Sparse Robust Regression for Explaining Classifiers

Anton Björklund, 37664
Masters Thesis in Computer Science
Supervisors: Kai Puolamäki
Emilia Oikarinen
Jan Westerholm
Faculty of Science and Engineering
Åbo Akademi University
August 2019

Abstract

A common characteristic of many datasets is the presence of outliers, items that do not follow the same structure as the rest of the data. If the outliers are not taken into account it will have negative consequences when the dataset is used, such as leading to the wrong conclusions. The field of robust statistics is concerned with finding and dealing with the outliers. This thesis introduces a novel algorithm for robust regression called SLISE, Sparse LInear Subset Explanations. SLISE is able to ignore outliers by finding the largest subset of data items that can be represented by a linear model to a given accuracy.

In this thesis SLISE is compared to existing robust regression methods both theoretically and empirically. We find that SLISE is as robust as state-of-the-art methods and is faster on large datasets, which is important with regards to the ever-growing sizes of modern datasets.

One of the most interesting applications for SLISE is to explain outcomes from black box models. With the increased use of machine learning these kinds of models become more and more prevalent, but in many situations the opaqueness limits their usefulness. Thus recently there have been a lot of research into explaining outcomes from black box models. SLISE gives explanations in the form of *local explanations*. Local explanations are only valid for one item or a subset of all possible items but this enables the explanations to focus on the important features for those specific items.

Similar to many other local explanation methods SLISE gives explanations in the form of linear models that locally approximate the black box model. An advantage with SLISE is that no new data or new outcomes are required contrary to many existing methods that have data-specific mutation processes. This allows SLISE to account for constraint and structures inherent to the data, such as conservation laws in physical systems.

Keywords: Robust Regression, Interpretability, Local Explanations

Preface

The algorithm that is presented in this thesis was first introduced in a paper [9] and I would like to thank my co-authors of that paper Andreas Henelius, Emilia Oikarinen, Kimmo Kallonen, and Kai Puolamäki. Furthermore, the work leading up to both the paper and this thesis has been done in the *Exploratory Data Analysis* group at University of Helsinki and I am grateful for that opportunity. A further thanks to my colleagues for providing a pleasant work environment.

I would also like to thank my supervisors Kai Puolamäki, Emilia Oikarinen, and Jan Westerholm for all the great advise and insights during the writing of this thesis.

Finally, all of this have been possible due to the financial support by *Academy of Finland* (decisions 326280 and 326339). Also crucial for honing the algorithm presented in this thesis are the computational resources provided by *Finnish Grid and Cloud Infrastructure* [22].

Contents

Abbreviations

Anchors	Local explanations based on rules [50] (Explainer)
C-Steps	Concentration steps [54] (Optimisation)
CNN	Convolutional Neural Network (Classifier)
EMNIST	Extended MNIST [15]: Dataset of handwritten letters (Dataset)
F1-Score	Measure of classification quality (Evaluation)
Fast-LTS	Faster LTS [54] (Robust Regression)
FCGI	Finnish Grid and Cloud Infrastructure [22] (Computer Cluster)
GDPR	General Data Protection Regulation [3] (EU Regulation)
IMDB	Internet Movie DataBase [41] (Dataset)
L1-Norm	Sum of absolute values, $\|x\|_1 = \sum_i \|x_i\|$ (Regularisation)
L2-Norm	Sum of squared values, $\|x\|_2 = \sum_i x_i^2$ (Regularisation)
LAD	Least Absolute Deviation (Regression)
LAD-LASSO	LAD with LASSO regularisation [66] (Regression)
LASSO	Least Absolute Shrinkage and Selection Operator [62] (Regularisation)
LIME	Local Interpretable Model-agnostic Explanations [49] (Explainer)
LORE	LOcal Rule-based Explanations [27] (Explainer)
LR	Logistic Regression (Classifier)
LTS	Least Trimmed Squares [52] (Robust Regression)
M-Estimator	Maximum likelihood type estimates (Robust Regression)

MES	Model Explanation System [63] (Explainer)
MM-Estimator	Robust M-ESTIMATOR [67] (Robust Regression)
MM-LASSO	MM-ESTIMATOR with LASSO regularisation [59] (Robust Regression)
MNIST	Dataset of handwritten digits [36] (Dataset)
NN	Neural Network (Classifier)
OLS	Ordinary Least-Squares (Regression)
Quasi-Newton	Approximation of the Newton method (Optimisation)
S-Estimator	Robust estimates of scales [51] (Robust Regression)
SHAP	SHapley Additive exPlanations [40] (Explainer)
SLISE	Sparse LInear Subset Explanations [9] (Robust Regression)
Sparse-LTS	FAST-LTS with LASSO regularisation [4] (Robust Regression)
SVM	Support Vector Machine [16] (Classifier)

1. Introduction

Practically all real-world datasets contain outliers, items that do not adhere to the same structure as the rest of the data. These outliers are problematic when modelling or analysing the data, since even a single outlier can have a large negative impact on the outcome [32]. It is thus important to use methods that are robust to outliers. In this thesis I present a novel robust regression method termed SLISE, Sparse LInear Subset Explanations. Specifically, SLISE finds *the largest subset of data items that can be represented by a linear model to a given accuracy.*

Modern datasets tend to be quite large, both in terms of the number of items and in the number of variables. Efficient algorithms are crucial, especially for interactive use, and SLISE is able to outperform many of the existing robust regression methods on large datasets (shown in Section 4.2). Having many variables not only affects the running time, but also the interpretability. Sparse models (models where some of the coefficients are zero) are easier to interpret since the zero-coefficients can be ignored [38]. SLISE uses LASSO regularisation [62] to produce sparse models.

Another feature of modern data science is the increased use of black box models, such as neural networks. These kinds of models are used since they can handle more complex data and thus provide better predictions. The downside is that the inner workings of black box models are often incomprehensible. Thus there is a growing need of explanations for what the models are doing. The *Explanation* part of the the name SLISE comes from that one of the most interesting applications of SLISE is to explain outcomes from black box models.

Existing explanation methods usually aim for either explaining everything the black box model might do (a global explanation, i.e., turn it into a transparent model) or just explaining individual decisions (local explanations). The explanations provided by SLISE are local and SLISE does not require any modifications to the black box models for it to work.

Existing methods in this niche usually work by perturbing the item being

explained and then fitting a simpler model to this "neighbourhood". The simple model approximates the black box model (locally) and is presented as the explanation. SLISE does not require the user to design "reasonable" data-perturbations, but instead uses real data to fit the simple model. Specifically, SLISE finds *the largest subset of data items that can be approximated by a linear model centred on a selected item*. This has the added benefit that the subset can be used to measure how widely applicable the same explanation is to different items.

1.1 Disclosure

This thesis is based on a paper [9] where the SLISE algorithm was introduced for the first time. I am the first author of that paper and have contributed to all parts of both the paper and the algorithm. This includes the design, the implementation, and the writing. This thesis is a new and stand-alone text but it shares some content with the paper, namely some definitions, some proofs, and some experimental results.

Compared to the paper [9] this thesis contains more theory and related works for the two main domains, robust regression and opaque model explanations. The thesis also includes improvements to the algorithm in the form of more proofs and different initialisation alternatives (of at least one is more robust). Furthermore the thesis extends the experiments with new parameter experiments and more ways of utilising explanations.

1.2 Contributions and Structure

This thesis presents a novel robust regression method, SLISE, that can be used for explaining black box decisions. The problem SLISE solves can be described as *finding the largest subset of data items that can be represented by a linear model to a given accuracy*. This problem is **NP**-hard but SLISE is able to solve it with a constant approximation ratio. Empiric evaluations on both real and synthetic datasets show that SLISE outperforms some state-of-the-art robust regression methods. Furthermore SLISE provides sensible explanations for black box models and extends existing explanation methods primarily in the use of real data instead of mutations.

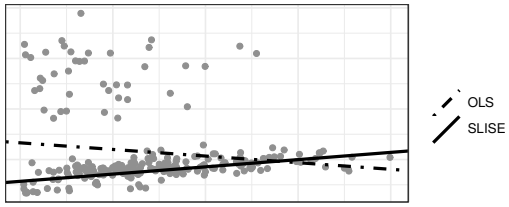


Figure 1.1: Robust regression example. The outliers in the upper left corner causes OLS regression to find a suboptimal model while SLISE is able to ignore the outliers.

The introductory Chapter 1 gives a quick overview of the method and the two primary domains. The introduction concludes with two example scenarios where SLISE can be used. Chapter 2 introduces the background for these two domains and discusses related works for both. The formal definition of the problem follows in Chapter 3 together with the proof that the problem is **NP**-hard. The same chapter also presents the algorithm and the numerical approximations required by the algorithm. SLISE is evaluated empirically in Chapter 4, using both synthetic and real-world datasets. The goal of the experiments is threefold, to select optimal parameters, to compare SLISE to other robust regression methods, and to show how SLISE can be used for explanations. This thesis ends with conclusions in Chapter 5 and a summary in Swedish in Chapter 6.

1.3 Examples

Next follows two simple examples of using SLISE in the two domains. This not only demonstrates the usefulness of SLISE but also introduces some topics that will be expanded upon in the following chapters.

Robust Regression Figure 1.1 shows a two-dimensional dataset with outliers in the top-left. These outliers causes ordinary least-squares (OLS) regression to give a model that is a bad fit for most of the points. SLISE, on the other hand, is largely unaffected by the outliers, by ignoring them, and finds a model that fits a large subset of the data.

Explaining Black Box Models Table 1.1 shows the probabilities for having a high income, in a toy dataset, according to a simple classifier. Assume that the dataset consists of mostly people with a high education, for example, a faculty

	Education	
Age	*low*	*high*
young	0.07	0.31
old	0.22	0.61

Table 1.1: Explanation example. Probabilities of having a *high income* according to a simple classifier.

of a university department. A local explanation task could be to determine which factor is more important, age or education, for the classification of an old professor according to the classifier.

Looking at only the class probabilities in Table 1.1 it appears that education is more important, and this is indeed the explanation, e.g., the state-of-the-art local explanation method LIME [49] finds. However, knowing the education level for this person actually gives little information about the class since the dataset contains almost only people with high education. Instead, *in this dataset*, age is a better determinant of high income for the old professor, and this is what SLISE returns.

The model in Table 1.1 is actually a simple logistic regression[1]. LIME finds the largest coefficient in the logistic regression model, whereas SLISE finds the behaviour of the model in the dataset. The interaction between the model and the data is thus important, but can make the interpretation of even simple models non-trivial if the data has complex structures.

This insight is significant because it suggests that one cannot always cleanly separate the model from the data. An example of this is conservation laws in physical systems. If the data is accurate then the observations will never violate such laws and the model has no incentive to learn structures outside the constraints. Thus predictions for data breaking the constraints must be considered either undefined or random. One may therefore find explanations that violate the laws of physics if the explanation method does not adhere to the constraints, or the data. SLISE satisfies these kinds of constraints automatically, because the explanations are based on observing how the model behaves in the dataset, instead of randomly sampling (possibly non-physical) points around the item of interest (as in, e.g., [49, 50, 40, 27]).

[1]Probability of high income is given by $p = \sigma(-2.53 + 1.73 \cdot \text{education} + 1.26 \cdot \text{age})$, where $\sigma(x) = 1/(1 + e^{-x})$ is a sigmoid function.

2. Background

As described in the introduction SLISE is a sparse robust regression method with applicability to explain decisions made by black box models. The next chapter contains the in-depth definition of SLISE, while this chapter focuses on the theory of those two fields, as well as related methods for both.

2.1 Robust Regression

Normal regression is concerned with finding the relationship between one or more independent variables and one dependent variable. There are many ways to accomplish this, but one of the most basic methods is *ordinary least squares* regression (OLS). OLS regression finds a linear model by minimising the sum of squared residuals. However it is very common for datasets to have samples with divergent values, called outliers. The reason these outliers exists can be anything from measurement errors to structures in the data that the regression model cannot describe. By squaring the residuals OLS gives a lot of weight to the outliers, which will affect the final model.

Robust regression methods are regression methods where the presence of outliers does not affect the results (too much) [53]. This means robust regressors minimise the effects the outliers have on the final model. The example in Figure 1.1 from the previous chapter shows a clear difference between robust and non-robust regression. OLS regression is not robust and thus the outliers have a large impact on the final model. There are multiple possible ways of accomplishing robustness, but normally the outliers are either de-emphasised or outright ignored. How the outliers are detected also differs between robust regression methods.

A popular measure of robustness is the *breakdown value* [32] which measures how many of the samples in a dataset must be replaced by outliers in order to cause an arbitrarily large change in the model. For OLS regression a single

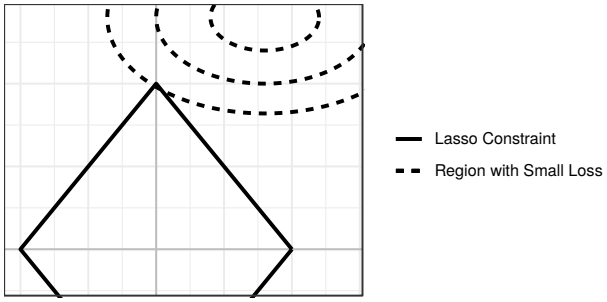


Figure 2.1: Geometric interpretation of Lasso

outlier is enough to cause an arbitrarily large change, which corresponds to a breakdown value of $1/n$, where n is the number of samples [32]. The best possible breakdown value is 1, but that implies that the model is independent of the data, which is not very useful. In practise the best breakdown value is 0.5, since if more than half of the data is replaced then the there is no way of telling which subset is the original one. But methods with smaller breakdown values may also perform well if the data does not contain a lot of outliers.

2.1.1 Sparsity

Another feature of some regression methods, including SLISE, is the ability to produce sparse models. Sparse models are models with some of the coefficients being zero. This is especially useful for large datasets with many variables. The sparsity makes the models more interpretable, since sparsity helps you focus your attention on the variables that matter (those with coefficients other than zero).

Sparsity is often accomplished using a L1-norm on the model coefficients ($\|\alpha\|_1 = \sum_i |\alpha_i|$), which is also known as LASSO regularisation [62]. LASSO (Least Absolute Shrinkage and Selection Operator) regularisation can either be used as a constraint $\|\alpha\|_1 \leq t$ or by adding it to a loss function with a Lagrange multiplier $\lambda\|\alpha\|_1$. Here α is a vector that contains the coefficients for the regression model and $t, \lambda \geq 0$ are regularisation parameters.

The easiest way to show how LASSO regularisation produces sparse models is through a geometric interpretation, shown in Figure 2.1. The square represents the allowed area where $\|\alpha\| \leq t$ and if the optimal non-sparse solution is outside the constraint then the boundary of the low-loss region, exemplified by the dashed-lines, is more likely to tangent the corners (sparse solutions) than the sides [62]. This is in contrast to using a L2-norm (Tikhonov regularisation

Method	Robust	Breakdown	Sparse	Outliers
OLS	No	$1/n$	No	Emphasised
LASSO [62]	No	$1/n$	Yes	Emphasised
LAD	Maybe	$1/n$	No	De-emphasised
LAD-LASSO [66]	Maybe	$1/n$	Yes	De-emphasised
MM-ESTIMATOR [67]	Yes	0.5	No	Penalised
MM-LASSO [59]	Yes	0.5	Yes	Penalised
LTS [54]	Yes	0.5	No	Ignored
SPARSE-LTS [4]	Yes	0.5	Yes	Ignored
SLISE [9]	Yes	≤ 0.5	Yes	Ignored

Table 2.1: Summary of some related robust regression methods

/ Ridge regression) where the boundary $\|\alpha\|_2 \leq t$ is a circle and no point is more likely to tangent than the rest.

2.1.2 Related Methods

Below follows descriptions for a couple of robust regression methods. They are selected to be well known, but also representative of different ways of handling outliers. The discussion naturally includes their breakdown value and mentions variants which include LASSO regularisation in order to provide sparsity. Table 2.1 shows a quick overview of some related methods (OLS, LASSO and SLISE included for comparison). For a more in-depth survey see, e.g., [53, 59].

Least Absolute Deviation Least Absolute Deviation (LAD) is very similar to ordinary least squares but replaces the sum of squared residuals with a sum of absolute residuals. By not squaring the errors LAD is in practise more robust, which can be seen in, e.g., the robustness experiment in Section 4.2.2. But LAD has the same breakdown value as OLS ($1/n$) so it is not robust in a general case (also shown in Section 4.2.2). Using absolute values introduces two complications. The solutions are not unique and LAD cannot be solved analytically [2]. There are, however, multiple efficient iterative methods for solving LAD, e.g., [56, 8].

Similarly to how OLS regression can be extended with LASSO regularisation to create LASSO [62] (from where the name origins), LASSO regularisation can

also be added to LAD to create the sparse LAD-LASSO algorithm [66].

MM-Estimator One of the earliest attempts at creating a robust regression algorithm is the M-ESTIMATOR which avoids outliers based on maximum likelihood estimates. However, it was soon discovered that it has as low a breakdown value as OLS, which suggests that in theory it is not robust [31]. This lead to the development of the S-ESTIMATOR [51] where the residual scales are robustly estimated (with similar calculations to the ones in M-ESTIMATOR). S-ESTIMATOR is a robust regression algorithm with a breakdown value of 0.5, but it is inefficient [67]. There are many other proposed alternative M-ESTIMATORs that address the robustness, some even quite recent ones [39, 11], but the most well known is MM-ESTIMATOR [67]. By using ideas from S-ESTIMATOR for initialisation MM-ESTIMATOR achieves the same breakdown value of 0.5, but still keeps the efficiency of M-ESTIMATOR.

The application of LASSO regularisation to create sparse variants of algorithms is something that continues with MM-ESTIMATOR, resulting in MM-LASSO [59].

Least Trimmed Squares Least Trimmed Squares [52] (LTS) takes a different approach by working with a subset S, of a given size $|S| = k$ where $n/2 < k \leq n$. LTS finds the linear model with the least sum of squared residuals for k samples (the subset), the rest $n - k$ samples are ignored. Since LTS considers a subset of included items, rather than smoothly penalising outliers, it is a combinatorial problem. LTS has no closed-form solution, but algorithms provide good approximations. Another problem with (early) LTS was that the approximation was inefficient, but that was solved with the development of the FAST-LTS variant [54].

FAST-LTS starts from a number of candidate subsets then iteratively improves them with concentration steps (C-steps). For each step and candidate a linear model is calculated for the subset using OLS, then the subset is updated to contain the k samples with the least residuals. This procedure is guaranteed to converge since the candidates either remain the same or are improved. When all candidates have converged (stopped improving) the best model is selected.

FAST-LTS has also been further enhanced with sparsity by the addition of LASSO regularisation to create SPARSE-LTS [4].

2.2 Explaining Classifiers

A surprising application of SLISE is in the field of explaining black box models. The phrase *black box* usually refers to machine learning models that are so complex that it is infeasible to understand their inner workings. But it could just as well be human experts refusing to offer motivations for their decisions. This section focuses on answering why you would need explanation, what questions these explanations answer, and how a couple of existing methods work. For a more detailed review see, e.g., [28, 45].

2.2.1 The Need for Explanations

The most obvious reason for explanations is to build trust. The name black box in itself suggest that there is some opaque process going on, so one way to increase trust is to give insight into what is happening. While trust and understanding is important in general it is especially important in safety critical applications, such as in medicine [10]. Beside building trust there are also some practical reasons for needing explanations.

The European Union recently enacted the General Data Protection Regulation (GDPR [3]) which among other things stipulates the right for citizens to get an explanation for algorithmic decisions [26]. This is not limited to black box models, but obviously also covers them. The aim of this requirement is not only to increase transparency but also to avoid systematic biases and discrimination. Discrimination is a known problem in machine learning [37] and has real consequences, e.g., Amazon did not deploy a new hiring tool after it showed signs of discrimination [17]. Methods exists for mitigating discrimination [29], but explanations can play a key role in detecting such biases [28, 49].

One popular type of black box models, deep learning, has been shown to outperform humans on tasks normally requiring expert knowledge, e.g., in lung cancer detection [19]. At the same time some established deep learning models have been shown to be fragile, where imperceptible changes to the input can cause drastically different results [60, 46, 25]. This would motivate a hybrid approach with both a trained model and a human expert. This has been successfully used in, e.g., [65, 42] where the breast cancer detection accuracy increased from 92.5 % to 99.5 % by introducing a human expert. For the

		Interpretation	
Locality	Globally Interpretable Model	Global Post-hoc Explanation	Global Model Inspection
	Locally Interpretable Model	Local Post-hoc Explanation	Local Model Inspection

Table 2.2: Explanation categories

hybrid approach to work the model has to share more information than just the prediction. An explanation would naturally constitute such additional information.

A black box that is able to give accurate predictions probably has some internal representation of the data. Being able to extract that information could help with verifying that what the models is doing is reasonable. Furthermore, if the data is not fully understood then this information could be used to extend human knowledge. Both of these phenomena could be really beneficial in, e.g., scientific use-cases, such as when classifying particle-jets in high energy physics [18, 33].

2.2.2 Types of Explanations

Explanation of opaque models is currently a hot research topic, so naturally there are a lot of different approaches [28], but they can generally be categorised based on what they are trying to explain and the scope of the explanations. The categories are summarised in Table 2.2.

A natural start is to not create black boxes in the first place, and instead use simpler, interpretable models. Examples of interpretable models would be all kinds of simple models, such as decision trees and various nearest neighbours algorithms, but also some more advanced, e.g., Super-sparse Linear Integer Models [64], and Interpretable Decision Sets [34]. The big drawback with interpretable models is the same as the reason why we use black boxes; complex models can solve some (complex) problems better than interpretable models, and some problems are not at all solvable with current interpretable models.

If you do not want to sacrifice accuracy by simplifying the model then the next option is to use complex models and try to create explanations for the

outcomes after training. Contrary to the interpretable models above these explanations usually do not fit all data (global explanation), but rather focus on a single outcome (local explanation) [49, 28]. Not requiring an explanation for the whole black box might seem like a big limitation, but in many situation it is actually the preferred problem to solve. Compare listing all the things that might lead to cancer to listing the specific risk factors for a certain patient, if you were that patient then you would most likely be more interested in your personal factors, i.e., the local explanation.

Note that the explanation example from Section 1.3 uses a globally interpretable model (logistic regression). But due to structures in the data the variable with most weight in the global model was not the most important one in all *local* cases. Thus local explanations are still valuable even if you have a simple global model, especially if the local explanations are able to account for structures in the data.

Post-hoc local explanations are usually created by either finding a saliency map showing the important features or by approximating the complex model with an interpretable one [28]. The disadvantage with post-hoc explanations is that they are approximations (also applies to saliency maps), and the explanations can only be as good and expressive as the approximations. Furthermore when fitting approximations you often need a "neighbourhood" in which the approximation is valid, and defining this "neighbourhood" is not trivial [45, 27].

It is possible to create explanations for black boxes without using approximations, but they will naturally have more limited scope than any of the approaches above. Instead of explaining the outcomes *model inspection* focuses on specific subprocedures of the black box (local or global). For example, Sensitivity Analysis measures how sensitive the outcome is to changes in the input variables [28], while Activation Maximisation [21] shows what kind of images would maximally activate specific layers in a convolutional neural network. For a survey of different ways of visualising layers in a convolutional neural network see, e.g., [48].

This brings in a third dimension for the categories [28] (not shown in Table 2.2); are the explanations model agnostic or model aware? Model agnostic explanation methods can be used on any black box, even humans, while model aware explanation methods uses some feature tied to specific types of models, e.g., neural networks offer gradients while random forests do not. Interpretable models can all be considered model aware (since the model is the explanation), but there exists both agnostic and aware methods for post-hoc explanations

and inspections. Explanation methods can also be be specialised on different kinds of data, but that is largely independent of the categories mentioned previously.

2.2.3 Interpretability

All of the categories mentioned in the previous section have the same goal, transferring insights about the black box to a human. This requires the results to be interpretable. It is impossible to give an exact definition of interpretability, since it is subjective [38]. Generally a model is considered interpretable if it is simple and small [28, 38], e.g., linear models is easier to interpret than neural networks but a linear model with thousands of coefficients is usually not interpretable. The exact level of simplicity and size obviously depends on the persons doing the interpretation and their expertise [24]. This makes comparisons between model-families difficult.

Transparency is often used as a synonym to interpretability, and while these two concepts are not exactly the same thing, transparency can give additional tools for concretising interpretability. Transparency consists of three concepts; simulatability, decomposability, and algorithmic transparency [38]. *Simulatability* refers to the ability for a human to apply a model without the help of a computer, e.g., follow a decision tree. *Decomposability* states that all components, e.g., variables and calculations, should have a natural description and a reason for being used. Finally, *Algorithmic Transparency* extends the transparency to the training of the model [38]. Intuitively, using a black box model to create (transparent) explanations for another black box model creates distrust. Algorithmic transparency is this intuition quantified as a requirement.

Furthermore, the authors of [24, 38] note that the presentation also has an impact on the interpretability and that the correct choice of presentation depends on the context. However interpretability is not only about making the transfer of information easy, but sometimes also about maximising the amount of information transferred. For example, an expert might not trust too simple explanations [35]. It is also important to consider how the explanations will affect the behaviour of the humans seeing them. Seemingly good explanations will raise the trust in the model, even if it is misplaced. This can lead to more errors, such as obvious misclassifications, not being detected [47].

While there is much that requires consideration on a case by case basis, a

couple of high-level properties that explanations preferably should have can be found in, e.g., [28, 23, 38, 49]. Satisfying all of them at the same time might not be possible, and selecting the best trade-off depends on the situation.

Interpretability Keeping in mind the discussion above about the difficulty of measuring the interpretability, the target audience should at least be able to interpret the explanations.

Accuracy/Fidelity The explanations should match the black box. In the case of interpretable models this means fitting the data, while approximations should imitate the black box model. Both cases can be measured with standard accuracy measures, e.g., F_1-scores.

Informativeness The explanations should give as much information as possible, with the given representation. This definition is kept vague on purpose since it depends on the interpreters skills and because this might contradict the interpretability.

Stability/Generality Explaining similar samples should *usually* lead to similar explanations. Furthermore local approximations should be applicable to more samples than just the one being explained.

2.2.4 Related Methods

Following the categories outlined Section 2.2.2, SLISE creates model-agnostic, local, post-hoc explanations. Thus all the related methods presented in this section are from this niche. A quick summary of the methods can be seen in Table 2.3. For a more complete review of explanations, including methods from other categories, see, e.g., [28, 45].

LIME `Local Interpretable Model-agnostic Explanations` [49]. LIME creates a neighbourhood by mutating the sample to be explained. The mutation is dependent on the data-type, e.g., images are mutated by grouping similar pixels into "super-pixels" (dimensionality reduction) and randomly replacing individual super-pixels with a neutral colour (grey). Then all mutations are evaluated by the black box model and weighted according to the distance to the selected sample. Finally LIME fits a weighted sparse linear model to the neighbourhood.

Method	Explanation	Neighbourhood Generation
LIME [49]	Sparse linear model	Data-specific mutations
SHAP [40]	Sparse linear model	Data-specific mutations
ANCHORS [50]	Rules	Data-specific mutations
LORE [27]	Rules	Genetic algorithm
SLISE [9]	Sparse linear subset	Linear subset

Table 2.3: Summary of some related model-agnostic, local, post-hoc explanation methods

Intuitively, changing the features the model uses for classification will cause a change in the prediction, which LIME utilises in the explanation.

SHAP SHapley Additive exPlanations [40]. SHAP generalises LIME and a couple of other [7, 58] *additive feature attribution* methods. SHAP extends the previous works by setting parameters based on theoretical Shapley values, rather than heuristically. This leads to more stable and accurate explanations.

Anchors ANCHORS [50] addresses two of the issues with LIME; a linear model is not suitable for all kinds of models and it is unclear if the explanations are applicable to other samples than the one being explained. The name, ANCHORS, refers to the explanations that are in the form of rules constructed such that almost all samples for which the rules hold have the same classification. In the creation of the rules ANCHORS uses a similar mutation process to the mutations in LIME.

LORE LOcal Rule-based Explanations [27]. The authors of LORE criticise LIME, and similar methods, for having very uncontrolled mutations. LORE uses a genetic algorithm for controlling the mutations, mostly by penalising mutations far away from the explained sample according to a normalised Euclidean distance measure. This has the added advantage of letting LORE present counterfactuals, i.e., the minimum alternation needed to change the classification.

Other The methods above is just a sample of local, model-agnostic, post-hoc explainers and, e.g., MES [63] could replace ANCHORS since both give explanations in the form of rules, but with different algorithms for finding the rules. Furthermore by not requiring the methods to be model-agnostic there are more features that the explainers can use, such as when [23] uses

gradients from a neural network to find good perturbations of the data item being explained. Finally the authors of [30] states that one needs to be careful when interpreting linear models. This is especially important for users that are not familiar with inference, since the model coefficients show how the model uses the data to infer the outcome and *not* which (latent) features are important for the outcome. [30] further offers a way of using the linear model to find the important latent features.

3. The SLISE Algorithm

SLISE is a robust linear regression method that finds *the largest subset of data items that can be represented by a linear model to a given accuracy.* The solution to this problem has applications to both *global robust linear regression* and *local linear regression* that approximates the decision surface of a black box model in the vicinity of a data item. The problem is formally defined in Section 3.1.

Of the existing robust regression methods SLISE is most closely related to LTS, and its variants. Both utilise a subset and derive their robustness from ignoring outliers. However, there is a major difference; in LTS the size of the subset is fixed a priori while in SLISE the size is dynamic. This breaks the assumptions needed for the FAST-LTS algorithm and thus a novel algorithm (SLISE) is developed in Section 3.2 and 3.3.

3.1 Problem Definition

Let $(X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^n)$ be a dataset that consists of n pairs $\{(x_i, y_i)\}_{i=1}^n$, where x_i (the *predictor*) denotes the ith d-dimensional item (row) in X and y_i (the *response*) denotes the ith element in Y. Furthermore let $\varepsilon \geq 0$ be the largest tolerable error and $\lambda \geq 0$ be a regularisation coefficient. With this the main problem can now be stated:

Problem 3.1. *Given $X \in \mathbb{R}^{n \times d}$, $Y \in \mathbb{R}^n$, and non-negative $\varepsilon, \lambda \in \mathbb{R}$, find the regression coefficients $\alpha \in \mathbb{R}^d$ minimising the loss function*

$$\texttt{Loss}(X, Y, \varepsilon, \lambda, \alpha) = \sum_{i=1}^{n} H\left(\varepsilon^2 - r_i^2\right)\left(r_i^2/n - \varepsilon^2\right) + \lambda \|\alpha\|_1, \tag{3.1}$$

where the residual errors are given by $r_i = y_i - \alpha^\intercal x_i$, $H(\cdot)$ is the Heaviside step function satisfying $H(u) = 1$ if $u \geq 0$ and $H(u) = 0$ otherwise, and $\|\alpha\|_1 = \sum_{i=1}^d |\alpha_i|$ denotes the L1-norm. If necessary, the data matrix X has been augmented with a column of all ones to accommodate the intercept *term of the model.*

Note that this is a combinatorial problem in disguise where we try to find the largest possible subset S. Using the short form $[n] = \{1, \ldots, n\}$ Equation 3.1 can be rewritten, in combinatorial form, as

$$\texttt{Loss}(X, Y, \varepsilon, \lambda, \alpha) = \sum_{i \in S} \left(r_i^2/n - \varepsilon^2 \right) + \lambda \|\alpha\|_1 \text{ where } S = \{i \in [n] \mid r_i^2 \leq \varepsilon^2\}. \tag{3.2}$$

The loss function of Equation 3.1 (and Equation 3.2) thus consists of three parts; the maximisation of subset size $\sum_{i \in S} \varepsilon^2 = |S|\varepsilon^2$, the minimisation of the residuals $\sum_{i \in S} r_i^2/n \leq \varepsilon^2$, and the LASSO-regularisation $\lambda \|\alpha\|_1$. The main goal is to maximise the subset and this is reflected in the loss function, since any decrease of the subset size has an equal or greater impact on the loss than all the residuals combined. At the limit of $\varepsilon \to \infty$ it follows that $S = [n]$ and Problem 3.1 becomes equivalent to LASSO [62].

3.1.1 Breakdown Value

In the beginning of this chapter it is mentioned that Problem 3.1 requires a new algorithm, but it also requires a new proof for the breakdown value.

Theorem 3.1. *The breakdown value of* SLISE *is* $|S_0|/(2n) \leq 0.5$*, where* S_0 *is the subset given by* SLISE *when the dataset contains no outliers.*

Proof. Following the definition from [32, 53, 20] the breakdown value can be found by starting with a uncorrupted dataset (X_0, Y_0) with no outliers and then adversarially change the values of a fraction v of the data items into adversarial values. The breakdown value is the minimum v that can cause an arbitrarily large deviation in the model.

The subset given by SLISE on (X_0, Y_0) is S_0 and on the corrupted dataset (X_v, Y_v) the subset is S_v. SLISE breaks down when S_v consists of corrupted items ($|S_v|/n = v$). This can happen when $|S_v| \geq |S_0 \setminus S_v|$.[1] Rearranging yields $v \geq |S_0 \setminus S_v|/n \geq |S_0|/2/n$. □

The breakdown value for SLISE is thus dependent on the data. If, for example, the second largest (independent) linear subset is larger than $|S_0|/2$ then the SLISE breaks down with $v > |S_0|/(2n)$. However the breakdown value is a measure of worst case, so the theorem still holds. SLISE achieves the largest possible data-dependent breakdown value if the relation between X_0 and Y_0

[1] $S_0 \setminus S_v = \{i \in [n] \mid i \in S_0 \wedge i \notin S_v\}$

is linear (within an error tolerance of ε). Then $|S_0| = n$ and the breakdown value becomes 0.5, which is the practical upper limit for robustness.

3.1.2 Explaining Classifiers

Local explanations require the model to be local to the data item being explained, (x_k, y_k) where $k \in [n]$. This can be accomplished with SLISE by adding an additional constraint to Problem 3.1, requiring that the linear model passes through this item, i.e., $y_k - \alpha^\intercal x_k = r_k = 0$. Centring the data on this item by using $y_i \to y_i - y_k$ and $x_i \to x_i - x_k$ for all $i \in [n]$ causes $r_k = (y_k - y_k) - \alpha^\intercal(x_k - x_k) = 0$, which fulfils the constraint. Note that this also eliminates any potential intercept by setting it to zero. Hence it is sufficient to only consider Problem 3.1 for both finding global regression models (robust regression) and local regression models (local explanations).

As noted in Section 3.1.1 SLISE works best if the relation between X and Y is close to linear. Thus, in practise, if a classifier outputs class probabilities $P \in [0.0, 1.0]^n$ we transform them into linear values using a logit transformation, $y_i = \log(p_i/(1 - p_i))$, for all $i \in [n]$. This yields the vector $Y \in \mathbb{R}^n$ that is used with SLISE (or $Y - y_k$ in the case of local explanations). Note that this linear model is comparable to the linear model obtained using normal logistic regression.

The explanations offered by SLISE thus consist of a linear/logistic regression that approximates the black box model around the outcome being explained (x_k, y_k). This is comparable to many of the other local explanation methods described in Section 2.2.4. Additionally, SLISE also offers a subset of the data items where the approximation is valid, allowing for an error of maximally ε. Further details and examples of explanations are given in Section 4.3.

3.1.3 Complexity

An algorithm approximating Problem 3.1 is developed in the following sections. The approximation is needed when the number of items grows beyond a trivial amount due to the complexity of Problem 3.1.

Theorem 3.2. *Problem 3.1 is* **NP***-hard and hard to approximate.*

Proof. This can be proven by a reduction to the MAXIMUM SATISFYING LINEAR SUBSYSTEM problem [6, Problem MP10], which is known to be **NP**-hard.

The MAXIMUM SATISFYING LINEAR SUBSYSTEM problem is defined as finding $\alpha \in \mathbb{Q}^d$ such that as many equations as possible in $X\alpha = y$ are satisfied, where $X \in \mathbb{Z}^{n\times d}$ and $y \in \mathbb{Z}^n$. This is equivalent to Problem 3.1 with $\varepsilon = 0$ and $\lambda = 0$. The MAXIMUM SATISFYING LINEAR SUBSYSTEM problem is not approximable within n^γ for some $\gamma > 0$, according to [5]. □

3.2 Numerical Optimisation

Since Problem 3.1 is **NP**-hard it has to, in practice, be solved approximately for all but trivial datasets. The combinatorial Problem 3.1 is relaxed into an optimisation problem by replacing the Heaviside function with a sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ and a rectifier function $\phi(u) \approx \min(0, u)$. This allows us to compute the gradient and find α by minimising

$$\beta\text{-}\texttt{Loss}(X, Y, \varepsilon, \lambda, \alpha) = \sum_{i=1}^{n} \sigma\left(\beta(\varepsilon^2 - r_i^2)\right) \phi\left(r_i^2/n - \varepsilon^2\right) + \lambda\|\alpha\|_1, \quad (3.3)$$

where the parameter β determines the steepness of the sigmoid. The rectifier function ϕ is required in order to ensure that large residuals ($> n\varepsilon^2$) do not affect the loss, since the sigmoid function never gives zeroes for finite values. Furthermore the rectifier function needs to be continuous and differentiable, and is defined as $\phi(u) = u$ for $u \leq -\omega$, $\phi(u) = -(u^2/\omega + \omega)/2$ for $-\omega < u < 0$, and $\phi(u) = -\omega/2$ for $0 \leq u$, where $\omega > 0$ is a small constant. Equation 3.3 is a smoothed variant of Equation 3.1 (in Problem 3.1) and becomes equivalent to Equation 3.1 when $\beta \to \infty$ and $\omega \to 0^+$.

3.2.1 Graduated Optimisation

The minimisation of Equation 3.3 is done using *graduated optimisation*. The idea behind graduated optimisation is to iteratively solve a difficult optimisation problem by progressively increasing the complexity [44]. A natural way to increase the complexity of Equation 3.3 is by gradually increasing the β parameter. With $\beta = 0$ the problem is convex and equivalent to LASSO, while with $\beta \to \infty$ it corresponds to Problem 3.1, which is **NP**-hard. At each step (of increasing β values) the previous value of α is used as a starting point for finding the new minimum of Equation 3.3.

It is important that consecutive solutions (with increasing β values) are close enough for graduated optimisation to work, which is why we derive an

approximation ratio between solutions with different values of β. The derivation starts with the observation that the minimisation of Equation 3.3 can be rewritten as a maximisation of $-\beta$-$\texttt{Loss}(X, Y, \varepsilon, \lambda, \alpha)$. Furthermore the L1-regularisation term is unaffected by β and is omitted for simplicity.

Theorem 3.3. *Given $\beta_1, \beta_2 \geq 0$, such that $\beta_1 < \beta_2$, 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 $r_i = y_i - \alpha^\intercal x_i$ and $j \in \{1, 2\}$. The inequality $G_2(\alpha_2) \leq KG_2(\alpha_1)$ always holds where $\alpha_1 = \arg\max_\alpha G_1(\alpha)$, $\alpha_2 = \arg\max_\alpha G_2(\alpha)$, and $K = G_1(\alpha_1)/\left(G_2(\alpha_1) \min_r f_1(r)/f_2(r)\right)$ is the approximation ratio.*

Proof. Let us first argue the non-negativity of f_1 and f_2. 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 $r_i^* = y_i - \alpha_2^\intercal x_i$ and $k = \min_r f_1(r)/f_2(r)$, which allows us the rewrite and approximate:

$$G_1(\alpha_2) = \sum_{i=1}^{n} f_1(r_i^*) = \sum_{i=1}^{n} f_2(r_i^*) f_1(r_i^*)/f_2(r_i^*) \geq kG_2(\alpha_2).$$

Then $G_2(\alpha_2) \leq G_1(\alpha_2)/k \leq G_1(\alpha_1)/k \leq G_2(\alpha_1)G_1(\alpha_1)/(kG_2(\alpha_1))$, and the inequality from the theorem holds. □

The approximation ratio K in Theorem 3.3 can be used to select the sequence of β values ($\beta_0 = 0, \beta_1, \beta_2 \ldots$). At each step the next β value is chosen so that K stays within a bound specified by the parameter $r_{\max}$ in Algorithm 3.1.

3.2.2 Stopping Criteria

Iterating until $\beta \to \infty$ is not possible, so at some point the algorithm has to stop. The stop should trigger when $\sigma(\beta(\varepsilon^2 - r^2)) \approx H(\varepsilon^2 - r^2)$, i.e. the stop is dependent on the shape of the sigmoid function. The shape is determined by both β and ε. However ε is expected to change regularly so a stopping shape that is independent of ε is needed.

The goal is to find a function for $\beta_{\max}$ that works with arbitrary values of ε. Assume there exists a pair of values β_{opt} and ε_{opt} that results in an optimal stopping shape for the sigmoid function. The two sigmoid functions have the same *relative* shape if and only if $\sigma(\beta_{\max}(\varepsilon^2 - (p\varepsilon)^2)) = \sigma(\beta_{\text{opt}}(\varepsilon_{\text{opt}}^2 - (p\varepsilon_{\text{opt}})^2))$ for every value of $p \in \mathbb{R}$. The sigmoid function is strictly increasing and can

trivially be removed from both sides

$$\begin{aligned}\sigma(\beta_{\max}(\varepsilon^2 - (p\varepsilon)^2)) &= \sigma(\beta_{\text{opt}}(\varepsilon_{\text{opt}}^2 - (p\varepsilon_{\text{opt}})^2)) \\ \beta_{\max}\varepsilon^2(1-p^2) &= \beta_{\text{opt}}\varepsilon_{\text{opt}}^2(1-p^2) \\ \beta_{\max} &= \beta_{\text{opt}}\varepsilon_{\text{opt}}^2/\varepsilon^2 = c/\varepsilon^2,\end{aligned} \tag{3.4}$$

where $c = \beta_{\text{opt}}\varepsilon_{\text{opt}}^2$ is a constant. The actual value of c is empirically determined in Section 4.1.2.

3.2.3 Approximation Ratio

Theorem 3.2 not only shows that Problem 3.1 is **NP**-hard, but also that it is hard to approximate. Using the approximation ratio between two β-`Losses` we derive a new approximation ratio between Equation 3.3 and Equation 3.1. To do that we set $\beta_2 \to \infty$ so that $f_2(r) = -H(\varepsilon^2 - r^2)(r^2/n - \varepsilon^2)$. Additionally we introduce an ε^* such that $f_2^*(r) = -H(\varepsilon^{*2} - r^2)(r^2/n - \varepsilon^{*2})$, $G_2^*(\alpha) = \sum_{i=1}^n f_2^*(y_i - \alpha^\intercal x_i)$, and $\alpha_2^* = \arg\min_\alpha G_2^*(\alpha)$. This leads to a new approximation ratio K_{ε^*}:

Lemma 3.1. *The approximation ratio between α_1 and α_2^* is $K_{\varepsilon^*} = G_1(\alpha_1)/\left(G_2^*(\alpha_1)k_{\varepsilon^*}\right)$ 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 3.3 with the observation that $k_{\varepsilon^*} = \min_r f_1(r)/f_2^*(r) = \min_{r \le \varepsilon^*} -f_1(r)/(r^2/n - \varepsilon^{*2})$ which leads to $k_{\varepsilon^*} = \sigma(\beta_1(\varepsilon^2 - \varepsilon^{*2}))\phi(\varepsilon^{*2}/n - \varepsilon^2)/(\varepsilon^{*2}/n - \varepsilon^{*2})$. □

The first goal is to find a value of ε^* such that the approximation ratio K_{ε^*} is minimised. This can be written as $\varepsilon^* = \arg\min_{\varepsilon^*} K_{\varepsilon^*} = \arg\max_{\varepsilon^*} G_2^*(\alpha_1)k_{\varepsilon^*}$, which leads to

$$\varepsilon^* = \arg\max_{\varepsilon^*} -\sum_{i=1}^n H(\varepsilon^{*2} - r_i^2)(r_i^2/n - \varepsilon^{*2})\frac{\sigma(\beta_1(\varepsilon^2 - \varepsilon^{*2}))\phi(\varepsilon^{*2}/n - \varepsilon^2)}{\varepsilon^{*2}/n - \varepsilon^{*2}} \tag{3.5}$$

where $r_i = y_i - \alpha_1^\intercal x_i$. Thanks to the non-continuity of the Heaviside function the maximum can be found at $\varepsilon^* = r_j$ for some $j \in [n]$. Furthermore, with a large large enough n, so that $1/n \approx 0$, Equation 3.5 can be simplified to $\varepsilon^* \approx \arg\max_{r_j} \sum_{i=1}^n H(r_j^2 - r_i^2)\sigma(\beta_1(\varepsilon^2 - r_j^2))$. We can also assume, without a loss of generality, that the residuals are sorted so that $r_1^2 \le r_2^2 \le \ldots \le r_n^2$, which means that $\sum_{i=1}^n H(r_j^2 - r_i^2) = j$ and $\varepsilon^* \approx \arg\max_{r_j} j \cdot \sigma(\beta_1(\varepsilon^2 - r_j^2))$.

We call α_2^* (the optimum for Problem 3.1 with ε^*) the matching solution. If the data is subsampled to a constant size then Equation 3.3 has a constant approximation ratio for the matching solution.

Theorem 3.4. *The matching solution satisfies the inequality* $G_2^*(\alpha_2^*) \leq K^* G_2^*(\alpha_1)$ *where* $K^* = \mathcal{O}(\log n)$ *is the approximation ratio.*

Proof. By definition $K_{r_t} \geq K_{\varepsilon^*}$ where $t \in [n]$, $K_{r_t} = G_1(\alpha_1)/(-\sum_{i=1}^n H(r_t^2 - r_i^2)(r_i^2/n - r_t^2)k_{r_t})$, and $k_{r_t} = \sigma(\beta_1(\varepsilon^2 - r_t^2))\phi(r_t^2/n - \varepsilon^2)/(r_t^2/n - r_t^2)$. This is the same as in Lemma 3.1. Assuming the residuals are sorted so that $r_1^2 \leq r_2^2 \leq \ldots \leq r_n^2$ (the same as above), $K_{r_t} \geq K_{\varepsilon^*}$ can be rearranged as

$$\begin{aligned}\sigma(\beta_1(\varepsilon - r_t^2)) &\leq G_2^*(\alpha_1)k_{\varepsilon^*}/(-\sum_{i=1}^{n} H(r_t^2 - r_i^2)(r_i^2/n - r_t^2)\phi(r_t^2/n - \varepsilon^2)/(r_t^2/n - r_t^2)) \\ &\leq G_2^*(\alpha_1)k_{\varepsilon^*}/(-t(r_t^2/n - r_t^2)\phi(r_t^2/n - \varepsilon^2)/(r_t^2/n - r_t^2)) \\ &= -G_2^*(\alpha_1)k_{\varepsilon^*}/(t\phi(r_t^2/n - \varepsilon^2)).\end{aligned}$$

Inserting this into G_1 yields

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

And combined with K_{ε^*} from Lemma 3.1 this leads to
$K_{\varepsilon^*} = G_1(\alpha_1)/(G_2^*(\alpha_1)k_{\varepsilon^*}) \leq G_2^*(\alpha_1)k_{\varepsilon^*}(\log n+1)/(G_2^*(\alpha_1)k_{\varepsilon^*}) = \log n+1.$ □

3.3 Algorithm

This section presents the algorithm, SLISE, that approximately solves Problem 3.1, using the approximations described above. Following the high-level pseudocode in Algorithm 3.1, SLISE starts with selecting initial values for the linear model α and the sigmoid steepness β (line 3). The second phase of SLISE is the graduated optimisation (lines 4–7).

3.3.1 Initialisation

In total four alternative initialisation schemes are presented as pseudocode in Algorithm 3.2 and 3.3. These are later compared empirically in Section 4.1.1. The first initialisation alternative (Algorithm 3.2, lines 2–5) is to use LASSO-regression. LASSO-regression is the non-robust counterpart to SLISE. It is also straight forward to implement since it only requires $\beta = 0$ (LASSO is a convex

1 **Parameters:** (1) Dataset $X \in \mathbb{R}^{n \times d}$, $Y \in \mathbb{R}^n$,
(2) error tolerance ε,
(3) regularisation coefficient λ,
(4) maximum sigmoid steepness $\beta_{\max}$,
(5) target approximation ratio $r_{\max}$
2 **Function** `SLISE`(X, Y, ε, λ, $\beta_{\max}$, $r_{\max}$)
3 $\alpha, \beta \leftarrow$ `Initialise`(X, Y, ε, $r_{\max}$)
4 **while** $\beta < \beta_{\max}$ **do**
5 $\alpha \leftarrow$ `OWL-QN`(β-`Loss`, X, Y, ε, λ, α)
6 $\beta \leftarrow \beta'$ such that `AppoximationRatio`(X, Y, ε, β, β', α) $= r_{\max}$
7 $\alpha \leftarrow$ `OWL-QN`(β-`Loss`, X, Y, ε, λ, α)
8 **Result:** α

Algorithm 3.1: The SLISE algorithm.

problem so the initial α does not matter). However setting $\beta = 0$ removes all notion of a subset, which has a major impact on the loss function.

With $\beta > 0$ the problem is no longer convex and the choice of α becomes important. The second initialisation scheme (Algorithm 3.2, lines 6–9) uses OLS-regression to find α and the approximation ratio (Theorem 3.3) to select β (further explained below). However neither OLS nor LASSO is robust.

The third alternative (Algorithm 3.2, lines 10–13) is similar to the second, but uses a constant α instead. The constant α consists of only zeros, i.e. starting from a super-sparse solution. The β is also chosen based on the approximation ratio (Theorem 3.3).

The final alternative (Algorithm 3.3) is inspired by another robust regression method, FAST-LTS [54]. A number u_{init} of candidates are generated of which the best one is selected. The candidates are generated by drawing random subsets (X_S, Y_S) and fitting linear models to them (using OLS-regression, lines 6–7). 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 the number of dimensions, and u the number of candidates. If d is large then u would also have to be large to compensate for the decreased probability. To alleviate this issue we create smaller subsets (line 9) and fit the model using PCA (line 10) when d is large. The best candidate is the one that minimises the β-`Loss` (lines 11-14).

1 **Parameters:** (1) Dataset $X \in \mathbb{R}^{n \times d}$, $Y \in \mathbb{R}^n$,
(2) error tolerance ε,
(3) target approximation ratio $r_{\max}$
2 **Function** `InitialiseLasso`(X, Y)
3 $\alpha \leftarrow$ `OrdinaryLeastSquares`(X, Y)
4 $\beta \leftarrow 0$
5 **Result:** α, β

6 **Function** `InitialiseOLS`(X, Y, ε, $r_{\max}$)
7 $\alpha \leftarrow$ `OrdinaryLeastSquares`(X, Y)
8 $\beta \leftarrow \beta'$ such that `AppoximationRatio`(X, Y, ε, *0*, β', α) $= r_{\max}$
9 **Result:** α, β

10 **Function** `InitialiseZeros`(X, Y, ε, $r_{\max}$)
11 $\alpha \leftarrow \mathbf{0}$
12 $\beta \leftarrow \beta'$ such that `AppoximationRatio`(X, Y, ε, *0*, β', α) $= r_{\max}$
13 **Result:** α, β

Algorithm 3.2: Deterministic initialisation schemes.

3.3.2 Optimisation

With the initial values for α and β we now perform graduated optimisation, where we gradually increase the value of β until we reach $\beta_{\max}$ (Algorithm 3.1, lines 4–6). At each iteration we find the α that minimises the β-`Loss` from Equation 3.3 using the current value of β (line 5). This optimisation is done with OWL-QN [57] which is a quasi-Newton optimisation method with built-in L1-regularisation. We then increase β (line 6) such that the approximation ratio K from Theorem 3.3 between the new and old β equals $r_{\max}$. The approximation ratio K is translated into code in Algorithm 3.4.

In Equation 3.3 we use the rectifier function ϕ to ensure negativity. This function requires a constant ω that we choose to have a magnitude less than the smallest positive value that can be expressed with a floating-point number. This makes $\phi(u)$ numerically equivalent to $\min(0, u)$, which is used in Algorithm 3.4. A minor side-effect of this simplification is that lines 8–10 are needed to avoid division by zero (which would not happen with the proper ϕ).

Parameters: (1) Dataset $X \in \mathbb{R}^{n \times d}$, $Y \in \mathbb{R}^n$,
(2) error tolerance ε,
(3) target approximation ratio $r_{\max}$,
(4) number of candidates u_{init}
Function `InitialiseCandidates`(X, Y, ε, $r_{\max}$, u_{init})
  $\alpha, \beta, l \leftarrow \mathbf{0}, 0, 0$
  for $i = 1, \ldots, u_{\text{init}}$ **do**
    if $d \leq 12$ **then**
      $X_S, Y_S \leftarrow$ `RandomSubset`(X, Y, $size = d$)
      $\alpha' \leftarrow$ `OrdinaryLeastSquares`(X_S, Y_S)
    else
      $X_S, Y_S \leftarrow$ `RandomSubset`(X, Y, $size = 10$)
      $\alpha' \leftarrow$ `InversePCA(OrdinaryLeastSquares(PCA(`X_S`),` Y_S`))`
    if β-`Loss`(X, Y, ε, 0, α') $< l$ **then**
      $\alpha \leftarrow \alpha'$
      $\beta \leftarrow \beta'$ such that `AppoximationRatio`(X,Y, ε,0,β',α)=$r_{\max}$
      $l \leftarrow \beta$-`Loss`(X, Y, ε, 0, α)
  Result: α, β

Algorithm 3.3: Non-deterministic initialisation scheme.

3.3.3 Complexity

The complexity of Algorithm 3.1 is different than the complexity of Problem 3.1 and can be derived from pseudocode above. The initialisation has at most a complexity of $\mathcal{O}(nd^2u_{\text{init}})$ which is less than the complexity of the optimisation. There are three main contributors to the complexity of the optimisation; the loss function, OWL-QN, and the graduated optimisation. The evaluation of the loss function has a complexity of $\mathcal{O}(nd)$, due to the multiplication between the linear model α and the data-matrix X. OWL-QN has a complexity of $\mathcal{O}(dp_o)$, where p_o is the number of iterations. Graduated optimisation is also an iterative method $\mathcal{O}(p_g)$, but it only adds the approximation ratio calculation, which is not dominant $\mathcal{O}(nd)$. Combining these complexities yields a complexity of $\mathcal{O}(nd^2p)$ for SLISE, where $p = p_o + p_g$ is the total number of iterations.

Parameters: (1) Dataset $X \in \mathbb{R}^{n \times d}$, $Y \in \mathbb{R}^n$,
(2) error tolerance ε,
(3) sigmoid steepness β_1, β_2,
(4) linear model α
Function `AppoximationRatio(`X, Y, ε, β_1, β_2, α`)`
  $f \leftarrow$ `function(`r, β`)` : $-\sigma(\beta(\varepsilon^2 - r^2))$
  $\phi \leftarrow$ `function(`r`)` : $\min(0, r^2/n - \varepsilon^2)$
  $k \leftarrow$ `minimize(`$f(r, \beta_1)/f(r, \beta_2)$, by adjusting r`)`
  for $i = 1, \ldots, n$ **do**
    $r_i \leftarrow y_i - \alpha^\intercal x_i$
  if $\sum_{i=1}^n \phi(r_i) = 0$ **then**
    $K \leftarrow \sum_{i=1}^n f(r_i, \beta_1)/(\sum_{i=1}^n f(r_i, \beta_2) \cdot k)$
  else
    $K \leftarrow \sum_{i=1}^n f(r_i, \beta_1)\phi(r_i)/(\sum_{i=1}^n f(r_i, \beta_2)\phi(r_i) \cdot k)$
  Result: K

Algorithm 3.4: Approximation ratio calculation.

4. Experiments

The experiments have three goals; find good default parameters, compare SLISE to other robust regression methods, and show that SLISE can produce reasonable explanations for black box models. This chapter is thus split into three parts corresponding to those goals. In Section 4.1 suitable default values for the parameters and a robust default initialisation method are determined empirically. Section 4.2 demonstrates that SLISE is a robust regression method, that it scales to large datasets better than competing methods, and that it gives the best solution to Problem 3.1. Finally in Section 4.3 different ways of utilising SLISE for explaining black box decisions are demonstrated, this also includes a brief comparison to another local explanation method.

SLISE and all the experiments are implemented in R [43] (version 3.5.1). The experiments were run on a high-performance cluster [22], where each job were allocated 4 cores from an Intel Xeon E5-2680 processor running at 2.4 GHz and 16 GB of RAM. The source code for SLISE and all the experiments is released under an open source license and is available from `http://www.github.com/edahelsinki/slise`.

Datasets The experiments use a combination of real and synthetic datasets. The real datasets are handwritten digits (EMNIST [15]), movie reviews (IMDB [41]), and particle jets (PHYSICS, [1]). SYNTHETIC datasets are used when specific dimensions are needed and are generated as follows. The data matrix $X \in \mathbb{R}^{n \times d}$ is created by sampling from a normal distribution with zero mean and unit variance. The response vector $Y \in \mathbb{R}^n$ is created by $y_i \leftarrow a^\intercal x_i$ plus some normal noise with zero mean and 0.05 variance, where $a \in \mathbb{R}^d$ is one of nine linear models drawn from a uniform distribution between -1 and 1. Each of the nine model creates 10% of the Y-values, except one that creates 20% of the Y-values. This larger chunk should enable robust regression methods to find the corresponding model. An overview of all the datasets is shown in Table 4.1.

Dataset	Items	Dimensions	Type	Classifier
EMNIST	40 000	672	image	CNN
IMDB	25 000	1000	text	LR, SVM
PHYSICS	260 000	4 / 324	tabular / image	NN/ CNN
SYNTHETIC	n	d	–	–

Table 4.1: The datasets used in the experiments. The synthetic dataset can be generated to any desired size.

Pre-Processing SLISE uses LASSO regularisation for both introducing sparsity and regularisation. Since LASSO sums the absolute values of the coefficients it is important that the variables have roughly the same magnitude [62]. Normalising the variables also makes it easier to compare the relative importance of values when interpreting the explanations. Thus the data matrices X for all datasets have been normalised.

EMNIST images have 28×28 pixels and the values of the pixels are scaled to be in the range $[-1, 1]$. Some of the pixels have the same value in all of the images (mostly in the corners), these pixels are removed and the images are flattened to vectors of length 672. The tabular version of the PHYSICS dataset is normalised column-wise by subtracting the mean and dividing by the standard deviation, while the image version of the PHYSICS dataset is flattened the same way as the EMNIST dataset. The pipeline for the texts in the IMDB dataset starts with case normalisation, removal of punctuation, removal of stop words and stemming. Then, the 1000 most common words are used to form a bag-of-words model. The obtained word frequencies are divided by the most common word in each review to account for different review lengths.

The Y-vectors for all datasets are also normalised to allow for easier selection of error tolerance ε. The values are linearly scaled to be approximately within $[-0.5, 0.5]$ based on the distance between the 5^{th} and 95^{th} quantiles. Note that this is only possible to do automatically if one can be absolutely certain that less than 5% of the y-values have abnormally large or small values (such as with classifiers giving probabilities from the range $(0, 1)$).

Classifiers Five high-performance classifiers have been fitted to the real-world datasets (see Table 4.1). The classifiers are two convolutional neural networks (CNN), a normal neural network (NN), a logistic regression (LR), and a support vector machine (SVM). The outputs from these classifiers are used in every

Dataset	**Task**	ε	λ	**Subsampling**
EMNIST	explanation	0.1	2.0	Digits 2, 4, 5, 7, and 8
IMDB	explanation	0.1	1.0	Subsampled to 10 000 $\times$ 1000
PHYSICS	explanation	0.1	1.0	Subsampled to 10 000 $\times$ 4
SYNTHETIC	regression	0.1	0.5	Size 1000 $\times$ 30

Table 4.2: The datasets used for the parameter experiments.

experiments involving the datasets, not only the explanations. The reason for this is that regression (SLISE) works better if the response is real-valued rather than categorical. Categorical values miss the nuances and additional detail offered by spread-out real-values. As described in Section 3.1.2 the class probabilities p_i from the classifiers are transformed into real numbers using the logit transformation $y_i = \log(p_i/(1 - p_i))$.

4.1 Parameters

The two most important parameters for SLISE are the error tolerance ε and the sparsity coefficient λ. These depend on the use-case and the dataset, and therefore they *must* be manually adjusted. Most of the experiments in this thesis use $\varepsilon = 0.1$, which can be seen as a 10 % error tolerance due to the scaling of Y described above, and the default sparsity is $\lambda = 0$, i.e. no sparsity.

The default values of the other parameters, and which initialisation scheme to recommend, are selected based on empirical evidence. Specifically, the selection is based on the value of the loss function and the running time. Furthermore the datasets in Table 4.1 are randomly subsampled. The use of multiple smaller datasets better captures the variability of the losses and the running times with respect to different choices. The properties of the smaller datasets are given in Table 4.2. For each parameter value the experiments have been run 160 times (40 times for each of the 4 primary datasets).

4.1.1 Initialisation

In Section 3.3.1 four different schemes for selecting the initial values for the linear model α and sigmoid steepness β are presented. Table 4.3 shows medians along with the 5^{th} and 95^{th} quantiles for the loss and the running times for every

Dataset		**Loss**			**Time [s]**	
Method	5^{th}	Median	95^{th}	5^{th}	Median	95^{th}
emnist						
LASSO	-45.18	-41.86	-38.43	95.24	118.59	182.60
OLS	-45.22	-41.77	12.15	5.09	100.62	160.65
ZEROS	-45.18	-41.79	-38.54	97.21	113.08	180.42
CANDIDATES	-45.16	-41.87	-38.58	91.15	106.62	167.13
imdb						
LASSO	-45.80	-45.42	-44.50	144.68	187.65	314.81
OLS	-45.86	-45.39	-44.81	119.51	148.57	203.66
ZEROS	-45.78	-45.48	-44.71	131.66	183.28	282.87
CANDIDATES	-45.84	-45.43	-44.69	134.85	171.77	278.32
physics						
LASSO	-6.88	-6.56	-5.71	0.05	0.09	0.34
OLS	-6.90	-6.56	-5.71	0.04	0.07	0.13
ZEROS	-6.88	-6.56	-5.71	0.05	0.08	0.18
CANDIDATES	-6.90	-6.56	-5.71	0.11	0.15	0.22
synthetic						
LASSO	-3.95	-3.53	-3.33	0.10	0.13	0.35
OLS	-3.95	-3.53	-3.33	0.10	0.12	0.15
ZEROS	-3.95	-3.53	-3.33	0.10	0.12	0.14
CANDIDATES	-3.95	-3.53	-3.33	0.30	0.33	0.50

Table 4.3: Comparing different initialisation methods for SLISE. Lower values are better for both losses and times.

combination of dataset and initialisation scheme. No particular method stands out, which indicates that the combination of graduated optimisation and OWL-QN yields good performance overall. However, SLISE is an approximation and can be only be guaranteed to find a local optima (in contrast to finding the global optima), so knowing possible pitfalls is important when selecting the initialisation.

Both LASSO and OLS are non-robust so even a single outlier can lead to arbitrarily large deviations [32], which may lead to inescapable local optima. This can be seen in Table 4.3 where the loss value for SLISE with the OLS initialisation is positive (95^{th} quantile). This is a clear indication of a bad

local optima. Note that LASSO reduces to OLS with $\lambda = 0$, so neither of these initialisation schemes can be recommended as default.

Starting from a vector of ZEROS is more likely to end up in a sparse local optima (since the starting point is super sparse). But with large enough λ the initial zero-vector becomes a local optima. This makes it easy to detect when the optimisation has failed to escape a local optima ($\|\alpha\|_1 \approx 0$).

Another sign of a bad local optima is if the progression of growing βs is really short, i.e., if the optimum for the β-`Loss` with $\beta \approx 0$ is almost the optimum for the β-`Loss` with $\beta \approx \beta_{\max}$. If the relation between X and Y is very linear then this is not an issue, but otherwise this is usually a bad local optima that the optimisation cannot escape. This can be detected by the β-`Loss` being close to zero, or even positive, due to the subset being tiny.

The initialisation schemes above only provide a single starting point so even if a bad local optima could be detected there is no solution. The idea behind the CANDIDATES initialisation scheme is based on detecting very bad local optima and trying to select the best of the generated CANDIDATES, based on the β-`Loss`. But that criteria might not be the best indicator of optimality, since the matching ε^* for all starting β:s will be large (see Section 3.2.3) which means that the local optima might be in a region better suited to a larger ε.

The number of initial CANDIDATES is a parameter u_{init}. A larger number increases the likelihood that a good candidate is found. The results in the rightmost column of Figure 4.1 show that it is easy to find a sufficiently large number. The default is to use $u_{\text{init}} = 500$ (the same as in FAST-LTS [54]), which is large enough but still results in a reasonably low running time.

To summarise the findings in Table 4.3: When $n \gg d$ all initialisation schemes lead to the same result, but even with the high-dimensional datasets no initialisation is significantly better than the others. The OLS and LASSO initialisation schemes are non-robust. The CANDIDATES initialisation scheme is non-deterministic and adds quite a bit of overhead to the small datasets, but the running times for those are still well below one second. The ZEROS initialisation is another viable option, but it cannot correct for clearly bad initialisations. Thus we recommend the CANDIDATES initialisation scheme.

4.1.2 Iterations

SLISE incorporates two iterative optimisation methods, OWL-QN and graduated optimisation. Increasing the number of iterations leads to better results, but

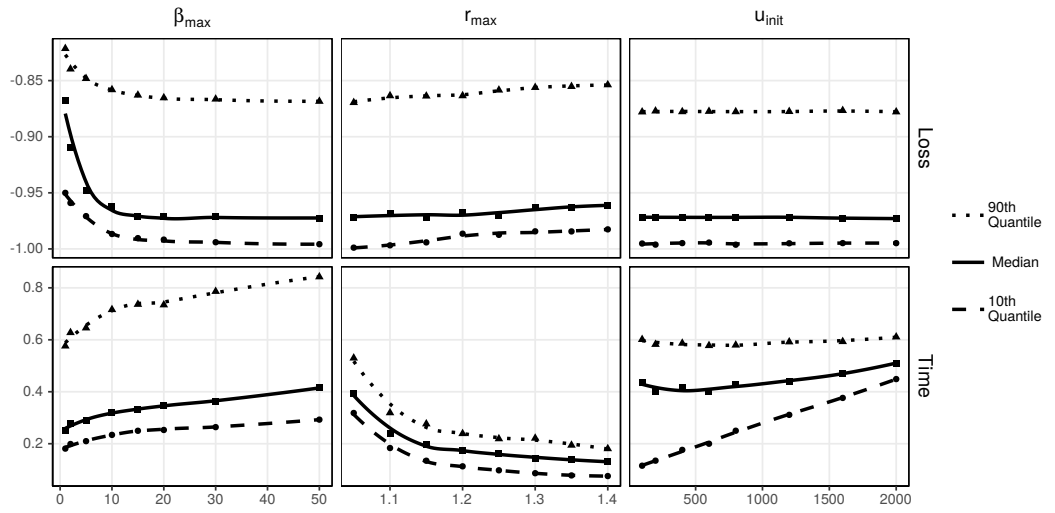


Figure 4.1: Losses and running times for different parameter values. The values have been scaled dataset-wise to facilitate comparisons (division by the maximum absolute value). In each graph all parameters are kept constant at default values, and only the parameter being investigated is varied. Lower values are better.

beyond a point there are clear diminishing returns. The number of iterations in graduated optimisation is determined by the target approximation ratio $r_{\max} > 1$. A larger value results in fewer iterations, but it has to be small enough to ensure that consecutive iterations have similar optima. However the optima are found using OWL-QN which is powerful enough to tolerate larger values of $r_{\max}$.

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 limit 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 converge each iteration.

The results in the second column of Figure 4.1 indicate that values larger than $r_{\max} \geq 1.2$ looses accuracy. However $r_{\max}$ has to be determined together with $i_{\max}$ due to the interaction between the two optimisation methods. Figure 4.2 shows the results from an experiment varying both parameters. The combination of $r_{\max} = 1.15$ and $i_{\max} = 200$ yields good results while still being quite time efficient. Note that the last OWL-QN optimisation (when $\beta = \beta_{\max}$) is allowed to run for four times as many iterations as the optimisation with smaller β:s, since the result of that optimisation is the result of SLISE.

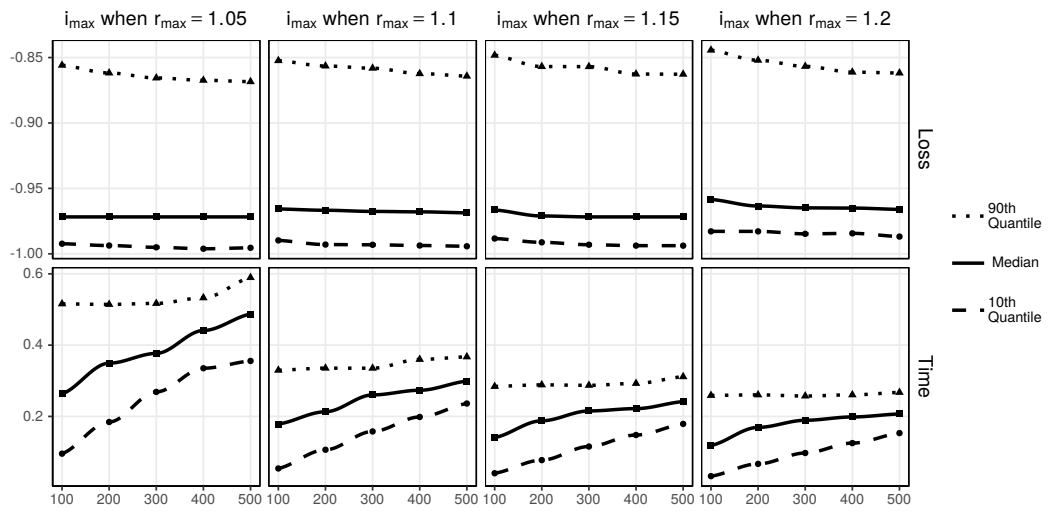


Figure 4.2: Losses and running times for different values of the two parameters that control the number of iterations. The values have been scaled dataset-wise by dividing with the maximum absolute value. Lower values are better.

Stopping parameter The iterative optimisation ends when β reaches a specified value. It is sufficient that this stopping parameter $\beta_{\max}$ is large enough to make the shape of the sigmoid function essentially equivalent to a Heaviside function. As shown in Section 3.2.2, in order to make the shape of the sigmoid only depend on $\beta_{\max}$ it has to be defined as $\beta_{\max} = c/\varepsilon^2$ where c is a constant. The results in the leftmost column of Figure 4.1 show that the loss saturates before $\beta_{\max} = 25/\varepsilon^2$, and any larger value just adds running time.

4.1.3 Summary

Based on the empirical evaluations above, the CANDIDATES initialisation scheme is the most robust one and is the recommended one for SLISE. The default values for the parameters are $\beta_{\max} = 25/\varepsilon^2$, $r_{\max} = 1.15$, $i_{\max} = 200$, and $u_{\text{init}} = 500$. These defaults are used in all the experiments in this thesis.

4.2 Robust Regression

The goal of the regression experiments is to compare SLISE to other robust regression methods and show that SLISE optimises Problem 3.1. The other methods in the comparisons are introduced in Section 2.1, but a short summary can be seen in Table 4.4. All methods are used with default settings, except

Algorithm	Robust	Sparse	R-Package
SLISE	yes	yes	
FAST-LTS [54]	yes	no	robustbase
SPARSE-LTS [4]	yes	yes	robustHD
MM-ESTIMATOR [67]	yes	no	MASS
MM-LASSO [59]	yes	yes	pense
LAD-LASSO [66]	maybe	yes	MTE
LASSO [62]	no	yes	glmnet

Table 4.4: Properties of the regression methods used in the experiments.

that the LTS variants are used with subsets of size $n/2$ for the largest possible breakdown value. Not all methods methods support sparsity, and when they do, finding an equivalent regularisation parameter, λ, is difficult. Hence, unless otherwise noted, all methods are used with *almost* no sparsity ($\lambda = 10^{-6}$).

4.2.1 Scalability

First the scalability of the different methods is investigated. Most of the methods have the same theoretical complexity as SLISE $\mathcal{O}(nd^2p)$, but even then the number of iterations p varies wildly. Thus an empirical evaluation is merited. The emphasis is on scaling to large datasets, since that is when the differences start to show and matter.

The experiment uses the SYNTHETIC dataset with varying numbers of either samples n or dimensions d while the other one is fixed. The methods that support sparsity are used with different $\lambda \in \{0, 0.01, 0.1, 0.5\}$. Each dataset, parameter, and size combination is run five times and the mean running times are presented. We use a time limit of 10 minutes so any so any calculation that takes more than that, or more than 16GB of RAM, is cancelled.

In the left plot of Figure 4.3 the number of samples is varied ($n \in \{500,$ 1 000, 5 000, 10 000, 50 000, 100 000$\}$) and the number of dimensions is fixed ($d = 100$). SLISE outperforms all robust regression methods except FAST-LTS on large datasets. This is due to FAST-LTS being the only one to subsample the data, keeping the running time constant when the number of samples increases.

Figure 4.3 (right) shows the result from varying the number of dimensions ($d \in \{10,$ 50, 100, 500, 1 000$\}$) with a fixed number of samples ($n = 10\ 000$). SLISE outperforms all other robust regression methods when $d > 10$ and is able

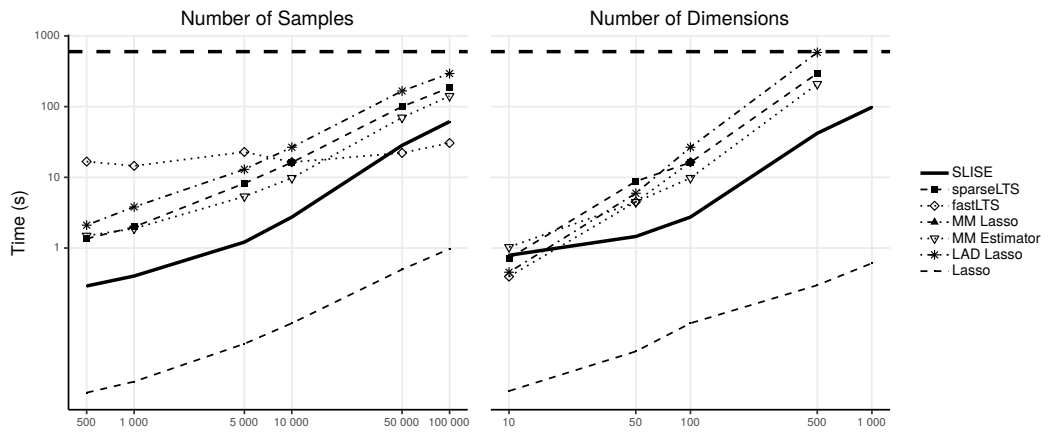


Figure 4.3: Running times in seconds. Left: Varying the number of samples n with a fixed number of dimensions $d = 100$. Right: Varying the number of dimensions d with a fixed number of samples $n = 10\ 000$. The cut-off time is shown using a dashed horizontal line at $t = 600$ seconds.

to finish in less than 100 seconds, even for a massive $10\ 000 \times 1\ 000$ dataset (none of the other robust regression methods finished within the time limit).

4.2.2 Robustness

The breakdown value from Section 2.1 gives theoretical values for robustness. To test the robustness in practise the PHYSICS dataset is used. In the left plot of Figure 4.4 a fraction of the items are corrupted by replacing the response variable with random noise (uniformly distributed between $\min(Y)$ and $\max(Y)$). Each of the robust regression methods are trained on the corrupted dataset and then the sum residuals for the uncorrupted dataset is calculated for each model. If the method is robust then the sum of residuals will not increase as the noise fraction increases, until the fraction becomes too large and the method breaks down (as expected).

The Y-values in the PHYSICS dataset stem from a non-linear classifier, which means that the linear robust regression methods might detect outliers even before explicit outliers are added. This is the reason why the MM-methods start breaking down before their theoretical 0.5 and why SLISE initially gives larger residuals than the rest (since internally SLISE ignores some residuals). Another interesting observation in the left plot of Figure 4.4 is that LAD-LASSO seems to be quite robust, but this is due to the outliers having the same magnitude as the non-outliers. SLISE is able to handle a larger noise fraction

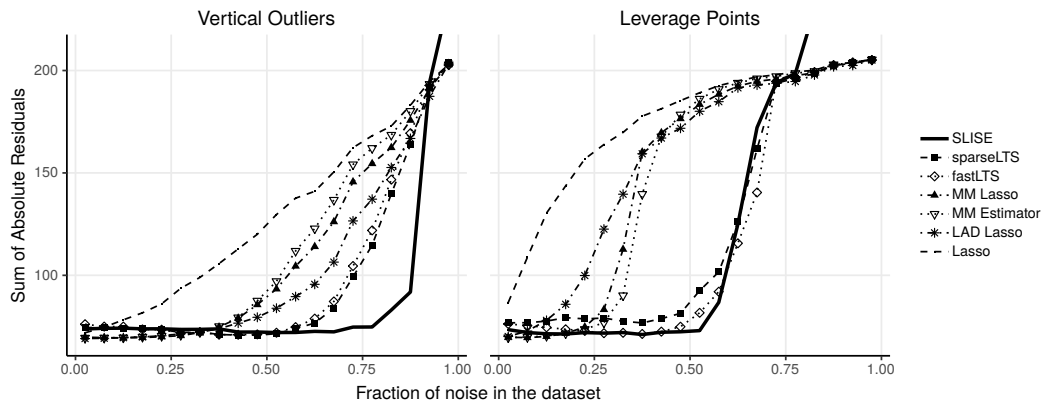


Figure 4.4: Robustness to outliers. The x-axis shows the fraction of noise and the y-axis the sum of the residuals. Small residuals as the noise increases indicate a robust method.

than any other method due to the varying size of the subset.

The left plot in Figure 4.4 showcases *vertical outliers*, i.e. changed Y-values, but the literature also suggests other types of outliers, such as *leverage points* [55] where the X-values are changed. In the right plot the X-values are replaced with values sampled from a normal distribution with both the mean and the standard deviation being 4. SLISE and LTS performs best, and both break down slightly above 0.5. Figure 4.4 also shows why a worst case measure such as the breakdown value is useful since LAD-LASSO breaks down almost immediately with leverage points.

4.2.3 Optimality

The two experiments above demonstrate that SLISE is competitive with other robust regression methods. Left to demonstrate is that SLISE finds a good solution to Problem 3.1. Since the optimal solution is unknown SLISE is compared to the other methods with the assumption that it provides a significantly better solution. The experiment uses a SYNTHETIC dataset of size $n = 1\ 000$ and $d = 30$. All robust regression methods are fitted to this dataset and the resulting models are evaluated using Equation 3.1 with different values of ε. The results in Figure 4.5 are normalised with respect to the LASSO model (all loss-values are divided by the corresponding LASSO-loss) and hence the loss for LASSO appears constant. SLISE finds the best solutions for $\varepsilon = 0.1$, which is expected since that is the goal of the algorithm.

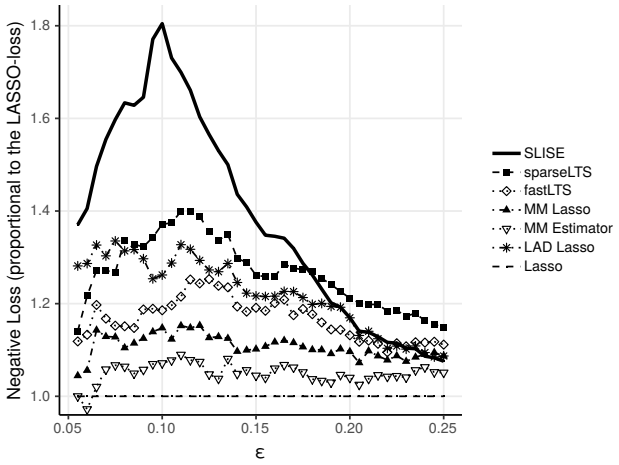


Figure 4.5: Optimality of SLISE. Negative loss-values are shown, normalised with respect to the corresponding loss for OLS. Higher values are better, especially for SLISE around $\varepsilon = 0.1$.

4.3 Explanations

One of the most interesting applications of SLISE is to explain outcomes from black box models. As is outlined in Section 2.2 SLISE creates post-hoc local explanations. Following a pattern from previous works the explanations are local approximations of the complex model using a simple model. Two desired properties for these simple models are simplicity and smallness [38], which SLISE accomplishes through sparse linear models. Setting SLISE apart from previous works is the adherence to the data and the meaningful neighbourhoods in the form of subsets.

The first experiment is a quick comparison to a state-of-the-art explanation method to show that SLISE provides similar explanations, while the rest of the experiments aim to demonstrate different ways of utilising SLISE for explanations. The standard way of presenting local explanations would be to show the approximating model, which allows the user to infer properties that are important for the classification. With SLISE the demonstrations go beyond that and also incorporate the subset and properties of the algorithm in order to extract more information from the black box model.

The experiments use three different types of data; text, image, and tabular. The previous works presented in Section 2.2.4 usually mutate the sample being explained with mutations that are often specific to the data type, or even dataset. In contrast, SLISE works the same way for any type of data, only

lime Although it might seem a bit bizarre to see a ...
simply enjoy the fun. Mary is a street kid ... older William ...
at the time & just on the cusp ... too much of the plot ...
great fun to watch ... are very good ... street scenes ...

slise Although it might seem a bit bizarre to see a ...
simply enjoy the fun. Mary is a street kid ... older William ...
at the time & just on the cusp ... too much of the plot ...
great fun to watch ... are very good ... street scenes ...

Figure 4.6: Comparing the explanations from LIME (top) and SLISE (bottom) on the IMDB dataset with a logistic regression classifier. Parts without any weight from either model are left out for brevity.

requiring the samples to be turned into real-vectors. This allows for a more consistent behaviour across data domains.

4.3.1 Classification of Text

The most cited post-hoc local explanation method is LIME [49], which also provides explanations in terms of sparse linear models. Thus a comparison between SLISE and LIME is the first step in showing the viability of SLISE as an explanation method. The comparison uses the IMDB dataset and a logistic regression classifier trying to determine whether the reviews are positive or negative. Since both methods provide sparse linear models the sparsity parameters have been chosen to approximately match (for SLISE $\lambda = 0.75$ and for LIME the number of features is 8).

Figure 4.6 show abbreviated explanations from both methods on a review from the dataset. The explanations are quite similar and mainly highlight the same words, but the LIME-explanation surprisingly shows that the word `street` is important. Street actually has a positive coefficient in the logistic regression model, but in the dataset the word is quite rare only occurring in 2.6% of the reviews. SLISE takes this into account and priorities more general words such as `enjoy`, `fun`, and `great`. If the frequency of the word `street` were to change (e.g. by adding or removing review with the word) SLISE could adapt while the LIME explanation stays the same.

Figure 4.7 shows the SLISE explanation for another review. Since the data is in a bag-of-words form the classifier, a support vector machine, is not able to model the interaction between the words `not` and `bad`. This causes the SVM to

slise ... in reality, Shemp wasn't really that bad. ...
At least he wasn't as bad as Joe Besser. ...
The slapstick gags are hilarious, especially this one scene ...

Figure 4.7: SLISE explaining how the SVM does not model the phrase `not bad` as positive.

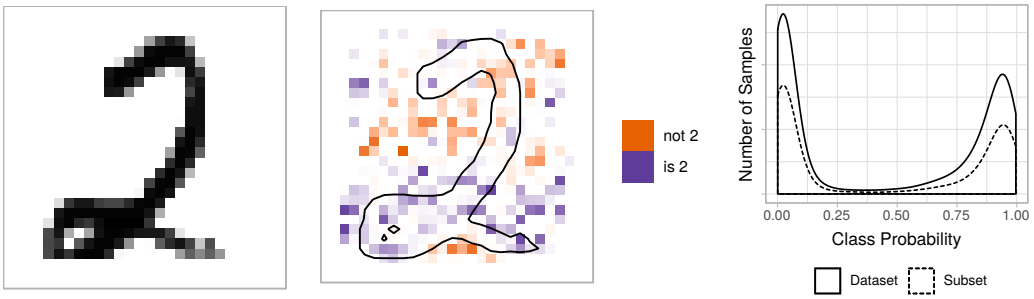


Figure 4.8: Left: The digit being explained. Middle: Salience map showing the regression weights of the linear model found using SLISE. The instance being explained is overlaid in the image. Purple colour indicates a weight supporting positive classification of a 2, and orange colour indicates a weight not in support of classifying the item as a 2. Right: Class probability distributions for the full dataset and for the found subset S.

falsely classify this review as negative. The explanation is able to explain this mistake by giving negative weight to the words `wasn't` and `bad` in contexts where the meaning is actually positive.

4.3.2 Classification of Images

The EMNIST dataset is both high-dimensional and complex while at the same time easy to visualise (as images) since it consists of handwritten characters (of which only the digits are used in this thesis, matching the original MNIST dataset [36]). With this motivation EMNIST is the primary dataset when demonstrating different ways of utilising SLISE for explanations.

The black box providing the classifications is a simple convolutional neural network with batch normalisation and max-pooling. Most experiments explain the same digit, a 2 shown in Figure 4.8. This is done deliberately so that the insights from every explanation can be compared to and added to previous insights. SLISE is used with default parameters, $\varepsilon = 0.1$, and $\lambda = 2$. Furthermore the dataset is subsampled so that 50% of the images are 2s and 50% are of other digits (`0-1`, `3-9`).

Approximation as explanation SLISE approximates the opaque model with a sparse linear model. The model coefficients can be used to deduce which features are important for the classification. The middle image in Figure 4.8 shows a *saliency map* where every pixel corresponds to a coefficient in the α-vector. The colour of the saliency map indicate whether a black pixel supports (purple) or opposes (orange) the classification as a 2. The saturation tells how important the pixel is for classification (the relative weight in the α-vector).

The most striking feature in the saliency map is the horizontal line at the bottom. This is indeed quite characteristic for 2s, so it is only natural if the classifier uses it to detect 2s. The other feature in the saliency map is the orange area from the middle left to the top right, where 2s tend to be mostly empty. Note that this explanation is not applicable to all 2s, e.g., the small orange area at the bottom slightly contradicts 2s with a straight bottom line.

In order to really be able to deduce features that distinguish one class (e.g., 2) from others one need to verify that the subset given by SLISE contains items from both classes (e.g., both 2s and other digits). The rightmost plot in Figure 4.8 shows the class probability distribution for the dataset and for the found subset, and the subset does indeed contain items from both classes. All of the explanations shown in this paper have subsets that contain both classes. In case that does not happen one can try increasing ε, decreasing λ, or spreading out the Y-values, e.g., by applying a logit transformation to probabilities (as described in Section 3.1.2).

Subset as explanation An unique feature of SLISE compared to other local explanation methods is that the "neighbourhood", or subset, consists of real data items. It is thus interesting to examine how this subset can be used to extract information. The approximation is valid for all items in the subset, with an error tolerance of ε. Figure 4.9 shows six digits from the subset overlaid on the saliency map. The 9 and the 8 are both clearly not classified as 2s (the probabilities are 0.008 and 0.188) but the figure also shows why the classifier considers the 8 to be more like a 2 than the 9 (less of the digit is orange and more of it is purple).

Another interesting question is when is the approximation not valid, in other words which images are not in the subset. Figure 4.10 shows a scatterplot of images from the EMNIST dataset. The y-axis are probabilities given by the classifier while the x-axis are predictions given by the model that SLISE found. The item being explained is shown on a black background and the subset

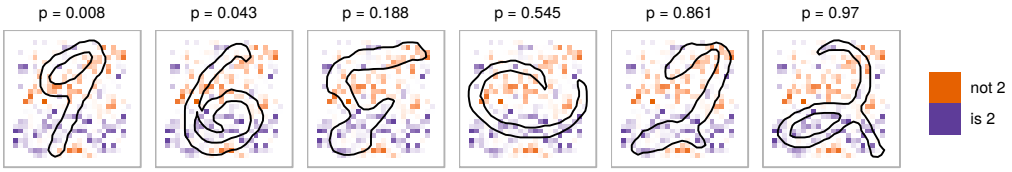


Figure 4.9: Exploring how the model given by SLISE interacts with other digits in the subset (than the one being explained).

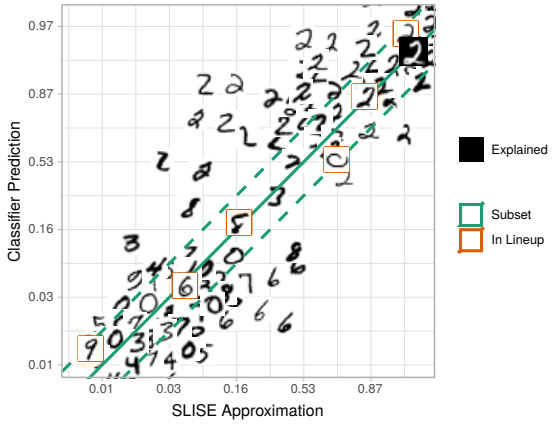


Figure 4.10: Exploring how the approximating model found by SLISE interacts with other digits, both inside and outside the subset (showing 100 digits from the dataset). The diagonal represents samples well approximated by the linear model, while those in the top left are undervalued and in the bottom right overvalued.

lie within the corridor marked with dashed green lines around the diagonal (where the classifier and approximation agrees). The data items in the top left and bottom right are not part of the subset and the model is not a valid approximation for these items. In particular Z-like 2s and L-like 6s are ill-suited for this approximation.

Modifying the subset size The subset size controls how local the explanations are. Large subsets lead to more general explanations while smaller subsets tend to have models with features unique to the subset, which has similar effects to overfitting. With SLISE the size of the subset controlled by ε, a larger ε leads to a larger subset. Figure 4.11 shows a multiple saliency maps of approximations with varying parameters in order of decreasing subset size. The explanations have only have slight variations, since they all explain the same outcome using the same dataset. This means the explanation is quite stable which is one of the desired properties of explanations from Section 2.2.3. Note that when $\varepsilon \to \infty$ SLISE becomes equivalent to logistic regression through the item being explained.

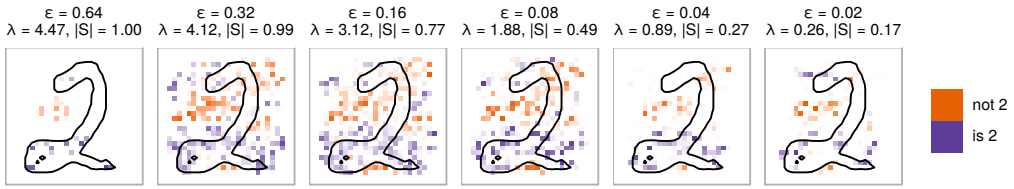


Figure 4.11: Exploring how the variation of parameters (primarily the error tolerance ε) affects the explanation.

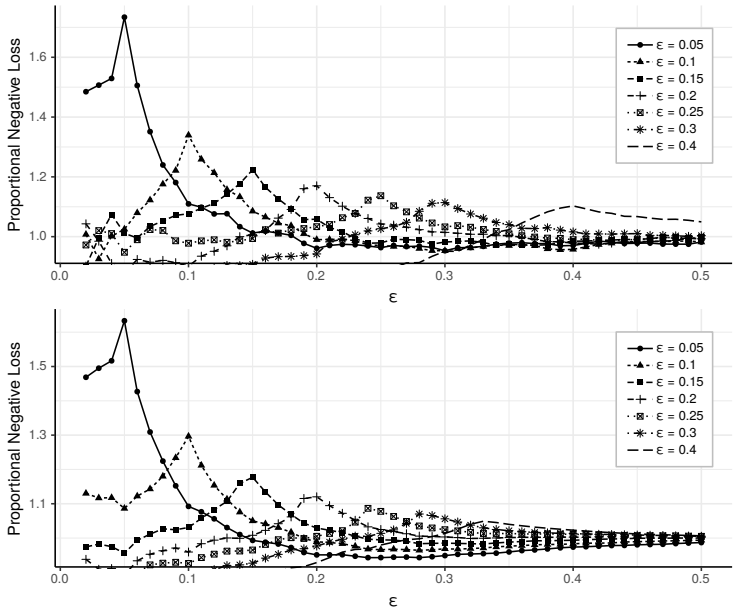


Figure 4.12: Comparing the optimality of different epsilon values. Top: A SYNTHETIC dataset of size $n = 1000$ and $d = 30$. Bottom: The EMNIST dataset.

Another way of investigating the effect of ε is with a plot similar to the optimality experiment (Figure 4.5). Instead of different algorithms Figure 4.12 uses SLISE with different values for ε ($\lambda = 0$). Instead of using LASSO as a baseline the mean loss at each point is used, i.e. the plots show the improvements over the mean. This avoids the potentially erratic behaviour of LASSO at low ε-values. The expected behaviour is that models with larger ε will be more general, meaning the peaks will be both wider and shallower. This is indeed the case with SYNTHETIC data (the top plot).

The bottom plot of Figure 4.12 shows results from the EMNIST dataset. The models with $\varepsilon > 0.2$ are shallower than expected and the peaks are not at the assigned ε values. The simplest reason for this is that the bulk of the items in the subsets have residuals that are less than the value of the peak and only a handful items have residuals larger than the peak (and less than ε). Thus the conclusion is that for this dataset $\varepsilon \leq 0.2$ would suffice and be

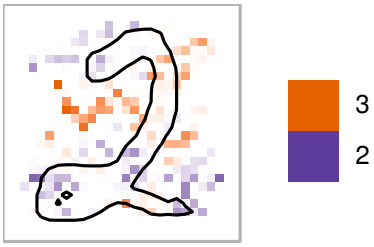


Figure 4.13: Answering different questions by changing the dataset. Here we see what differentiates this 2 from being a 3.

more robust.

Modifying the dataset The dataset used by SLISE has a big impact on the explanation. This is a deliberate design choice with a motivating example in Section 1.3. Modifying the dataset is thus also something that can be used to extract additional information from the classifier. For example, restricting the dataset to only 2s and 3s, as in Figure 4.13, shows what separates the 2 from a 3. Compared to the saliency map from Figure 4.8, in this explanation the bottom curve of 3s "splits" the bottom horizontal line of 2s and the middle horizontal line of 3s is more emphasised.

Explaining internal layers Since SLISE only requires the data to be in vector-form it is easy to combine SLISE with other methods. Of the categories from Section 2.2.2 the model inspection methods are of particular interest, especially those that visualise internal layers of a neural network. In Figure 4.14 SLISE is combined with *Activation Maximisation* [21]. Activation maximisation is used to visualise what kind of images would cause the largest activations for the nodes of an internal layer. This explains the first half of the CNN, up until the fully connected layers, and SLISE explains the latter half (with $\lambda = 9$). Note that the digit being explained is not the same 2 as above but a 3, in order to show that SLISE also works with different digits.

Activation Maximisation usually starts from random noise that is updated through gradient descent to maximise the activation of specific nodes. The CNN classifier contains pooling layers, which means that one can find multiple, equally good, maximisations where the only difference is spatial shifts. This is normally not an issue, but in order to make the explanation local to the data item being explained we prefer to show the shift that is best aligned with that item. This is accomplished without any alignment constraint on the convergence by starting the descent from the image being explained with some

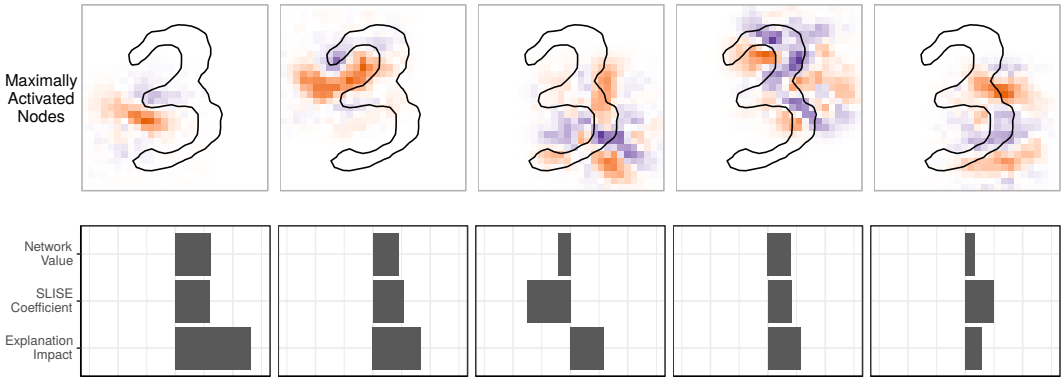


Figure 4.14: Explanation by combining SLISE with activation maximisation that visualises nodes from an internal layer of the neural network.

added normal noise (with a variance of 0.05).

In Figure 4.14 the 3 being explained is shown as a outline on top of the images that would maximally activate the internal nodes. Note that even if the colour scheme is the same as above the meaning is slightly different, purple means that the pixels should be filled in order to achieve a large activation while orange marks pixels that should be empty. The plots below show how much the 3 activates the node, the weight given by SLISE, and the combined impact. The impact is a multiplication of the node value and the SLISE weight and might be a more intuitive representation, since a large negative value combined with a large negative weight actually has a large positive effect (impact) on the classification rather than the opposite (as seen with the third node).

The first and second node (from the left in Figure 4.14) check for the characteristic empty spaces in 3s. The third node seems to check for a pattern that is not at all related to 3s, which results in the 3 getting a negative activation value *and* a negative weight in α (i.e. a positive value would indicate that this is not a 3). The fourth node matches curves from 3s and 8s. The fifth node does not perfectly fit the explained 3 (small value), but it is still useful in distinguishing 3s from other digits (large weight).

The problem with these kinds of explanations is that one has to interpret not only a linear model but also multiple saliency maps. In Figure 4.14 only the nodes with an absolute impact larger than 5% of the total sum of absolute impacts are shown, in order to reduce the amount of cognitive load required for the interpretation. Furthermore, the nodes (and thus the saliency maps) might be matching multiple features and a feature might require multiple nodes, making the interpretation more difficult. However, Figure 4.14 at least shows that the features the CNN finds and uses are reasonable.

	Pt	QG_ptD	QG_axis2	QG_mult
Jet	1196	0.935	0.002	16
$\boldsymbol{\alpha}$	0.02	0.15	-0.09	0

Table 4.5: SLISE explanation for why a NN classifies this jet as 94% likely to be a quark jet. The dataset is of size $n = 100\ 000$ and $d = 4$, and the sparsity regularisation is $\lambda = 997$.

4.3.3 Classification of Particle Jets

Some datasets follows a strict generating model that limits which samples are possible. Explanation methods that do not adhere to the generating model may produce explanations that also do not follow the generating model, which would cause domain-experts to distrust the black box model. Furthermore, the black box model is only trained on data that follows the generating model, which means that the outcomes for randomly sampled data items will be more or less random.

One such dataset is the PHYSICS dataset were the laws of physics must not be violated. SLISE automatically adheres to this constraint by only using real data to construct the explanations. The dataset contains particle jets from simulated proton-proton collisions [13]. Jets are created when unstable particles decay into multiple stable particles and the classification task is to determine whether the initiating particle was *quark* or a *gluon*. Quark and gluon jets are very similar, but there are some statistical differences [12]. Gluon jets tend to be wider while quark jets tend to have fewer particles carrying the most of the energy.

In the first experiment the jets are represented in tabular form with four physically motivated variables; the number or particles `QG_mult`, the secondary axis of an ellipse encompassing the jet `QG_axis2`, the energy distribution variable `QG_ptD`, and the total transverse momentum `Pt`. Table 4.5 shows the local explanation for a quark jet. The explanation has been confirmed by a particle physicist to be supported by the underlying physical theory of quark and gluon jets [9]. Note that a positive weight for `QG_ptD` means that increasing the value makes the jet more quark-like while the negative weight for `QG_axis2` means that a small value makes the jet more quark-like. The zero-weight for `QG_mult` demonstrates the ability of SLISE to create sparse explanations, but it is not really needed with only four variables.

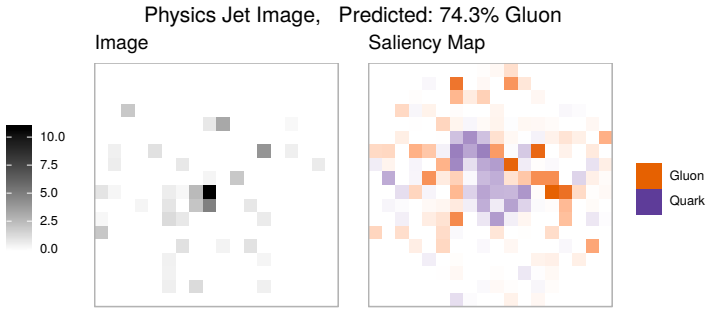


Figure 4.15: Using SLISE to explain the decision of a CNN. The image is a jet from the PHYSICS dataset where each pixel represents the energy passing through it. The dataset is of size $n = 100\,000$ and $d = 18 \times 18$, and the sparsity regularisation is $\lambda = 50$.

Jets can be represented in multiple different ways. A representation that has been successful for classification [14, 18, 33] is to map an image to a 2D plane in a particle detector. The left image of Figure 4.15 shows one such image where the value of each pixels is equal to the sum of the energies from all the particles passing through it. The right image shows a saliency map as the explanation. The explanation follows the reasoning above, gluon-jets tend to be wider while quark jets tend to have a very energetic centre.

5. Conclusion

This thesis introduces the SLISE algorithm, a novel robust regression method. SLISE extends existing robust regression methods especially in terms of scalability, which is important with large datasets. SLISE also introduces a novel way of discarding outliers by finding a subset of variable size. Additionally SLISE yields sparse solutions through LASSO-regularisation.

This thesis also demonstrates how SLISE can be used to find meaningful and interpretable explanations for outcomes from black box models. During the explanation process it is important to take the data distribution into account and sometimes it is even crucial not to perturb the data, such as when the data has a strict generating model. SLISE is able to handle both requirements by only using real data for the explanations instead of relying on mutations. Furthermore by using real data the interaction between the model and the data can be accounted for. This is important because even simple models might have non-trivial local explanations due to that interaction. Additionally, SLISE works the same way for all data types. This simplicity is important as it provides consistent operation across data domains.

5.1 Weaknesses and Mitigations

All robust regression methods have their own set of trade-offs that are more or less appropriate depending on the situation. SLISE achieves robust regression by ignoring outliers, but may end up also ignoring non-outliers if the value of ε is too small. SLISE is an approximative algorithm with a breakdown value that is dependent on the data, although the experiments show that SLISE consistently gives good solutions.

One of the advantages of SLISE when creating local explanations is the inclusion of real data in the explanations, but it is also one of the biggest weaknesses. If the data is skewed it will affect the explanations. This is

mitigated in this thesis by subsampling the datasets so that both classes have the same number of items. Dense clusters of data items are also problematic and this is mitigated with the use of the logit transformation and additionally the neural networks use label smoothing [61].

Another strength of SLISE is how well it scales, but it still has a complexity of $\mathcal{O}(nd^2p)$. The number of dimensions scales quadratically and can be problematic for sizes much larger than the 1000 dimensions (the IMDB dataset) that are demonstrated in this thesis. To speed up such situations one could use some kind of dimensionality reduction method. Or when explaining the outcome from a neural network the explanation could start from an internal layer with fewer dimensions (as demonstrated in Section 4.3.2).

Complex datasets pose an additional problem, SLISE is a robust *linear* regression algorithm, which means that if response cannot be predicted by a linear combination of the variables then SLISE is not applicable. Examples of such datasets would be anything with a temporal and or spatial structure. However, if these structures are handled by the data then SLISE could still be used. This is, e.g., the case with EMNIST where there are clear spatial structures, but since all images have been centred and scaled SLISE can still be useful.

5.2 Future Work

There are a couple of directions that future research into SLISE could take. Anecdotally it seems like the sparsity decreases during the graduated optimisation. This would suggest that λ is dependent on β, but finding this dependency is left to future work. The problem could also be solved by not using the Lagrange multiplier λ and instead use the constraint $\|\alpha\|_1 \leq t$.

In Section 3.1 it is shown how the loss function always prioritises maximising the subset over minimising the residuals, future work could see what happens if this requirement is relaxed.

Currently SLISE finds one subset and one model, a prospective future generalisation would be to find multiple subsets with corresponding models. This would create a completely new type of explanations, global explanations consisting of multiple local explanations.

5.3 Open Source

SLISE is implemented in R and released under an open source license. The source code, and the code for all the experiments, is available from `http://www.github.com/edahelsinki/slise`.

6. Sammanfattning

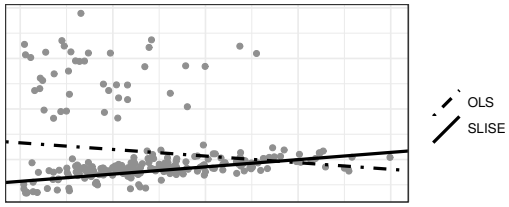
Kännetecknande för många datamängder är utstickare, datapunkter som inte följer samma mönster som resten av datamängden. Dessa punkter är problematiska när man anpassar matematiska modeller till datamängden, eftersom redan en utstickare kan leda till oanat stora avvikelser [32]. Lösningen är att använda robusta metoder som kan hantera utstickare. Denna avhandling presenterar en ny robust regressionsmetod kallad SLISE (Sparse LInear Subset Explanations).

SLISE introducerades för första gången i en artikel [9] där jag är den första författaren. Avhandlingen bygger på artikeln och använder således delvis samma definitioner och bevis. Avhandlingen är dock en fristående text som går djupare in på detaljer samt inkluderar förbättringar och tillägg utöver vad som finns i artikeln.

6.1 Robust regression

Robust regression skiljer sig från vanlig regression genom förmågan att hantera utstickare. En demonstration av detta visas i Figur 6.1. Olika robusta regressionsmetoder hanterar utstickare på olika sätt, somliga låter bli att prioritera dem (t.ex. [66]), vissa bestraffar utstickare (t.ex. [51, 59]), och en del helt och hållet ignorerar dem (t.ex. [52, 54, 4]).

SLISE hittar den största möjliga delmängd som består av datapunkter som följer samma linjära modell till en given precision. SLISE ignorerar således alla punkter utanför denna delmängd, som antas vara utstickare. SLISE skiljer sig från tidigare robusta regressions metoder som också ignorerar utstickare genom att storleken på delmängden tillåts variera istället för att vara given på förhand som i, t.ex., [52, 54, 4].



Figur 6.1: Utstickarna i övre vänstra hörnet gör att en minsta kvadratregression (OLS) ger ett sämre resultat än SLISE (som är en robust regressions metod).

6.2 Förklaringar

Ett av de mest intressanta användningsområdena för SLISE är att förklara resultat från svarta lådor. Med svarta lådor menas maskininlärda modeller som är så komplexa att en människa inte kan förstå vad de gör. Användningen av dessa modeller är begränsad i många situationer på grund av avsaknaden av transparens, bland annat för medicinska syften [10]. Detta kan hjälpas genom förklaringar på vad som leder till de givna resultaten. För en detaljerad översikt över olika förklaringsmetoder se, exempelvis, [28].

SLISE ger lokala förklaringar för alla sorters modeller utan att ändra på modellerna. Lokala förklaringar beskriver inte hela modellen, utan endast vad som leder till enskilda resultat. Detta är dock ofta vad som önskas, jämför exempelvis en lista på allt som kan orsaka cancer till personliga riskfaktorer. Andra metoder i denna nisch är, t.ex., [49, 40, 27, 50].

SLISE förklarar svarta lådor genom att approximera de komplicerade modellerna med en enkel linjär modell. Förklaringen är giltig för de datapunkter där både den komplexa och den enkla modellen ger samma resultat. Att approximera komplicerade modeller med enklare ligger som grund för många förklaringsmetoder, t.ex. [49, 40, 27, 50]. Det som skiljer SLISE från de andra metoderna är att approximeringarna bygger på äkta data istället för att muterad data. Fördelen är att mutationer kan leda till omöjliga situationer, exempelvis genom att bryta mot fysikens lagar, vilket gör sådana förklaringar mindre pålitliga.

6.3 Algoritm

Givna en datamängd ($X \in \mathbb{R}^{n\times d}, Y \in \mathbb{R}^n$) bestående av n par av $\{(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})\}_{i=1}^n$, den maximala tillåtna avvikelsen $\varepsilon \geq 0$, samt en regulariseringsparameter $\lambda \geq 0$, kan problemet som SLISE löser formellt beskrivas som att hitta den linjära modell $\alpha \in \mathbb{R}^d$ som minimerar uttrycket

$$\sum_{i=1}^{n} H\left(\varepsilon^2 - r_i^2\right)\left(r_i^2/n - \varepsilon^2\right) + \lambda \sum_{i=1}^{d} |\alpha_i|, \tag{6.1}$$

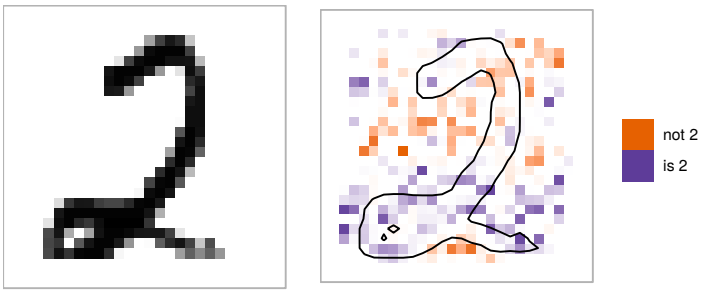
där $r_i = y_i - \alpha^\intercal x_i$ är residualer och $H(\cdot)$ är Heavisidefunktionen ($H(u) = 1$ om $u \geq 0$ och $H(u) = 0$ om $u < 0$). Problemet består således av tre delar; maximering av antalet $r_i^2 \leq \varepsilon^2$, minimering av r_i^2 för de $r_i^2 \leq \varepsilon^2$, och LASSO [62] regularisering $\sum_{i=1}^{d} |\alpha_i|$.

Detta problem är **NP**-hårt och måste således approximeras för icke-triviala datamängder. Approximeringen börjar med att ersätta Heavisidefunktionen med en sigmoidfunktion $\sigma(u) = 1/(1 + e^{-u})$. Då kan problemet lösas genom *gradvis optimering* [44] som är en iterativ metod som turvis optimerar den linjära modellen α och turvis ökar sigmoidfunktionens lutning, vilket gör den allt mer lik Heavisidefunktionen.

6.4 Experiment

För den empiriska evalueringen av SLISE används både riktiga datamängder [15, 1, 41] och syntetiska datamängder. I experimenten där SLISE jämförs med andra robusta regressionsmetoder [66, 51, 59, 54, 4] är SLISE snabbare på stora datamängder än de andra, dessutom lyckas SLISE matcha de andras robusthet.

I förklaringsexperimenten används både olika typer av data och olika klassifikationsalgoritmer. SLISE ger liknande förklaringar som andra metoder i samma nisch och skillnaderna kommer från att SLISE använder riktig data istället för muterad data för att hitta förklaringarna. Detta leder inte endast till att SLISE följer alla begränsningar i datamängden (fysikexemplet från ovan) men också att SLISE beaktar strukturer i datamängden. Tillsynes enkla modeller, t.ex. logistisk regression, kan ge oväntat bra resultat för komplicerad data genom att utnyttja dessa strukturer, så om strukturerna inte beaktas i förklaringarna förloras implicita delar av modellerna.



Figur 6.2: Till höger är en prominenskarta som visar vikterna för de olika pixlarna. Vikterna beräkna när SLISE förklarar klassifikationen av siffran till vänster. Lila pixlar innanför konturen betyder att siffran är mer lik en tvåa, medan orangea pixlar innanför betyder att siffran är mindre lik en tvåa. De flesta orangea pixlarna är utanför medan de flesta lila är innanför så detta är högst sannolikt en tvåa.

I Figur 6.2 visas ett exempel på en förklaring, varför ett faltningsnätverk anser att siffran är en tvåa. SLISE förklarar att nätverket har identifierat att tvåor brukar ha ett horisontellt streck i nedre kanten (markerat med lila).

6.5 Slutsatser

SLISE kan mäta sig med andra robusta regressionsmetoder, framförallt när det kommer till stora datamängder. Dessutom ger SLISE meningsfulla förklaringar till resultat från svarta lådor och erbjuder ett nytt perspektiv i och med att datamängden tas i beaktande. Största svagheten med SLISE är också beroendet av datamängden, så om datamängden är skev kommer det påverka resultatet. Denna design är dock ett medvetet beslut för att fånga strukturer i datamängden. Skevhet kan motverkas genom att balansera och normalisera datamängden på förhand.

Källkoden för SLISE och alla experiment finns tillgänglig som öppen källkod från `http://www.github.com/edahelsinki/slise`.

7. Bibliography

[1] Helsinki OpenData Tuples. URL `https://hot.hip.fi/`.

[2] *Least Absolute Deviation Regression*, pages 299–302. Springer New York, New York, NY, 2008. ISBN 978-0-387-32833-1. DOI 10.1007/978-0-387-32833-1_225.

[3] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, 57:1–88, 2016. URL `https://eur-lex.europa.eu/eli/reg/2016/679/oj`.

[4] Andreas Alfons, Christophe Croux, and Sarah Gelper. Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *The Annals of Applied Statistics*, 7(1):226–248, 03 2013. DOI 10.1214/12-AOAS575.

[5] Edoardo Amaldi and Viggo Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1):181–210, 1995. ISSN 0304-3975. DOI 10.1016/0304-3975(94)00254-G.

[6] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer, 2nd edition, 1999. ISBN 9783642584121. DOI 10.1007/978-3-642-58412-1.

[7] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance

propagation. *PLOS ONE*, 10(7):1–46, 07 2015. DOI 10.1371/journal.pone.0130140.

[8] I. Barrodale and F. D. K. Roberts. An improved algorithm for discrete l_1 linear approximation. *SIAM Journal on Numerical Analysis*, 10(5): 839–848, 1973. ISSN 00361429. URL `http://www.jstor.org/stable/2156318`.

[9] Anton Björklund, Andreas Henelius, Emilia Oikarinen, Kimmo Kallonen, and Kai Puolamäki. Sparse robust regression for explaining classifiers. In *Proceedings of the 22nd International Conference on Discovery Science*. Springer, 2019. To appear.

[10] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730. ACM, 2015. ISBN 978-1-4503-3664-2. DOI 10.1145/2783258.2788613.

[11] Olivier Catoni. Challenging the empirical mean and empirical variance: A deviation study. *Annales de l'I.H.P. Probabilités et statistiques*, 48(4): 1148–1185, 2012. DOI 10.1214/11-AIHP454.

[12] CMS Collaboration. Performance of quark/gluon discrimination in 8 TeV pp data. Technical report, CMS-PAS-JME-13-002, 2013. URL `http://inspirehep.net/record/1260880`.

[13] CMS Collaboration. Simulated dataset QCD_Pt-15to3000_TuneZ2star_Flat_8TeV_pythia6 in AODSIM format for 2012 collision data. *CERN Open Data Portal*, 2017. DOI 10.7483/OPENDATA.CMS.7Y4S.93A0.

[14] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwartzman. Jet-Images: Computer Vision Inspired Techniques for Jet Tagging. *Journal of High Energy Physics*, 02:118, 2015. DOI 10.1007/JHEP02(2015)118.

[15] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. 2017. URL `https://arxiv.org/abs/1702.05373`.

[16] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-78019-5.

[17] Jeffrey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women. *Reuters*, Oct 2018. URL `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-i` (Last read 20.06.2019).

[18] Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-images — deep learning edition. *Journal of High Energy Physics*, 7:69, 2016. DOI 10.1007/JHEP07(2016)069.

[19] Diego Ardila, Atilla P. Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J. Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, David P. Naidich, and Shravya Shetty. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*, 25(6):954–961, 2019. ISSN 1546-170X. DOI 10.1038/s41591-019-0447-x.

[20] David Donoho and Peter Huber. The notion of breakdown point. *A festschrift for Erich L. Lehmann*, 157184, 1983.

[21] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[22] FGCI. Finnish Grid and Cloud Infrastructure. urn:nbn:fi:research-infras-2016072533.

[23] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457, Oct 2017. DOI 10.1109/iccv.2017.371.

[24] Alex A. Freitas. Comprehensible classification models: A position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, March 2014. ISSN 1931-0145. DOI 10.1145/2594473.2594475.

[25] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. 2014. URL https://arxiv.org/abs/1412.6572.

[26] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38 (3):50–57, Oct. 2017. DOI 10.1609/aimag.v38i3.2741.

[27] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. 2018. URL https://arxiv.org/abs/1805.10820.

[28] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42, Aug 2018. ISSN 0360-0300. DOI 10.1145/3236009.

[29] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6374-equality-of-opportunity-in-supervised-learning.pdf.

[30] Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, 87:96 – 110, 2014. ISSN 1053-8119. DOI 10.1016/j.neuroimage.2013.10.067.

[31] Peter J. Huber. *Robust Statistics*, pages 1248–1251. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-04898-2. DOI 10.1007/978-3-642-04898-2_594.

[32] Mia Hubert and Michiel Debruyne. Breakdown value. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):296–302, 2009. DOI 10.1002/wics.34.

[33] Patrick T. Komiske, Eric M. Metodiev, and Matthew D. Schwartz. Deep learning in color: towards automated quark/gluon jet discrimination. *Journal of High Energy Physics*, 01:110, 2017. DOI 10.1007/JHEP01(2017)110.

[34] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1675–1684. ACM, 2016. ISBN 978-1-4503-4232-2. DOI 10.1145/2939672.2939874.

[35] Nada Lavrač. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16(1):3 – 23, 1999. ISSN 0933-3657. DOI 10.1016/S0933-3657(98)00062-1.

[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, November 1998. URL `http://yann.lecun.com/exdb/mnist/`.

[37] Matthias Leese. The new profiling: Algorithms, black boxes, and the failure of anti-discriminatory safeguards in the european union. *Security Dialogue*, 45(5):494–511, 2014. DOI 10.1177/0967010614544204.

[38] Zachary C. Lipton. The mythos of model interpretability. 2016. URL `https://arxiv.org/abs/1606.03490`.

[39] Po-Ling Loh. Scale calibration for high-dimensional robust regression. 2018. URL `https://arxiv.org/abs/1811.02096`.

[40] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

[41] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, June 2011. URL `http://www.aclweb.org/anthology/P11-1015`.

[42] Bertalan Mesko. The role of artificial intelligence in precision medicine. *Expert Review of Precision Medicine and Drug Development*, 2(5):239–241, 2017. DOI 10.1080/23808993.2017.1380516.

[43] Microsoft and R Core Team. *Microsoft R Open*. Microsoft, 2018. URL `https://mran.microsoft.com/`.

[44] Hossein Mobahi and John W. Fisher. On the link between gaussian homotopy continuation and convex envelopes. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 43–56. Springer International Publishing, 2015. ISBN 978-3-319-14612-6. DOI 10.1007/978-3-319-14612-6_4.

[45] Christoph Molnar. *Interpretable Machine Learning – A Guide for Making Black Box Models Explainable*. 2019. URL `https://christophm.github.io/interpretable-ml-book/index.html`.
Last read 20.06.2019.

[46] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security - ASIA CCS '17*, 2017. DOI 10.1145/3052973.3053009.

[47] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. Manipulating and measuring model interpretability. 2018. URL `https://arxiv.org/abs/1802.07810`.

[48] Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. How convolutional neural networks see the world — a survey of convolutional neural network visualization methods. *Mathematical Foundations of Computing*, 1(2):149–180, 2018. ISSN 2577-8838. DOI 10.3934/mfc.2018008.

[49] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 1135–1144, 2016. DOI 10.1145/2939672.2939778.

[50] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982/15850`.

[51] P. Rousseeuw and V. Yohai. Robust regression by means of S-estimators. In *Robust and nonlinear time series analysis*, pages 256–272. 1984. ISBN 978-1-4615-7821-5. DOI 10.1007/978-1-4615-7821-5_15.

[52] Peter J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984. DOI 10.1080/01621459.1984.10477105.

[53] Peter J Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):73–79, 2011. DOI 10.1002/widm.2.

[54] Peter J. Rousseeuw and Katrien Van Driessen. An algorithm for positive-breakdown regression based on concentration steps. In *Data Analysis: Scientific Modeling and Practical Application*, pages 335–346. Springer, 2000. ISBN 978-3-642-58250-9. DOI 10.1007/978-3-642-58250-9_27.

[55] Peter J. Rousseeuw and Bert C. van Zomeren. Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*, 85(411):633–639, 1990. DOI 10.1080/01621459.1990.10474920.

[56] E. J. Schlossmacher. An iterative technique for absolute deviations curve fitting. *Journal of the American Statistical Association*, 68(344):857–859, 1973. DOI 10.1080/01621459.1973.10481436.

[57] Mark Schmidt, Ewout Berg, Michael Friedlander, and Kevin Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 456–463. PMLR, Apr 2009. URL `http://proceedings.mlr.press/v5/schmidt09a.html`.

[58] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3145–3153. JMLR. org, 2017. URL `http://dl.acm.org/citation.cfm?id=3305890.3306006`.

[59] Ezequiel Smucler and Victor J Yohai. Robust and sparse estimators for linear regression models. *Computational Statistics & Data Analysis*, 111: 116–130, 2017. ISSN 0167-9473. DOI 10.1016/j.csda.2017.02.002.

[60] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2013. URL https://arxiv.org/abs/1312.6199.

[61] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. DOI 10.1109/cvpr.2016.308.

[62] Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. DOI 10.1111/j.2517-6161.1996.tb02080.x.

[63] Ryan Turner. A model explanation system. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, Sep 2016. DOI 10.1109/MLSP.2016.7738872.

[64] Berk Ustun, Stefano Traca, and Cynthia Rudin. Supersparse linear integer models for interpretable classification. 2014. URL https://arxiv.org/abs/1306.6677.

[65] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H Beck. Deep learning for identifying metastatic breast cancer. 2016. URL https://arxiv.org/abs/1606.05718.

[66] Hansheng Wang, Guodong Li, and Guohua Jiang. Robust regression shrinkage and consistent variable selection through the LAD-Lasso. *Journal of Business & Economic Statistics*, 25(3):347–355, 2007. DOI 10.1198/073500106000000251.

[67] Victor J Yohai. High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, pages 642–656, 1987. DOI 10.1214/aos/1176350366.