCV error of fold 1 from (19) is 2.4, while the lower bound from (15) is $9/7<2.4$. Thus, (15) can be weaker than (19) even when using the optimal value of the MIO and perspective relaxation for upper and lower bounds in (15).