



# Robust regression via error tolerance

Anton Björklund<sup>1</sup> · Andreas Henelius<sup>1,2</sup> · Emilia Oikarinen<sup>1</sup> ·  
Kimmo Kallonen<sup>3</sup> · Kai Puolamäki<sup>1,4</sup>

Received: 16 February 2021 / Accepted: 4 January 2022 / Published online: 27 January 2022  
© The Author(s) 2022

## Abstract

Real-world datasets are often characterised by outliers; data items that do not follow the same structure as the rest of the data. These outliers might negatively influence modelling of the data. In data analysis it is, therefore, important to consider methods that are robust to outliers. In this paper we develop a robust regression method that finds the largest subset of data items that can be approximated using a sparse linear model to a given precision. We show that this can yield the best possible robustness to outliers. However, this problem is NP-hard and to solve it we present an efficient approximation algorithm, termed SLISE. Our method extends existing state-of-the-art robust regression methods, especially in terms of speed on high-dimensional datasets. We demonstrate our method by applying it to both synthetic and real-world regression problems.

**Keywords** Robust Regression · Robust Statistics · Outlier Detection · Sparsity

---

Responsible editor: Johannes Fürnkranz

---

✉ Anton Björklund  
anton.bjorklund@helsinki.fi

Andreas Henelius  
andreas.henelius@op.fi

Emilia Oikarinen  
emilia.oikarinen@helsinki.fi

Kimmo Kallonen  
kimmo.kallonen@helsinki.fi

Kai Puolamäki  
kai.puolamaki@helsinki.fi

<sup>1</sup> Department of Computer Science, University of Helsinki, Helsinki, Finland

<sup>2</sup> OP Financial Group, Helsinki, Finland

<sup>3</sup> Helsinki Institute of Physics, University of Helsinki, Helsinki, Finland

<sup>4</sup> Institute for Atmospheric and Earth System Research (INAR), University of Helsinki, Helsinki, Finland

## 1 Introduction

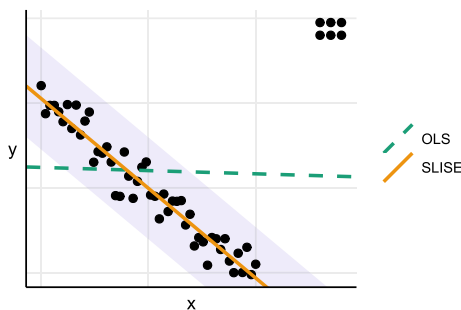
In practically all analyses of real-world data we encounter outliers, i.e., data items that do not follow the same patterns as the majority of the data. Such items are problematic, since they may negatively influence modelling of the data. This is observed, for instance, in ordinary least-squares (OLS) regression where already a single outlier may lead to arbitrarily large errors (Donoho and Huber 1983). It is, hence, important to consider robust methods that effectively avoid the influence of outliers.

Robust regression methods can be used as almost drop-in replacements for linear regression, which is still widely used because of the inherent interpretability and simplicity. Linear regression is also often used as a part of other machine learning or data mining algorithms, e.g., in explainable artificial intelligence (Ribeiro et al. 2016; Björklund et al. 2019). Furthermore, robust regression can be used to search for outliers by investigating the data items that do not adhere to the robust model.

In this paper we present a sparse robust regression method, termed SLISE (Sparse Linear Subset Explanations), that achieves the highest possible theoretical robustness and outperforms many existing state-of-the-art robust regression methods in terms of scalability on large datasets. Specifically, we consider *finding the largest subset of data items that can be represented by a linear model to a given accuracy*.

To illustrate the need for robust regression methods, consider the dataset shown in Fig. 1 containing a cluster of outliers in the top right corner. Here ordinary least-squares linear (OLS) regression clearly fails due to the influence of the outliers. In contrast, our SLISE method yields high robustness by ignoring the outliers and finds a linear model that approximates a (large) subset of the data items.

The two main goals of the SLISE algorithm can be stated in brief as follows. The primary goal of the algorithm is to find a maximum subset of points that can be modelled by a linear model to a given maximum error. This subset is visualised in Fig. 1 by the shaded area around the orange SLISE line. The width of this “corridor” shows the (adjustable) error tolerance. The secondary goal of SLISE is to minimise the residuals of the items within the subset marked by the corridor.



**Fig. 1** Outliers can cause ordinary least-squares regression to give unusable results (green dashed line). Robust regression mitigates the effect of the outliers. SLISE (orange solid line) accomplishes this by finding a maximal subset of items which can be modelled by a linear model to a given maximum error. This maximum error is illustrated as the lightly shaded “corridor” around the SLISE regression line (Color figure online)

In the analysis of multidimensional datasets it is also important to consider the interpretability of the model. SLISE incorporates LASSO regularisation and can, hence, yield sparse models, i.e., models where some of the coefficients are zero and have no impact on the model, making interpreting the results easier (Ribeiro et al. 2016).

## 1.1 Related work

One of the reasons for the non-robustness of OLS is the minimisation of the sum of squared residuals in the loss function, meaning that outliers highly affect the final fit. One remedy for this is to use a loss function without squaring, e.g., absolute deviation (Giloni et al. 2006) or Huber-loss (Huber 1964). Although this is, in practice, often more robust than OLS, theoretically these loss functions are just as susceptible to outliers as OLS (Rousseeuw and Hubert 2011).

Linear regression can also be solved by finding a maximum likelihood estimate for the parameters. The likelihood function can then be chosen such that the estimation becomes robust. Early examples include S-Estimators (Rousseeuw and Yohai 1984) and MM-Estimators (Yohai 1987). These have recently been further developed into, e.g., MM-LASSO (Smucler and Yohai 2017), MTE-LASSO (Qin et al. 2017), and SMDM (Koller and Stahel 2017).

Another approach to robust modelling is to fit the model only to non-outliers, instead of considering the full dataset. However, in order to ignore the outliers, these must first be found. One approach is to ignore a fixed fraction of the items, typically up to half, and simultaneously optimise for both the regression model and the subset of included data items. This is the idea behind the *Least Trimmed Squares* (LTS) algorithm (Rousseeuw 1984), which also has more recent improvements (Rousseeuw and Van Driessen 2000; Alfons et al. 2013).

This can also be done by randomly selecting subsets until a subset of only non-outliers is found. This subset can then be expanded to include all non-outliers, e.g., by fitting a linear model to the subset and selecting all data items with small-enough errors. This is what the RANSAC algorithm does (Fischler and Bolles 1981). RANSAC has gained popularity in the field of computer vision with multiple recent improvements (Barath and Matas 2018; Barath et al. 2020). However, these improvements operate on different assumptions and data structures, since the task is to match pixels between two images.

Another way of achieving robustness is to replace non-robust parts of the algorithms with more robust parts, e.g., by replacing means with medians (Hubert and Debruyne 2009). This is the idea behind *Quantile Regression* (Koenker and Hallock 2001), for which the most recent development is CONQUER (Fernandes et al. 2021).

Beside robustness, linear regression methods can have other characteristics, such as sparsity. In this context, sparsity means that some of the coefficients are (deliberately) zero. A common way of achieving sparsity is through LASSO regularisation (Tibshirani 1996), and this has also been added to various robust regression methods (Alfons et al. 2013; Smucler and Yohai 2017; Wang et al. 2007).

Of the robust regression methods, SLISE is most closely related to LTS, or rather to the sparse variant SPARSE-LTS (Alfons et al. 2013). Both LTS and SLISE fit linear

models for a subset of the items. The difference is that LTS requires the size of subset to be fixed and specified a priori while SLISE defines the subset based on a maximum tolerable error. This requires a new algorithm that actually scales better with regards to the number of dimension (see Sect. 4.2) without sacrificing any theoretical robustness (see Sect. 2.2). The SLISE algorithm uses graduated optimisation to find solutions, which makes it both fast and robust against noise and outliers.

## 1.2 Contributions

We present a novel robust regression method, SLISE, by considering the problem of *finding the largest subset that can be approximated by a sparse linear model to a given accuracy* (Problem 1). We show that this can yield the best possible breakdown value (Sect. 2.2), but that the problem is **NP-hard** (Theorem 2). We present an approximative algorithm for solving it (Algorithm 1) and demonstrate that it also works empirically using both synthetic and real-world datasets (Sects. 5–7). Compared to other robust regression methods SLISE yields high robustness and good scalability to high-dimensional datasets.

The initial version of the SLISE algorithm was presented in a conference paper (Björklund et al. 2019). That paper has a greater focus on one specific use case for the robust regression method, namely, explaining outcomes from black box supervised learning models. (The word “Explanations” in the name of the SLISE algorithm stems from this application.) When applied to the problem of explaining models, the idea is to find a simple interpretable linear model that approximates a more complex supervised learning model in the neighbourhood of a data item of interest. For this task SLISE must be able to find good solutions even for small error tolerances. The advantage of using SLISE for explaining supervised learning models is that no resampling of datasets is required and that the explanations (linear models) obey constraints imposed by the data. In the present paper, however, we focus solely on robust regression.

Compared to Björklund et al. (2019), the discussion on the problem characteristics and numeric approximation is substantially extended with additional proofs in Sects. 2 and 3. We also provide novel initialisation schemes for the SLISE algorithm in Sect. 4, and evaluate their effect, and the effect of all the other parameters, on the stability and performance of the algorithm in Sect. 6. We find that one of the new schemes is more robust than the LASSO initialisation used previously, and provide recommendations for suitable default values for the other parameters. Furthermore, we perform a more thorough empirical comparison to related methods in Sect. 7.

## 1.3 Organisation

In Sect. 2 we formalise our robust regression problem, and show its complexity and breakdown value. We then discuss the practical numeric optimisation and its approximation properties in Sect. 3. The algorithm, with different initialisation schemes and asymptotic complexity, is presented in Sect. 4. The empirical evaluation, which consists of the default parameter selection and the comparison to related methods, follows in Sects. 5–7. We end with conclusions in Sect. 8.

## 2 Problem definition

Let  $(X, Y)$ , where  $X \in \mathbb{R}^{n \times d}$  and  $Y \in \mathbb{R}^n$ , be a dataset consisting of  $n$  pairs  $\{(x_i, y_i)\}_{i=1}^n$  where we denote the  $i$ th  $d$ -dimensional item (row) in  $X$  by  $x_i \in \mathbb{R}^d$  (the *predictor*) and similarly the  $i$ th element in  $Y$  by  $y_i \in \mathbb{R}$  (the *response*). We use the shorthand  $[n] = \{1, \dots, n\}$ .

Our goal is to develop a linear regression method that is robust to outliers by *finding the largest subset of data items that can be described by a sparse linear model to a given precision*, as exemplified in the introduction.

We now state the main problem in this paper:

**Problem 1** Given  $X \in \mathbb{R}^{n \times d}$ ,  $Y \in \mathbb{R}^n$ , the error tolerance  $\varepsilon \in \mathbb{R} \geq 0$ , and the regularisation strength  $\lambda \in \mathbb{R} \geq 0$ ; find the regression coefficients  $\alpha \in \mathbb{R}^d$  minimising the  $(\varepsilon, \lambda)$ -loss function

$$\text{LOSS}(\varepsilon, \lambda, X, Y, \alpha) = \sum_{i=1}^n H(\varepsilon^2 - r_i^2) (r_i^2/n - \varepsilon^2) + \lambda \|\alpha\|_1, \tag{1}$$

where the residual errors are given by  $r_i = y_i - \alpha^T x_i$ ,  $H(\cdot)$  is the Heaviside step function satisfying  $H(u) = 1$  if  $u \geq 0$  and  $H(u) = 0$  otherwise, and  $\|\alpha\|_1$  denotes the L1-norm. In the remainder of the paper we will use the L1-norm given by  $\|\alpha\|_1 = \sum_{i=1}^d |\alpha_i|$ , even though the theoretical results would be valid for L2 or other norms as well. The benefits of L1 or Lasso regularization include that it leads to sparse solutions, which is beneficial for interpretability and explanations (Björklund et al. 2019). If necessary, the data matrix  $X$  can be augmented with a column of all ones to accommodate the *intercept* term of the model.

Alternatively, the Lagrangian term  $\lambda \|\alpha\|_1$  in Eq. (1) can be replaced by a constraint  $\|\alpha\|_1 \leq t$  for some  $t$ . Now we can rewrite Eq. (1) as

$$\text{LOSS}(\varepsilon, \lambda, X, Y, \alpha) = \sum_{i \in S} (r_i^2/n - \varepsilon^2) \tag{2}$$

under the constraint  $\|\alpha\|_1 \leq t$ , and where

$$S = \{i \in [n] \mid r_i^2 \leq \varepsilon^2\}. \tag{3}$$

Here  $S$  is the subset of data items assumed to be non-outliers and the complement,  $S^c = [n] \setminus S$ , is the subset of *potential* outliers. As can be seen in Eq. (2), we only consider the non-outliers when fitting the linear model.

At the limit of  $\varepsilon \rightarrow \infty$  it follows that  $S = [n]$  and Problem 1 becomes equivalent to LASSO (Tibshirani 1996). When  $\varepsilon$  is small Problem 1 becomes a combinatorial problem in disguise, where we try to find a maximal subset  $S$ , due to the subtraction of  $\varepsilon^2$  in Eq. (2).

**Theorem 1** *Minimising the loss in Eq. (2) maximises the size of the subset  $S$  in Eq. (3).*

**Proof** The subset size term in Eq. (2) is  $-\sum_{i \in S} \varepsilon^2 = -|S|\varepsilon^2$ , while for the residual term it holds that  $\sum_{i \in S} r_i^2/n \leq \varepsilon^2$ . This means that any change in the subset size has at least as large an impact on the loss as all the residuals combined, and for any  $\alpha$  and  $\alpha^*$  satisfying  $\text{Loss}(\varepsilon, \lambda, X, Y, \alpha) \leq \text{Loss}(\varepsilon, \lambda, X, Y, \alpha^*)$ , it then follows that  $|S| \geq |S^*|$ .  $\square$

## 2.1 Complexity class

Due to the combinatorial nature, finding an exact solution to Problem 1 is difficult. By showing that Problem 1 is a generalisation of a known NP-hard problem we can give a lower bound for the complexity class.

**Theorem 2** *Problem 1 is NP-hard and hard to approximate.*

**Proof** We prove the theorem by a reduction to the MAXIMUM SATISFYING LINEAR SUBSYSTEM problem (Ausiello et al. 1999, Problem MP10), which is known to be NP-hard. In MAXIMUM SATISFYING LINEAR SUBSYSTEM we are given the system  $X\alpha = y$ , where  $X \in \mathbb{Z}^{n \times m}$  and  $y \in \mathbb{Z}^n$  and we want to find  $\alpha \in \mathbb{Q}^m$  such that as many equations as possible are satisfied. This is equivalent to Problem 1 with  $\varepsilon = 0$  and  $\lambda = 0$ . Additionally, the problem is not approximable within  $n^\gamma$  for some  $\gamma > 0$ , according to Amaldi and Kann (1995).  $\square$

## 2.2 Breakdown value

The robustness of robust regression methods is often measured using the *breakdown value* (Donoho and Huber 1983; Rousseeuw and Hubert 2011), which is defined as the theoretical minimum fraction of (adversarial) outliers that can cause an arbitrarily large deviation in the model. This can, e.g., be measured with (Alfons et al. 2013):

$$v_{\text{breakdown}} = \min\{v : \sup\|X\alpha_v - Y\| \rightarrow \infty\},$$

where  $v$  is the fraction of outliers and  $\alpha_v$  is the linear model that fits the dataset  $(X_v, Y_v)$  where  $v$  of the items have been replaced by items with arbitrary values (outliers).

Non-robust regression methods, such as ordinary least-squares, have a breakdown value of  $1/n$  (Hubert and Debruyne 2009), i.e., a single outlier is enough to cause a breakdown. The practical upper limit for the breakdown value is 0.5, since any value larger than that cannot be *guaranteed*, without prior information.

**Theorem 3** *The breakdown value of Problem 1 is 0.5.*

**Proof** Following the definition, the breakdown value can be found as follows. We start from an uncorrupted (i.e., no outliers) dataset  $(X_0, Y_0)$  with  $n$  items obeying a linear model parametrised by  $\alpha_0$  and where all data items are within the corridor defined by  $\varepsilon$ . A fraction  $v$  of the data items are then changed into adversarial outliers, yielding the corrupted dataset  $(X_v, Y_v)$ .

The subset given by Problem 1 on  $(X_0, Y_0)$  is  $S_0$  and  $|S_0| = n$ . With a finite  $\varepsilon$  the breakdown occurs when the subset  $S_v$  for  $(X_v, Y_v)$  contains corrupted items. This

requires  $|S_v| \geq |S_0|/2$ , since otherwise the uncorrupted subset is larger,  $|S_0 \setminus S_v| > |S_v|$ , and is selected. Thus, the breakdown value is  $v = |S_v|/n = (|S_0|/2)/n = 0.5$ .  $\square$

In Sect. 7.2 we empirically validate the breakdown value.

### 3 Numeric approximation

In order to solve the NP-hard Problem 1 efficiently in the general case, we relax the problem by replacing the Heaviside function with a sigmoid function  $\sigma(u) = 1/(1 + e^{-u})$  and a continuous and differentiable rectifier function  $\phi(u) \approx \min(0, u)$ . This allows us to compute the gradient and find  $\alpha$  by minimising

$$\beta\text{-LOSS}(\varepsilon, \lambda, X, Y, \alpha) = \sum_{i=1}^n \sigma(\beta(\varepsilon^2 - r_i^2)) \phi(r_i^2/n - \varepsilon^2) + \lambda \|\alpha\|_1, \quad (4)$$

where the parameter  $\beta$  determines the steepness of the sigmoid and the rectifier function  $\phi$  is parametrised by a small constant  $\omega > 0$  such that

$$\phi(u) = \begin{cases} u, & u \leq -\omega, \\ -(u^2/\omega + \omega)/2, & -\omega < u < 0, \\ -\omega/2, & u \geq 0. \end{cases}$$

It is easy to see that Eq. (4) is a smoothed variant of Eq. (1) and that the two become equal when  $\beta \rightarrow \infty$  and  $\omega \rightarrow 0^+$ .

#### 3.1 Graduated optimisation

We perform the optimisation of Eq. (4) using *graduated optimisation* (Mobahi and Fisher 2015). Graduated optimisation iteratively solves a difficult optimisation problem by progressively increasing the complexity. A natural parametrisation for the complexity of our problem is via the  $\beta$  parameter, since  $\beta = 0$  corresponds to a convex optimisation problem equivalent to LASSO and when  $\beta \rightarrow \infty$  the problem becomes equivalent to Problem 1. This yields an iterative optimisation strategy.

At each step we use the previous optimal value of  $\alpha$  as a starting point for minimisation of Eq. (4), and then increase the value of  $\beta$ . It is important that the optima of consecutive  $\alpha$ :s are ‘‘close enough’’. This is why we derive an approximation ratio between solutions with different values of  $\beta$ . We observe that our problem can be rewritten as a maximisation of  $-\beta\text{-LOSS}(\varepsilon, \lambda, X, Y, \alpha)$ . The choice of  $\beta$  does not affect the L1-norm and we omit it for simplicity ( $\lambda = 0$ ).

**Theorem 4** *Given  $\varepsilon, \beta_1, \beta_2 \geq 0$ , such that  $\beta_1 \leq \beta_2$ , and the functions*

$$f_j(r) = -\sigma(\beta_j(\varepsilon^2 - r^2))\phi(r^2/n - \varepsilon^2),$$

and  $G_j(\alpha) = \sum_{i=1}^n f_j(r_i)$  where  $r_i = y_i - \alpha^T x_i$  and  $j \in \{1, 2\}$ . For  $\alpha_1 = \arg \max_{\alpha} G_1(\alpha)$  and  $\alpha_2 = \arg \max_{\alpha} G_2(\alpha)$  the inequality

$$G_2(\alpha_2) \leq K G_2(\alpha_1) \tag{5}$$

always holds, where  $K = G_1(\alpha_1) / (G_2(\alpha_1) \min_r f_1(r) / f_2(r))$  is the approximation ratio.

**Proof** The functions  $f_1$  and  $f_2$  are both always non-zero and positive: the inequalities  $\sigma(u) > 0$  and  $\phi(u) < 0$  hold for all  $u \in \mathbb{R}$ , thus  $f_j(r) > 0$ . Now, by definition,  $G_1(\alpha_2) \leq G_1(\alpha_1)$ . We denote  $r_i^* = y_i - \alpha_2^T x_i$  and  $k = \min_r f_1(r) / f_2(r)$ , which allows us the rewrite and bound:

$$G_1(\alpha_2) = \sum_{i=1}^n f_1(r_i^*) = \sum_{i=1}^n f_2(r_i^*) f_1(r_i^*) / f_2(r_i^*) \geq k G_2(\alpha_2).$$

Then  $G_2(\alpha_2) \leq G_1(\alpha_2) / k \leq G_1(\alpha_1) / k \leq G_2(\alpha_1) G_1(\alpha_1) / (k G_2(\alpha_1))$ , and the inequality from the theorem holds. □

We use Theorem 4 in the graduated optimisation to choose the sequence of increasing  $\beta$  values ( $\beta_1, \beta_2, \dots, \beta_i > \beta_{i-1}$ ), so that the approximation ratio, defined as  $K$ , stays constant.

### 3.2 Stopping criterion

Iterating until  $\beta \rightarrow \infty$  is not possible in practice, so we need a stopping criterion for the algorithm. The iterations should be stopped when  $\sigma(\beta(\varepsilon^2 - r^2)) \approx H(\varepsilon^2 - r^2)$ , i.e., the stopping criterion is dependent on the shape of the sigmoid function. The sigmoid shape is determined by both  $\beta$  and  $\varepsilon$ . However,  $\varepsilon$  is expected to change often, depending on both the dataset and the task, so a stopping criterion independent of  $\varepsilon$  would be preferable.

**Observation 1** *Setting  $\beta_{\max} \propto 1/\varepsilon^2$  makes the stopping criterion independent of  $\varepsilon$ .*

**Proof** Assume that there exists a pair of values  $\beta_c$  and  $\varepsilon_c$ . We say that a sigmoid function parametrised by some  $\beta_{\max}$  and  $\varepsilon$  has the same *relative shape* if and only if

$$\sigma(\beta_{\max}(\varepsilon^2 - (p\varepsilon)^2)) = \sigma(\beta_c(\varepsilon_c^2 - (p\varepsilon_c)^2))$$

for every value of  $p \in \mathbb{R}$ . Since the sigmoid function is strictly increasing, it can trivially be removed from both sides of the equation and hence

$$\begin{aligned} \sigma(\beta_{\max}(\varepsilon^2 - (p\varepsilon)^2)) &= \sigma(\beta_c(\varepsilon_c^2 - (p\varepsilon_c)^2)) \\ \beta_{\max} \varepsilon^2 (1 - p^2) &= \beta_c \varepsilon_c^2 (1 - p^2) \\ \beta_{\max} &= \beta_c \varepsilon_c^2 / \varepsilon^2 = c / \varepsilon^2, \end{aligned} \tag{6}$$

where  $c = \beta_c \varepsilon_c^2$  is a constant. □

We empirically determine a good default value for  $c$  in Sect. 6.5.



### 3.3 Approximation ratio for the numeric approximation

By using the approximation ratio between two  $\beta$ -Losses (Theorem 4) we can derive a new approximation ratio between the losses of Eqs. (4) and (1) (the problem definition and the numeric approximation).

We set  $\beta_2 \rightarrow \infty$  and  $\omega \rightarrow 0^+$  so that  $f_2(r) = -H(\varepsilon^2 - r^2)\phi(r^2/n - \varepsilon^2)$ . Additionally we introduce an  $\varepsilon^*$  such that  $f_2^*(r) = -H((\varepsilon^*)^2 - r^2)\phi(r^2/n - (\varepsilon^*)^2)$ ,  $G_2^*(\alpha) = \sum_{i=1}^n f_2^*(y_i - \alpha^\top x_i)$ , and  $\alpha_2^* = \arg \min_{\alpha} G_2^*(\alpha)$ . This leads to a new approximation ratio  $K_{\varepsilon^*}$  derived in the following lemma.

**Lemma 1** *The approximation ratio between the losses parametrised by  $\alpha_1$  and  $\alpha_2^*$  is  $K_{\varepsilon^*} = G_1(\alpha_1) / (G_2^*(\alpha_1)k_{\varepsilon^*})$  where*

$$k_{\varepsilon^*} = \sigma(\beta_1(\varepsilon^2 - (\varepsilon^*)^2))\phi((\varepsilon^*)^2/n - \varepsilon^2) / ((\varepsilon^*)^2/n - (\varepsilon^*)^2).$$

**Proof** The proof is omitted since it exactly mirrors Theorem 4 with the observation that  $k_{\varepsilon^*} = \min_r f_1(r) / f_2^*(r) = \min_{r \leq \varepsilon^*} (-f_1(r) / (r^2/n - \varepsilon^{*2}))$ , which leads to

$$k_{\varepsilon^*} = \sigma(\beta_1(\varepsilon^2 - (\varepsilon^*)^2))\phi((\varepsilon^*)^2/n - \varepsilon^2) / ((\varepsilon^*)^2/n - (\varepsilon^*)^2). \quad \square$$

We call  $\alpha_2^*$  the *matching solution*, since it is the optimum for Eq. (1) closest to  $\alpha_1$ . Note that  $\alpha_2^*$  has a different  $\varepsilon$  (namely  $\varepsilon^*$ ) that we need to specify in order to fully define the matching solution.

**Lemma 2** *For  $\varepsilon^*$  minimising the approximation ratio  $K_{\varepsilon^*}$ , it holds*

$$\varepsilon^* = \arg \max_{\varepsilon^*} \left( - \sum_{i=1}^n H((\varepsilon^*)^2 - r_i^2)(r_i^2/n - (\varepsilon^*)^2) \frac{\sigma(\beta_1(\varepsilon^2 - (\varepsilon^*)^2))\phi((\varepsilon^*)^2/n - \varepsilon^2)}{(\varepsilon^*)^2/n - (\varepsilon^*)^2} \right) \quad (7)$$

where  $r_i = y_i - \alpha_1^\top x_i$ .

**Proof** Let us denote

$$\varepsilon^* = \arg \min_{\varepsilon^*} K_{\varepsilon^*} = \arg \min_{\varepsilon^*} G_1(\alpha_1) / (G_2^*(\alpha_1)k_{\varepsilon^*}) = \arg \max_{\varepsilon^*} G_2^*(\alpha_1)k_{\varepsilon^*},$$

from which Eq. (7) follows. □

Due to the non-continuity of the Heaviside function, the maximum can be found at  $\varepsilon^* = r_j$  for some  $j \in [n]$ . We can further assume, without loss of generality, that the residuals are sorted so that  $r_1^2 \leq r_2^2 \leq \dots \leq r_n^2$ , which means that  $\sum_{i=1}^n H(r_j^2 - r_i^2) = j$ . With a large enough  $n$ , so that  $1/n \approx 0$ , Eq. (7) can be simplified to

$$\varepsilon^* \approx \arg \max_{r_j} j \cdot \sigma(\beta_1(\varepsilon^2 - r_j^2)). \quad (8)$$

Now, if the data is subsampled to a constant size, then Eq. (4) has a constant approximation ratio for the matching solution.

**Theorem 5** *The matching solution  $\alpha_2^*$  satisfies the inequality  $G_2^*(\alpha_2^*) \leq K_{\varepsilon^*} G_2^*(\alpha_1)$  where  $K_{\varepsilon^*} = \mathcal{O}(\log n)$  is the approximation ratio.*

**Proof** Since  $\varepsilon^* = r_j$  for some  $j \in [n]$ , it follows that  $K_{\varepsilon^*} \leq K_{r_t}$  for all  $t \in [n]$ , where the definition of  $K_{r_t} = G_1(\alpha_1)/(G_2^{r_t}(\alpha_1)k_{r_t})$  follows Theorem 1 (with  $r_t$  instead of  $\varepsilon^*$ ). Hence,

$$1/K_{r_t} = G_2^{r_t}(\alpha_1)k_{r_t}/G_1(\alpha_1) \leq (G_2^*(\alpha_1)k_{\varepsilon^*})/G_1(\alpha_1) = 1/K_{\varepsilon^*},$$

and we can derive, by rearranging the terms in the inequality,

$$k_{r_t} \leq G_2^*(\alpha_1)k_{\varepsilon^*}/G_2^{r_t}(\alpha_1).$$

Assuming that the residuals are sorted so that  $r_1^2 \leq r_2^2 \leq \dots \leq r_n^2$ , then

$$G_2^{r_t}(\alpha_1) = - \sum_{i=1}^n H(r_i^2 - r_t^2)(r_i^2/n - r_t^2) \geq -t(r_t^2/n - r_t^2),$$

and further following the definition in Theorem 1

$$k_{r_t} = \sigma(\beta_1(\varepsilon^2 - r_t^2))\phi(r_t^2/n - \varepsilon^2)/(r_t^2/n - r_t^2),$$

which, by rearranging, lets us derive

$$\begin{aligned} -\sigma(\beta_1(\varepsilon^2 - r_t^2))\phi(r_t^2/n - \varepsilon^2) &= k_{r_t} \cdot (-r_t^2/n - r_t^2) \\ &\leq G_2^*(\alpha_1)k_{\varepsilon^*}/G_2^{r_t}(\alpha_1) \cdot (-r_t^2/n - r_t^2) \\ &\leq G_2^*(\alpha_1)k_{\varepsilon^*}/(-t(r_t^2/n - r_t^2)) \cdot (-r_t^2/n - r_t^2) \\ &= G_2^*(\alpha_1)k_{\varepsilon^*}/t. \end{aligned}$$

Inserting this into  $G_1$  yields

$$\begin{aligned} G_1(\alpha_1) &= - \sum_{i=1}^n \sigma(\beta_1(\varepsilon^2 - r_i^2))\phi(r_i^2/n - \varepsilon^2) \\ &\leq \sum_{i=1}^n G_2^*(\alpha_1)k_{\varepsilon^*}/i \\ &\leq G_2^*(\alpha_1)k_{\varepsilon^*}(\log n + 1), \end{aligned}$$

and when combined with  $K_{\varepsilon^*}$  from Theorem 1 we have

$$K_{\varepsilon^*} = G_1(\alpha_1)/(G_2^*(\alpha_1)k_{\varepsilon^*}) \leq G_2^*(\alpha_1)k_{\varepsilon^*}(\log n + 1)/(G_2^*(\alpha_1)k_{\varepsilon^*}) = \log n + 1,$$

which completes the proof. □

### 4 The SLISE algorithm

In this section we describe an approximate numeric algorithm, SLISE, for solving Problem 1. We start by introducing the algorithm, and then continue by discussing

different initialisation schemes. Finally we demonstrate the asymptotic complexity of the SLISE algorithm.

The SLISE algorithm (Algorithm 1) takes as input the data and the optimisation parameters. The algorithm starts by selecting initial values for the linear model  $\alpha$  and the sigmoid steepness  $\beta$  (line 3). There are several potential initialisation schemes, and we will present and discuss four alternatives later in this section (see Sect. 4.1 and Algorithm 3).

```

1 Parameters: (1) dataset  $X \in \mathbb{R}^{n \times d}$ ,  $Y \in \mathbb{R}^n$ ,
                (2) error tolerance  $\varepsilon$ ,
                (3) regularisation coefficient  $\lambda$ ,
                (4) maximum sigmoid steepness  $\beta_{\max}$ ,
                (5) approximation ratio  $K$ ,
                (6) additional parameters to the initialisation function ...
2 Function SLISE( $X, Y, \varepsilon, \lambda, \beta_{\max}, K, \dots$ )
3    $\alpha, \beta \leftarrow \text{Initialise}(X, Y, \varepsilon, K, \dots)$ 
4   while  $\beta < \beta_{\max}$  do
5      $\alpha \leftarrow \text{OWL-QN}(\beta\text{-Loss}, X, Y, \varepsilon, \lambda, \alpha)$ 
6      $\beta \leftarrow \beta'$  such that  $\text{ApproximationRatio}(X, Y, \varepsilon, \beta, \beta', \alpha) = K$ 
7    $\alpha \leftarrow \text{OWL-QN}(\beta\text{-Loss}, X, Y, \varepsilon, \lambda, \alpha)$ 
8   Result:  $\alpha$ 
    
```

**Algorithm 1:** The SLISE algorithm. The  $\beta$ -Loss is given by Eq. (4).

The main part of SLISE consists of graduated optimisation (lines 4–7), that optimise the values for  $\alpha$  and  $\beta$ . This is done by alternating between optimising  $\alpha$ , and increasing  $\beta$  until we reach  $\beta_{\max}$ . At each iteration, we need to find the  $\alpha$  that minimises the  $\beta$ -Loss in Eq. (4), using the current value of  $\beta$  (line 5). This optimisation is done with OWL-QN (Schmidt et al. 2009) which is a quasi-Newton optimisation method with built-in L1-regularisation. We then increase  $\beta$  (line 6) such that the approximation ratio between consecutive steps, as defined in Theorem 4, equals  $K$ .

The pseudocode for the approximation ratio calculation is provided in Algorithm 2. In Eq. (4) we use the rectifying  $\phi$  function to ensure negativity. This function requires a constant  $\omega$  and its value can be chosen to be smaller than machine epsilon. Hence, in the approximation ratio calculation (line 4),  $\phi(u)$  is numerically equivalent to  $\min(0, u)$ . As a minor computational side-effect of this, we have to make sure not to divide by zero if all  $\phi(r_i) = 0$  (lines 8–11).

### 4.1 Initialisation schemes

In Algorithm 3 we introduce four alternative initialisation schemes, and later in Sect. 6.3 we empirically compare the proposed approaches.

The first initialisation scheme (lines 2–5) is to use the non-robust counterpart to SLISE, i.e., LASSO-regression, by setting  $\beta = 0$ . Since LASSO is a convex problem, the initial  $\alpha$  does not matter, and here we use ordinary least squares regression to obtain the initial  $\alpha$ .

```

1 Parameters: (1) dataset  $X \in \mathbb{R}^{n \times d}$ ,  $Y \in \mathbb{R}^n$ ,
                (2) error tolerance  $\varepsilon$ ,
                (3) sigmoid steepnesses  $\beta_1, \beta_2$ ,
                (4) linear model  $\alpha$ 
2 Function ApproximationRatio ( $X, Y, \varepsilon, \beta_1, \beta_2, \alpha$ )
3    $f \leftarrow \text{function}(r, \beta) : -\sigma(\beta(\varepsilon^2 - r^2))$ 
4    $\phi \leftarrow \text{function}(r) : \min(0, r^2/n - \varepsilon^2)$ 
5    $k \leftarrow \text{Minimise}(f(r, \beta_1)/f(r, \beta_2))$  by adjusting  $r$ 
6   for  $i = 1, \dots, n$  do
7      $r_i \leftarrow y_i - \alpha^\top x_i$ 
8   if  $\sum_{i=1}^n \phi(r_i) \neq 0$  then
9      $K \leftarrow \sum_{i=1}^n f(r_i, \beta_1)\phi(r_i) / (k \cdot \sum_{i=1}^n f(r_i, \beta_2)\phi(r_i))$ 
10  else
11     $K \leftarrow \sum_{i=1}^n f(r_i, \beta_1) / (k \cdot \sum_{i=1}^n f(r_i, \beta_2))$ 
12  Result:  $K$ 

```

**Algorithm 2:** Approximation ratio calculation.

With  $\beta > 0$  the problem is no longer convex and the choice of initial  $\alpha$  becomes important. The next two initialisation schemes (lines 6–9 and lines 10–13) offer two different alternatives. In the first scheme the initial  $\alpha$  (line 7) is given by non-sparse OLS, while in the second scheme  $\alpha$  (line 11) is a super-sparse constant vector of only zeros. In both cases the approximation ratio (Algorithm 2) is used to select an initial value for  $\beta$ .

The final scheme (lines 14–27) is inspired by the initialisation used in FAST-LTS (Rousseeuw and Van Driessen 2000) and RANSAC (Fischler and Bolles 1981), which are related robust regression methods. The idea is to generate  $u_{\text{init}}$  initial candidates, and heuristically select the best one. The candidates are generated by drawing random subsets  $(X_S, Y_S)$  of size  $d + 1$ , i.e.,  $X_S \in \mathbb{R}^{d \times d+1}$  and  $Y_S \in \mathbb{R}^d$ , and fitting linear models to them (using OLS, lines 20–22). Note that the probability that at least one of the subsets is free from outliers is given by  $p = 1 - (1 - (1 - o)^d)^u$ , where  $o$  is the fraction of outliers,  $d$  is the number of dimensions, and  $u$  is the number of candidates. If  $d$  is large, then  $u$  would also have to be large to compensate. To alleviate this potential issue we use PCA to temporarily reduce the number of dimensions when  $d$  is larger than a threshold  $t_{\text{pca}}$  (lines 17–19). Finally, the best candidate  $\alpha$  is the one that minimises the  $\beta$ -LOSS and  $\beta$  is updated to match the currently best  $\alpha$  (line 23–26).

## 4.2 Asymptotic complexity

The evaluation of the loss function (and its gradient) has a time complexity of  $\mathcal{O}(nd)$ , due to the multiplication between the linear model  $\alpha$  and the data-matrix  $X$ . The approximation ratio calculation also has a complexity of  $\mathcal{O}(nd)$  for the same reason. This means that the complexity of the initialisation is dominated by the complexity of OLS, which is  $\mathcal{O}(\min(nd^2, d^3))$ .

The optimisation consists of OWL-QN and graduated optimisation. OWL-QN is a variant of LBFGS and has a time complexity of  $\mathcal{O}(md)$  (Schmidt et al. 2009), where  $m$

```

1 Parameters: (1) dataset  $X \in \mathbb{R}^{n \times d}$ ,  $Y \in \mathbb{R}^n$ ,
                (2) error tolerance  $\varepsilon$ ,
                (3) target approximation ratio  $K$ ,
                (4) number of candidates  $u_{\text{init}}$ ,
                (5) threshold for using PCA  $t_{\text{pca}}$ 
2 Function InitialiseLasso( $X, Y$ )
3    $\alpha \leftarrow \text{OrdinaryLeastSquares}(X, Y)$ 
4    $\beta \leftarrow 0$ 
5   Result:  $\alpha, \beta$ 
6 Function InitialiseOLS( $X, Y, \varepsilon, K$ )
7    $\alpha \leftarrow \text{OrdinaryLeastSquares}(X, Y)$ 
8    $\beta \leftarrow \beta'$  such that  $\text{ApproximationRatio}(X, Y, \varepsilon, 0, \beta', \alpha) = K$ 
9   Result:  $\alpha, \beta$ 
10 Function InitialiseZeros( $X, Y, \varepsilon, K$ )
11    $\alpha \leftarrow \mathbf{0}$ 
12    $\beta \leftarrow \beta'$  such that  $\text{ApproximationRatio}(X, Y, \varepsilon, 0, \beta', \alpha) = K$ 
13   Result:  $\alpha, \beta$ 
14 Function InitialiseCandidates( $X, Y, \varepsilon, K, u_{\text{init}}, t_{\text{pca}}$ )
15    $\alpha, \beta, l \leftarrow \mathbf{0}, 0, 0$ 
16   for  $i = 1, \dots, u_{\text{init}}$  do
17     if  $d > t_{\text{pca}}$  then
18        $X_S, Y_S \leftarrow \text{RandomSubset}(X, Y, \text{size} = t_{\text{pca}})$ 
19        $\alpha' \leftarrow \text{InversePCA}(\text{OrdinaryLeastSquares}(\text{PCA}(X_S), Y_S))$ 
20     else
21        $X_S, Y_S \leftarrow \text{RandomSubset}(X, Y, \text{size} = d)$ 
22        $\alpha' \leftarrow \text{OrdinaryLeastSquares}(X_S, Y_S)$ 
23     if  $\beta\text{-Loss}(X, Y, \varepsilon, 0, \alpha') < l$  then
24        $\alpha \leftarrow \alpha'$ 
25        $\beta \leftarrow \beta'$  such that  $\text{ApproximationRatio}(X, Y, \varepsilon, 0, \beta', \alpha) = K$ 
26        $l \leftarrow \beta\text{-Loss}(X, Y, \varepsilon, 0, \alpha)$ 
27   Result:  $\alpha, \beta$ 

```

**Algorithm 3:** Schemes for initialising  $\alpha$  and  $\beta$ .

is the size of the “memory” for approximating the inverse Hessian. Additionally, OWL-QN requires the loss value and gradient to be calculated every iteration. In practice, the number of iterations can be limited by a constant upper bound (see Sect. 6.4). The graduated optimisation only adds the approximation ratio calculation, the cost of which is negligible.

The total asymptotic time complexity of SLISE is a combination of the initialisation and the optimisation complexities:  $\mathcal{O}(\min(nd^2, d^3) + ndp)$ , where  $p$  is an upper bound for the total number of optimisation iterations. However, in most cases  $p \gg d$  and the OLS term becomes vanishingly small (see Sect. 7.1) and in cases where  $d > p$  the exact OLS solution can be replaced by an optimisation, e.g., using LBFGS ( $\mathcal{O}(ndi_{\text{max}})$ ), since the initialisation does not have to be exact. In this case the complexity of SLISE becomes  $\mathcal{O}(ndp)$ , or rather  $\mathcal{O}(nd)$  since  $p$  is constant. Also note that any given dataset

might require fewer iterations than  $p$ , but that is linked to the difficulty of finding the largest linear subset, rather than the size of the dataset.

The memory complexity of SLISE is at least  $\mathcal{O}(nd)$ , i.e., the memory required to store the dataset itself. The loss function adds  $\mathcal{O}(\max(n, d))$ , the approximation ratio  $\mathcal{O}(n)$ , and OWL-QN  $\mathcal{O}(md)$ . This makes the total asymptotic memory complexity of SLISE  $\mathcal{O}(nd)$ .

## 5 Experimental setup

We divide the empirical evaluation into three sections. In this section we describe the datasets and environment used for the experiments. In Sect. 6, we empirically determine suitable default values for the parameters of SLISE, and which initialisation method to recommend. Later, in Sect. 7 we compare SLISE to competing methods by demonstrating that (i) SLISE scales better on high-dimensional datasets than competing methods, (ii) SLISE is very robust to noise, and (iii) the solutions found using SLISE are optimal.

The experiments were run using R (Microsoft and R Core Team 2019, v. 3.5) on a high-performance computer cluster (FGCI 2021), using 4 cores from an Intel Xeon E5-2680 at 2.4 GHz, and reserving 16 GB of RAM. Our implementation of the SLISE algorithm, the code to run all the experiments, and the data processing are released as open source (Björklund et al. 2021).

### 5.1 Datasets

For the empirical evaluation we use both real and synthetic datasets. An overview of all the datasets is shown in Table 1. The real datasets are three regression datasets from the *UCI Machine Learning Repository* (Dua and Graff 2019), namely STUDENT (Cortez and Silva 2008), AIR QUALITY (De Vito et al. 2008), and SUPERCONDUCTIVITY (Hamidieh 2018).

As additional real datasets we use some classification datasets for which we create regression tasks with the help of complex classifiers. The datasets and classifier pairs are handwritten digits (Cohen et al. 2017, EMNIST) with a convolutional neural network, movie reviews (Maas et al. 2011, IMDB) with a support vector machine, and particle jets (HIP CMS Experiment 2019, PHYSICS) with a neural network. The predictions from the classifiers are probabilities, which we turn into real values using the logit function:  $y'_i = \log(y/(1 - y))$ . EMNIST has ten classes (digits), which yields ten different regression tasks. Whenever one of these tasks are used, we randomly subsample the dataset such that 50% of the items are of the “correct” digit and 50% are from all the other digits. This creates datasets with lots of potential outliers.

Synthetic datasets are generated as follows. The data matrix  $X \in \mathbb{R}^{n \times d}$  is created by sampling from ten normal distribution with randomised means and variances ( $\mu \sim \mathcal{N}(0, 2)$  and  $\sigma^2 \sim \mathcal{U}(0, 1)$ ). The response vector  $Y \in \mathbb{R}^n$  is created by  $y_i = a_{k_0} + a_k^\top x_i + e$ , where  $e$  is normal noise with zero mean and unit variance, and  $a_k \in \mathbb{R}^{d+1}$  is one of seven linear models with the coefficients drawn from a uniform distribution

**Table 1** The datasets used in the experiments. The synthetic dataset can be generated to any desired size

Dataset	Items	Variables	References
STUDENT	1044	40	Cortez and Silva (2008)
AIR QUALITY	6941	11	De Vito et al. (2008)
SUPERCONDUCTIVITY	21,263	81	Hamidieh (2018)
IMDB	25,000	1000	Maas et al. (2011)
PHYSICS	260,000	4	HIP CMS Experiment (2019)
EMNIST	40,000	672	Cohen et al. (2017)
SYNTHETIC	$n$	$d$	Björklund et al. (2021)

between  $-1$  and  $1$ . Each of the seven models  $a_k$  is used to create 10% of the  $Y$ -values, except one that creates 40% of the  $Y$ -values. The code for creating the synthetic datasets is available in the repository (Björklund et al. 2021) for full reproducibility.

## 5.2 Pre-processing

SLISE uses LASSO regularisation both for introducing sparsity and regularisation. Since the LASSO penalty sums the absolute values of the coefficients, it is important that the variables have roughly the same magnitude (Tibshirani 1996). Normalising the variables also makes it easier to compare the relative importance of values when interpreting the results. Thus, the variables in all datasets, except IMDB and EMNIST, have been centred around (subtracted) the median and scaled (divided) by the MAD (median absolute deviation) since this is a more robust form of normalisation than using means and standard deviations (Rousseeuw and Hubert 2011). Furthermore, we add an intercept column to the datasets after the normalisation. For all datasets we also normalise the target ( $Y$ ) in the same way, to make the  $\varepsilon$ :s comparable.

In the STUDENT dataset we remove the grades for the first and second period, since these are very correlated with the target (the grades for the third period), as noted by Cortez and Silva (2008).

For the EMNIST dataset the targets ( $Y$ ) are robustly scaled as described above, whereas the pixel values of the input images are just linearly scaled from  $[0, 255]$  to  $[-1, 1]$ . Some of the pixels have the same value in all of the images (i.e., the corners), which is problematic for some of the comparison methods (in Sect. 7), so these are removed and the images are flattened to vectors of length 672.

The text data in the IMDB dataset is transformed into real-valued vectors by using a bag-of-words model, after case normalisation, removal of stop words, removal of punctuation, and stemming. The obtained word frequencies are divided by the most frequent word in each review to adjust for different review lengths, yielding real-valued vectors of length 1000.

**Table 2** In the experiments we use the following default values for the parameters of SLISE, if nothing else is mentioned

Parameter	Default value	Description
$\varepsilon$	$\implies  S /n = 0.3$	Error tolerance
$\lambda$	$10^{-4} \cdot n\varepsilon^2$	Regularisation coefficient
$\beta_{\max}$	$20/\varepsilon^2$	Stopping parameter
$K$	1.15	Approximation ratio
$i_{\max}$	300	Maximum number of OWL- QN iterations
Initialisation	CANDIDATES	Initialisation scheme
$u_{\text{init}}$	500	Number of CANDIDATES
$t_{\text{pca}}$	10	Threshold for using PCA

## 6 Parameter experiments

The SLISE algorithm presented in Sect. 4 has multiple parameters that must be set. In addition, we also presented four different initialisation schemes. In this section we find good default values for the parameters, and determine which initialisation scheme to recommend.

The two most important parameters for SLISE are the error tolerance  $\varepsilon$  and the sparsity coefficient  $\lambda$ . These depend on both the use case and the dataset. Therefore, they should be manually adjusted whenever SLISE is used. The default values for the other parameters, and which initialisation scheme to recommend, are selected based on empirical evidence. Specifically, we base the selection on the value of the loss function and the running time.

All experiments are run ten times per dataset (with different random seeds) in order to better capture the variance (this means that EMNIST is run a hundred times, due to the ten different tasks/digits). Furthermore, any dataset with  $n > 10,000$  is randomly subsampled to  $n = 10,000$ . If nothing else is mentioned, the values in Table 2 are used as default values for the parameters in the experiments.

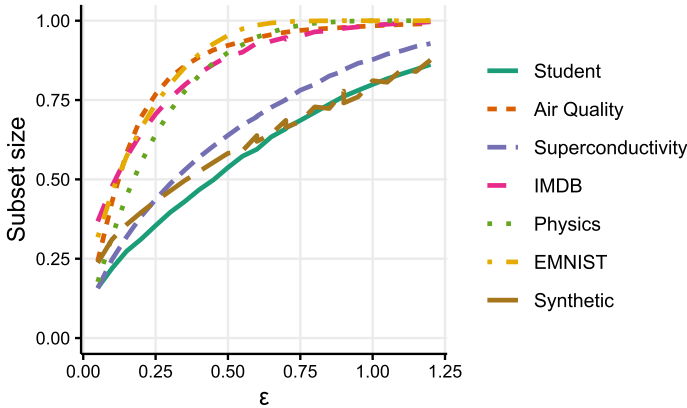
### 6.1 Error tolerance

The role of the error tolerance is to enable detection and ignoring of outliers. Depending on the dataset or task there might be some obvious limits for the error which should be used when available. The question is hence how to select the value for  $\varepsilon$  without this knowledge?

The value of  $\varepsilon$  directly affects the size of the subset that fits the resulting model. Ideally, we want a value of  $\varepsilon$  such that the subset contains a large amount of non-outliers and no outliers. In order to better understand how to set the value for  $\varepsilon$  we consider, in Fig. 2, how  $\varepsilon$  corresponds to subset sizes for the datasets.

To be able to guarantee the maximum possible robustness we could proceed as Rousseeuw (1984) and select the  $\varepsilon$ :s that correspond to subset sizes of 50%. However, since none of the datasets considered here contain that many outliers another possibility





**Fig. 2** Measuring how the  $\epsilon$  value affects the subset size. Small values result in (too) small subsets, while large values have a diminishing return (due to the natural sparseness at the edges of the distributions and to outliers)

**Table 3** Selection of  $\epsilon$  based on the resulting subset size

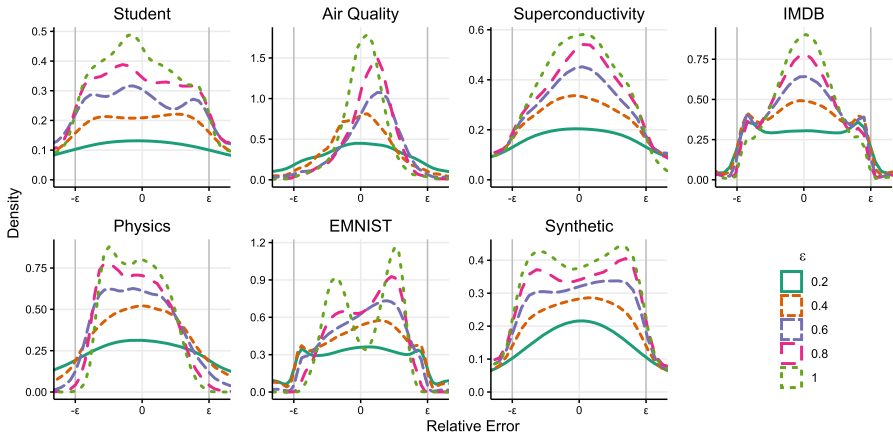
Dataset	$\epsilon$ for subset size		
	75%	50%	30%
STUDENT	0.90	0.45	0.18
AIR QUALITY	0.24	0.12	0.07
SUPERCONDUCTIVITY	0.70	0.30	0.14
IMDB	0.29	0.11	0.05
PHYSICS	0.35	0.21	0.09
EMNIST	0.28	0.13	0.05
SYNTHETIC	0.92	0.36	0.11

is to be less strict and choose subset sizes of, e.g., 75% (Alfons et al. 2013). However, decreasing  $\epsilon$  makes the task more difficult and we want to test parameters under adverse conditions. Hence, in this paper we use 30% for all datasets. Furthermore, in the previous paper on SLISE (Björklund et al. 2019) the main use case is explaining outcomes from black box models, where smaller values of  $\epsilon$  might be preferable. The corresponding  $\epsilon$  values for all datasets can be seen in Table 3.

Another way to investigate the choice of  $\epsilon$  is to plot the distribution of errors relative to the size of  $\epsilon$ . When  $\epsilon$  is small only the peak fits within  $[-\epsilon, \epsilon]$  and when  $\epsilon$  is large parts of the tails are included. It is natural for distributions to have tails, and not all items in the tails are outliers, but this could nonetheless be used as another criterion for selecting  $\epsilon$ , which can be seen in Fig. 3.

### 6.2 Regularisation

The regularisation coefficient  $\lambda$  is dependent on the use case and dataset. With LASSO-type methods it is common to scale the regularisation by the number of items  $n$  (or



**Fig. 3** The distributions of errors relative to  $\epsilon$  can give information on the the effect of  $\epsilon$ . When  $\epsilon$  becomes large some of the tails of the error distributions fit inside  $[-\epsilon, \epsilon]$ . This is a sign that  $\epsilon$  might be too large. For example,  $\epsilon > 0.8$  with the PHYSICS dataset include tails within the interval (selected subset)

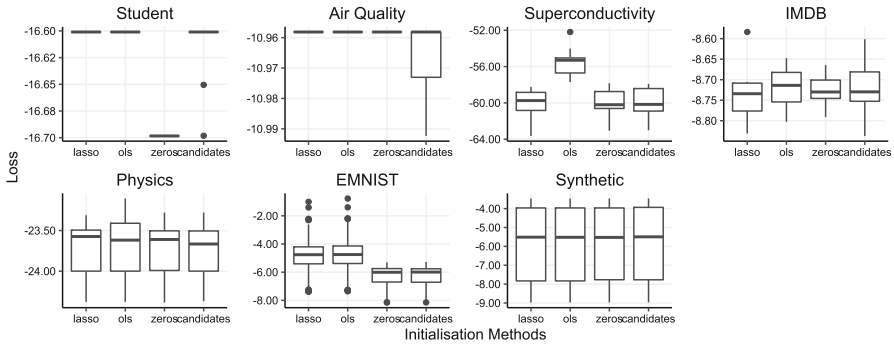
by dividing the rest of the loss by  $n$ ). For SLISE the size of the subset  $|S|$  would be a better choice but that is not known in advance. Furthermore, the rest of the loss function is proportional to  $\epsilon^2$ , so selecting  $\lambda \propto n\epsilon^2$  makes the transition between dataset sizes and parameter values easier. For the purpose of the experiments, we only use a minimal regularisation by setting  $\lambda = 10^{-4} \cdot n\epsilon^2$ , since we are not looking for a specific sparsity.

### 6.3 Initialisation

In Sect. 4 we present four different schemes for selecting the initial values for the linear model  $\alpha$  and sigmoid steepness  $\beta$ . Figure 4 shows the results from comparing the four initialisation schemes. No particular method stands out, which indicates that the combination of graduated optimisation and OWL-QN yields good performance overall. Furthermore, there are no major differences in running time, according to Table 4. However, SLISE can only be guaranteed to find a local optimum (in contrast to finding the global optimum), so we need to consider possible failure modes.

Both LASSO and OLS are non-robust so even a single outlier can lead to arbitrarily large deviations (Donoho and Huber 1983). This means that the starting points might be heavily influenced by outliers. Starting from a vector of ZEROS is good for sparsity, but with a large enough  $\lambda$  it becomes a local optimum. It is, however, easy to detect when the optimisation has failed to escape this kind of local optimum, by checking if  $\|\alpha\|_1 \approx 0$ .

Another problem with using a fixed starting point (which is what the LASSO, OLS, and ZEROS initialisation schemes do) is that if the starting point is bad, then there is no way to detect and recover from it. This can be seen in Fig. 5, where the fixed starting  $\alpha$ :s are all very close to a cluster of outliers, which creates a local optimum that the optimisation cannot escape. The CANDIDATES initialisation is designed to detect



**Fig. 4** Losses for the different initialisation methods. Lower is better

**Table 4** Median times (for the whole algorithm) when trying different initialisation methods. Lower values are better

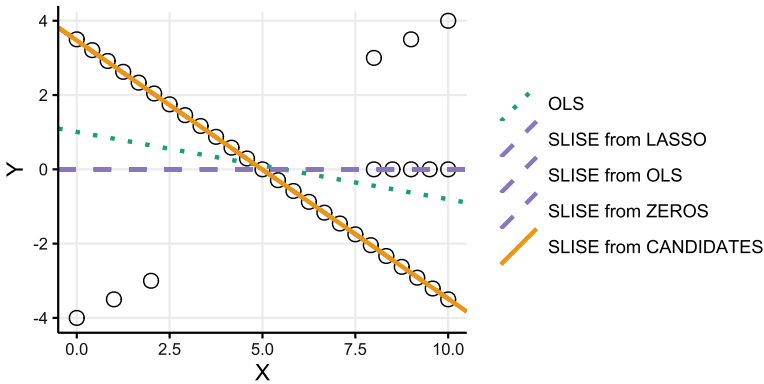
Dataset	Median time [s]			
	LASSO	OLS	ZEROS	CANDIDATES
STUDENT	1.5	1.6	1.4	1.9
AIR QUALITY	10.9	10.6	10.7	11.8
SUPERCONDUCTIVITY	39.7	30.0	35.8	35.4
IMDB	350.8	372.5	345.0	325.4
PHYSICS	15.1	15.4	14.3	15.9
EMNIST	323.7	304.1	257.7	312.4
SYNTHETIC	4.3	3.9	3.9	4.4

and discard these bad local optima early, by selecting a candidate based on the best  $\beta$ -Loss.

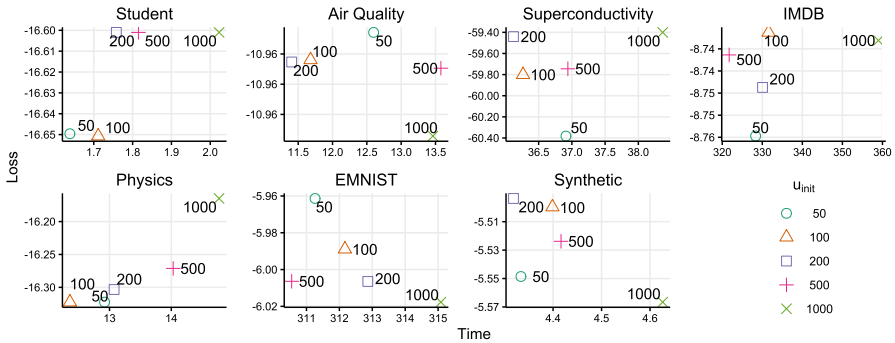
The CANDIDATES initialisation scheme is non-deterministic, and requires the number of candidates  $u_{init}$  as a parameter. A larger number increases the likelihood that a good candidate is found, but also increases the running time. However, the results from Fig. 6 show that the number of candidates only has a small impact, and also that the time differences are negligible. FAST-LTS (Rousseeuw and Van Driessen 2000) has a similar notion of candidates, and by default they use  $u_{init} = 500$ , which seems to be a reasonable choice also in this case.

With  $u_{init} = 500$  fixed, we can find a value for the threshold  $t_{pca}$  for using PCA, that is independent of any dataset. The formula for the probability of at least one candidate having no outliers is  $p = 1 - (1 - (1 - o)^d)^u$ , where  $o$  is the fraction of outliers,  $d = t_{pca}$ , and  $u = u_{init} = 500$ . A larger threshold leads to less information lost in the PCA while a smaller value increases  $p$ . The curves for different  $t_{pca}$  can be seen in Fig. 7. If  $t_{pca} = 10$  then  $p = 0.38$  when  $o = 0.5$ , which should be sufficient in most scenarios, since including one outlier does not automatically lead to an inescapable and bad local optimum.

Based on the results in Fig. 4, Table 4, and Fig. 5, we choose CANDIDATES as the recommended initialisation scheme. It is the only stochastic initialisation scheme,



**Fig. 5** Example showing when some of the initialisation schemes fail. The data is constructed such that the starting  $\alpha$ :s from LASSO, OLS, and ZEROS all pass by a cluster of outliers. These outliers create a local optimum which only the CANDIDATES initialisation scheme is able to avoid. SLISE is used with  $\epsilon = 0.2$  and  $\lambda = 0.01$

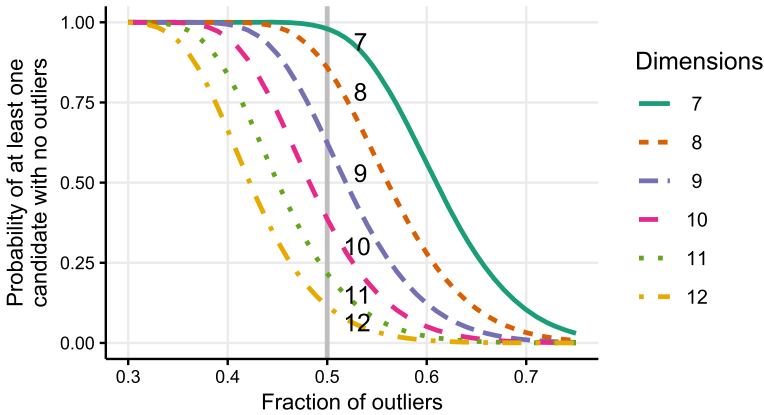


**Fig. 6** Losses and running times for different number of initial candidates in the initialisation. For a given dataset, the losses and running times are for all practical purposes equal for different number of initial candidates. The results from multiple runs are aggregated using the mean. Lower is better for both time and loss

which means that it does not have a fixed (potentially bad) starting point. Alternatively, the failure mode of ZEROS is easy to detect, and the initialisation is fast, so it is our second choice if a deterministic algorithm is desired.

### 6.4 Iterations

SLISE incorporates two iterative optimisation methods, OWL- QN and graduated optimisation. Increasing the number of iterations leads to better results for both methods, but beyond a point there are clear diminishing returns. The number of iterations in graduated optimisation is determined by the target approximation ratio  $K$  (where  $K > 1$ ). A larger value results in fewer iterations. However, we require the steps to be small enough that consecutive iterations have similar optima.



**Fig. 7** The probability of finding at least one subset without outliers for different fractions of outliers and different dimensions for the PCA threshold. This can be calculated with the formula  $p = 1 - (1 - (1 - o)^d)^u$ . Both higher probabilities and more dimensions are better

The number of iterations in OWL- QN can be controlled by several different convergence criteria, but the simplest one is to simply limit the number of iterations  $i_{max}$ . This has the advantage of ensuring an upper bound on the number of iterations in the worst-case scenario. Additionally, since OWL- QN is run multiple times on similar problems there will be a lot of wasted resources if it is forced to fully converge each graduated optimisation iteration.

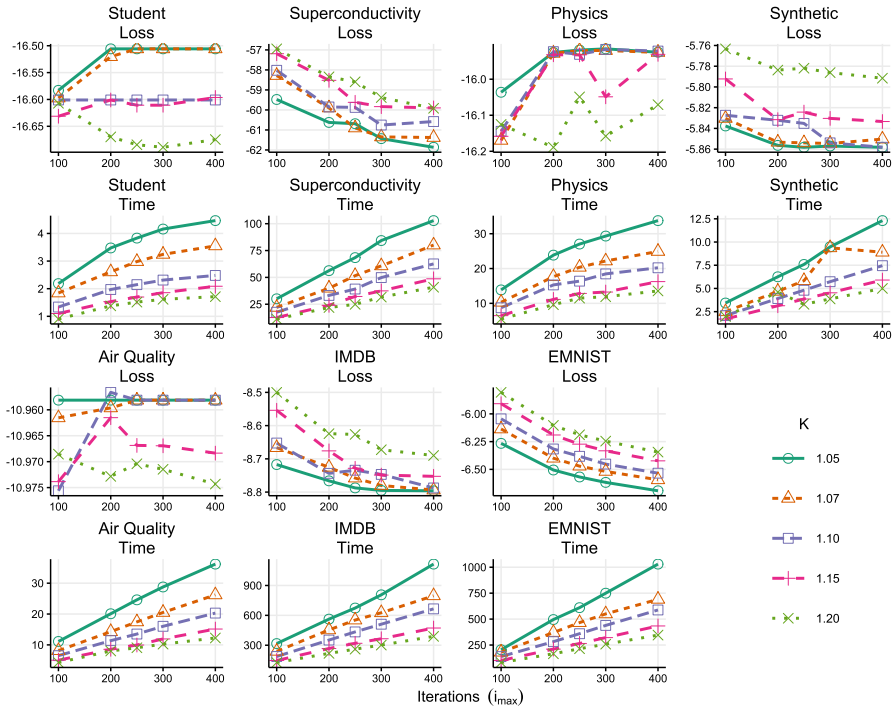
As can be seen in Fig. 8,  $K$  and  $i_{max}$  complements each other, in that a decrease in  $K$  can be compensated for by a decrease in  $i_{max}$  while preserving time and loss values. The choice of values for these parameters can have a massive impact on the running time, while the impact on the loss at times is minimal. Based on the results, the combination of  $K = 1.15$  and  $i_{max} = 300$  is a good default trade-off between time and loss. Furthermore, the last optimisation of OWL- QN is not an intermediate step and is, therefore, allowed extra time to converge, by multiplying  $i_{max}$  with four ( $i_{max} = 1200$  when  $\beta = \beta_{max}$ ).

### 6.5 Stopping parameter

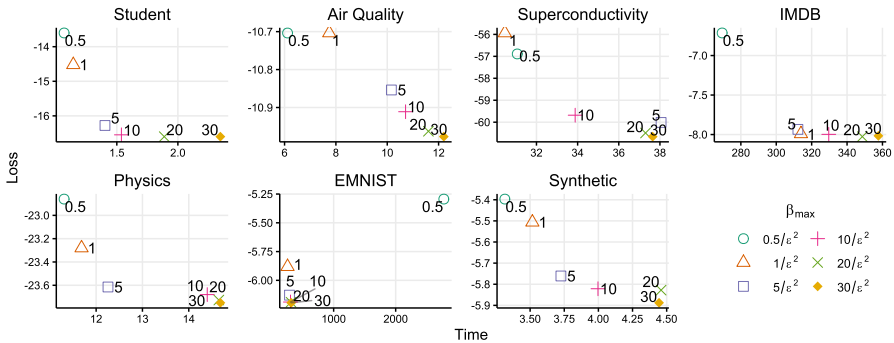
It is sufficient that the stopping parameter  $\beta_{max}$  is large enough to make the sigmoid function essentially equivalent to a Heaviside function. As shown in Sect. 3.2, in order to make the shape of the sigmoid only depend on  $\beta_{max}$  it has to be defined as  $\beta_{max} = c/\epsilon^2$ . The results in Fig. 9 show that  $c = 20$  is sufficiently large, and any larger value merely increases the running time.

## 7 Robust regression experiments

In this section we compare SLISE to five robust regression methods: SPARSE-LTS (Alfons et al. 2013), SMDM (Koller and Stahel 2017), CONQUER (Fernandes et al.



**Fig. 8** Losses and running times for different values of the two parameters that control the number of iterations. The results from multiple runs are aggregated using the mean. Lower values are better for both time and losses



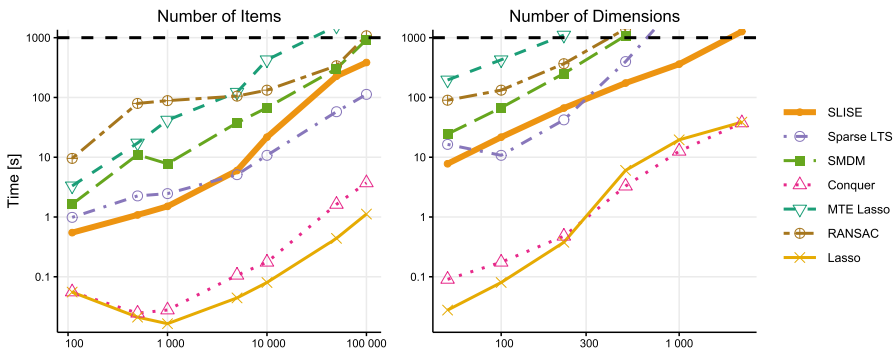
**Fig. 9** Losses and running times for various values for the  $\beta_{max}$  parameter. Lower values are better

2021), MTE- LASSO (Qin et al. 2017), and RANSAC (Fischler and Bolles 1981). LASSO (Tibshirani 1996) is also included as a non-robust baseline. To make the comparison maximally useful we compare against implementations found in established software libraries. See Table 5 for an overview of all methods.

All algorithms have been used with default settings, with the exception of SPARSE-LTS, which has been used with a subset size of  $n/2$  for maximal robustness, and

**Table 5** Properties of the regression methods compared in the experiments

Algorithm	Robust	Sparse	References	Software
SLISE	Yes	Yes	<i>The method presented in this paper</i>	slise (R)
SPARSE- LTS	Yes	Yes	Alfons et al. (2013)	robustHD (R)
SMDM	Yes	No	Koller and Stahel (2017)	robustbase (R)
CONQUER	Maybe	No	Fernandes et al. (2021)	conquer (R)
MTE- LASSO	Maybe	Yes	Qin et al. (2017)	MTE (R)
RANSAC	Maybe	No	Fischler and Bolles (1981)	scikit-learn (Python)
LASSO	No	Yes	Tibshirani (1996)	glmnet (R)



**Fig. 10** Running times in seconds. Left: Varying the number of samples  $n$  with a fixed number of dimensions  $d = 100$ . Right: Varying the number of dimensions  $d$  with a fixed number of samples  $n = 10,000$ . The cut-off time is shown using a dashed horizontal line at  $t = 1000$  s. Lower is better

RANSAC, where we use 20,000 trials and the same error threshold as for SLISE. In the case of SLISE, the parameter values are the same as above and can be found in Table 2.

### 7.1 Scalability

First, we investigate the scalability of the methods. Many of the methods have the same asymptotic time complexity,  $\mathcal{O}(nd^2)$ , but almost all are iterative methods and the number of iterations are not accounted for in the complexity. We empirically determine the running time on the SYNTHETIC datasets with (i) a fixed number of dimensions ( $d = 100$ ) with an increasing number of items, and (ii) a fixed number of items ( $n = 10,000$ ) with an increasing number of variables. The results are aggregated from ten different runs. We also limit the calculations to 1000 s. The results are shown in Fig. 10.

We observe that SLISE performs comparable to the other robust regression methods when the number of items increases (the left plot of Fig. 10). However, when the number of variables increases (the right plot of Fig. 10) SLISE is faster than most robust regression methods. The only exception is CONQUER which is almost as fast as LASSO.

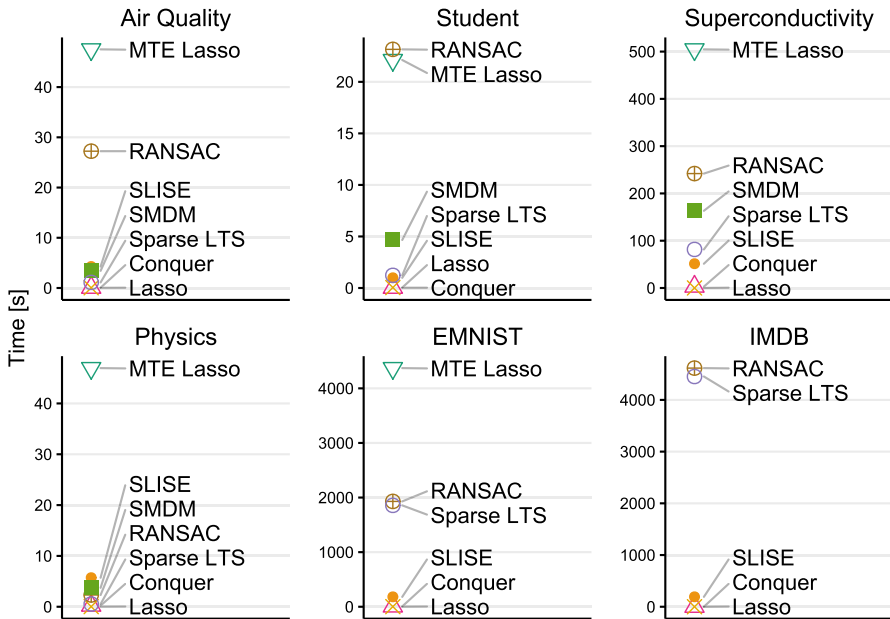


Fig. 11 Running times on real datasets. Lower is better

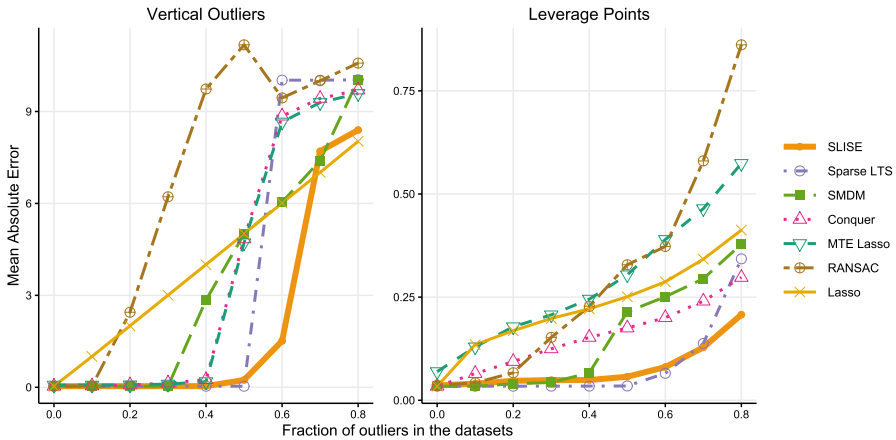
The scalability experiment only tests running times on one, SYNTHETIC, dataset. To get a broader perspective we also evaluate the running times on the real datasets. The results in Fig. 11 show that SLISE is comparable or only a couple of seconds slower than most methods on the small datasets. However, on the larger datasets, SUPERCONDUCTIVITY, IMDB, and EMNIST, SLISE is much faster than the other robust methods, except for CONQUER. On the larger datasets SLISE is actually faster than Fig. 10 would suggest, which demonstrates how the difficulty of any given dataset or task affects the running time.

## 7.2 Robustness

Next, we empirically compare the methods' robustness to outliers. To do this we corrupt datasets by replacing a fraction of the items with outliers. We utilise two types of outliers commonly found in literature (Rousseeuw and van Zomeren 1990; Alfons et al. 2013): *vertical outliers* and *leverage points*. The dataset we use is a variant of the SYNTHETIC dataset, where the  $Y$  values are from only one model (so that we can be sure that there are no inherent outliers).

All methods are trained on datasets corrupted by outliers and evaluated on the uncorrupted datasets. If a method is robust to a certain fraction of noise then the residuals for the uncorrupted data will be small. The results are shown in Fig. 12. The breakdown value is the point where the curves start trending upwards, and at high outlier fractions all methods are expected to break down.





**Fig. 12** Robustness to outliers. The  $x$ -axis shows the fraction of outliers and the  $y$ -axis the mean absolute error on the clean dataset. Consistently small residuals as the number of outliers increases indicate a robust method

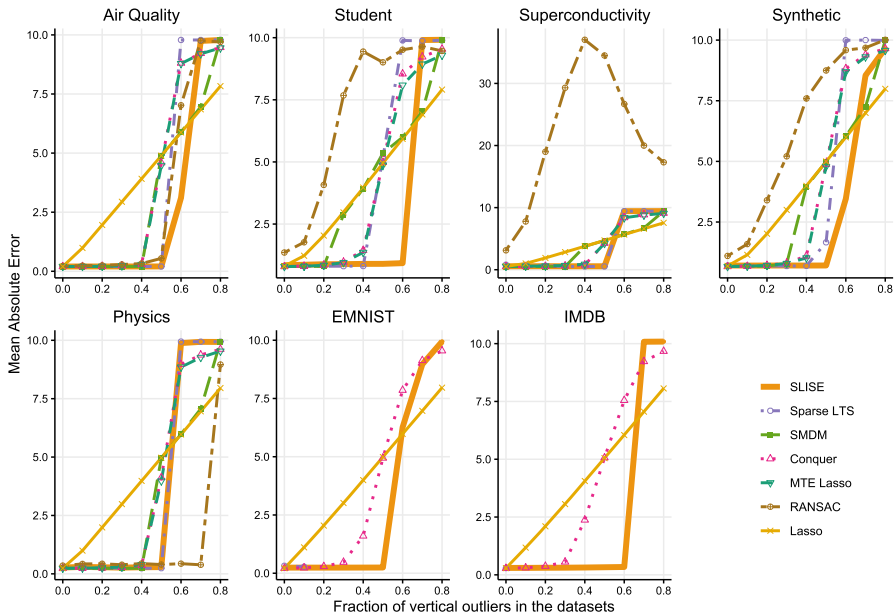
Vertical outliers are outliers where the target value is different from the rest. We create a vertical outlier by taking a non-outlier item  $i$  and replace  $y_i$  with  $y'_i \sim \mathcal{N}(10, 1)$ . As we can see in the left plot of Fig. 12, SLISE and SPARSE- LTS are the most robust ones. However, vertical outliers are generally considered to be an easier type of outliers (Alfons et al. 2013).

Leverage points are outliers where the variable values are unusual. Here we create a leverage point by taking a non-outlier item  $i$  and replace  $x_i$  with  $x'_i \sim \mathcal{N}(10, 1)$ . When the fraction of leverage points is high, most of the correlation between the predictors and target is broken, so regression methods tend to converge towards constant predictions, rather than breaking down. Nonetheless, in the right plot of Fig. 12 we can see that LASSO and MTE- LASSO break down immediately, while CONQUER and RANSAC follow soon thereafter. The absolute errors on the clean data from SMDM, SLISE, and SPARSE- LTS stay low for larger fraction of outliers indicating that they are more robust towards leverage points.

Since the robustness experiment is performed on a rather strict dataset we also consider the robustness to outliers on the real datasets. In Fig. 13 we can see how adding *vertical outliers* affects the behaviour of the regression methods on real datasets. RANSAC fares well on the low-dimensional datasets, PHYSICS and AIR QUALITY, but fails on even moderately sized datasets. This is because the chance of randomly finding a set of non-outliers shrinks exponentially with the number of dimensions (Fischler and Bolles 1981), so even the high number of trials (20,000) is not enough. On the contrary, SLISE *consistently* achieves a breakdown value of at least 0.5.

### 7.3 No outliers

Robust regression methods should also work in situations where there are no outliers. To evaluate this we perform 10-fold cross validation on the real datasets (with no added



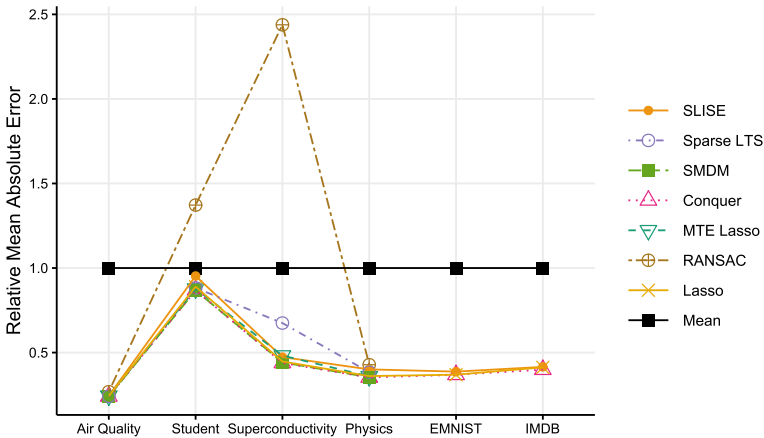
**Fig. 13** Robustness to outliers on real datasets. The  $x$ -axis shows the fraction of outliers and the  $y$ -axis the mean absolute error on the clean dataset. Some methods were not evaluated for the IMDB and EMNIST datasets, due to their time requirements. Lower error is better

outliers). As a baseline we include a dummy model that always predicts the mean  $y$ -value from the training data. In Fig. 14 we see that most robust regression methods (including SLISE) perform about as well as the non-robust LASSO, clearly better than the mean model, the exception being RANSAC on high-dimensional datasets.

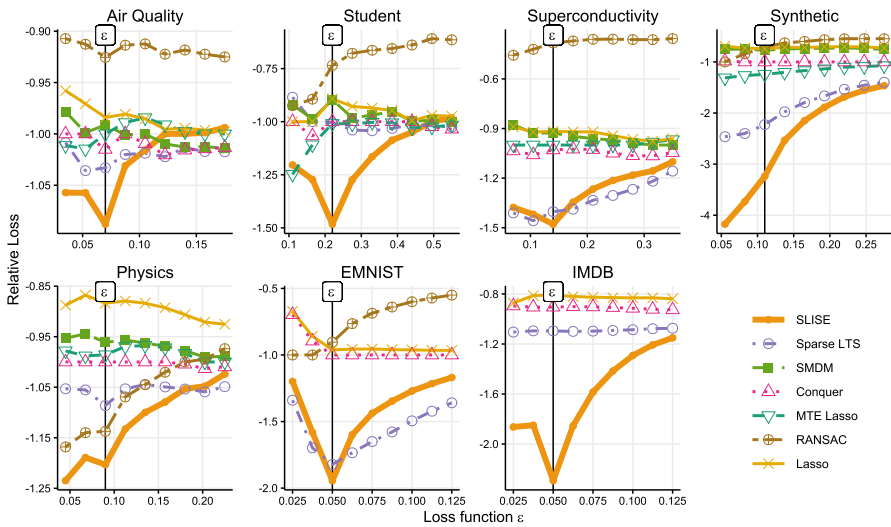
## 7.4 Optimality

Finally, we demonstrate that the solution found by SLISE optimises the loss of Eq. (1). The SLISE algorithm is designed to find the largest subset such that the residuals are upper-bounded by  $\varepsilon$ . To investigate if the model found using SLISE is optimal, we determine regression models (i.e., obtain the coefficient vectors  $\alpha$ ) using each algorithm. We then calculate the value of the loss-function in Eq. (1) for every model with varying values of  $\varepsilon$ .

The results are shown in Fig. 15. All loss-values have been normalised with respect to the absolute median for each value of  $\varepsilon$  and dataset. SLISE consistently reaches the smallest loss in the region around  $\varepsilon$  used for training, as expected. For SUPERCONDUCTIVITY and EMNIST the loss curves for SPARSE-LTS is very close to the curves for SLISE, but SLISE and SPARSE-LTS should actually give equally good, or even identical, results if the error tolerance  $\varepsilon$  in SLISE happens to match the subset size  $h$  in SPARSE-LTS.



**Fig. 14** Cross validation (10-fold) on the real datasets with no added outliers. SPARSE-LTS, SLISE, and RANSAC use subset sizes/error tolerances corresponding to 50% of the data. All values have been divided by the corresponding mean absolute error for the reference (mean) model. Some methods were not evaluated for the IMDB and EMNIST datasets, due to their time requirements. Lower error is better



**Fig. 15** Finding the best solution to Problem 1. The loss-values are normalised by dividing by the absolute median loss per dataset and  $\epsilon$  value. The  $\epsilon$  used for training SLISE and RANSAC is marked with a vertical line. Some methods were not evaluated for the IMDB and EMNIST datasets, due to their time requirements. Lower values are better

### 8 Conclusions

This paper refines the SLISE algorithm for robust regression. SLISE introduces a novel way of detecting and discarding outliers; find a subset of non-outliers where the error is less than an adjustable error tolerance ( $\epsilon$ ), for fitting the regression model. This flexible subset size (based on  $\epsilon$ ) is in contrast to other methods (primarily the LTS

family) where the subset size is fixed. Additionally, SLISE yields sparse solutions through built-in regularisation.

Although finding an exact solution to the problem definition (Problem 1) is **NP**-hard (see Sect. 2.1), the combination of graduated optimisation with a quasi-Newton optimiser (OWL- QN ) yields an effective approximation (see Sects. 3 and 7). Adding a stochastic initialisation further mitigates the risk of unfortunate starting conditions (see Sect. 6.3).

When comparing to other robust regression methods, SLISE is able to achieve robustness levels that are among the best possible, which we show both theoretically (see Sect. 2.2) and empirically (see Sect. 7.2). Furthermore, SLISE scales better to high-dimensional data than many alternative methods (see Sects. 4.2 and 7.1).

Future work could investigate how the choice of  $\lambda$  affects the sparsity, especially during the graduated optimisation. Another direction would be to try changing the balance between maximising the subset and minimising the residuals, or to introduce different weights for the data items.

In an earlier paper (Björklund et al. 2019) we show how SLISE can be used to explain outcomes from black box models in a way that respects constraints in the data. Along this line we could further investigate the utility of selecting the data used for the explanations in order to answer specific questions in an interactive manner. This would give better insight into the learned models and their behaviour.

The explanations given by SLISE are local, i.e. for specific outcomes, and an interesting follow up would be to combine these local explanations into one global explanation. Furthermore, SLISE is a robust regression method and, therefore, quite generic, which means that it can readily be integrated into, or combined with, other explanation methods.

Our implementation of the SLISE algorithm is released under an open source license. It is available in both Python (Björklund 2021) and R (Björklund et al. 2021), which also includes the code for running all the experiments.

**Acknowledgements** Supported by Academy of Finland (decisions 326280 and 326339). Supported by the Doctoral Programme in Computer Science at University of Helsinki. Computational resources provided by Finnish Grid and Cloud Infrastructure (FGCI).

**Funding** Open Access funding provided by University of Helsinki including Helsinki University Central Hospital.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

Alfons A, Croux C, Gelper S (2013) Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *Ann Appl Stat* 7(1):226–248. <https://doi.org/10.1214/12-AOAS575>

- Amaldi E, Kann V (1995) The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theor Comput Sci* 147(1):181–210. [https://doi.org/10.1016/0304-3975\(94\)00254-G](https://doi.org/10.1016/0304-3975(94)00254-G)
- Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M (1999) Complexity and approximation: combinatorial optimization problems and their approximability properties, 2nd edn. Springer, Berlin. <https://doi.org/10.1007/978-3-642-58412-1>
- Barath D, Matas J (2018) Graph-cut RANSAC. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). <https://arxiv.org/abs/1706.00984v2>
- Barath D, Noskova J, Ivashchkin M, Matas J (2020) Magsac++, a fast, reliable and accurate robust estimator. In: Proceedings of the IEEE/VF conference on computer vision and pattern recognition (CVPR). <https://arxiv.org/abs/1912.05909>
- Björklund A (2021) SLISE—sparse linear subset explanations (Python version). <https://github.com/edahelsinki/pyslise>
- Björklund A, Henelius A, Oikarinen E, Kallonen K, Puolamäki K (2019) Sparse robust regression for explaining classifiers. In: *Discovery science*. Springer, Berlin, pp 351–366. [https://doi.org/10.1007/978-3-030-33778-0\\_27](https://doi.org/10.1007/978-3-030-33778-0_27)
- Björklund A, Puolamäki K, Henelius A (2021) SLISE—sparse linear subset explanations (R version). <https://github.com/edahelsinki/slise>
- Cohen G, Afshar S, Tapson J, van Schaik A (2017) EMNIST: an extension of MNIST to handwritten letters. [arXiv:170205373](https://arxiv.org/abs/170205373) <https://arxiv.org/abs/1702.05373>
- Cortez P, Silva AMG (2008) Using data mining to predict secondary school student performance. In: Proceedings of 5th Future Business Technology Conference (FUBUTEC 2008)
- De Vito S, Massera E, Piga M, Martinotto L, Di Francia G (2008) On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sens Actuators B Chem* 129(2):750–757. <https://doi.org/10.1016/j.snb.2007.09.060>
- Donoho DL, Huber PJ (1983) The notion of breakdown point. A festschrift for Erich L Lehmann, pp 157–184
- Dua D, Graff C (2019) UCI machine learning repository. <http://archive.ics.uci.edu/ml>
- Fernandes M, Guerre E, Horta E (2021) Smoothing quantile regressions. *J Bus Econ Stat* 39(1):338–357. <https://doi.org/10.1080/07350015.2019.1660177>
- FGCI (2021) Finnish grid and cloud infrastructure. [Urn:nbn.fi:research-infras-2016072533](https://nbn.fi/research-infras-2016072533)
- Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24(6):381–395. <https://doi.org/10.1145/358669.358692>
- Giloni A, Simonoff JS, Sengupta B (2006) Robust weighted lad regression. *Comput Stat Data Anal* 50(11):3124–3140. <https://doi.org/10.1016/j.csda.2005.06.005>
- Hamidieh K (2018) A data-driven statistical model for predicting the critical temperature of a superconductor. *Comput Mater Sci* 154:346–354. <https://doi.org/10.1016/j.commatsci.2018.07.052>
- HIP CMS Experiment (2019) Helsinki OpenData Tuples. <https://hot.hip.fi/>
- Huber PJ (1964) Robust estimation of a location parameter. *Ann Math Stat* 35(1):73–101. <https://doi.org/10.1214/aoms/1177703732>
- Hubert M, Debruyne M (2009) Breakdown value. *Wiley Interdiscip Rev Comput Stat* 1(3):296–302. <https://doi.org/10.1002/wics.34>
- Koenker R, Hallock KF (2001) Quantile regression. *J Econ Perspect* 15(4):143–156. <https://doi.org/10.1257/jep.15.4.143>
- Koller M, Stahel WA (2017) Nonsingular subsampling for regression estimators with categorical predictors. *Comput Stat* 32(2):631–646. <https://doi.org/10.1007/s00180-016-0679-x>
- Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, pp 142–150. <http://www.aclweb.org/anthology/P11-1015>
- Microsoft and R Core Team (2019) Microsoft R Open. <https://mran.microsoft.com/>
- Mobahi H, Fisher JW (2015) On the link between gaussian homotopy continuation and convex envelopes. In: *Energy minimization methods in computer vision and pattern recognition*. Springer, pp 43–56
- Qin Y, Li S, Li Y, Yu Y (2017) Penalized maximum tangent likelihood estimation and robust variable selection. [arXiv:170805439](https://arxiv.org/abs/170805439) <http://arxiv.org/abs/1708.05439>
- Ribeiro MT, Singh S, Guestrin C (2016) Why should I trust you?: Explaining the predictions of any classifier. In: *SIGKDD*, pp 1135–1144

- Rousseeuw PJ (1984) Least median of squares regression. *J Am Stat Assoc* 79(388):871–880. <https://doi.org/10.1080/01621459.1984.10477105>
- Rousseeuw PJ, Hubert M (2011) Robust statistics for outlier detection. *Wiley Interdiscip Rev Data Min Knowl Discov* 1(1):73–79. <https://doi.org/10.1002/widm.2>
- Rousseeuw P, Yohai V (1984) Robust regression by means of S-estimators, vol 26. Springer, New York, pp 256–272. [https://doi.org/10.1007/978-1-4615-7821-5\\_15](https://doi.org/10.1007/978-1-4615-7821-5_15)
- Rousseeuw PJ, Van Driessen K (2000) An algorithm for positive-breakdown regression based on concentration steps. In: *Data analysis*. Springer, pp 335–346. [https://doi.org/10.1007/978-3-642-58250-9\\_27](https://doi.org/10.1007/978-3-642-58250-9_27)
- Rousseeuw PJ, van Zomeren BC (1990) Unmasking multivariate outliers and leverage points. *J Am Stat Assoc* 85(411):633–639. <https://doi.org/10.1080/01621459.1990.10474920>
- Schmidt M, Berg E, Friedlander M, Murphy K (2009) Optimizing costly functions with simple constraints: a limited-memory projected quasi-newton algorithm. In: *Artificial intelligence and statistics*, vol 5, pp 456–463. <http://proceedings.mlr.press/v5/schmidt09a.html>
- Smucler E, Yohai VJ (2017) Robust and sparse estimators for linear regression models. *Comput Stat Data Anal* 111:116–130. <https://doi.org/10.1016/j.csda.2017.02.002>
- Tibshirani R (1996) Regression shrinkage and selection via the Lasso. *J R Stat Soc Ser B (Methodol)* 58(1):267–288. <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>
- Wang H, Li G, Jiang G (2007) Robust regression shrinkage and consistent variable selection through the LAD-Lasso. *J Bus Econ Stat* 25(3):347–355. <https://doi.org/10.1198/073500106000000251>
- Yohai VJ (1987) High breakdown-point and high efficiency robust estimates for regression. *Ann Stat* 15(2):642–656. <https://doi.org/10.1214/aos/1176350366>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.