Abstract

Two of the most common methods for robust regression are least trimmed squares (LTS) and least median of squares (LMS) regression. Both of these methods require sorting the squared residuals. Because sorting is not a differentiable operation, end-to-end optimalisation with gradient-based methods is not stable. Furthermore, existing algorithms for estimating LTS and LMS regressors rely on multiple random initial starting points. We propose and investigate two potential improvements to LTS and LMS: (1) the use of soft differentiable sorting in the loss functions and (2) deterministic initialisation of the estimators using the wrapping transformation. The first improvement aims to tackle the drawbacks of using hard sorting by introducing an alternative loss function that can be optimised using gradient based optimisation schemes, while the latter improvement aims to remove the need for multiple random initial starting points, leading to both improved accuracy as well as faster convergence. We perform an extensive experiment on both simulated and real world datasets and compare the performance of our introduced methods with well known baseline methods for robust regression. We show that the deterministic initialisation has significant benefits for LTS and LMS, both for predictive accuracy and for computational speed. The soft loss function mostly benefits LMS, as it makes it possible to apply iterative optimisation schemes to the LMS loss function. We also demonstrate the potential application of the Soft LTS loss function to non-linear regression problems using neural networks.

Smart initialisation and approximating loss function for robust regression

Thomas Servotte[1, 4], Jakob Raymaekers[2, 3], and Tim Verdonck[1, 2]

[1]Department of Mathematics, University of Antwerp
[2]Department of Mathematics, KU Leuven
[3]Department of Quantitative Economics, University of Maastricht
[4]Silverfin

1 Introduction

Regression is one of the most common problems in statistics and machine learning. Many techniques have been developed to find a relation between a set of (continuous) predictor variables and a continuous target variable, ranging from the well known Ordinary Least Squares (OLS) regression for fitting a straight line to complex deep neural networks that can fit any arbitrary (non-linear) function [11]. Unfortunately, most of these established techniques suffer from a sensitivity to anomalies or outliers in the data.

Robust statistics is a sub-discipline of statistics that tries to address this sensitivity to outliers. Many methods have been developed to perform robust linear regression, see for example [1, 24, 31, 16]. Some of the most popular approaches are M-estimators, S-estimators, and MM-estimators. M-estimators are generalisations of the maximum likelihood estimator and use the following objective:

$$\hat{\boldsymbol{\beta}}_M = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \rho\left(\frac{r_i(\boldsymbol{\beta})}{\hat{\sigma}}\right)$$

with $r_i(\boldsymbol{\beta})$ the residual of the ith observation, $\hat{\sigma}$ a (robust) preliminary scale estimate and $\rho : \mathbb{R} \to \mathbb{R}$. E.g. if $\rho(t) = t^2$ we obtain least squares regression and if $\rho(t) = |t|$ we obtain least absolute deviation regression. The most well-known M-estimator is probably Huber regression [18]. S-estimators [22] are based on (robust) estimators of scale and use the objective

$$\hat{\boldsymbol{\beta}}_S = \underset{\beta}{\operatorname{argmin}}\, \hat{\sigma}(r_i(\boldsymbol{\beta}))$$

where $\hat{\sigma}(r_i(\boldsymbol{\beta}))$ is a robust scale such as the M-estimator of scale defined as the solution to

$$\frac{1}{n}\sum_{i=1}^{n} \rho\left(\frac{r_i(\boldsymbol{\beta})}{\hat{\sigma}(\boldsymbol{\beta})}\right) = \delta$$

with ρ a symmetric and continuously differentiable function and $\rho(0) = 0$. MM-estimators [32] combine S- and M-estimators. An initial regression is estimated with an S-estimator. Subsequently a regression M-estimator is computed using the fixed scale estimate $\hat{\sigma}_S$ and initial estimate $\hat{\boldsymbol{\beta}}_S$ from the first step. This results in an estimate with a high breakdown value (as a result of using the S-estimator results) and high efficiency (property of M-estimators). Two popular special cases of S-estimators, where the robust scale is based on the order statistics of the residuals, are the Least Trimmed Squares (LTS) and Least Median Squares (LMS) [23]. LTS considers the mean of the h smallest squared residuals and LMS considers the median squared residual as loss function.

A drawback of the LTS and LMS estimators is that they require sorting the residuals, which is a piece-wise linear function and gradient based optimisation is thus mostly unstable. As a result, most algorithms that have been developed for these estimators are iterative and less efficient than those for M-estimators.

Some researchers have tried to define approximate sorting operators that are smooth and thus differentiable everywhere, which allows the use of sorting in end-to-end optimisations. [5] define a differentiable proxy for sorting and ranking by translating the operator to an optimal transport problem with entropic regularisation. [2] propose differentiable sorting and ranking operators that achieve $O(nlogn)$ time and $O(n)$ space complexity. They do this by defining the operators as projections on the permutahedron which are then reduced to isotonic optimisation. The authors also propose the application of their soft sorting algorithm to robust regression. In particular, they demonstrate its potential by defining a soft version of LTS regression and apply it to a set of real datasets with artificial outliers.

The idea of approximating the (robust) loss function in order to improve computational performance has also been explored in [10]. In this work, the authors propose an approximate optimisation scheme for robust M-regression losses and demonstrate that this approach can lead to improved robustness at a desirable computational cost. However, in general, literature on the use of approximating loss function for robust statistics is very scarce.

In this paper, we build on the ideas proposed by [2]. We introduce soft sorting for LMS, yielding a differentiable objective function approximating the classical LMS loss. Furthermore, we design a more extensive experiment on simulated data as well as real data with artificial outliers to properly evaluate soft LMS and soft LTS and compare them with their benchmarks. Finally, we consider the application of soft sorting for non-linear regression by using the Soft LTS loss as objective for a multi-layer perceptron (MLP) trained on a simulated dataset. In [15], the 'hard' LTS objective was used to train neural networks with an iteratively reweighted algorithm. Using the Soft LTS objective, we can train the network end-to-end by directly passing the Soft LTS loss to the gradient based optimiser (e.g. Adam [14]).

Another drawback of the algorithms used to the LTS and LMS estimators (and many robust estimators in general), is that they often rely on random starting points. E.g. FAST LTS [25] uses a certain number of random subsets of the data to determine multiple starting points. Random starting points are very common in robust statistics, and the use of deterministic alternatives has already been proposed for some common robust estimation techniques, especially in the context of location and covariance estimation. DetMCD and its extensions

([13, 3, 7]) rely on the use of a small number of estimators that yield determinstic starting points as opposed to multiple random starting points in the FastMCD algorithm ([21]). Similarly, DetS ([12]) is a deterministic alternative for FastS ([27]), an iterative algorithm for fitting an S-estimator. In this paper we propose to mitigate the problem of random starting points for the Fast LTS and LMS by starting from weights that are initialised by first transforming the data and then fitting a classical regression estimator. We transform both the response and the predictors using wrapping, which was proposed in [19] as a transformation of the data that allows fast and robust estimation of the covariance matrix for high dimensional datasets. By using a single starting value, we can greatly improve on the computational speed of the algorithms based on random subsampling.

Our contributions can be summarised as follows: (1) We perform an extensive experiment to investigate the impact of soft sorting on robust regression, more specifically its potential application to LTS and LMS, (2) we analyse the impact of smartly initialising the weights in LTS and LMS by using the wrapped covariance [19] and (3) we investigate the potential generalisation of Soft LTS to non-linear regression.

Throughout the paper we will consider a linear regression model where the expected conditional response $E[Y|\boldsymbol{X}]$ is modeled by a linear combination of a p-variate random vector $\boldsymbol{X}$: $E[Y|\boldsymbol{X}] = \alpha + \boldsymbol{X}\boldsymbol{\beta}$. We assume that we observe a sample $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ of size n, where each $\boldsymbol{x}_j$ is an observed p-variate vector. We denote the residual for observation i by $r_i(\boldsymbol{\beta}) = y_i - \boldsymbol{x}_i\boldsymbol{\beta}$, and the i-th smallest value residual (i.e. the i-th order statistic) by $r_{[i]}$. We omit the intercept here for notational simplicity, but it can be added by including a constant in the predictor matrix.

In section 2 we give more background information on the techniques used in this paper, i.e. LMS, LTS, soft sorting and wrapping. In section 3 we outline how we implement soft versions of LMS and LTS as well as our approach to wrapping initialisation. Section 4 outlines our experimental set-up, including a description of the datasets and section 5 presents the results from the experiments. Finally, we discuss our conclusions and suggestions for further research in section 6.

2 Preliminaries

2.1 Least Median Squares

Least Median of Squares (LMS) regression was first proposed in [23]. Instead of using the sum of squared residuals as the loss function, LMS tries to minimize the median of the squared residuals, i.e.:

$$\hat{\boldsymbol{\beta}}_{\text{LMS}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, r^2_{[\lceil \frac{n}{2} \rceil]}(\boldsymbol{\beta})$$

It has been shown that LMS has a breakdown value of 50%, but low statistical efficiency (converging at rate $n^{-1/3}$). However, the main drawback of this method is the lack of efficient algorithms [29] [24]. The main approach to fit an LMS regression is to select N (e.g. 3000) random subsets of size $p+1$ where p is the number of features in the dataset, fit an OLS regression on each of these subsets and select the weights of the model with the lowest median squared residual on the full dataset.

Despite some of its disadvantages, in practice LMS is still often used as an initial estimator for more efficient estimators and as a data-analytic tool because of its high breakdown value and optimal robustness properties [17].

2.2 Least Trimmed Squares

Similar to LMS, Least Trimmed Squares (LTS) regression also does not consider all observations in the loss function. Instead, it considers the sum of the h observations with the smallest residuals:

$$\hat{\boldsymbol{\beta}}_{\text{LTS}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{h} r_{[i]}^2(\boldsymbol{\beta}).$$

The most popular algorithm to compute the LTS coefficients is FAST LTS [25, 26]. The idea is to use so-called *concentration-steps (C-steps)* on a number of random starting points until convergence. A *C-step* consists of selecting the h-subset of observations with the smallest residuals and fitting an OLS regression on this subset. Applying *C-steps* iteratively guarantees a monotone decrease in the objective function. In FAST LTS, random starting points are obtained by selecting random subsets of size $p+1$ (where p is the dimensionality of the dataset). A predetermined set of C-steps is applied to each of these random initial subsets, after which the n best models are selected and further *C-steps* are performed until convergence. Finally, the best model (with the lowest LTS train loss) is selected.

2.3 Soft sorting

Many robust algorithms use sorting and ranking. The drawback of these methods is that these operations are not smooth functions, making it harder to optimize loss functions that depend on them. [2] proposed a new approximating algorithm for sorting and ranking. In this paper we focus on soft sorting, as both LTS and LMS rely on the sorting operator.

The soft sorting operator proposed in [2] relies on projections onto the permutahedron, the convex hull of permutations, and using a reduction to isotonic optimization.

Specifically, soft sorting is defined as

$$s_{\epsilon\Psi} = P_\Psi(\rho/\epsilon, \theta)$$

where P_Ψ is a regularised projection onto the permutahedron, ρ is the reversing permutation ($\rho := (n, n-1, ..., 1)$), $\epsilon > 0$ is a regularisation parameter and θ is the score vector we want to sort. The authors propose 2 alternative forms of regularisation Ψ, quadratic and entropic. In this paper we only consider quadratic regularisation, i.e.

$$P_\Psi(\rho, \theta) = P_Q(\rho, \theta) = \underset{\mu \in P(\theta)}{\operatorname{argmin}} \|\mu - \rho\|^2$$

The parameter ϵ acts as interpolator between 'hard' sorting and 'soft' sorting. E.g. for the application of soft sorting to LTS, $\epsilon \to 0$ can be interpreted as regular LTS (with a hard objective) and $\epsilon \to \infty$ results in regular least squares regression (as the sorted scores will just approximate the mean of all scores).

The authors empirically show that their approach is significantly faster than previously existing approaches (achieving $O(nlogn)$ time and $O(n)$ space complexity).

2.4 Wrapping

The wrapping transformation was initially proposed in [9] as the optimal transformation of the data for robust location estimation. This transformation was dubbed 'wrapping' in [19], where it was used for efficient estimation of robust correlations. The transformation has the following form:

$$\Psi_{b,c}(z_w) = \begin{cases} z & if\ 0 \leq |z| < b \\ q_1 \tanh\left(q_2(c - |z|)\right) \operatorname{sign}(z) & if\ b \leq |z| \leq c \\ 0 & if\ c < |z| \end{cases}$$

z is the standardized variable ($z = (x - m_x)/s_x$), using a robust location (m_x) and scale (s_x) estimate (e.g. median and median absolute deviation). The cutoff values b and c are usually set to default values 1.5 and 4. The parameter b is similar to the corner value in the Huber function, as the influence of values more than 1.5 standard deviations away is limited. The parameters q_1 and q_2 can be derived by requiring continuity of $\Psi_{b,c}$ and optimal V-robustness for the normal distribution. Since we set $b = 1.5$ and $c = 4$, the optimal values for q_1 and q_2 are 1.540793 and 0.8622731 respectively, as shown in [19]. Figure 1 shows the wrapping function $\Psi_{b,c}$.

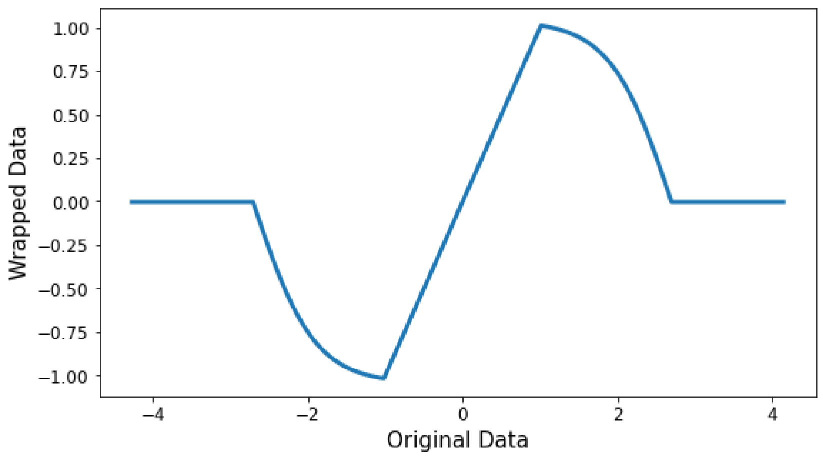

Figure 1: Wrapping function $\Psi_{b,c}$

Using the wrapping function, the 'wrapped' variable is finally constructed as $x_w = z_w s_x + m_x$, with m_x and s_x the original robust location and scale estimates. These transformed variables can then be used as input to a classical method, such as the estimation of covariance matrices. The resulting covariance matrix estimate has been found to be biased, mainly because the marginal distributions of the variables have been squashed. This can be partly mitigated by re-scaling the data, i.e. $x_w = z_w \frac{s_x}{s_{z_w}} + m_x - m_{z_w} \frac{s_x}{s_{z_w}}$ where m_{z_w} and s_{z_w} are the (non-robust) mean and standard deviation of z_w. The resulting (non-robust) location and scale of the wrapped variables are the same as the initial robust location and scale estimates (m_x and s_z).

3 Methodology

This section is structured as follows: in section 3.1 we elaborate in the implementation of soft version for LTS and LMS. In section 3.2 the use of wrapping for smart initialisation is discussed and section 3.3 shows how we apply Soft LTS to non-linear regression.

3.1 Soft LTS and Soft LMS

To investigate the impact of using soft sorting, we implemented Soft LTS and Soft LMS. Both algorithms are very similar and only differ in the loss function. Soft LTS uses the soft version of the LTS loss, i.e.

$$\hat{\boldsymbol{\beta}}_{\text{sLTS}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{h} s_{\epsilon\Psi}(r^2(\boldsymbol{\beta}))_{[i]}.$$

and Soft LMS uses the soft version of the LMS loss, i.e.

$$\hat{\boldsymbol{\beta}}_{\text{sLMS}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, s_{\epsilon\Psi}(r^2(\boldsymbol{\beta}))_{[\lceil \frac{n}{2} \rceil]}$$

where $\boldsymbol{\beta}$ are the regression parameters (intercept and coefficients), n is the size of the dataset, h is the size of the subset (often set to $0.5n$) and $s_{\epsilon\Psi}$ is the soft sort operator with regularisation type Ψ [2]. We use quadratic regularisation in both cases (i.e. $\Psi = Q$) and fix the soft sort regularisation parameter ϵ to $1e-4$ (experimentation with a range of values for $epsilon$ indicated that this is a reasonable default value). The algorithms start with a number of random subsets with size $n\_features + 1$. Initial regression weights are estimated with OLS on these subsets. For each initial subset, the weights are optimised with the L-BFGS-B algorithm (as implemented in the Scipy package [1] [4]) for a fixed number of iterations (10 in our experiments). Subsequently, the k subsets ($k = 10$ in our experiments) with the lowest hard loss (i.e. regular LTS or LMS loss) are selected and are further optimised with L-BFGS-B until convergence or a maximum of 300 iterations is reached.

We considered alternative gradient-based optimization algorithms for the soft loss such as Adam [14], but because there was no significant difference in performance with the L-BFGS-B algorithm, we do not report results with these alternative optimisers.

[1]https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html

3.2 Wrapping initialisation

Most algorithms for LTS, LMS and other robust regressors require starting values on which an iterative algorithm is applied. The choice of starting value is very important, as it can influence both the speed as well as the quality of convergence. Random initialisation is most common, but has a significant drawback: to mitigate sensitivity to noise and outliers, many random initialisations are needed, which causes a significant computational cost. Furthermore, there is no trivial way to determine a minimum number of random starting points to ensure decent results.

In this paper, we use the wrapping transformation to derive initialisation weights for LTS and LMS estimators. This can be done in 2 ways: Xy-wrapping or X-wrapping. For Xy-wrapping, the features X and targets y are first concatenated. Wrapping is then applied to the concatenated data. Subsequently, the subset of the (concatenated) data with the h smallest Mahalanobis distances to the mean and location of the wrapped data is selected. Finally, a regular OLS estimator is fitted on the selected subset. X-wrapping is very similar, but wrapping is only applied to the features. Again, a subset of the data is selected with the h smallest Mahalanobis distances to the mean and covariance of the transformed data. Since wrapping is only applied to the features, a Huber regressor is fitted on this subset instead of a regular OLS.

Figure 2 shows the application of Xy-wrapping regression to simulated data with 1 feature sampled from a univariate standard gaussian and 20% outliers. Even though wrapping regression does not result in a perfect fit to the clean data, it is significantly better than simple OLS. Combined with the fact that the wrapping transformation has low computational cost, wrapping regression proves to be a suitable initialisation method for robust regression.

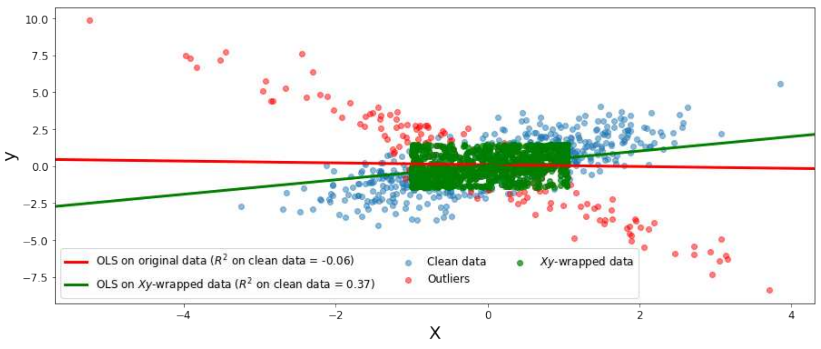

Figure 2: Example of wrapping regression on simulated data

We only report results for Xy-wrapping as in most cases this proved to be more robust than X wrapping.

3.3 Non-linear Soft LTS

Robust non-linear regression is very challenging, and many of the ideas that work for linear regression cannot be directly translated to the non-linear setting. Using a LTS- or LMS-type loss function for non-linear regression is possible, but the resulting optimization problem is even more sensitive to starting values. Using many random initialisations, which is common for linear robust regressors, also becomes more difficult as training non-linear regression models is typically much more costly. Furthermore, as extrapolation outside of the range of observed values is very unreliable for non-linear regression (as opposed to linear regression), ignoring part of the data during fitting can result in models that perform very poorly at the edges of the data range [20].

While [15] empirically showed that traditional C-steps can give decent results for non-linear neural network regression, there is no theoretical basis to guarantee convergence (as opposed to linear regression). Furthermore, as neural networks are very sensitive to local minima, using traditional C-steps very likely results in sub-optimal results. We apply the Soft LTS objective to non-linear regression modeled by a neural network. The differentiable loss function allows us to leverage the power of deep learning to fit any non-linear function in an end-to-end way, without relying on C-steps.

4 Experiment

4.1 Data

4.1.1 Linear regression

In order to evaluate our proposals, we use both simulated datasets as well as real datasets from the UCI Machine Learning Repository and StatLib[2] [8].

For the simulated data, we consider 2 settings.

In the first setting, we generate the features $\boldsymbol{X}$ from a 10-dimensional multivariate standard normal distribution with $\mathbf{0}$ mean and unit covariance matrix. The target $\boldsymbol{y}$ is then calculated as follows:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Where $\boldsymbol{\beta} = \mathbf{1}_p$ is the vector of clean regression coefficients and $\boldsymbol{\epsilon}$ is a vector of length n sampled from normal distribution with 0 mean and standard deviation of 1. To add leverage points, we simply multiply $\boldsymbol{X}$ with a factor $d_x > 1$. To add vertical outliers, we calculate the targets as $\boldsymbol{y} = d_y\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ with $d_y \leq -1$. In our experiments, we set $d_y = -5$ and $d_x = 5$. We refer to this setting as simulated data with uncorrelated features.

The second setting is similar to what the authors in [6] refer to as the *A09* type simulation. We generate $\boldsymbol{X}$ from a 10-dimensional multivariate normal distribution with $\mathbf{0}$ mean and a covariance matrix Σ where the elements are equal to $\rho^{|i-j|}$, with ρ a constant between $[-1, 1]$, and i, j are row and column indices. Following [6], we set $\rho = -0.9$ in our experiments. $\boldsymbol{y}$ is then calculated the same way as the setting with uncorrelated features described above, but with $\boldsymbol{\epsilon}$ sampled from a normal distribution with 0 mean and a standard deviation of 0.1 instead of 1. We chose a lower standard deviation for this simulation as the signal-to-noise ratio would otherwise be too small given the more difficult simulation approach with correlated features. Leverage points are generated

[2]http://lib.stat.cmu.edu/datasets/

from a new multivariate standard normal distribution, with the same covariance matrix as the clean data, but a contaminated location vector that is determined as follows:

$$\boldsymbol{\mu}_{cont} = \frac{\boldsymbol{v} d_x \sqrt{\chi^2_{m-1,0.975}}}{\sqrt{(\boldsymbol{v} - \boldsymbol{\mu}_{\boldsymbol{clean}})^\top \Sigma^{-1} (\boldsymbol{v} - \boldsymbol{\mu}_{\boldsymbol{clean}})}}$$

$\boldsymbol{v}$ is the smallest eigenvector of Σ, m is the number of features and d_x is a factor greater than or equal to 1. The covariance matrix can be multiplied by a factor to either spread the outliers further apart or centralise them. In our experiment we put all leverage points at a single location $\boldsymbol{\mu}_{cont}$, i.e. we multiply the covariance by 0. This specific way of generating outliers in the predictor space is motivated by the fact that these types of outliers generate the strongest bias on the classical covariance matrix of the predictors. Vertical outliers are created by shifting the clean targets with a factor d_y. In our experiments we set $d_x = 5$ and $d_y = -5$.

For both simulations, we sample 1000 observations, (800 for training, 200 for testing). The added outliers are always split equally over bad leverage points and vertical outliers (i.e. if 20% outliers are added, this consists of 10% vertical outliers and 10% bad leverage points).

For the real data sets, we consider the following datasets from StatLib and the UCI Machine Learning repository: Bodyfat[3], Computer Hardware[4], Combined Cycle Power Plant (CCPP) Data Set[5][30] and Average Localization Error[6][28].

Table 1: Overview of used datasets.

Dataset	**# observations**	**# features**
Bodyfat	252	14
Computer Hardware	209	6
CCPP	9568	4
Average Localization Error	107	4

For each of these datasets, we first split the clean data in a train and test set, 80% of the data is used for training, 20% is used for testing. Subsequently we add vertical outliers and bad leverage points only to the training set. Note that this procedure implicitly assumes that the original data only contains clean samples. As the selected datasets are well-known benchmark datasets, extensively used to illustrate non-robust algorithms, we believe this is a safe assumption.

We generate vertical outliers by simply adding a factor d_y to the clean targets. Leverage points are created by adding a factor d_x in the direction of the smallest eigenvector of the empirical covariance matrix of the clean data ($\boldsymbol{X}_{\boldsymbol{cont}} = \boldsymbol{X}_{\boldsymbol{clean}} + \boldsymbol{v} d_x$, with $\boldsymbol{v}$ the smallest eigenvector of $cov(\boldsymbol{X})$). In our

[3]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/regression/bodyfat
[4]https://archive.ics.uci.edu/ml/datasets/Computer+Hardware
[5]https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
[6]https://archive.ics.uci.edu/ml/datasets/Average+Localization+Error+%28ALE%29+in+sensor+node+localization+proce

experiments we set $d_y = 10$ and $d_x = 5$ Here as well, we split the outliers equally over vertical outliers and bad leverage points.

4.1.2 Non-linear regression

A key attribute of the Soft loss functions is that we can more easily generalise its application to non-linear data. To demonstrate this, we generate a simple dataset with two features of size 1000. The features $\boldsymbol{X}$ are generated by sampling from a 2-dimensional gaussian with $\mathbf{0}$ mean and covariance $\Sigma = \left(\begin{smallmatrix} 1 & 0.9 \\ 0.9 & 1 \end{smallmatrix}\right)$. We add 20% outliers in the direction of smallest eigenvector of the clean covariance matrix by sampling from a 2-dimensional gaussian distribution with the same covariance matrix as the clean data, but the following mean vector:

$$\boldsymbol{\mu_{cont}} = \frac{\boldsymbol{v}\sqrt{\chi^2_{1,0.975}}}{\sqrt{\boldsymbol{v}^T\Sigma^{-1}\boldsymbol{v}}}$$

with $\boldsymbol{v}$ the eigenvector of Σ with the smallest eigenvalue. $\boldsymbol{y}$ is then calculated as follows:

$$\boldsymbol{y_{clean}} = \sin\left(1 + \boldsymbol{X_{1,clean}} + \boldsymbol{X_{2,clean}}\right) + \boldsymbol{\epsilon}$$

$$\boldsymbol{y_{cont}} = 10 + \sin\left(1 + \boldsymbol{X_{1,cont}} + \boldsymbol{X_{2,cont}}\right) + \boldsymbol{\epsilon}$$

with $\boldsymbol{\epsilon}$ noise sampled from a univariate gaussian with 0 mean and a standard deviation of 0.1. Figure 3 shows what the generated data looks like.

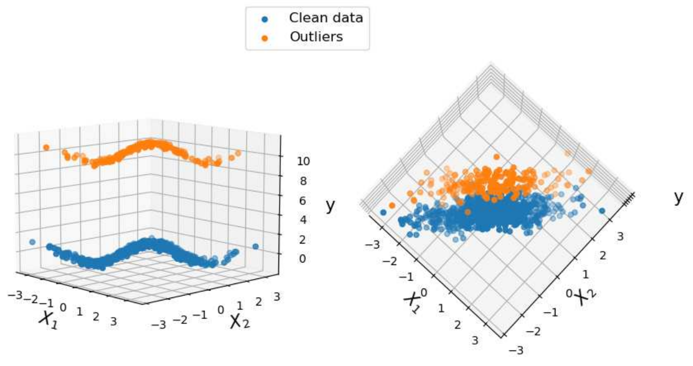

Figure 3: Simulated data for non-linear regression

4.2 Set-up

4.2.1 Linear regression

We consider the following algorithms as baseline: simple OLS regression, default FAST LTS (i.e. 500 initial subsets, 2 initial C-steps and 10 best models that are fit until convergence, with a tolerance of $1e-15$), FAST LTS with a single (random) subset that is fit until convergence and default classic LMS (i.e. 3000 subsets of which the one with the lowest loss function is selected). We have also included the MM-estimator ([32]) with default settings as implemented in the *lmrob* function of the R-package *robustbase*[7].

These baseline models are compared to Soft LTS with 20 random starts, Soft LTS with a single random start, Soft LTS with a single start using Xy-wrapping initialisation, Fast LTS with a single start using Xy-wrapping initialisation, Soft LMS with 20 random starts, Soft LMS with a single random start and Soft LMS with a single start using Xy-wrapping initialisation.

For each dataset we let the outlier percentage vary between 0.1 and 0.5 and fit each model 10 times with different random seeds (which impacts the train/test split as well as the selection of initial random subsets for most algorithms). For all algorithms we fix the parameter α to 0.5 (i.e. the h-subset is half of the training set). All other hyper-parameters as described above also remain fixed during the experiment. We report the mean R^2 value on the test set for each dataset and each outlier percentage across iterations as well as the mean fit duration for each algorithm. The experiments were executed on a MacBook Pro with a 2,3 GHz Quad-Core Intel Core i5 processor.

4.2.2 Non-linear regression

In order to demonstrate the impact of using Soft LTS for non-linear regression, we train a multi-layer perceptron (MLP) on the generated data with 3 different loss functions: mean squared error (MSE), mean absolute error (MAE) and Soft LTS loss with $\alpha = 0.7$. The architecture of the MLP is the same in all cases: 2 hidden layers with 100 neurons each and ReLU activation. The MSE and MAE networks are trained for 2000 epochs with a batch size of 50 and an Adam optimiser with a learning rate of 0.001, the Soft LTS network is trained for 10 000 epochs with a batch size equal to the size of the dataset and an Adam optimiser with a learning rate of 0.01.

For each of the networks, we also show the impact of using y-wrapping to initialise the weights of the neural network. The initialisation weights are the same for each network and are obtained by training an MLP with the same architecture and MSE objective on the y-wrapped data.

We report the R^2 value on the clean data for each model.

5 Results

5.1 Simulation data

5.1.1 Least Trimmed Squares

Figure 4 shows the impact of applying soft sorting and Xy-wrapping initialisation for LTS on the simulated dataset. On the X-axis of the top 2 panels, the outlier percentage varies from 10% to 50% outliers and the Y-axis shows the mean R^2 on the test set. The Y-axis of the bottom panel represents the average total fit duration in seconds on a logarithmic scale across iterations and across all outlier percentages. In terms of robustness, we can see that FAST LTS with a single random start does not perform well on either dataset, while

[7]https://rdrr.io/cran/robustbase/man/lmrob.html

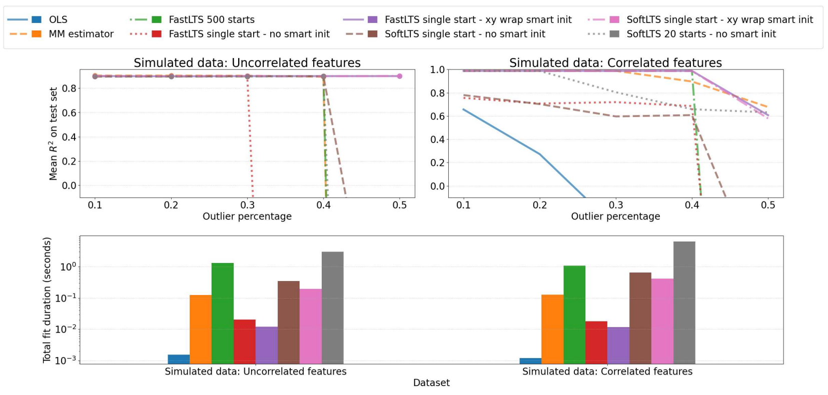

Figure 4: Impact of soft loss function and Xy-wrapping initialisation for LTS on simulated data (left: simulation with uncorrelated features, right: simulation with correlated features)

the soft version seems to have a higher breakdown value for the data with uncorrelated features and similar performance for the data with correlated features. If we compare FAST LTS and Soft LTS with multiple random starts we see that Soft LTS with multiple random starts performs equally on the data with uncorrelated features and slightly worse on the data with correlated features. The comparison is not entirely fair as Soft LTS only has 20 random starts. The reason why we do not consider 500 random starts is clear when we look at the fit duration in the bottom panel of figure 4, as even with only 20 starts, Soft LTS is already significantly slower than FAST LTS with 500 starts.

It is clear that there is a large positive effect of applying the proposed initialisation based on Xy-wrapping. In particular, both FAST LTS and Soft LTS improve greatly when starting from the weights retrieved from the regression on the wrapped data, compared with those from a single random start. On most datasets, the single starts with Xy-wrapping even outperform FAST LTS with 500 random starts and Soft LTS with 20 random starts. Besides the positive impact on robustness, there is also an improvement in speed. We can see a small speed-up when comparing the single start alternatives. This seems counterintuitive, but can be explained by the fact that a more robust starting value may require fewer C-steps to converge. When compared with the more standard FAST LTS with 500 random starts, the Xy-wrapping can lead to speed improvements up to a factor 100 for similar or better performance.

Comparison with the baseline MM-estimator shows that FAST LTS with Xy-wrapping initialisation and Soft LTS with Xy-wrapping initialisation have similar performance in terms of robustness, but FAST LTS with Xy-wrapping initialisation is up to a factor 10 times faster than the MM-estimator and Soft LTS with Xy-wrapping initialisation

5.1.2 Least Median Squares

Figure 5 shows the results for the experiments with the LMS objective. Classic LMS with 3000 random subsets appears to be more robust than Soft LMS

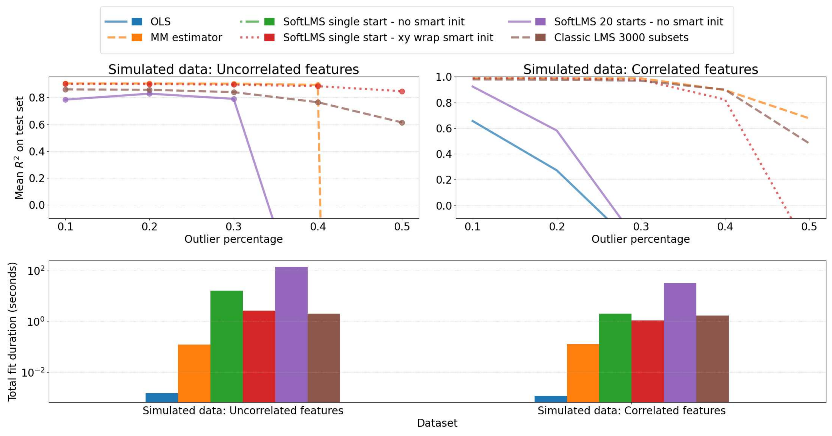

Figure 5: Impact of soft loss function and Xy-wrapping initialisation for LMS on simulated data

with 20 starts for both datasets. Soft LMS with a single random start is not competitive at all (it is not even visible in figure 5, as its R^2 value already drops below 0 for 10% outliers). Furthermore, as can be seen in the bottom panel of figure 5, Soft LMS with multiple starts is almost a factor 100 times slower than Classic LMS.

Using Xy-wrapping initialisation leads to much more competitive results, as it outperforms Classic LMS on the data with uncorrelated features and has only a slightly worse performance for high outlier percentages on the data with correlated features. It is even competitive with the MM estimator results on both datasets. If we look at fit duration, we see similar times for Soft LMS with Xy-wrapping initialisation and Classic LMS with 3000 subsets. Nonetheless, in terms of fit duration, all LMS approaches are still outperformed by the MM estimator. It is clear that the soft loss function is only valuable when combined with smart initialisation for LMS.

5.1.3 Non-linear LTS

Figure 6 shows the results for the experiment on non-linear simulated data. It is clear that both MSE and MAE loss are not capable of handling the bad leverage points. They even have negative R^2 values, indicating that the fitted models perform worse than a horizontal plane. Using Soft LTS loss results in a nearly perfect fit to the clean data. If we apply y-wrapping initialisation, we see a slight additional improvement. If we look at the training loss history in figure 7, we can also see that using the smart initialisation results in a slightly faster convergence.

5.2 Real data

5.2.1 Least Trimmed Squares

Figure 8 shows the results on real datasets for the LTS variants. Similar conclusions can be drawn as for the simulated data: FAST LTS and Soft LTS initialised

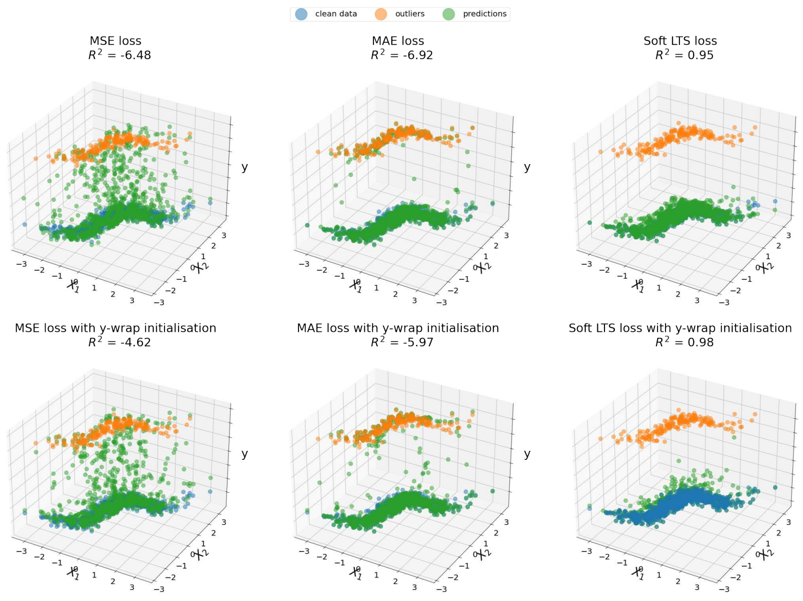

Figure 6: Results non-linear regression

with Xy-wrapping together with the MM estimator and FAST LTS with 500 random starts have the best performance on most datasets. Both FAST LTS and Soft LTS with a single random start are consistently outperformed by the variants with smart initialisation and multiple random starts.

If we look at the fit durations, we can see that Soft LTS does not scale well with the dataset size. For the largest dataset (CCPP with 9568 observations), Soft LTS is more than a factor 100 times slower than FAST LTS. We do however see the impact of applying smart initialisation, which leads to faster convergence for bath FAST LTS and Soft LTS. While the MM-estimator is faster than FAST LTS with 500 starts, it is up to a factor 10 times slower than FAST LTS with smart initialisation for the smaller datasets. For the largest dataset (CCPP), the difference in fit duration shrinks, which indicates that the LTS algorithm does not scale as well with the dataset size as the MM estimator.

5.2.2 Least Median Squares

Figure 9 shows the results on real datasets for the LMS variants. Classic LMS with 3000 starts appears to be the most robust on all datasets, closely followed by the MM estimator and Soft LMS with Xy-wrapping initialisation (except for the ALE dataset and outlier percentages of 50%, where Soft LMS with Xy-wrapping initialisation is significantly worse). Soft LMS with a single random start is again not visible on the plots as this leads to very poor results, even for small outlier percentages.

Looking at the duration we see a similar pattern as Soft LTS: for larger datasets, Soft LMS is slower than Classic LMS with 3000 starts, even with Xy-wrapping initialisation. All LMS variants are significantly slower than the MM estimator

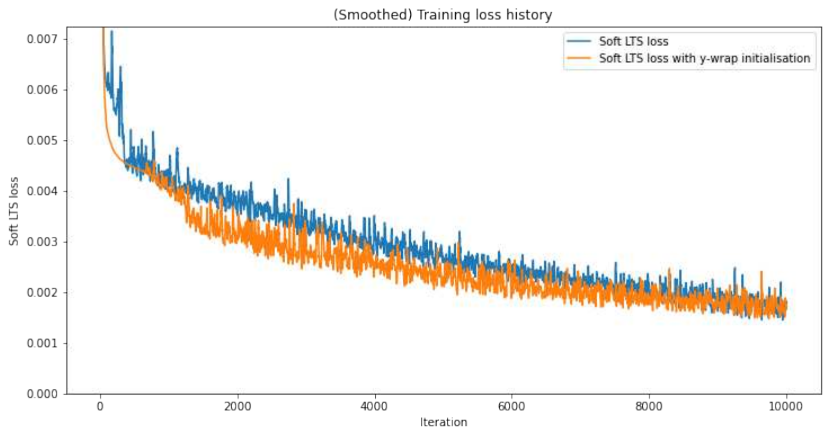

Figure 7: Training loss for Soft LTS MLP

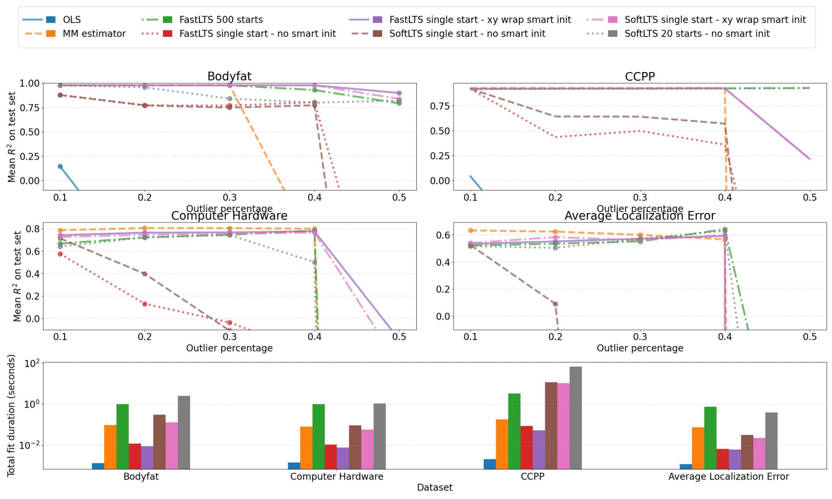

Figure 8: Impact of soft loss function and Xy-wrapping initialisation on LTS on real data

6 Conclusion

LTS and LMS are well known methods for robust regression. Both of these methods rely on sorting of the squared residuals, which is a piece-wise linear function, making it difficult to use gradient-based optimisation approaches. The most popular existing algorithms for fitting an LTS or LMS regression use a large number of random initialisations. In this paper, we tried to improve upon these algorithms both in terms of robustness as well as fit duration by (1) using soft sorting in the objective function to construct 'Soft LTS' and 'Soft LMS' estimators and (2) using a deterministic initialisation by applying wrapping regression.

For LTS, we saw substantial improvements as a result of applying Xy-wrapping initialisation. On almost all datasets, FAST LTS with Xy-wrapping initialistion

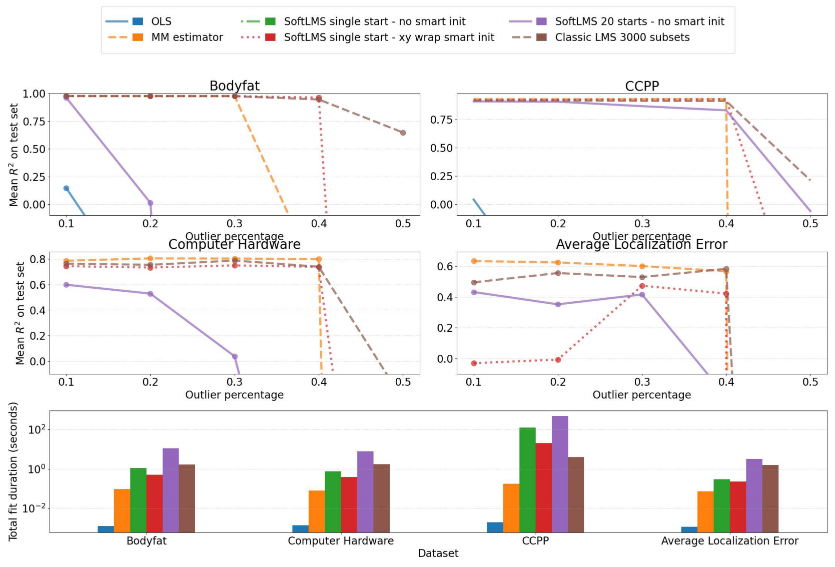

Figure 9: Impact of soft loss function and Xy-wrapping initialisation for LMS on simulated data

obtained the best performance, often outperforming FAST LTS with 500 random starts and the MM estimator. Furthermore, there is a significant speed-up in fit duration. Fitting FAST LTS with a single start, initialised with Xy-wrapping was a factor 100 times faster than FAST LTS with 500 random starts in our experiments and a factor of 10 times faster than the MM estimator. On the other hand, the use of soft sorting did not result in any improvements when it comes to robustness. In most cases a comparable performance as FAST LTS could be achieved, but due to the higher fit durations, we can conclude that Soft LTS is not a competitive alternative for FAST LTS.

Our experiments with LMS also confirmed that Xy-wrapping has its benefits. However, the use of the soft loss objective is a requirement in this case, as there is no alternative iterative optimisation method (where FAST LTS uses C-steps). On the simulated datasets, Soft LMS with Xy-wrapping initialisation achieved a better performance than Classic LMS with 3000 starts, with on almost equal fit duration. On the real datasets, Classic LMS with 3000 subsets was slightly more robust than Soft LMS with Xy-wrapping initialisation, as well as up to a factor 10 times faster on the largest dataset (9568 observations). All LMS variants were slower than the MM estimator.

Finally, we also demonstrated the potential use of the Soft LTS loss for non-linear regression. On our simulated dataset with so-called bad leverage points (i.e. points that are outlying both in X and y), an MLP with Soft LTS objective was able to achieve an almost perfect fit on the clean data, while the same network with MSE and MAE loss were not able to handle the outliers. Unfortunately, in real applications, the choice of α is not so straight forward to make, as the percentage of outliers is not known. While we can use a low α of 0.5 by default in many cases, this is not the case for non-linear regression. With linear regression, it is safer to assume reliable extrapolation at the edges of the distribution, while this is not possible for non-linear regression, where there is no straightforward way to extrapolate the learned function outside of the known domain.

Further research could be done on alternative deterministic initialisation methods. Following the example of DetMCD [13], multiple deterministic starting points could be selected, increasing the potential to find the 'optimal' solution while significantly reducing the number of initial starting points and thus also the fit duration.

The use of Soft LTS for non-linear regression also requires further in-depth analysis. Datasets suited for non-linear regression with known outliers could be used to further validate the potential of Soft LTS as a loss function.

7 Acknowledgements

This research received funding from the Flemish Government, through Flanders Innovation and Entrepreneurship (VLAIO, project HBC.2020.2285).

References

[1] Andersen, R., 2008. Modern methods for robust regression. 152, Sage.

[2] Blondel, M., Teboul, O., Berthet, Q., Djolonga, J., 2020. Fast differentiable sorting and ranking, in: Proceedings of the 37th International Conference on Machine Learning, PMLR. pp. 950–959. URL: https://proceedings.mlr.press/v119/blondel20a.html.

[3] Boudt, K., Rousseeuw, P.J., Vanduffel, S., Verdonck, T., 2020. The minimum regularized covariance determinant estimator. Statistics and Computing 30, 113–128.

[4] Byrd, R.H., Lu, P., Nocedal, J., Zhu, C., 1995. A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16, 1190–1208. URL: https://doi.org/10.1137/0916069, doi:10.1137/0916069, arXiv:https://doi.org/10.1137/0916069.

[5] Cuturi, M., Teboul, O., Vert, J.P., 2019. Differentiable ranking and sorting using optimal transport, in: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/d8c24ca8f23c562a5600876ca2a550ce-Paper.pdf.

[6] De Ketelaere, B., Hubert, M., Raymaekers, J., Rousseeuw, P., Vranckx, I., 2020-04-15. Real-time outlier detection for large datasets by rt-detmcd.

[7] De Ketelaere, B., Hubert, M., Raymaekers, J., Rousseeuw, P.J., Vranckx, I., 2020. Real-time outlier detection for large datasets by rt-detmcd. Chemometrics and Intelligent Laboratory Systems 199, 103957.

[8] Dua, D., Graff, C., 2017. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

[9] Hampel, F.R., Rousseeuw, P.J., Ronchetti, E., 1981. The change-of-variance curve and optimal redescending m-estimators. Journal of the American Statistical Association 76, 643–648. URL: https://doi.org/10.1080/01621459.1981.10477698, doi:10.1080/01621459.1981.10477698, arXiv:https://doi.org/10.1080/01621459.1981.10477698.

[10] Holland, M.J., Ikeda, K., 2017. Robust regression using biased objectives. Machine Learning 106, 1643–1679. URL: https://doi.org/10.1007/s10994-017-5653-5, doi:10.1007/s10994-017-5653-5.

[11] Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks 2, 359–366.

[12] Hubert, M., Rousseeuw, P., Vanpaemel, D., Verdonck, T., 2015. The dets and detmm estimators for multivariate location and scatter. Computational Statistics & Data Analysis 81, 64–75. URL: https://www.sciencedirect.com/science/article/pii/S0167947314002175, doi:https://doi.org/10.1016/j.csda.2014.07.013.

[13] Hubert, M., Rousseeuw, P.J., Verdonck, T., 2012. A Deterministic Algorithm for Robust Location and Scatter. Journal of Computational and Graphical Statistics 21, 618–637. URL: https://doi.org/10.1080/10618600.2012.672100, doi:10.1080/10618600.2012.672100. publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2012.672100.

[14] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. URL: http://arxiv.org/abs/1412.6980.

[15] Lin, Y.L., Hsieh, J.G., Jeng, J.H., Cheng, W.C., 2015. On least trimmed squares neural networks. Neurocomputing 161, 107–112. URL: https://www.sciencedirect.com/science/article/pii/S0925231215002313, doi:https://doi.org/10.1016/j.neucom.2015.02.059.

[16] Maronna, R.A., Martin, R.D., Yohai, V.J., Salibián-Barrera, M., 2019. Robust statistics: theory and methods (with R). John Wiley & Sons.

[17] Martin, R.D., Yohai, V.J., Zamar, R.H., 1989. Min-max bias robust regression. The Annals of Statistics 17, 1608–1630.

[18] Owen, A.B., 2006. A robust hybrid of lasso and ridge regression. Technical Report. URL: https://artowen.su.domains/reports/hhu.pdf.

[19] Raymaekers, J., Rousseeuw, P.J., 2021. Fast robust correlation for high-dimensional data. Technometrics 63, 184–198.

[20] Riazoshams, H., Midi, H., Ghilagaber, G., 2018. Robust nonlinear regression (with applications using r) —— 10.1002/9781119010463. doi:10.1002/9781119010463.

[21] Rousseeuw, P., Driessen, K., 1999. A fast algorithm for the minimum covariance determinant estimator. Technometrics 41, 212–223. doi:10.1080/00401706.1999.10485670.

[22] Rousseeuw, P., Yohai, V., 1984. Robust regression by means of s-estimators, in: Franke, J., Härdle, W., Martin, D. (Eds.), Robust and Nonlinear Time Series Analysis, Springer US, New York, NY. pp. 256–272.

[23] Rousseeuw, P.J., 1984. Least median of squares regression. Journal of the American Statistical Association 79, 871–880. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1984.10477105, doi:10.1080/01621459.1984.10477105, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1984.10477105.

[24] Rousseeuw, P.J., Leroy, A., 1987. Robust Regression and Outlier Detection. Wiley Series in Probability and Statistics, Wiley.

[25] Rousseeuw, P.J., Van Driessen, K., 2000. An Algorithm for Positive-Breakdown Regression Based on Concentration Steps. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 335–346. URL: https://doi.org/10.1007/978-3-642-58250-9_27, doi:10.1007/978-3-642-58250-9_27.

[26] Rousseeuw, P.J., Van Driessen, K., 2006. Computing lts regression for large data sets. Data mining and knowledge discovery 12, 29–45.

[27] Salibian-Barrera, M., Yohai, V.J., 2006. A fast algorithm for s-regression estimates. Journal of computational and Graphical Statistics 15, 414–427.

[28] Singh, A., Kotiyal, V., Sharma, S., Nagar, J., Lee, C.C., 2020. A machine learning approach to predict the average localization error with applications to wireless sensor networks. IEEE Access 8, 208253–208263. doi:10.1109/ACCESS.2020.3038645.

[29] Steele, J., Steiger, W., 1986. Algorithms and complexity for least median of squares regression. Discrete Applied Mathematics 14, 93–100. URL: https://www.sciencedirect.com/science/article/pii/0166218X86900090, doi:https://doi.org/10.1016/0166-218X(86)90009-0.

[30] Tüfekci, P., 2014. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems 60, 126–140. URL: https://www.sciencedirect.com/science/article/pii/S0142061514000908, doi:https://doi.org/10.1016/j.ijepes.2014.02.027.

[31] Wilcox, R.R., 2022. Chapter 10 - robust regression, in: Wilcox, R.R. (Ed.), Introduction to Robust Estimation and Hypothesis Testing (Fifth Edition). fifth edition ed.. Academic Press, pp. 577–651. URL: https://www.sciencedirect.com/science/article/pii/B9780128200988000166, doi:https://doi.org/10.1016/B978-0-12-820098-8.00016-6.

[32] Yohai, V.J., 1987. High breakdown-point and high efficiency robust estimates for regression. The Annals of Statistics 15, 642–656. URL: http://www.jstor.org/stable/2241331.