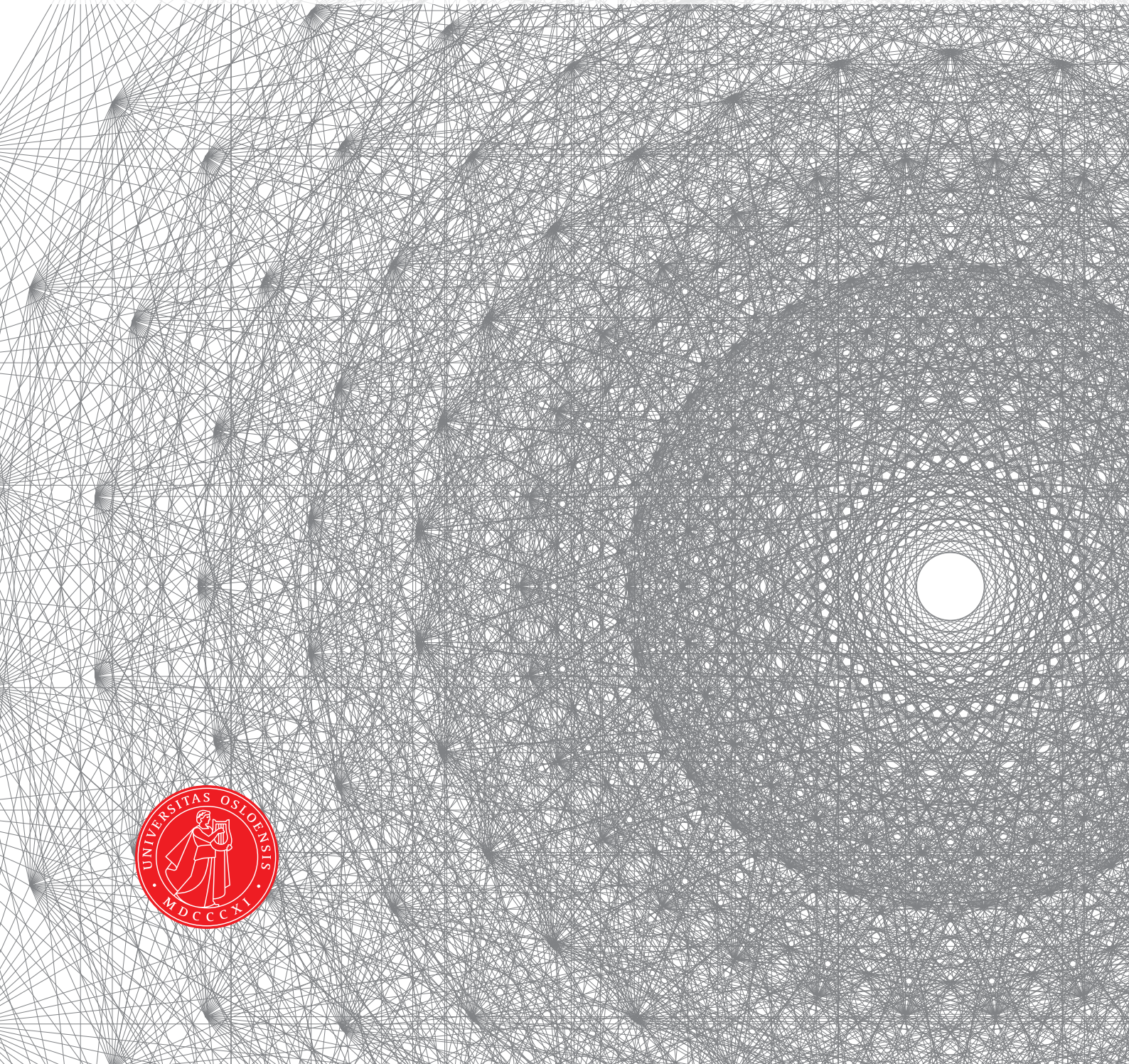
UiO : Department of Mathematics
University of Oslo

Computational barriers in statistical estimation and reconstruction

Can solutions to the LASSO be computed?

Martine Tan

Master's Thesis, Spring 2021



This master's thesis is submitted under the master's programme *Computational Science*, with programme option *Applied Mathematics and Risk Analysis*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

Lasso is an optimization problem that is favored by both statisticians and data scientists, and algorithms for solving it is offered in most statistical computing and machine learning software. However, new research into the fundamental barriers in the theory of computation shows that several optimization methods, including lasso, are generally non-computable. It has been shown that, when the input matrices are assumed to have more columns than rows, the following phenomenon occurs: (1) For any $K \geq 1$, one can construct an input class for lasso such that any algorithm will fail to compute K or more correct digits of the true solution for at least one input. This means that lasso is non-computable in the traditional Turing sense. (2) There is an algorithm that computes $K - 1$ correct digits for the same input class, but any such algorithm requires an arbitrarily long time to do so. (3) There is an algorithm that computes $K - 2$ correct digits, which is polynomial in the number of variables in the inputs. Similar results to (1), (2), and (3) are shown for randomized algorithms as well.

This thesis establishes the impossibility results outlined above, for lasso in the form typical of regression situations, where the input matrices are assumed to have more rows than columns. The impossibility results are then generalized to hold for a continuous error $\epsilon \geq 0$ instead of the number of correct digits K. This leads to phase transitions for lasso that are reminiscent of the phase transitions that appear in hardness of approximation. We call the values where the phase transitions occur for a given lasso input class the strong breakdown-epsilon, and the weak breakdown-epsilon. Specifically, for a given input class, solutions with error less than the strong breakdown-epsilon are non-computable in general. Solutions with error less than the weak breakdown-epsilon are computable, but require an arbitrary amount of time in the worst case. Solutions with error greater than the weak breakdown-epsilon are computable in polynomial time (in the number of input variables). We extend the results to hold for randomized algorithms as well, and establish the values of the probabilistic versions of the breakdown-epsilons for the same input class. These results show that it is sometimes impossible to compute solutions to lasso by use of algorithms.

Acknowledgements

First and foremost I want to thank my supervisors Øyvind Ryan and Anders C. Hansen. Øyvind introduced me to compressed sensing, which is what started me on the journey to this master's thesis – although the thesis ended up being about something else entirely. Øyvind has provided invaluable feedback on my drafts, and has always been available to advise and help me along the way. It was due to Anders' suggestions that I ended up with my final research topic. I have been allowed to read his work-in-progress through several iterations of the final (but not really final) draft, and to base the topic of this thesis on their results. Anders has been a source of inspiration and fruitful discussions on both the small details, and the greater picture of these results.

I would also like to thank Eva C. Berner and Erlend Ø. Amsen for their assistance in proofreading the manuscript, and my parents and family for their encouragement and optimism. Lastly, I would like to thank Kyrre H. Tveiten for being a great sounding board and for correcting my language mistakes. He has been patient and understanding throughout the COVID-19 lock-down, and my long hours in the home office.

Martine Tan
May 2021

Preface

This thesis is based on work that is, at the time of writing, unpublished and still under development. The main contributor is [BHV], and the reader may wish to request a copy to read alongside this thesis. Note that any references made in this thesis to specific results in [BHV], may be outdated by the time it is published. For example, theorem 3.3 in [BHV] may very well end up being theorem 4.1 in the final published version. If this is the case, an errata list will be made available to go along with this thesis, which will correct any such references.

Contents

CHAPTER 1

Introduction

The *least absolute shrinkage and selection operator*, commonly known as *lasso* was introduced as a regression tool in 1996 by R. Tibshirani [Tib96], and has since made its way into most statistical computing and machine learning toolboxes. Originally formulated as a method for estimating coefficients in linear regression models, lasso has the ability to both shrink and perform subset selection of the coefficients. The latter part is a result of lasso's tendency to set some of the coefficient estimates to be exactly zero, effectively excluding the corresponding variable from the linear model. This makes lasso a popular and widely used tool. Lasso is also used for signal reconstruction in compressed sensing, where it is closely related to *quadratically constrained basis pursuit* and *basis pursuit denoising* [AH21; FR13].

We take basis in the theory from [HTJ07; Tib96]. Consider a typical regression situation, where we have $m \in \mathbb{N}$ data points $\{(\mathbf{a}_i, y_i)\}_{i=1}^m$, such that $\mathbf{a}_i = (a_{i1}, \dots, a_{iN})^\top$ contain values for N independent variables and y_i is the dependent variable or outcome for the i-th case. We follow the practice of assuming that the a_{ij} are standardized, in the sense that $(1/m)\sum_i a_{ij} = 0$ and $(1/m)\sum_i a_{ij}^2 = 1$ are satisfied. One also typically assumes that the data points $(\mathbf{a}_i, y_i)$ are independent, or that the outcomes y_i are conditionally independent given the a_{ij}. The lasso estimate is defined as any solution of

$$\underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \sum_{i=1}^m \left(y_i - x^0 - \sum_{j=1}^N x_1^1 a_{ij}\right)^2 \quad \text{subject to } \sum_{j=1}^N |x_j^1| \le t \tag{1.1}$$

where $t \ge 0$ is a free parameter that determines the degree of regularization. Letting $A = \begin{pmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_m\end{pmatrix}^\top \in \mathbb{R}^{m\times N}$, we can write (1.1) in its Lagrangian form

$$\underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \sum_{i=1}^m \left(y_i - x^0 - \sum_{j=1}^N x_1^1 a_{ij}\right)^2 + \lambda \sum_{j=1}^N |x_j^1| \tag{1.2}$$

$$= \underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \|y - x^0\mathbf{1}_m - Ax^1\|_2^2 + \lambda\|x^1\|_1 \tag{1.3}$$

where λ is the regression parameter, and has a one-to-one correspondence with t. Lasso of the form (1.1) is sometimes called constrained lasso, while (1.2) is called unconstrained lasso. Sometimes, the ℓ_2 term in the minimization problem is multiplied by a factor $1/2$ or $1/(m)$, to simplify the expression of its derivatives

or to make the term equivalent to the mean squared error of the linear model built from the (x^0, x^1) estimates. This difference is inconsequential, since λ is a free parameter and can be chosen to counteract any such change. The lasso function[1] provided by MATLAB solves the problem

$$\underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|y - x^0\mathbf{1}_m - Ax^1\|_2^2 + \lambda\|x^1\|_1 \tag{1.4}$$

where the factor $1/(2m)$ is chosen. Sometimes, we may assume that the intercept component x^0 of the model is zero, and therefore consider the minimization problem

$$\underset{x\in\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|y - Ax\|_2^2 + \lambda\|x\|_1 \tag{1.5}$$

instead. Another way to exclude the intercept component x_0 from the expression in (1.4) is to integrate it into Ax by letting $x = (x^0, x_1^1, \ldots, x_N^1)^\top$ and $A = \begin{pmatrix}\mathbf{1}_m & \begin{pmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_m\end{pmatrix}^\top\end{pmatrix}$ in (1.5). Therefore, the problem of solving (1.4) and (1.5) can be considered equivalent. In this thesis we will consider the lasso problem in the forms given by (1.4) and (1.5).

In essence, lasso is just an optimization problem; we want to find the coefficients that minimize the mean squared error of the linear model, subject to some ℓ_1 constraint. Computing lasso solutions is a quadratic programming problem, for which there have been developed several algorithms [Tib96]. However, new research in an upcoming paper by A. Bastounis, A. C. Hansen, and V. Vlačić [BHV] shows that despite their widespread application, several optimization problems, including lasso, suffer from the same limitation: They are non-computable in a computational model that allows for approximate, real inputs and calculations with round-off errors. In other words, there are cases where the lasso minimization problem can not be solved to arbitrary precision by any algorithm, even randomized. Nevertheless, successful computing of lasso estimates in practice is commonplace.

A vast majority of modern computers are based on floating-point arithmetic. However, even a rational number such as $1/3$ can not be represented by a finite number of digits in the common base 2 floating-point representation. The same holds for irrational numbers like π and $\sqrt{2}$. These are some of the underlying limitations of computers that traditional numerical analysis of the lasso solution algorithms fails to consider. In S. Smale's list of problems for the 21-century [Sma98], he asks for "[Computational] models which process approximate inputs and which permit round-off computations". The computational model that has been developed over a series of papers [Ben+15a; Ben+15b; Ben+20; BHV; Han11] fulfills this requirement, and has been used to establish computational impossibility results for linear programming, basis pursuit, and constrained and unconstrained lasso. Even though linear programming problems are solvable in polynomial time for rational inputs [Blu+98; Kar84; Kha80], the outlook is more bleak when we allow the inputs to be real. In [BHV], it is shown that for any of the problems listed above, and any natural number $K \geq 2$, there exists a class of inputs Ω, for which no algorithm can produce an estimate of the solution with K correct digits for all the inputs in Ω. There is an algorithm

[1] https://se.mathworks.com/help/stats/lasso.html

that can provide estimates with $K-1$ correct digits for all the inputs in Ω, but any such algorithm will need an arbitrarily long runtime to do so. However, there is an algorithm that can compute estimates with $K-2$ correct digits for all the inputs in Ω, that has runtime polynomial in the number of variables in the input. The input classes Ω constructed in the proof of this result all feature matrices $A \in \mathbb{R}^{m,N}$ where $N > m$. Furthermore, the input classes are well-conditioned and bounded. Nevertheless, estimates of the solutions can not always be computed to arbitrary precision. The definition of the algorithm used is purposefully made as general as possible, so the impossibility results established are universal across all models of computation. In particular, the impossibility results hold in the Turing model as well.

This thesis further explores the computational barriers of the lasso problem (1.5). A similar result to the central theorem of [BHV], summarized above, is established for lasso where the matrices $A \in \mathbb{R}^{m,N}$ featured in the input set Ω satisfy $m > N$. The impossibility result holds in the usual regression situation, where the matrix A is standardized before fitting the model. The result is then generalized to hold for a continuous error $\epsilon \geq 0$ rather than the number of correct digits K, which is discrete. This leads to phase transitions reminiscent of the field of hardness of approximation [AB09; Pap94]. We shall see that, for the input class Ω, the problem of computing solutions to lasso transitions between

1. being computable in polynomial time,
2. being computable, but requiring an arbitrarily long runtime, and
3. being non-computable,

depending on how small error $\epsilon \geq 0$ is required on the solution. Unlike the problems that appear in hardness of approximation however, the phase transitions established in this thesis are independent of whether P $\neq$ NP. Here, P and NP refer to the complexity classes containing problems solvable in polynomial time by a deterministic Turing machine, and a non-deterministic Turing machine, respectively [AB09; Ko91; Pap94; Sip13].

We will generalize the impossibility results to hold for randomized algorithms as well. In the computational model developed in [Ben+15a; Ben+15b; Ben+20; BHV; Han11], allowing randomized algorithms the possibility of having a non-zero probability of not halting makes them more powerful. An interesting result is that even though the problem of approximating lasso solutions for the inputs in Ω is non-computable for small enough $\epsilon \geq 0$ (if we require the algorithms to always halt), there is a randomized algorithm that can compute approximations for arbitrarily small $\epsilon \geq 0$ with probability 2/3. However, this algorithm has a non-zero probability of not halting. Furthermore, we will show that the probability 2/3 of success can not be improved.

Original work

This thesis contains several results that, to the best of my knowledge, are original. These results, and their relation to the results in [BHV], are summarized here.

Theorem 3.2.1 can be considered an extension to the deterministic results in theorem 3.3 in [BHV]. There, they have established impossibility results for lasso of the form

$$\text{argmin}_{x\in\mathbb{R}^N} \|Ax - y\|_2^2 + \lambda\|x\|_1,$$

where it is assumed that the input matrix $A \in \mathbb{R}^{m,N}$ satisfies $N > m$. In this paper, we will establish similar impossibility results for lasso of the form

$$\text{argmin}_{x\in\mathbb{R}^N} \frac{1}{2m}\|Ax - y\|_2^2 + \lambda\|x\|_1,$$

where we assume that the input matrix $A \in \mathbb{R}^{m,N}$ satisfies $m > N$ and is standardized. This is a typical regression situation, while the setup in [BHV] is more reminiscent of applications in compressed sensing and signal reconstruction [AH21].

Theorem 3.5.1 can be considered a generalization of theorem 3.2.1. More specifically, theorem 3.2.1 is a 'quantised' version of theorem 3.5.1, where the approximation accuracy is measured in the number of correct digits K, instead of the actual distance to the true solution (in some given metric). Theorem 4.1.3 is an extension of theorem 3.5.1, and establishes 'unquantised' impossibility results for randomized algorithms. There is no 'unquantised' version of theorem 3.3 in [BHV], although the possibility of such a result is mentioned. Therefore, these results can be considered original. Lastly, lemma 3.3.1 can also be considered my own.

CHAPTER 2

General algorithms and complexity theory

We start by introducing several concepts from complexity theory and the solvability complexity index (SCI) from [Ben+15a; Ben+15b; BHV; Han11]. Since there are several models of computation that are non-equivalent, we introduce the concept of the general algorithm which will encompass all reasonable models of computation. This means that all of the impossibility results shown for general algorithms in the later chapters, are universal across the different models of computation. We specify what is meant by a computational problem, explain how general algorithms deal with inexact input, and introduce the concept of breakdown epsilons. We will later extend the theory to include randomized general algorithms, and the corresponding probabilistic breakdown epsilons.

2.1 Preliminaries for the deterministic impossibility results

This section contains the exact definitions of a computational problem, the general algorithm, and the strong and weak breakdown epsilons. The definitions are taken from [BHV].

Definition 2.1.1 (Computational Problem)**.** Let Ω be a set of input values, and let Λ be a set of complex valued functions on Ω such that for $\iota_1, \iota_2 \in \Omega$ we have that $\iota_1 = \iota_2$ if and only if $f(\iota_1) = f(\iota_2)$ for all $f \in \Lambda$. Let $(\mathcal{M}, d)$ be a metric space, and let $\Xi : \Omega \to \mathcal{M}$. Then we call the collection $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ a *computational problem*. Furthermore, we call Ω the *domain*, Λ the *evaluation set*, and Ξ the *problem function* or *solution map*. If it is clear by context, we can omit $\mathcal{M}$ and Λ and simply write $\{\Xi, \Omega\}$ as a shorthand for the computational problem.

This definition is quite abstract, to allow a vast amount of computational problems to fit into the framework. The components of a computational problem can be more naturally explained in the following way. The domain Ω is the set of objects or inputs that induces the computational problem, while $\mathcal{M}$ is the metric space in which the solutions to the problem are given. The problem function $\Xi : \Omega \to \mathcal{M}$ is the solution map that we want to compute with some algorithm. Finally, the evaluation set Λ is the collection of functions we may

use to read information from the inputs in Ω. In particular, we can rewrite lasso as a computational problem $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$ where

- The *domain* $\Omega_{m,N}$ consists of pairs $\iota = (y, A)$ of vectors $y \in \mathbb{R}^m$ and matrices $A \in \mathbb{R}^{m \times N}$ of given dimensions $m, N \in \mathbb{N}$.
- The metric space $(\mathcal{M}_N, d)$ is $(\mathbb{R}^N \cup \{\infty\}, d)$, where d is the metric induced by the q-norm $\|\cdot\|_q$ for some $q \in [1, \infty]$. The q-norm is defined by $\|x\|_q \coloneqq (\sum_{i=1}^N |x_i|^q)^{1/q}$ for $x \in \mathbb{R}^N$. This is normally referred to as the p-norm, but in this paper we will call it the q-norm. This is to avoid confusion later, when p will be reserved for a probability variable.
- The *evaluation set* $\Lambda_{m,N}$ consists of coordinate functions. Specifically, $\Lambda_{m,N} = \{f_i^{\text{vec}} : f_i^{\text{vec}}(\iota) = y_i\}_{i=1}^m \cup \{f_{i,j}^{\text{mat}} : f_{i,j}^{\text{mat}}(\iota) = A_{ij}\}_{i=1,j=1}^{i=m,j=N}$.
- The *problem function* Ξ is the map giving the true lasso solutions for each $\iota = (y, A)$:
$$\Xi(\iota) = \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|Ax - y\|_2^2 + \lambda\|x\|_1 \tag{2.1}$$
In statistics and machine learning, the matrix A is typically required to be standardized. In many implementations of lasso, such as the one provided by MATLAB, the input matrix is automatically standardized before the minimizer is estimated. The problem function is then
$$\Xi(\iota) = \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|\tilde{A}x - y\|_2^2 + \lambda\|x\|_1 \tag{2.2}$$
where $\tilde{A}$ is the standardized version of the matrix A.

We want to investigate whether or not there exists an algorithm that, given any input to the lasso computational problem (2.2), computes an approximate solution. For this we need to define exactly what an algorithm is. The following definition of an algorithm is very general, and encompasses all other reasonable definitions. In particular, the Turing machine and the Blum-Shub-Smale (BSS) machine can both be considered special cases of the general algorithm.

Definition 2.1.2 (General Algorithm)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. A *general algorithm* for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ is a mapping $\Gamma : \Omega \to \mathcal{M} \cup \{\text{NH}\}$ such that for every $\iota \in \Omega$ the following conditions hold:

(i) there exists a nonempty subset of evaluations $\Lambda_\Gamma(\iota) \subset \Lambda$, and whenever $\Gamma(\iota) \neq \text{NH}$, we have $|\Lambda_\Gamma(\iota)| < \infty$,

(ii) the action of Γ on ι is uniquely determined by $\{f(\iota)\}_{f \in \Lambda_\Gamma(\iota)}$,

(iii) for every $\iota' \in \Omega$ such that $f(\iota') = f(\iota)\ \forall f \in \Lambda_\Gamma(\iota)$, it holds that $\Lambda_\Gamma(\iota') = \Lambda_\Gamma(\iota)$.

The case where $\Gamma(\iota) = \text{NH}$ is a non-halting output that signifies that the general algorithm Γ does not halt on the given input ι. We shall, with slight abuse of notation, write GA for the family of all general algorithms for a given computational problem.

Remark 2.1.3. The non-halting output NH is needed in the definition of the general algorithm when we later extend the definition to the *randomized general algorithm*. One could omit the possibility of outputting NH in the definition of the general algorithm, in which case it is simply a mapping $\Gamma : \Omega \to \mathcal{M}$. However, allowing a randomized general algorithm the possibility of not halting (i.e. not producing reasonable a output) makes it more powerful. This is affirmed by theorem 4.1.3 in chapter 4, and is also shown in [BHV].

To be able to evaluate the output of the general algorithm we have to extend the metric d on $\mathcal{M}$ to a metric on $\mathcal{M} \cup \{\text{NH}\}$. This is done in the following way:

$$\tilde{d}_{\mathcal{M}}(x,y) = \begin{cases} d_{\mathcal{M}}(x,y) & \text{if } x,y \in \mathcal{M} \\ 0 & \text{if } x = y = \text{NH} \\ \infty & \text{otherwise.} \end{cases} \tag{2.3}$$

However, the problem function Ξ of a given computational problem may be multi-valued. This can sometimes be the case for minimization problems such as lasso, if there are more than one minimizer for a given input $\iota \in \Omega$. In such cases, $\Xi(\iota)$ is the set of all valid minimizers, and the goal is to approximate any of the minimizers in $\Xi(\iota)$. If $\Gamma : \Omega \to \mathcal{M} \cup \{\text{NH}\}$ is a general algorithm for some computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, we measure the approximation error by

$$\text{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) = \inf_{\xi \in \Xi(\iota)} \tilde{d}_{\mathcal{M}}(\Gamma(\iota), \xi). \tag{2.4}$$

Given some $\epsilon \geq 0$, an output $\Gamma(\iota)$ that satisfies $\text{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) \leq \epsilon$ will in this paper be referred to as an ϵ-*approximation*.

How to deal with inexact input

We can not typically expect that all input can be exactly represented, when given to a general algorithm. Sometimes the inputs we want to use are solutions to separate computational problems of their own. Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, a general algorithm Γ is able to read an input $\iota \in \Omega$ though the functions in the evaluation set Λ. Suppose that $\Lambda = \{f_j\}_{j \in \beta}$, where β is some index set (which can be both finite or infinite). If, for instance, $f_j(\iota) = \pi$ for some $j \in \beta$, then $f_j(\iota)$ can not be accessed exactly. Instead, we can use a sequence $f_{j,n}(\iota) : \Omega \to \mathbb{D} + i\mathbb{D}$, such that

$$\|\{f_{j,n}(\iota)\}_{j \in \beta} - \{f_j(\iota)\}_{j \in \beta}\|_\infty \leq 2^{-n}, \quad \forall \iota \in \Omega.$$

Here $\mathbb{D}$ is the set of dyadic rational numbers, that is, $\mathbb{D} = \{a/2^b : a, b \in \mathbb{N}\}$. Then, $f_{j,n}(\iota) \to f_j(\iota)$ as $n \to \infty$, so we can instead get access to an approximation of $f_j(\iota)$ to the accuracy we need. This concept is formalized in the definition of Δ_1-*information* below.

Definition 2.1.4 (Δ_1-information)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem with $\Lambda = \{f_j\}_{j \in \beta}$. Suppose that for each $j \in \beta$ and $n \in \mathbb{N}$, there exists a function $f_{j,n} : \Omega \to \mathbb{D} + i\mathbb{D}$, and that

$$\|\{f_{j,n}(\iota)\}_{j \in \beta} - \{f_j(\iota)\}_{j \in \beta}\|_\infty \leq 2^{-n}, \quad \forall \iota \in \Omega. \tag{2.5}$$

holds for all $n \in \mathbb{N}$. Then we say that the set $\hat{\Lambda} = \{f_{j,n} : j \in \beta, n \in \mathbb{N}\}$ provides Δ_1-information for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$. We denote the family of all such sets $\hat{\Lambda}$ by $\mathcal{L}^1(\Lambda)$.

Remark 2.1.5. The reason the functions $f_{j,n}$ are mapped onto dyadic rationals, is to allow the algorithms designed in this paper to be run on Turing machines that operate with binary numbers (see definition 2.1.9). Turing machines, unlike general algorithms, are only able to work with numbers with finite representation. Since non-dyadic numbers do not have finite binary representations, we restrict the output of the functions to $\mathbb{D}$.

The definition of the general algorithm naturally also holds for computational problems $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ for a given $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$. However, we are usually interested in algorithms that that can be applied to $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ for *all* possible choices of $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$. For this, we have the following definition.

Definition 2.1.6 (Computational problem with Δ_1 information)**.** Suppose $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ is a computational problem with $\Lambda = \{f_j\}_{j\in\beta}$. The corresponding *computational problem with* Δ_1*-information* is defined as

$$\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} \coloneqq \{\tilde{\Xi}, \tilde{\Omega}, \mathcal{M}, \tilde{\Lambda}\}$$

where

$$\begin{aligned}\tilde{\Omega} = \big\{\tilde{\iota} = \{(f_{j,1}(\iota), f_{j,2}(\iota), \dots)\}_{j\in\beta} : \iota \in \Omega \text{ and} \\ f_{j,n} : \Omega \to \mathbb{D} + i\mathbb{D} \text{ satisfies (2.5) for all } n \in \mathbb{N}\big\}\end{aligned} \tag{2.6}$$

and $\tilde{\Xi}(\tilde{\iota}) = \Xi(\iota)$, and $\tilde{\Lambda} = \{\tilde{f}_{j,n}\}_{j\in\beta,n\in\mathbb{N}}$ where $\tilde{f}_{j,n}(\tilde{\iota}) = f_{j,n}(\iota)$. Then for each $\tilde{\iota} \in \tilde{\Omega}$, there is a unique $\iota \in \Omega$ such that $\tilde{\iota} = \{(f_{j,1}(\iota), f_{j,2}(\iota), \dots)\}_{j\in\beta}$. We say that $\iota \in \Omega$ corresponds to $\tilde{\iota} \in \tilde{\Omega}$.

By this definition, we can interpret $\tilde{\Omega}$ as the family of all sequences that approximate the inputs in Ω. Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$, a general algorithm must work for all inputs $\tilde{\iota} \in \tilde{\Omega}$. In other words, it must work for any sequence that approximates some $\iota \in \Omega$. In contrast, a general algorithm for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$, where $\hat{\Lambda}$ is defined as in definition 2.1.4, need only work for that particular choice of Δ_1-information.

Relation of the concepts to the Turing model

As mentioned earlier, we will use general algorithms to establish the general impossibility results for the various computational problems. However, for any results regarding the cases where satisfactory algorithms *do* exist, we need a more explicit model of computation for which the computational complexity of algorithms can be analyzed. In this paper, all such results are done for problems $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}, \mathcal{M}, \tilde{\Lambda}\}$, using algorithms that can be executed by a Turing machine (see [AB09; Ko91; Pap94] for a classical presentation of the Turing model, and [Sip13] for a more in-depth introduction to the Turing machine).

For our purposes, it suffices to think of a Turing machine as a control head working on an infinite tape divided into cells, that each can contain only one of a finite amount of symbols. The machine starts with the input written on the tape (with some form of encoding for vectors and matrices, in our case). As the machine starts, the control head can move along the tape, reading and writing on the cells, until only the desired output remains on the tape and the machine halts. The tape is infinite, so the control head may use as many cells as needed to store information while the machine operates, and to write the

final output on the tape. However, the input given to the machine must be of finite length. The runtime of a Turing machine on a given input is the number of steps the control head takes before the machine halts, where a step is defined as the control head moving from one cell to the neighboring cell to its left or right. The space complexity on a given input is the total number of cells the control head reads or writes to during computation.

Remark 2.1.7 (Oracle tape providing Δ_1-information to Turing machines). We need to make clear exactly how $\tilde{\iota} \in \tilde{\Omega}$ is given to an algorithm as input. In the Turing model, we suppose that the index set β for Λ is countable, and that the indices $j \in \beta$ are integers, or otherwise encoded in a finite alphabet. A Turing machine can access $\tilde{\iota} \in \tilde{\Omega}$ through an *oracle* $\mathcal{O}$ that, upon input $(j, n) \in \beta \times \mathbb{N}$, writes the unique finite binary string representing $\tilde{f}_{j,n}(\tilde{\iota}) = f_{j,n}(\iota) \in \mathbb{D}$ on a separate tape called the *oracle tape*. The Turing machine can make a query to the oracle at any time during the computation, at which point the oracle takes over until it has written the unique binary string for $f_{j,n}(\iota)$ on the oracle tape. We do not specify how the oracle provides $f_{j,n}(\iota)$, as we are interested in algorithms that work for all sequences $\tilde{\iota}$ approximating some $\iota \in \Omega$. We call a Turing machine with access to an oracle an *Oracle Turing machine*. See [Ko91] for a more in-depth explanation of oracle Turing machines.

Computational complexity in the Turing model

In the Turing model, the amount of space an algorithm uses is bounded by its runtime. This is because the number of cells a Turing machine reads or writes to can not be greater than the number of steps the control head takes. Therefore, we need an exact definition of the runtime of an oracle Turing machine, to be able to analyze the time and space complexity of the algorithms constructed in chapters 3 and 4.

Definition 2.1.8 (Runtime of an algorithm in the Turing model)**.** Given an oracle Turing machine Γ for the problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}, \mathcal{M}, \tilde{\Lambda}\}$ we define the runtime of Γ on $\tilde{\iota} \in \tilde{\Omega}$ by

$$\begin{aligned}\mathbf{Runtime}_\Gamma(\tilde{\iota}) = {} & \text{The number of steps performed by } \Gamma \text{ before halting} \\ & + \text{the combined cost of all the calls to the oracle } \mathcal{O} \text{ for } \tilde{\iota}\end{aligned} \tag{2.7}$$

where the cost of calling the oracle $\mathcal{O}$ for $\tilde{f}_{j,n}(\tilde{\iota}) = f_{j,n}(\iota)$ is $j + n$.

Note that this definition inherently accounts for the cost of making a query to, and reading the output of, the oracle. The following definition gives the number of bits needed to represent a dyadic rational number in binary.

Definition 2.1.9 (Binary representation and size of a dyadic rational)**.**
For a dyadic number $d \in \mathbb{D}$, its binary representation is

$$d = \pm s_n \cdots s_1 s_0 . t_1 t_2 \cdots t_m$$

where m is called its precision. The bit size of d is $m + n + 3$. If w is a vector of dyadic numbers, the bit size is defined as the sum of the bit sizes of the components of w.

The following definition gives the number of bits needed to represent integers and rational numbers.

Definition 2.1.10 (Bit encoding length)**.** For an integer $n \in \mathbb{Z}$, the encoding length is defined as

$$\text{Len}(n) = 1 + \lceil \log_2(|n| + 1) \rceil.$$

For a rational number $a/b \in \mathbb{Q}$, where a and $b > 0$ are coprime integers[1], the encoding length is defined as $\text{Len}(a/b) = \text{Len}(a) + \text{Len}(b)$. If w is a vector of rational numbers, the encoding length is defined as the sum of the encoding lengths of the components of w.

Remark 2.1.11 (Computing K correct digits)*.* Consider the computational problem for lasso which was defined on page 6. We will often discuss whether an algorithm Γ can compute K correct digits for the computational problems $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$, where $m, N \in \mathbb{N}$ are the dimensions of the inputs $\iota = (y, A) \in \mathbb{R}^m \times \mathbb{R}^{m \times N}$. What is really meant by this statement, is whether or not

$$\text{dist}_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}), \tilde{\Xi}(\tilde{\iota})) = \inf_{\xi \in \tilde{\Xi}(\tilde{\iota})} \|\Gamma(m, N, \tilde{\iota}) - \xi\|_q \leq 10^{-K}$$

for all $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ and all dimensions $m, N \in \mathbb{N}$. Here, the algorithm Γ takes the dimensions $m, N \in \mathbb{N}$ as input, and has access to $\tilde{\iota}$ corresponding to an $\iota \in \Omega_{m,N}$ though some oracle $\mathcal{O}$. Moreover, $\|\cdot\|_q$ is the q-norm we use to measure error, and $q \in [1, \infty]$. Whenever this inequality is fulfilled, each component of $\hat{x} = \Gamma(m, N, \tilde{\iota})$ is at most 10^{-K} away from the corresponding component of $\xi \in \tilde{\Xi}(\tilde{\iota})$. This means that all components of $\hat{x}$ have at least K correct digits after the decimal point, in base 10. The reason that we chose to count the number of correct digits in base 10, and not the more natural base 2, is to mimic the setup in [BHV].

Remark 2.1.12 (The complexity of an algorithm)*.* While the runtime of an algorithm Γ (in the Turing model) for a specific input $\tilde{\iota} \in \tilde{\Omega}$ is given by definition 2.1.8, we are more interested in the worst-case runtime of Γ given *any* input. We measure the overall time complexity of Γ in terms of the number of variables in the input (given by its dimensions m and N), and the accuracy required on the output. Above, the accuracy is stated in terms of the number of correct digits K, but we shall later see cases where it is given by some error threshold $\epsilon \geq 0$.

Breakdown epsilons

We now introduce the concept of the breakdown epsilons. For a given computational problem, the breakdown epsilons provide lower bounds on the accuracy attainable by general algorithms. There are two different versions with slight, but important, differences. We call them the *strong* and *weak* breakdown epsilon. The strong breakdown epsilon, defined below, can informally be described as the largest $\epsilon \geq 0$ for which no algorithm can produce a solution with accuracy better than ϵ.

Definition 2.1.13 (Strong breakdown epsilon)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. The *strong breakdown epsilon* is defined by

$$\epsilon_B^s = \sup\{\epsilon \geq 0 : \forall \Gamma \in \text{GA}, \exists \iota \in \Omega \text{ such that } \text{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) > \epsilon\}.$$

[1] Two integers are coprime is their only mutual divisor is 1.

On the other hand, the weak breakdown epsilon can informally be described as the largest $\epsilon \geq 0$ for which all algorithms need to use an arbitrarily large amount of input information to produce a solution with ϵ accuracy. To be able to define the weak breakdown epsilon, we need the following concept of the minimum amount of input information needed by an algorithm. In this definition, Λ is assumed to be countable and enumerated by some index $k \in \mathbb{N}$.

Definition 2.1.14 (Minimum amount of input information)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem where $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$, and let Γ be a general algorithm. The *minimum amount of input information* for Γ and $\iota \in \Omega$ is defined by

$$T_\Gamma(\iota) \coloneqq \sup\{m \in \mathbb{N} : f_m \in \Lambda_\Gamma(\iota)\}.$$

Note that if $\Gamma(\iota) = \text{NH}$, then the set $\Lambda_\Gamma(\iota)$ may be infinite by definition 2.1.2. In this case, $T_\Gamma(\iota) = \infty$.

Definition 2.1.15 (Weak breakdown epsilon)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem where $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$. The *weak breakdown epsilon* is defined by

$$\begin{aligned}\epsilon_B^w = \sup\{\epsilon \geq 0 : \forall \Gamma \in \text{GA} \text{ and } \forall M \in \mathbb{N}, \exists \iota \in \Omega \text{ such that} \\ \text{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) > \epsilon \text{ or } T_\Gamma(\iota) > M\}.\end{aligned}$$

Note that by this definition, the weak breakdown epsilon is independent of how we enumerate Λ.

At times, it will be convenient to have an alternative definition of the weak breakdown epsilon for computational problems with with some $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ that provides Δ_1-information. In this case, the amount of input information needed by an algorithm is more conveniently given by the input accuracy it requires. The definition below formalizes this in terms of the number of correct digits needed on the input.

Definition 2.1.16 (Number of correct 'digits' required on the input)**.** Suppose $\hat{\Lambda} = \{f_{k,m} : k \in \beta, m \in \mathbb{N}\}$ provides Δ_1-information to the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$. Let Γ be a general algorithm for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$. The *number of correct 'digits' required on the input* is given by

$$D_\Gamma(\iota) \coloneqq \sup\{m \in \mathbb{N} : \exists k \in \beta \text{ such that } f_{k,m} \in \hat{\Lambda}_\Gamma(\iota)\}$$

Consider a general algorithm Γ for a computational problem $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ like the one given in definition 2.1.16 above. Suppose $D_\Gamma(\iota_n) \to \infty$ as $n \to \infty$ for some sequence $\{\iota_n\}_{n=0}^\infty$ in Ω. Then, no matter what enumeration of $f_{k,m} \in \hat{\Lambda}$ is used to define T_Γ, we have that $T_\Gamma(\iota_n) \to \infty$ as $n \to \infty$ as well. If additionally Λ is finite, then $T_\Gamma(\iota_n) \to \infty$ as $n \to \infty$ for some sequence $\{\iota_n\}_{n=0}^\infty$ in Ω implies that $D_\Gamma(\iota_n) \to \infty$.

This means that, for a computational problem $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ where the Δ_1-information is derived from a finite evaluation set Λ, we have

$$\begin{aligned}\epsilon_B^w = \sup\{\epsilon \geq 0 : \forall \Gamma \in \text{GA} \text{ and } \forall M \in \mathbb{N}, \exists \iota \in \Omega \text{ such that} \\ \text{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) > \epsilon \text{ or } D_\Gamma(\iota) > M\}.\end{aligned} \tag{2.8}$$

Remark 2.1.17 (Connection between accuracy of input and Turing runtime). Let Γ be a Turing machine for the problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}, \mathcal{M}, \tilde{\Lambda}\}$.

Then the number of correct 'digits' required on the input $\tilde{\iota} \in \tilde{\Omega}$ is given by $D_\Gamma(\tilde{\iota}) = \sup\{m \in \mathbb{N} : \exists k \in \beta \text{ such that } \tilde{f}_{k,m} \in \tilde{\Lambda}_\Gamma(\iota)\} = \sup\{m \in \mathbb{N} : \exists k \in \beta \text{ such that } f_{k,m} \in \hat{\Lambda}_\Gamma(\iota)\}$, where $\hat{\Lambda}$ provides the particular Δ_1-information corresponding to $\tilde{\iota}$. It follows from definition 2.1.8 that

$$\mathbf{Runtime}_\Gamma(\tilde{\iota}) \geq D_\Gamma(\tilde{\iota}).$$

Consequently, if the weak breakdown epsilon as defined in (2.8) is greater than some $\epsilon > 0$, D_Γ can be used to show that any Turing machine will have arbitrarily high runtime when attempting to achieve ϵ accuracy. Note that this is not unique to the Turing model, as any reasonable complexity model should have a definition of the runtime of an algorithm such that the runtime is at least as high as the number of digits read from the input (see remark 8.23 in [BHV]).

2.2 Tools for proving the deterministic impossibility theorems

In this section we present the essential tool we need to establish the deterministic impossibility results. The tool is the following proposition, which will later be the key to proving lower bounds for the strong and weak breakdown epsilons. Its proof is a simplification of the proof of proposition 2.4.5, which will be stated in the next section.

Proposition 2.2.1. *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be an arbitrary computational problem with countable $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$. Let $\{\iota_n^1\}_{n=1}^\infty$ and $\{\iota_n^2\}_{n=1}^\infty$ be two sequences in Ω, and consider the following conditions.*

(a) *There exists sets $S^1, S^2 \subset \mathcal{M}$ and $\kappa > 0$ such that $\inf_{x_1 \in S^1,\, x_2 \in S^2} d_{\mathcal{M}}(x_1, x_2) \geq \kappa$ and $\Xi(\iota_n^1) \subset S^1$, $\Xi(\iota_n^2) \subset S^2$.*

(b) *For every $k \leq |\Lambda|$ there is a $c_k \in \mathbb{C}$ such that $|f_k(\iota_n^1) - c_k| \leq 1/4^n$ and $|f_k(\iota_n^2) - c_k| \leq 1/4^n$ for all $n \in N$.*

(c) *There is an $\iota^0 \in \Omega$ such that for every $k \leq |\Lambda|$, (b) is satisfied with $c_k = f_k(\iota^0)$.*

Depending of which of these conditions are fulfilled, there exists a $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ such that the following holds for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$.

(i) *$\epsilon_B^w \geq \kappa/2$ if* (a) - (b) *are satisfied.*

(ii) *$\epsilon_B^s \geq \kappa/2$ if* (a) - (c) *are satisfied.*

Proof. We start by proving that if (a) and (b) are satisfied, then (i) holds. We define a new sequence $\{\iota_n\}_{n=1}^\infty$ in Ω by setting $\iota_{2n} = \iota_{n+1}^1$ and $\iota_{2n-1} = \iota_{n+1}^2$. In light of lemma 2.4.2 we can assume without loss of generality that $\Omega = \{\iota_n : n \geq 1\}$. We want to construct a set $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ providing Δ_1-information for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ such that (i) holds. For positive integers m, n and $k \leq |\Lambda|$, let $d_k^{n,m} \in \mathbb{D}_m + i\mathbb{D}_m$ be such that $|f_k(\iota_n) - d_k^{n,m}| \leq 2^{-m}$, and

let $c_k^m \in \mathbb{D}_m + i\mathbb{D}_m$ be such that $|c_k^m - c_k| \leq 2^{-m}/\sqrt{2}$. Then for $k \leq |\Lambda|$ and $m \in N$, define the function $f_{k,m} \colon \Omega \to \mathbb{D}_m + i\mathbb{D}_m$ by

$$f_{k,m}(\iota_n) = \begin{cases} d_k^{n,m} & \text{if } 1 \leq n \leq m \\ c_k^m & \text{if } n > m \end{cases} \qquad \forall \iota_n \in \Omega. \tag{2.9}$$

Let $\hat{\Lambda} \coloneqq \{f_{k,m} : k \leq |\Lambda|, m \in \mathbb{N}\}$. Then we have that for all $n \geq 1$,

$$|c_k - f_k(\iota_n)| \leq 4^{-(\lceil n/2 \rceil + 1)} \leq 2^{-(n+2)}$$

by assumption (b) and our construction of $\{\iota_n\}_{n=1}^{\infty}$. Noting that $2^{-(n+2)} \leq 2^{-(m+3)}$ for all $m, n \in \mathbb{N}$ with $n > m$, we get that

$$\begin{aligned} |f_{k,m}(\iota_n) - f_k(\iota_n)| = |c_k^m - f_k(\iota_n)| &\leq |c_k^m - c_k| + |c_k - f_k(\iota)| \\ &\leq \frac{2^{-m}}{\sqrt{2}} + 2^{-(n+2)} \\ &\leq \frac{2^{-m}}{\sqrt{2}} + 2^{-(m+2)} \\ &= 2^{-m}(1/\sqrt{2} + 2^{-3}) < 2^{-m}, \end{aligned}$$

for all $n > m$. For $1 \leq n \leq m$, we also have

$$|f_{k,m}(\iota_n) - f_k(\iota_n)| = |d_k^{n,m} - f_k(\iota_n)| \leq 2^{-m},$$

by the definition of $d_k^{n,m}$. Thus $\hat{\Lambda}$ provides Δ_1-information for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ by definition 2.1.4.

We will prove that $\epsilon_B^w \geq \kappa/2$ for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ by contradiction. Therefore, assume that $\epsilon_B^w < \kappa/2$. By (2.8) there exists a general algorithm Γ and some $M \in \mathbb{N}$ such that

$$\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) \leq \epsilon_B^w < \kappa/2 \quad \text{and} \quad D_\Gamma(\iota) \leq M, \quad \forall \iota \in \Omega$$

In particular, we have $D_\Gamma(\iota_{M+1}) \leq M$, which implies that $m \leq M$ for all $f_{k,m} \in \hat{\Lambda}_\Gamma(\iota_{M+1})$. This and (2.9) gives $f_{k,m}(\iota_{M+1}) = c_k^m$. By the same argument, we also have $f_{k,m}(\iota_{M+2}) = c_k^m$. But then $\hat{\Lambda}_\Gamma(\iota_{M+1}) = \hat{\Lambda}_\Gamma(\iota_{M+2})$ by definition 2.1.2 (iii), which gives $\Gamma(\iota_{M+1}) = \Gamma(\iota_{M+2})$ by definition 2.1.2 (ii). This means that

$$\begin{aligned} &\mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{M+1}), \Xi(\iota_{M+2})) \\ &\quad \leq \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+1})) + \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+2})) \\ &\quad = \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+1})) + \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+2}), \Xi(\iota_{M+2})) \\ &\quad < \kappa. \end{aligned} \tag{2.10}$$

By the construction of $\{\iota_n\}_{n=1}^{\infty}$, we have that $\Xi(\iota_{M+1}) \in S^1$ if and only if $\Xi(\iota_{M+2}) \in S^2$, where S^1 and S^2 are as in (a). Thus,

$$\inf_{x_1 \in S^1,\, x_2 \in S^2} d_{\mathcal{M}}(x_1, x_2) \leq \mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{M+1}), \Xi(\iota_{M+2})) < \kappa$$

by (2.10). But this contradicts the assumption that condition (a) is satisfied. We conclude that $\epsilon_B^w \geq \kappa/2$ for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$.

The proof that (a) - (c) implies $\epsilon_B^s \geq \kappa/2$ for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ for some $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ is very similar. Let $\{\iota_n\}_{n=0}^{\infty}$ be defined as before for $n \geq 1$, and let $\iota_0 = \iota^0$ from condition (c). In light of lemma 2.4.2, we can assume without loss of generality that $\Omega = \{\iota_n : n \geq 0\}$. Furthermore, for $m, n \in \mathbb{N}$ and $k \leq |\Lambda|$ let $d_k^{n,m}$ and c_k^m be defined as before. Then for $k \leq |\Lambda|$ and $m \in \mathbb{N}$ we define $f_{k,m} \colon \Omega \to \mathbb{D}_m + i\mathbb{D}_m$ by

$$f_{k,m}(\iota_n) = \begin{cases} d_k^{n,m} & \text{if } 1 \leq n \leq m \\ c_k^m & \text{if } n = 0 \text{ or } n > m \end{cases} \qquad \forall \iota_n \in \Omega. \tag{2.11}$$

Let $\hat{\Lambda} \coloneqq \{f_{k,m} : k \leq |\Lambda|, m \in \mathbb{N}\}$. We already showed that $|f_{k,m}(\iota_n) - f_k(\iota_n)| < 2^{-m}$ whenever $n \geq 1$ in the proof of (i). For $n = 0$, we also have

$$|f_{k,m}(\iota_0) - f_k(\iota_0)| = |c_k^m - c_k| \leq 2^{-m}/\sqrt{2} < 2^{-m}.$$

Thus $\hat{\Lambda}$ provides Δ_1-information for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ by definition 2.1.4.

We once again argue by contradiction, so assume that $\epsilon_B^s < \kappa/2$. Then there exists a general algorithm Γ such that $\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) \leq \epsilon_B^s < \kappa/2$ for all $\iota \in \Omega$. The bound on the distance function implies that this Γ always halts; otherwise the distance would be ∞ by (2.3). Since Γ halts on all $\iota \in \Omega$, we have that $|\hat{\Lambda}_\Gamma(\iota)| < \infty$ for all $\iota \in \Omega$ by definition 2.1.2. Let $n = |\hat{\Lambda}_\Gamma(\iota_0)| < \infty$, and consider some $f_{k,m} \in \hat{\Lambda}_\Gamma(\iota_0)$. Then $m \leq n < n+1 < n+2$. This and (2.11) implies that $f_{k,m}(\iota_0) = f_{k,m}(\iota_{n+1}) = f_{k,m}(\iota_{n+2}) = c_k^m$ for all $f_{k,m} \in \hat{\Lambda}_\Gamma(\iota_0)$. But then $\hat{\Lambda}_\Gamma(\iota_0) = \hat{\Lambda}_\Gamma(\iota_{n+1})$ and $\hat{\Lambda}_\Gamma(\iota_0) = \hat{\Lambda}_\Gamma(\iota_{n+2})$ by definition 2.1.2 (iii), which gives $\Gamma(\iota_0) = \Gamma(\iota_{n+1}) = \Gamma(\iota_{n+2})$ by definition 2.1.2 (ii). As in equation (2.10), this means that $\mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{n+1}), \Xi(\iota_{n+2})) < \kappa$. By the construction of $\{\iota_n\}_{n=0}^{\infty}$, we have that $\Xi(\iota_{n+1}) \in S^1$ if and only if $\Xi(\iota_{n+2}) \in S^2$, where S^1 and S^2 are as in (a). Thus,

$$\inf_{x_1 \in S^1,\, x_2 \in S^2} d_{\mathcal{M}}(x_1, x_2) \leq \mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{n+1}), \Xi(\iota_{n+2})) < \kappa.$$

But this contradicts the assumption that condition (a) is satisfied. We conclude that $\epsilon_B^s \geq \kappa/2$ for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$. ■

2.3 Preliminaries for the randomized impossibility results

There are several modern methods of problem solving that depend on the ability to make random decisions; for example Monte Carlo methods. There is a built-in random number generator in most programming languages, for this purpose. Randomized algorithms are also frequently used in machine learning. It therefore makes sense to extend our theory to cover probabilistic models of computation. Before introducing the randomized version of the general algorithm, we give a brief description of the randomized version of the Turing machine. A probabilistic Turing machine can be seen as a (standard) Turing machine with the ability to make a so-called coin-flip. At each step in the computation, the control head chooses between two possible next moves, each with probability 1/2, depending on the outcome of the coin-flip. See [AB09; Pap94; Sip13] for a more in-depth description of the probabilistic Turing machine. The runtime of a probabilistic Turing machine on a given input, is

the worst case number of steps the control head takes before the machine halts. The space the machine uses on a given input, is the worst case total number of cells used on the tape. Like before, we let probabilistic Turing machines for computational problems with Δ_1-information access the inputs $\tilde{\iota} \in \tilde{\Omega}$ though an oracle $\mathcal{O}$ (see remark 2.1.7).

The randomized general algorithm is, like the general algorithm, purposefully made to be as general as possible, and encompasses all reasonable probabilistic models of computation. This includes the Turing model, as a probabilistic Turing machine can be seen as a special case of the randomized general algorithm [BHV].

Definition 2.3.1 (Randomized general algorithm [BHV])**.** Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ where $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$, a *randomized general algorithm* (RGA) is a collection X of general algorithms $\Gamma : \Omega \to \mathcal{M} \cup \{\text{NH}\}$, a σ-algebra $\mathcal{F}$ on X, and a family of probability measures $\{\mathbb{P}_\iota\}_{\iota \in \Omega}$ on $\mathcal{F}$ such that:

(i) For each $\iota \in \Omega$, the mapping $\Gamma^{\text{ran}}_\iota : (X, \mathcal{F}) \to (\mathcal{M} \cup \{\text{NH}\}, \mathcal{B})$ defined by $\Gamma^{\text{ran}}_\iota(\Gamma) = \Gamma(\iota)$ is a random variable. Here, $\mathcal{B}$ is the Borel σ-algebra on $\mathcal{M} \cup \{\text{NH}\}$.

(ii) For each $n \in \mathbb{N}$ and $\iota \in \Omega$, we have $\{\Gamma \in X : T_\Gamma(\iota) \leq n\} \in \mathcal{F}$

(iii) For all $\iota_1, \iota_2 \in \Omega$ and $E \in \mathcal{F}$ such that, for every $\Gamma \in E$ and every $f \in \Lambda_\Gamma(\iota_1)$, we have $f(\iota_1) = f(\iota_2)$, it holds that $\mathbb{P}_{\iota_1}(E) = \mathbb{P}_{\iota_2}(E)$.

As before, we shall with slight abuse of notation write RGA for the family of all randomized general algorithms for a given computational problem. We refer to the algorithms in RGA by Γ^{ran}.

It is worth noting that condition (ii) of definition 2.3.1 above ensures that the minimum amount of input information is a random variable. Specifically, we can for each $\iota \in \Omega$ define

$$T_{\Gamma^{\text{ran}}}(\iota) : X \to \mathbb{N} \cup \{\infty\} \text{ defined by } \Gamma \mapsto T_\Gamma(\iota). \tag{2.12}$$

Then $T_{\Gamma^{\text{ran}}}(\iota)$ is a valid random variable because of condition (ii).

Probabilistic breakdown epsilons

We can now define the probabilistic version of the strong breakdown epsilon.

Definition 2.3.2 (Probabilistic strong breakdown epsilon)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. The *probabilistic strong breakdown epsilon* $\epsilon^s_{\mathbb{P}\text{B}} : [0, 1) \to \mathbb{R}$ is defined by

$$\epsilon^s_{\mathbb{P}\text{B}}(p) = \sup\{\epsilon \geq 0 : \forall \Gamma^{\text{ran}} \in \text{RGA}\, \exists \iota \in \Omega \text{ s.t. } \mathbb{P}_\iota(\text{dist}_{\mathcal{M}}(\Gamma^{\text{ran}}_\iota, \Xi(\iota)) > \epsilon) > p\},$$

where $\Gamma^{\text{ran}}_\iota$ is as defined in definition 2.3.1(i).

The probabilistic strong breakdown epsilon of p can informally be described as the largest $\epsilon \geq 0$ for which all randomized algorithms will fail to produce a solution with ϵ accuracy with probability at least p, for at least one input. Another way to think of the probabilistic strong breakdown epsilon is that it is

the largest $\epsilon \geq 0$ for which no algorithm can produce an ϵ-approximation with probability greater than $1 - p$ for all inputs.

It turns out that the impossibility results for randomized algorithms differ when we consider randomized general algorithms that are required to halt on every input. Therefore, we introduce following definition of a halting randomized general algorithm, and the corresponding halting probabilistic strong breakdown epsilon.

Definition 2.3.3 (Halting randomized general algorithm [BHV])**.** A randomized general algorithm Γ^{ran} for a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ is called *halting randomized general algorithm* (hRGA) if for all $\iota \in \Omega$, $\mathbb{P}_\iota(\Gamma^{\text{ran}}_\iota = \text{NH}) = 0$.

As before, we write hRGA for the family of all halting randomized general algorithms for a given computational problem. Below is the corresponding strong breakdown epsilon for halting randomized general algorithms.

Definition 2.3.4 (Halting probabilistic strong breakdown epsilon)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. The *halting probabilistic strong breakdown epsilon* $\epsilon^s_{\mathbb{P}\text{hB}} : [0, 1) \to \mathbb{R}$ is defined by

$$\epsilon^s_{\mathbb{P}\text{hB}}(p) = \sup\{\epsilon \geq 0 : \forall \Gamma^{\text{ran}} \in \text{hRGA}\, \exists \iota \in \Omega \text{ s.t.} \\ \mathbb{P}_\iota(\text{dist}_{\mathcal{M}}(\Gamma^{\text{ran}}_\iota, \Xi(\iota)) > \epsilon) > p\},$$

where $\Gamma^{\text{ran}}_\iota$ is as defined in definition 2.3.1(i).

Lastly, we have the probabilistic version of the weak breakdown epsilon. As in the deterministic case, it involves the minimum amount of input information an algorithm needs to produce a solution.

Definition 2.3.5 (Probabilistic weak breakdown epsilon)**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem with $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$. The *probabilistic weak breakdown epsilon* $\epsilon^w_{\mathbb{P}\text{B}} : [0, 1) \to \mathbb{R}$ is defined by

$$\epsilon^w_{\mathbb{P}\text{B}}(p) = \sup\{\epsilon \geq 0 : \forall \Gamma^{\text{ran}} \in \text{RGA} \text{ and } M \in \mathbb{N}, \exists \iota \in \Omega \text{ s.t.} \\ \mathbb{P}_\iota(\text{dist}_{\mathcal{M}}(\Gamma^{\text{ran}}_\iota, \Xi(\iota)) > \epsilon \text{ or } T_{\Gamma^{\text{ran}}}(\iota) > M) > p\},$$

where $\Gamma^{\text{ran}}_\iota$ is as defined in definition 2.3.1(i), and $T_{\Gamma^{\text{ran}}}(\iota)$ is as defined in (2.12).

Both the probabilistic weak breakdown epsilon and the (deterministic) weak breakdown epsilon describes a weaker form of failure than their strong counterparts. The strong breakdown epsilons describe the barriers beyond which ϵ-approximations are non-computable (with a certain probability p in the probabilistic case). In contrast, the weak breakdown epsilons describe the barriers beyond which ϵ-approximations may be computable by algorithms, however any such algorithm will need an arbitrarily large amount of input information (for a certain probability p in the probabilistic case).

Remark 2.3.6. There are several so-called impossibility statements that can be made. In particular, for a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ with Δ_1-information, we have the following statements for fixed $\epsilon > 0$ and $p \in [0, 1)$:

(i) There exists a $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ such that for the computational problem $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$, we have $\epsilon^s_{\mathbb{P}\text{B}}(p) \geq \epsilon$

(ii) When considering the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$, we have that $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq \epsilon$

We will show statements of type (i), which says that there is a particular choice of $\hat{\Lambda}$ that provides Δ_1-information for the computational problem, such that no algorithm, given this specific Δ_1-information, is able to produce an ϵ-approximation for all inputs in Ω. Note that (i) implies (ii). Furthermore, the probabilistic strong breakdown epsilon in the statements can be replaced with any of the other breakdown epsilons we have defined so far, and we still have that (i) implies (ii).

2.4 Tools for proving the randomized impossibility theorems

As we did in the deterministic case, we will now present the main tool we need to establish lower bounds on the various probabilistic breakdown epsilons. This tool comes in the form of proposition 2.4.5, which can be seen as the "big brother" of proposition 2.2.1. However, we first give the following useful proposition which summarizes the relationships between the various breakdown epsilons that have been introduced so far.

Proposition 2.4.1. *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem with $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$. Suppose $p, q \in [0, 1)$ with $p \leq q$. Then*

$$\epsilon^s_{\mathbb{P}\mathrm{B}}(q) \leq \epsilon^s_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^s_B, \tag{2.13}$$

$$\epsilon^w_{\mathbb{P}\mathrm{B}}(q) \leq \epsilon^w_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^w_B, \tag{2.14}$$

$$\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^s_{\mathbb{P}\mathrm{hB}}(p), \tag{2.15}$$

$$\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^w_{\mathbb{P}\mathrm{B}}(p), \tag{2.16}$$

$$\epsilon^s_B \leq \epsilon^w_B. \tag{2.17}$$

Proof. First note that the last three inequalities follow directly from the definitions of the involved breakdown epsilons. In particular, (2.15) and (2.16) follow from definitions 2.3.2, 2.3.4 and 2.3.5, and (2.17) follows from definitions 2.1.13 and 2.1.15.

For (2.13), we have that $\epsilon^s_{\mathbb{P}\mathrm{B}}(q) \leq \epsilon^s_{\mathbb{P}\mathrm{B}}(p)$ by definition 2.3.2. Now, suppose for contradiction that $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) > \epsilon^s_B$. Then we can write $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) > \epsilon > \epsilon^s_B$ for some $\epsilon > 0$. By definition 2.3.2 of the probabilistic strong breakdown epsilon, this means that for all $\Gamma^{\mathrm{ran}} \in \mathrm{RGA}$ there exists an $\iota \in \Omega$ such that $\mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}_\iota, \Xi(\iota)) > \epsilon) > p$. On the other hand, $\epsilon > \epsilon^s_B$ implies that there exists an $\Gamma \in \mathrm{GA}$ for which $\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) \leq \epsilon$ for all inputs $\iota \in \Omega$. However, any general algorithm Γ can be seen as a randomized general algorithm. An RGA corresponding to Γ has $X = \{\Gamma\}$ and $\mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) > \epsilon) = 0$ for all $\iota \in \Omega$. But this contradicts our earlier statement of $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) > \epsilon$. Therefore, $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^s_B$ must hold instead.

The proof of (2.14) is identical, except it relies on definitions 2.1.15 and 2.3.5. ■

The three following lemmas will be useful in proving proposition 2.4.5. The first lemma states that if any breakdown epsilons can be established for a subset

of the input Ω of a computational problem, then the corresponding breakdown epsilons for the computational problem can not be any smaller.

Lemma 2.4.2. *Suppose we have computational problems* $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ *and* $\{\Xi, \Omega', \mathcal{M}, \Lambda'\}$*, where* $\Omega' \subset \Omega$ *and* $\Lambda' = \{f|_{\Omega'} : f \in \Lambda\}$*. Then any of the breakdown epsilons for* $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ *is at least as large as the corresponding breakdown epsilon for* $\{\Xi, \Omega', \mathcal{M}, \Lambda'\}$*.*

Proof. We prove the lemma for the probabilistic strong breakdown epsilon $\epsilon^s_{\mathbb{P}\mathrm{B}}$. The proof is analogous for $\epsilon^s_{\mathbb{P}\mathrm{hB}}$ and $\epsilon^w_{\mathbb{P}\mathrm{B}}$, and follows for the deterministic breakdown epsilons ϵ^s_B and ϵ^w_B because general algorithms can be written as randomized general algorithms.

Suppose $\{\Xi, \Omega', \mathcal{M}, \Lambda'\}$ has $\epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p) > 0$ for some $p \in [0, 1)$, but that $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ has $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) < \epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p)$. By definition 2.3.2 of the probabilistic strong breakdown epsilon, there exists some Γ^{ran} for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, consisting of a collection X of general algorithms, a sigma algebra $\mathcal{F}$ on X, and probability measures $\{\mathbb{P}_\iota\}_{\iota\in\Omega}$, such that for all $\iota \in \Omega$ we have $\mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}_\iota, \Xi(\iota)) > \epsilon) \leq p$ for $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) < \epsilon < \epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p)$. Let $\Gamma^{\mathrm{ran}\prime}$ be a randomized general algorithm for $\{\Xi, \Omega', \mathcal{M}, \Lambda'\}$ consisting of a collection X' of general algorithms, a sigma algebra $\mathcal{F}'$ on X', and probability measures $\{\mathbb{P}'_\iota\}_{\iota\in\Omega'}$, and let $X' = \{\Gamma|_{\Omega'} : \Gamma \in X\}$, $\mathcal{F}' = \mathcal{F}$, and $\mathbb{P}'_\iota = \mathbb{P}_\iota$ for all $\iota \in \Omega'$. Then we have that for all $\iota \in \Omega'$, $\mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}\prime}_\iota, \Xi(\iota)) > \epsilon) \leq p$ for $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) < \epsilon < \epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p)$, as well. However, this contradicts the fact that $\{\Xi, \Omega', \mathcal{M}, \Lambda'\}$ has probabilistic strong breakdown epsilon $\epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p)$. We conclude that if $\{\Xi, \Omega', \mathcal{M}, \Lambda'\}$ has probabilistic strong breakdown epsilon $\epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p) > 0$ for some $p \in [0, 1)$, then $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ has $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq \epsilon^s_{\mathbb{P}\mathrm{B}}{}'(p)$. ■

The following lemma states that condition (iii) of definition 2.3.1 of the randomized general algorithm holds no matter what enumeration of the evaluation set Λ is chosen.

Lemma 2.4.3. *Let* Γ^{ran} *be an* RGA *for a computational problem* $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ *with countable* $\Lambda = \{f_j : j \leq |\Lambda|\}$*. For* X*,* $\mathcal{F}$*, and* $\{\mathbb{P}_\iota\}_{\iota\in\Omega}$ *(as defined in Definition 2.3.1), we have the following.*

(i) *For each* $\iota \in \Omega$ *and* $f \in \Lambda$*, we have* $\{\Gamma \in X : f \in \Lambda_\Gamma(\iota)\} \in \mathcal{F}$*.*

(ii) *If* $\theta : \mathbb{N} \to \mathbb{N}$ *is a bijection and* $n \in \mathbb{N}$*, then* $\{\Gamma \in X : \Lambda_\Gamma(\iota) \subset \{f_{\theta(1)}, \ldots, f_{\theta(n)}\}\} \in \mathcal{F}$*. In particular, this means that* Γ^{ran} *is an* RGA *for* $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ *independently of the enumeration of the evaluation set* Λ*.*

Proof. For arbitrary $\iota \in \Omega$ and $f_j \in \Lambda$, we have that

$$\begin{aligned}\{\Gamma \in X : f_j \in \Lambda_\Gamma(\iota)\} &= \{\Gamma \in X : T_\Gamma(\iota) = j\} \\ &= \{\Gamma \in X : T_\Gamma(\iota) \leq j\} \setminus \{\Gamma \in X : T_\Gamma(\iota) \leq j - 1\}.\end{aligned}$$

Furthermore, $\{\Gamma \in X : T_\Gamma(\iota) \leq n\} \in \mathcal{F}$ for all $n \in \mathbb{N}$ by definition 2.3.1(ii). But then, $\{\Gamma \in X : T_\Gamma(\iota) \leq j\} \setminus \{\Gamma \in X : T_\Gamma(\iota) \leq j - 1\} \in \mathcal{F}$ since $\mathcal{F}$ is a σ-algebra. This establishes (i) of the lemma.

For (ii), let $\theta : \mathbb{N} \to \mathbb{N}$ be any bijection, and let $n \in \mathbb{N}$. Define $S = \{\theta(j) : j \in \mathbb{N}, j \leq n\}$. Then we have that

$$\{\Gamma \in X : \Lambda_\Gamma(\iota) \subset \{f_{\theta(1)}, \ldots, f_{\theta(n)}\}\} = X \setminus \bigcup_{m \in \mathbb{N} \setminus S} \{\Gamma \in X : f_m \in \Lambda_\Gamma(\iota)\} \in \mathcal{F}$$

because $\{\Gamma \in X : f_m \in \Lambda_\Gamma(\iota)\} \in \mathcal{F}$ by (i), and $\mathcal{F}$ is a σ-algebra. ■

This last lemma shows that the minimum number of digits required on the input $D_{\Gamma^{\text{ran}}}(\iota)$ is a random variable, just like the minimum amount of input information $T_{\Gamma^{\text{ran}}}(\iota)$.

Lemma 2.4.4. *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem with countable Λ, and suppose we have some $\hat{\Lambda} = \{f_{k,m} : k \leq |\Lambda|, m \in \mathbb{N}\} \in \mathcal{L}^1(\Lambda)$ and an* RGA Γ^{ran} *for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ (with X and $\mathcal{F}$ as in definition 2.3.1). Then for each $\iota \in \Omega$, the function $D_{\Gamma^{\text{ran}}}(\iota) : X \to \mathbb{N} \cup \{\infty\}$ defined by $\Gamma \mapsto D_\Gamma(\iota)$ is $\mathcal{F}$-measurable.*

Proof. Let $n \in \mathbb{N}$ be arbitrary, and consider the set $S = \{(k, m) \in \mathbb{N}^2 : k \leq |\Lambda|, m \leq n\}$ of indices for the elements $f_{k,m} \in \hat{\Lambda}$. Recalling definition 2.1.16 of D_Γ, we have that for each $\iota \in \Omega$,

$$\begin{aligned} \{\Gamma \in X : D_\Gamma(\iota) \leq n\} &= \{\Gamma \in X : \hat{\Lambda}_\Gamma(\iota) \subset \{f_{k,m} : (k, m) \in S\}\} \\ &= X \setminus \bigcup_{(k,m) \in \mathbb{N}^2 \setminus S} \{\Gamma \in X : f_{k,m} \in \hat{\Lambda}_\Gamma(\iota)\} \in \mathcal{F}, \end{aligned} \tag{2.18}$$

because $\mathcal{F}$ is a σ-algebra, and $\{\Gamma \in X : f_{k,m} \in \hat{\Lambda}_\Gamma(\iota)\} \in \mathcal{F}$ by lemma 2.4.3(i). But then we have that for all $n \in \mathbb{N}$, $\{\Gamma \in X : D_\Gamma(\iota) = n\} = \{\Gamma \in X : D_\Gamma(\iota) \leq n\} \setminus \{\Gamma \in X : D_\Gamma(\iota) \leq n - 1\} \in \mathcal{F}$, and furthermore

$$\{\Gamma \in X : D_\Gamma(\iota) = \infty\} = X \setminus \bigcup_{n \in \mathbb{N}} \{\Gamma \in X : D_\Gamma(\iota) \leq n\} \in \mathcal{F}.$$

This means that for any subset $A \subset \mathbb{N} \cup \{\infty\}$

$$\begin{aligned} (D_{\Gamma^{\text{ran}}}(\iota))^{-1}(A) &= \{\Gamma \in X : D_\Gamma(\iota) \in A\} \\ &= \bigcup_{a \in A} \{\Gamma \in X : D_\Gamma(\iota) = a\} \in \mathcal{F}. \end{aligned}$$

Thus $D_{\Gamma^{\text{ran}}}(\iota)$ is a measurable function. ■

We are now ready to state and prove the main result of this section. This proposition is a partial replication of proposition 9.5 in [BHV]. Furthermore, proposition 2.2.1 follows directly from this proposition, by proposition 2.4.1.

Proposition 2.4.5. *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be an arbitrary computational problem with countable $\Lambda = \{f_k : k \in \mathbb{N}, k \leq |\Lambda|\}$. Let $\{\iota_n^1\}_{n=1}^\infty$ and $\{\iota_n^2\}_{n=1}^\infty$ be two sequences in Ω, and consider the following conditions.*

(a) *There exists sets $S^1, S^2 \subset \mathcal{M}$ and $\kappa > 0$ such that $\inf_{x_1 \in S^1,\, x_2 \in S^2} d_{\mathcal{M}}(x_1, x_2) \geq \kappa$ and $\Xi(\iota_n^1) \subset S^1$, $\Xi(\iota_n^2) \subset S^2$.*

(b) *For every* $k \leq |\Lambda|$ *there is a* $c_k \in \mathbb{C}$ *such that* $|f_k(\iota_n^1) - c_k| \leq 1/4^n$ *and* $|f_k(\iota_n^2) - c_k| \leq 1/4^n$ *for all* $n \in N$.

(c) *There is an* $\iota^0 \in \Omega$ *such that for every* $k \leq |\Lambda|$, *(b) is satisfied with* $c_k = f_k(\iota^0)$.

Depending of which of these conditions are fulfilled, there exists a $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ *such that the following holds for* $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$.

(i) $\epsilon_B^w \geq \epsilon_{\mathbb{P}\mathrm{B}}^w(p) \geq \kappa/2$ *for* $p \in [0, 1/2)$ *if* (a) - (b) *are satisfied.*

(ii) $\epsilon_B^s \geq \epsilon_{\mathbb{P}\mathrm{hB}}^s(p) \geq \kappa/2$ *for* $p \in [0, 1/2)$ *and* $\epsilon_{\mathbb{P}\mathrm{B}}^s(p) \geq \kappa/2$ *for* $p \in [0, 1/3)$ *if* (a) - (c) *are satisfied.*

The proof of this proposition will be separated in two parts, for each of the results (i) and (ii).

Proof of 2.4.5 (i). Assume conditions (a) and (b) hold. Just as in the proof of proposition 2.2.1, we define a new sequence $\{\iota_n\}_{n=1}^{\infty}$ in Ω by setting $\iota_{2n} = \iota_{n+1}^1$ and $\iota_{2n-1} = \iota_{n+1}^2$. In light of lemma 2.4.2, we can assume without loss of generality that $\Omega = \{\iota_n : n \geq 1\}$. For $m, n \in \mathbb{N}$ and $k \leq |\Lambda|$, let $d_k^{n,m} \in \mathbb{D}_m + i\mathbb{D}_m$ be such that $|f_k(\iota_n) - d_k^{n,m}| \leq 2^{-m}$, and let $c_k^m \in \mathbb{D}_m + i\mathbb{D}_m$ be such that $|c_k^m - c_k| \leq 2^{-m}/\sqrt{2}$. For $k \leq |\Lambda|$ and $m \in N$, we define the function $f_{k,m} : \Omega \to \mathbb{D}_m + i\mathbb{D}_m$ by

$$f_{k,m}(\iota_n) = \begin{cases} d_k^{n,m} & \text{if } 1 \leq n \leq m \\ c_k^m & \text{if } n > m \end{cases} \qquad \forall \iota_n \in \Omega. \tag{2.19}$$

Let $\hat{\Lambda} \coloneqq \{f_{k,m} : k \leq |\Lambda|, m \in \mathbb{N}\}$. It was shown in the proof of proposition 2.2.1 that this $\hat{\Lambda}$ provides Δ_1-information for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$.

We will prove that $\epsilon_{\mathbb{P}\mathrm{B}}^w(p) \geq \kappa/2$ for all $p \in [0, 1/2)$ for the computational problem $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ by contradiction. Therefore, assume that $\epsilon_{\mathbb{P}\mathrm{B}}^w(p) < \kappa/2$ for some $p \in [0, 1/2)$. By definition 2.3.5 of the probabilistic weak breakdown epsilon, this means that there exists some $\Gamma^{\mathrm{ran}} \in \mathrm{RGA}$ and some $N \in \mathbb{N}$ such that

$$\mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma_\iota^{\mathrm{ran}}, \Xi(\iota)) \geq \kappa/2 \text{ or } T_{\Gamma^{\mathrm{ran}}}(\iota) > N) \leq p, \tag{2.20}$$

for all inputs $\iota \in \Omega$. Recall from definition 2.3.1 that this Γ^{ran} consists of a collection X of general algorithms, a sigma-algebra $\mathcal{F}$ on X, and a family of probability measures $\{\mathbb{P}_\iota\}_{\iota \in \Omega}$ on $\mathcal{F}$. We will work with this Γ^{ran} and $N \in \mathbb{N}$.

Notice first that by definitions 2.1.14 and 2.1.16 of T_Γ and D_Γ, there must exist some $M \in \mathbb{N}$ such that the conditional statement

$$D_\Gamma(\iota) > M \implies T_\Gamma(\iota) > N \tag{2.21}$$

holds for all $\Gamma \in X$ and all $\iota \in \Omega$. Using this $M \in \mathbb{N}$, define the sets

$$F_1 \coloneqq \{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+1})) \geq \kappa/2 \text{ or } D_\Gamma(\iota_{M+1}) > M\},$$
$$F_2 \coloneqq \{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+2}), \Xi(\iota_{M+2})) \geq \kappa/2 \text{ or } D_\Gamma(\iota_{M+2}) > M\},$$
$$\hat{F}_2 \coloneqq \{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+2}), \Xi(\iota_{M+2})) \geq \kappa/2\}, \text{ and}$$
$$\mathring{F}_2 \coloneqq \{\Gamma \in X : D_\Gamma(\iota_{M+2}) > M\}.$$

By the continuity of the metric $d_{\mathcal{M}}$, condition (i) of definition 2.3.1, and lemma 2.4.4, we have that these sets are measurable. Furthermore, for $j = 1, 2$ we have

$$\begin{aligned}\mathbb{P}_{\iota_{M+j}}(F_j) &= \mathbb{P}_{\iota_{M+j}}(\{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+j}), \Xi(\iota_{M+j})) \geq \kappa/2 \\ &\qquad\qquad \text{or } D_\Gamma(\iota_{M+j}) > M\}) \\ &= \mathbb{P}_{\iota_{M+j}}(\{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+j}), \Xi(\iota_{M+j})) \geq \kappa/2 \\ &\qquad\qquad \text{or } T_\Gamma(\iota_{M+j}) > N\}) \\ &\leq p\end{aligned}$$

by (2.20) and (2.21).

We will use the three following statements to arrive at a contradiction that completes the proof of (i).

1. $X = F_1 \cup \hat{F}_2$
2. $\mathbb{P}_{\iota_{M+1}}(\hat{F}_2 \cap \mathring{F}_2^c) = \mathbb{P}_{\iota_{M+2}}(\hat{F}_2 \cap \mathring{F}_2^c)$
3. $\mathbb{P}_{\iota_{M+1}}(\mathring{F}_2) = \mathbb{P}_{\iota_{M+2}}(\mathring{F}_2)$

To prove statement 1, consider an arbitrary $\Gamma \in X$. Suppose that $\Gamma \notin F_1$ and $\Gamma \notin \hat{F}_2$. Then we have that $\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+1})) < \kappa/2$, $\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+2}), \Xi(\iota_{M+2})) < \kappa/2$, and $D_\Gamma(\iota_{M+1}) \leq M$. The last inequality implies that $m \leq M$ for all $f_{k,m} \in \hat{\Lambda}_\Gamma(\iota_{M+1})$. But then $f_{m,k}(\iota_{M+1}) = f_{m,k}(\iota_{M+2}) = c_k^m$ by (2.19). This implies that $\hat{\Lambda}(\iota_{M+1}) = \hat{\Lambda}(\iota_{M+2})$ by condition (iii) of definition 2.1.2, which again implies that $\Gamma(\iota_{M+1}) = \Gamma(\iota_{M+2})$ by condition (ii) of definition 2.1.2. But then we have that

$$\begin{aligned}\inf_{\substack{x_1 \in \Xi(\iota_{M+1}) \\ x_2 \in \Xi(\iota_{M+2})}} d_{\mathcal{M}}(x_1, x_2) &= \mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{M+1}), \Xi(\iota_{M+2})) \\ &\leq \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+1})) \\ &\qquad + \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+2})) \\ &= \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+1}), \Xi(\iota_{M+1})) \\ &\qquad + \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{M+2}), \Xi(\iota_{M+2})) \\ &< \kappa\end{aligned}$$

which contradicts assumption (a). Therefore, we must have that $\Gamma \in F_1 \cup \hat{F}_2$, and since Γ was arbitrary, we conclude that $X \subseteq F_1 \cup \hat{F}_2$. However, $F_1 \cup \hat{F}_2 \subseteq X$ by their definitions. Thus $X = F_1 \cup \hat{F}_2$.

To prove statement 2 and 3, we will show that $\mathbb{P}_{\iota_{M+1}}(E \cap \mathring{F}_2^c) = \mathbb{P}_{\iota_{M+2}}(E \cap \mathring{F}_2^c)$ for all $E \in \mathcal{F}$. If $E \cap \mathring{F}_2^c = \emptyset$, the equality holds because both sides are zero. Therefore, we only need to prove the equality for $E \in \mathcal{F}$ such that $E \cap \mathring{F}_2^c \neq \emptyset$. We have that for all $\Gamma \in E \cap \mathring{F}_2^c$ and $f_{k,m} \in \hat{\Lambda}_\Gamma(\iota_{M+2})$, $m \leq M$ because $\mathring{F}_2 \coloneqq \{\Gamma \in X : D_\Gamma(\iota_{M+2}) > M\}$. Together with (2.9), this implies that $f_{k,m}(\iota_{M+1}) = f_{k,m}(\iota_{M+2}) = c_k^m$. By definition 2.3.1 (iii), this means $\mathbb{P}_{\iota_{M+1}}(E \cap \mathring{F}_2^c) = \mathbb{P}_{\iota_{M+2}}(E \cap \mathring{F}_2^c)$ as claimed. Statement 2 follows by noting that $\hat{F}_2 \in \mathcal{F}$ since $\hat{F}_2$ is measureable. Statement 3 follows by using $E = X \in \mathcal{F}$ and observing that $\mathbb{P}_{\iota_{M+1}}(\mathring{F}_2) = 1 - \mathbb{P}_{\iota_{M+1}}(\mathring{F}_2^c) = 1 - \mathbb{P}_{\iota_{M+1}}(X \cap \mathring{F}_2^c) = 1 - \mathbb{P}_{\iota_{M+2}}(X \cap \mathring{F}_2^c) = 1 - \mathbb{P}_{\iota_{M+2}}(\mathring{F}_2^c) = \mathbb{P}_{\iota_{M+2}}(\mathring{F}_2)$.

We can now show that our assumption that $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) < \kappa/2$ for some $p \in [0, 1/2)$ leads to a contradiction. We have that

$$\begin{aligned}
1 &= \mathbb{P}_{\iota_{M+1}}(X) = \mathbb{P}_{\iota_{M+1}}(F_1 \cup \hat{F}_2) \\
&\leq \mathbb{P}_{\iota_{M+1}}(F_1) + \mathbb{P}_{\iota_{M+1}}(\hat{F}_2) \\
&= \mathbb{P}_{\iota_{M+1}}(F_1) + \mathbb{P}_{\iota_{M+1}}(\hat{F}_2 \cap \mathring{F}_2^c) + \mathbb{P}_{\iota_{M+1}}(\hat{F}_2 \cap \mathring{F}_2) \\
&= \mathbb{P}_{\iota_{M+1}}(F_1) + \mathbb{P}_{\iota_{M+2}}(\hat{F}_2 \cap \mathring{F}_2^c) + \mathbb{P}_{\iota_{M+1}}(\hat{F}_2 \cap \mathring{F}_2) \\
&\leq \mathbb{P}_{\iota_{M+1}}(F_1) + \mathbb{P}_{\iota_{M+2}}(\hat{F}_2 \cap \mathring{F}_2^c) + \mathbb{P}_{\iota_{M+1}}(\mathring{F}_2) \\
&= \mathbb{P}_{\iota_{M+1}}(F_1) + \mathbb{P}_{\iota_{M+2}}(\hat{F}_2 \cap \mathring{F}_2^c) + \mathbb{P}_{\iota_{M+2}}(\mathring{F}_2) \\
&\leq \mathbb{P}_{\iota_{M+1}}(F_1) + \mathbb{P}_{\iota_{M+2}}(F_2) \\
&\leq p + p < 1
\end{aligned}$$

where we have used statements 1, 2, and 3, the addition law of probability, as well as the facts that $(\hat{F}_2 \cap \mathring{F}_2^c) \cup \mathring{F}_2 = F_2$ and $p < 1/2$. We conclude that $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) \geq \kappa/2$ for all $p \in [0, 1/2)$. By proposition 2.4.1, we have $\epsilon^w_B \geq \epsilon^w_{\mathbb{P}\mathrm{B}}(p)$, and statement (i) follows.

■

Proof of 2.4.5 (ii). Assume conditions (a) through (c) hold. Let $\{\iota_n\}_{n=1}^{\infty}$ be defined as before, and let $\iota_0 = \iota^0$ from condition (c). By lemma 2.4.2, we can assume without loss of generality that $\Omega = \{\iota_n : n \geq 0\}$. For $m, n \in \mathbb{N}$ and $k \leq |\Lambda|$, let $d_k^{n,m}$ and c_k^m be defined as above as well. Then for $k \leq |\Lambda|$ and $m \in \mathbb{N}$ we define $f_{k,m} : \Omega \to \mathbb{D}_m + i\mathbb{D}_m$ by

$$f_{k,m}(\iota_n) = \begin{cases} d_k^{n,m} & \text{if } 1 \leq n \leq m \\ c_k^m & \text{if } n = 0 \text{ or } n > m \end{cases} \qquad \forall \iota_n \in \Omega. \tag{2.22}$$

Let $\hat{\Lambda} \coloneqq \{f_{k,m} : k \leq |\Lambda|, m \in \mathbb{N}\}$. It was shown in the proof of 2.2.1 (ii) that this $\hat{\Lambda}$ provides Δ_1-information for the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$.

We start by proving that for $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$, we have $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq \kappa/2$ for $p \in [0, 1/3)$. Assume that $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) < \kappa/2$ for some $p \in [0, 1/3)$ for contradiction. By definition 2.3.2 of the probabilistic strong breakdown epsilon, this means that there exists some $\Gamma^{\mathrm{ran}} \in \mathrm{RGA}$ such that $\mathbb{P}_\iota(\mathrm{dist}_M(\Gamma^{\mathrm{ran}}_\iota, \Xi(\iota)) \geq \kappa/2) \leq p$ for all $\iota \in \Omega$. Let $F^n \coloneqq \{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_n), \Xi(\iota_n)) \geq \kappa/2\}$ for all $n \in \mathbb{N} \cup \{0\}$. The F_n sets are measureable by the continuity of the metric $d_{\mathcal{M}}$. Then we have that

$$\mathbb{P}_{\iota_n}(F^n) \leq p \text{ for all } n \in \mathbb{N} \cup \{0\}. \tag{2.23}$$

Define $\mathcal{G}^n(\iota) \coloneqq \{\Gamma \in X : D_\Gamma(\iota) \leq n\}$ for $\iota \in \Omega$ and $n \in \mathbb{N} \cup \{0\}$, and notice that $\mathcal{G}^n(\iota) \subset \mathcal{G}^{n+1}(\iota)$. By lemma 2.4.4, the sets $\mathcal{G}^n(\iota)$ are measureable. Furthermore,

$$\begin{aligned}
X \setminus \bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0) &= X \setminus \bigcup_{n=1}^{\infty} \{\Gamma \in X : D_\Gamma(\iota_0) \leq n\} \\
&= \{\Gamma \in X : D_\Gamma(\iota_0) = \infty\} \\
&= \{\Gamma \in X : |\hat{\Lambda}_\Gamma(\iota_0)| = \infty\}
\end{aligned}$$

$$\begin{aligned}&= \{\Gamma \in X : \Gamma(\iota_0) = \mathrm{NH}\}\\ &\subseteq \{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_0), \Xi(\iota_0)) = \infty\}\\ &\subseteq F^0\end{aligned}$$

by definitions 2.1.16 and 2.1.2(i), as well as equation (2.3). Consequently,

$$\mathbb{P}_{\iota_0}\left(X \setminus \bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0)\right) \leq p \tag{2.24}$$

by (2.23). For all $\Gamma \in \mathcal{G}^n(\iota_0)$, we have that $f_{k,m} \in \hat{\Lambda}(\iota_0)$ implies $m \leq n$ by definition 2.1.16 of D_Γ. This means that $f_{k,m}(\iota_0) = f_{k,m}(\iota_{n+1}) = f_{k,m}(\iota_{n+2}) = c_k^m$ for all $f_{k,m} \in \hat{\Lambda}(\iota_0)$ by (2.22). But then $\hat{\Lambda}(\iota_0) = \hat{\Lambda}(\iota_{n+1}) = \hat{\Lambda}(\iota_{n+2})$ by definition 2.1.2 (iii), which gives $\Gamma(\iota_0) = \Gamma(\iota_{n+1}) = \Gamma(\iota_{n+2})$ by definition 2.1.2 (ii). Consequently,

$$\Gamma(\iota_{n+1}) = \Gamma(\iota_{n+2}) \text{ for all } \Gamma \in \mathcal{G}^n(\iota_0). \tag{2.25}$$

We will use the following statement to complete the proof of (ii); There exists an $n \in \mathbb{N}$ such that:

1. $\mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) > 2p$
2. $\mathcal{G}^n(\iota_0) \subset F^{n+1} \cup F^{n+2}$
3. $\mathbb{P}_{\iota_0}(F^{n+1} \cap \mathcal{G}^n(\iota_0)) = \mathbb{P}_{\iota_{n+1}}(F^{n+1} \cap \mathcal{G}^n(\iota_0))$
4. $\mathbb{P}_{\iota_0}(F^{n+2} \cap \mathcal{G}^n(\iota_0)) = \mathbb{P}_{\iota_{n+2}}(F^{n+2} \cap \mathcal{G}^n(\iota_0))$

Statement 1 is proved by contradiction. Suppose that $\mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) \leq 2p$ for all $n \in \mathbb{N}$, and recall that $\mathcal{G}^n(\iota_0) \subset \mathcal{G}^{n+1}(\iota_0)$ for all n. Then by continuity of measures [Lin18, Proposition 7.1.5],

$$\begin{aligned}\mathbb{P}_{\iota_0}\left(X \setminus \bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0)\right) &= 1 - \mathbb{P}_{\iota_0}\left(\bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0)\right)\\ &= 1 - \lim_{n\to\infty} \mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) \geq 1 - 2p.\end{aligned}$$

However, this implies that $p \geq 1 - 2p$ by (2.24), which is a contradiction since $p \in [0, 1/3)$. Therefore, there exists some $n \in \mathbb{N}$ for which $\mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) > 2p$. This establishes statement 1.

For statement 2, note that if $\Gamma \in \mathcal{G}^n(\iota_0)$ but $\Gamma \notin F^{n+1} \cup F^{n+2}$, then

$$\begin{aligned}\mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{n+1}), \Xi(\iota_{n+2})) &\leq \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{n+1}), \Xi(\iota_{n+1})) + \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{n+1}), \Xi(\iota_{n+2}))\\ &= \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{n+1}), \Xi(\iota_{n+1})) + \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_{n+2}), \Xi(\iota_{n+2}))\\ &< \kappa,\end{aligned}$$

by (2.25). By the construction of $\{\iota_n\}_{n=1}^{\infty}$, we have that $\Xi(\iota_{n+1}) \in S^1$ if and only if $\Xi(\iota_{n+2}) \in S^2$, where S^1 and S^2 are as in condition (a). Thus

$$\inf_{x_1 \in S^1,\, x_2 \in S^2} d_{\mathcal{M}}(x_1, x_2) \leq \mathrm{dist}_{\mathcal{M}}(\Xi(\iota_{n+1}), \Xi(\iota_{n+2})) < \kappa,$$

but this contradicts the assumption that condition (a) is satisfied. Therefore we have that any $\Gamma \in \mathcal{G}^n(\iota_0)$ also satisfies $\Gamma \in F^{n+1} \cup F^{n+2}$.

For statement 3, note that for any $\Gamma \in \mathcal{G}^n(\iota_0) \cap F^{n+1}$, we have $f_{k,m}(\iota_{n+1}) = f_{k,m}(\iota_0)$ for all $f_{k,m} \in \hat{\Lambda}(\iota_0)$ by (2.25). Furthermore, $\mathcal{G}^n(\iota_0) \cap F^{n+1} \in \mathcal{F}$ since both $\mathcal{G}^n(\iota_0)$ and F^{n+1} are measureable. But then $\mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0) \cap F^{n+1}) = \mathbb{P}_{\iota_{n+1}}(\mathcal{G}^n(\iota_0) \cap F^{n+1})$ by definition 2.3.1 (iii). The proof of statement 4 is identical.

We can now show that our assumption that $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) < \kappa/2$ for some $p \in [0, 1/3)$ leads to a contradiction. We have that

$$\begin{aligned}
2p < \mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) &= \mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0) \cap (F^{n+1} \cup F^{n+2})) \\
&= \mathbb{P}_{\iota_0}\big((\mathcal{G}^n(\iota_0) \cap F^{n+1}) \cup (\mathcal{G}^n(\iota_0) \cap F^{n+2})\big) \\
&\leq \mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0) \cap F^{n+1}) + \mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0) \cap F^{n+2}) \\
&= \mathbb{P}_{\iota_{n+1}}(\mathcal{G}^n(\iota_0) \cap F^{n+1}) + \mathbb{P}_{\iota_{n+2}}(\mathcal{G}^n(\iota_0) \cap F^{n+2}) \\
&\leq \mathbb{P}_{\iota_{n+1}}(F^{n+1}) + \mathbb{P}_{\iota_{n+2}}(F^{n+2}) \\
&\leq 2p
\end{aligned} \tag{2.26}$$

where we have used statement 1, 2, 3, and 4, the addition law of probability, and (2.23). This is clearly a contradiction, and therefore we must have that $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq \kappa/2$ for all $p \in [0, 1/3)$.

It now remains to show that $\epsilon^s_B \geq \epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq \kappa/2$ for all $p \in [0, 1/2)$ for the same $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$. Note that by proposition 2.4.1, $\epsilon^s_B \geq \epsilon^s_{\mathbb{P}\mathrm{hB}}(p)$. Therefore we need only establish $\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq \kappa/2$.

Assume for contradiction that $\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) < \kappa/2$ for some $p \in [0, 1/2)$. By definition 2.3.4 of the halting probabilistic strong breakdown epsilon, this means that there exists some $\Gamma^{\mathrm{ran}} \in \mathrm{hRGA}$ such that $\mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}_\iota, \Xi(\iota)) \geq \kappa/2) \leq p$ for all $\iota \in \Omega$. Here, Γ^{ran} is a halting randomized general algorithm consisting of a collection X and a sigma-algebra $\mathcal{F}$, as well as probability measures $\{\mathbb{P}_\iota\}_{\iota \in \Omega}$ as defined in definition 2.3.1. Furthermore, $\mathbb{P}_\iota(\Gamma^{\mathrm{ran}}_\iota = \mathrm{NH}) = 0$ for all $\iota \in \Omega$. Define $F^n \coloneqq \{\Gamma \in X : \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota_n), \Xi(\iota_n)) \geq \kappa/2\}$ for all $n \in \mathbb{N} \cup \{0\}$ and $\mathcal{G}^n(\iota) \coloneqq \{\Gamma \in X : D_\Gamma(\iota) \leq n\}$ for $\iota \in \Omega$ and $n \in \mathbb{N} \cup \{0\}$ as before. Then we have that

$$\mathbb{P}_{\iota_n}(F_n) \leq p \text{ for all } n \in \mathbb{N} \cup \{0\}. \tag{2.27}$$

and

$$\Gamma(\iota_{n+1}) = \Gamma(\iota_{n+2}) \text{ for all } \Gamma \in \mathcal{G}^n(\iota). \tag{2.28}$$

by the same arguments used to establish (2.23) and (2.24). Moreover,

$$\begin{aligned}
X \setminus \bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0) &= X \setminus \bigcup_{n=1}^{\infty} \{\Gamma \in X : D_\Gamma(\iota_0) \leq n\} \\
&= \{\Gamma \in X : D_\Gamma(\iota_0) = \infty\} \\
&= \{\Gamma \in X : |\hat{\Lambda}_\Gamma(\iota_0)| = \infty\} \\
&= \{\Gamma \in X : \Gamma(\iota_0) = \mathrm{NH}\}
\end{aligned}$$

so instead of (2.24), we get

$$\mathbb{P}_{\iota_0}\left(X \setminus \bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0)\right) = \mathbb{P}_{\iota_0}(\{\Gamma \in X : \Gamma(\iota_0) = \mathrm{NH}\}) \tag{2.29}$$

$$= \mathbb{P}_{\iota_0}(\{\Gamma_{\iota_0}^{\mathrm{ran}} = \mathrm{NH}\}) = 0. \tag{2.30}$$

Suppose that $\mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) \leq 2p$ for all $n \in \mathbb{N}$. Then by continuity of measures [Lin18, Proposition 7.1.5],

$$0 = \mathbb{P}_{\iota_0}\left(X \setminus \bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0)\right) = 1 - \mathbb{P}_{\iota_0}\left(\bigcup_{n=1}^{\infty} \mathcal{G}^n(\iota_0)\right)$$

$$= 1 - \lim_{n\to\infty} \mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) \geq 1 - 2p.$$

because $\mathcal{G}^n(\iota_0) \subset \mathcal{G}^{n+1}(\iota_0)$ for all $n \in \mathbb{N} \cup \{0\}$. Since this is a contradiction, we have that there exists some $n \in \mathbb{N}$ for which $\mathbb{P}_{\iota_0}(\mathcal{G}^n(\iota_0)) < 2p$. Thus statement 1 holds for this Γ^{ran} as well. Furthermore, statement 2, 3 and 4 hold by the exact same arguments as before. This means that we can derive the contradiction in (2.26) by the exact same steps. We conclude that $\epsilon^{\mathrm{s}}_{\mathbb{P}\mathrm{hB}}(p) \geq \kappa/2$ for all $p \in [0, 1/2)$. ■

CHAPTER 3

Deterministic computational barriers for lasso

3.1 A practical example

Consider the following lasso problem

$$\underset{x\in\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|\tilde{A}x - y\|_2^2 + \lambda\|x\|_1 \tag{3.1}$$

for $m = 3$, $N = 2$, $\lambda \in (0, 1/\sqrt{3}]$, where $y = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \end{pmatrix}^\top$, and

$$A = \begin{pmatrix} \frac{1}{\sqrt{2}} - \alpha & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} - \alpha & -\frac{1}{\sqrt{2}} \\ 2\alpha & 0 \end{pmatrix} \tag{3.2}$$

for $\alpha > 0$, and $\tilde{A}$ is the standardized matrix for A. We want to compute a solution to this problem using MATLAB's *lasso*-function, and compare it to the true solution given by lemma 3.3.1, which is stated in the next section. The *lasso*-function standardizes the input matrix by centering and scaling the columns so they have mean zero and norm $\sqrt{m}$. Therefore, we can write

$$\tilde{A} = \begin{pmatrix} \mathbf{a}_1 - \bar{\mathbf{a}_1}\mathbf{1}_m & \cdots & \mathbf{a}_N - \bar{\mathbf{a}_N}\mathbf{1}_m \end{pmatrix} D$$

where $\bar{\mathbf{a}_i} = \frac{1}{m}\sum_{j=1}^m (\mathbf{a}_i)_j$ and D is the unique diagonal matrix such that the columns of $\tilde{A}$ have norm $\sqrt{m}$.

The *lasso*-function will by default standardize the so-called "design matrix" (A in our case) before fitting the model, and applies the regularization on x on the standardized scale. The estimated minimizer is returned on the original scale, however. Given the A and y as defined above as input, *lasso* should return Dx^*, where x^* is an estimate of the true minimizer of (3.1). Table 3.1 shows the result for decreasing values of α. The column labelled 'Warning' indicates whether the *lasso* function issued a warning after execution or not. If a warning was issued, then MATLAB met some sort of problem during the execution of the *lasso*-function. We see that lasso fails to compute a minimizer to (3.1) accurately. Moreover, the 'Warning' parameter does not reliably indicate whether the result is trustworthy or not. More often than not, no warning was issued even when the output was highly inaccurate.

What is happening here is not that MATLAB's *lasso*-function is implemented incorrectly. Rather, this is the manifestation of a deeper problem with the computability of lasso itself. Table 3.1 shows an interesting phenomenon where lasso is able to accurately approximate the true solution for the larger values of α, but completely fails at this task as α gets close to zero.

Default

α	**Error**	**Runtime**	**Warning**
2^{-1}	$2 \cdot 10^{-16}$	$< 0.01s$	0
2^{-7}	0.68	$< 0.01s$	0
2^{-15}	1.17	$< 0.01s$	0
2^{-20}	1.17	$< 0.01s$	0
2^{-24}	1.17	$< 0.01s$	0
2^{-26}	1.17	$< 0.01s$	0
2^{-28}	1.17	$< 0.01s$	0
2^{-30}	1.17	$< 0.01s$	0

'RelTol' $= \epsilon_{mach}$

α	**Error**	**Runtime**	**Warning**
2^{-1}	$2 \cdot 10^{-16}$	$< 0.01s$	0
2^{-7}	$6 \cdot 10^{-16}$	$0.02s$	0
2^{-15}	1.17	$0.27s$	1
2^{-20}	1.17	$0.28s$	1
2^{-24}	1.17	$0.27s$	1
2^{-26}	1.17	$0.27s$	1
2^{-28}	1.17	$< 0.01s$	0
2^{-30}	1.17	$< 0.01s$	0

'RelTol' $= \epsilon_{mach}$, 'MaxIter' $= \epsilon_{mach}^{-1}$

α	**Error**	**Runtime**	**Warning**
2^{-1}	$2 \cdot 10^{-16}$	$< 0.01s$	0
2^{-7}	$6 \cdot 10^{-16}$	$0.02s$	0
2^{-15}	$1 \cdot 10^{-11}$	$1522.7s$	0
2^{-20}	no output	$> 12h$	0
2^{-24}	no output	$> 12h$	0
2^{-26}	no output	$> 12h$	0
2^{-28}	1.17	$< 0.01s$	0
2^{-30}	1.17	$< 0.01s$	0

Table 3.1: Evaluating the performance of the *lasso*-function when applied to (3.1) with $\lambda = 0.1$. The first table shows the results using default settings. The middle and last tables show the results when changing the tolerance and maximum number of iterations allowed.

3.2 Computational barriers in terms of number of correct digits

The phenomenon demonstrated in section 3.1 is a result of theorem 3.2.1 listed below, which can be considered an addendum to theorem 3.3 in [BHV]. Given an $\lambda \in (0, 3/5]$, consider the lasso problem

$$\underset{x\in\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|\tilde{A}x - y\|_2^2 + \lambda\|x\|_1 \tag{3.3}$$

for $y \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times N}$, where $\tilde{A}$ is the unique standardized version of A. Theorem 3.2.1 says that, for any integer $K \geq 2$, there exists an input set Ω for (3.3) such that no algorithm can compute K correct decimal digits of the true solution for all the inputs in Ω. There is an algorithm that can compute $K-1$ digits for all the inputs in Ω, however any such algorithm will require arbitrary precision of the inputs, and therefore arbitrarily long runtime. Finally, there does exist an algorithm that has polynomial time and space complexity, that can compute $K-2$ digits for all the inputs in Ω.

The theorem is proved by construction of an input set Ω that gives rise to the aforementioned situation. However, the construction method makes it clear that the resulting Ω is only one of infinitely many input sets that exhibit similar behaviour. The matrices $A \in \mathbb{R}^{m\times N}$ appearing in the input set Ω that will be constructed in this paper, all satisfy $m > N \geq 2$. In contrast, the input set constructed in the proof of Theorem 3.3 in [BHV] use matrices $A \in \mathbb{R}^{m\times N}$ with $m < N$, and the result holds for lasso where A is not standardized.

Theorem 3.2.1. *Consider the solution map Ξ to the lasso problem (3.3) with $\lambda \in (0, 3/5]$ (and in the Turing case, let λ be rational as well). Let the metric on $\mathcal{M}_N$ be induced by the $\|\cdot\|_q$-norm, for an arbitrary $q \in [1, \infty]$. Let $K \geq 1$ be an integer. There exists a set of inputs*

$$\Omega = \bigcup_{\substack{m,N\in\mathbb{N}\\ m>N\geq 2}} \Omega_{m,N} \quad \textit{such that} \quad \Xi\colon \Omega_{m,N} \rightrightarrows \mathcal{M}_N \tag{3.4}$$

as well as Δ_1-information $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that

(i) *For $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ with $m, N \in \mathbb{N}$, $m > N \geq 2$, we have $\epsilon_B^s > 10^{-K}$.*

(ii) *If $K \geq 2$, we additionally have that $\epsilon_B^w > 10^{-(K-1)}$ for the same $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, with $m, N \in \mathbb{N}$ and $m > N \geq 2$. However, when considering the computational problems $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$, there exists an algorithm Γ that takes the dimensions m, N and any $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ as input, and satisfies*

$$dist_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}), \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-1)}.$$

(iii) *If $K \geq 2$, we also have that there exists an algorithm Γ that takes the dimensions m, N and any $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ as input, and satisfies*

$$dist_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}), \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-2)}.$$

Furthermore, in the Turing model, the runtime and space complexity of the Turing machine Γ are bounded by a polynomial in $n_{var} = mN + m$, and only a constant number of digits is read from the oracle tape.

The proof consists in constructing two types of lasso input sets: One that will induce a non-zero strong breakdown epsilon, and one that will induce a non-zero weak breakdown epsilon. For convenience we will call the members of the first set "strong" inputs, and the ones of the second set "weak" inputs. The union of these two sets across all valid input dimensions m, N results in the input set Ω in (3.4). In the next section, we define the inputs ι that make up Ω. The complete proof of this theorem is given in section 3.4.

Remark 3.2.2. In theorem 3.3 in [BHV], the size of the inputs $\iota = (y, A) \in \Omega_{m,N}$ are bounded in the sense that $\|y\|_\infty \leq 2$ and $\|A\|_{max} \leq 1$. Furthermore, the condition of the mapping Ξ, and the condition number of AA^* are both bounded. (See §8.1 as well as proposition 9.32 in [BHV]). Therefore, the phenomenon in theorem 3.2.1 is independent of whether or not the problem $\{\Xi, \Omega\}$ is well-conditioned. The same bounds can not be claimed for the input set Ω constructed in this paper, but this is merely a consequence of the initial choice of y and A in the experiment in section 3.1, which this Ω is based on.

3.3 Constructing the input set used to prove theorem 3.2.1

Inspired by the experiment conducted in section 3.1, we construct a family of matrices $A(\alpha, \beta, m, N) \in \mathbb{R}^{m\times N}$ and a family of vectors $y^A(y_c, m) \in \mathbb{R}^m$, where $\alpha, \beta \geq 0$ and $y_c > 0$, and $m > N \geq 2$. Let

$$A(\alpha,\beta,m,N) = \begin{pmatrix} \frac{1}{\sqrt{2}} - \alpha & \frac{1}{\sqrt{2}} - \beta \\ -\frac{1}{\sqrt{2}} - \alpha & -\frac{1}{\sqrt{2}} - \beta \\ 2\alpha & 2\beta \end{pmatrix} \oplus \begin{pmatrix} I_{N-2} \\ 0_{m-N-1\times N-2} \end{pmatrix} \tag{3.5}$$

$$y^A(y_c, m) = y_c \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & \cdots & 0 \end{pmatrix}^\top. \tag{3.6}$$

When there is no ambiguity, we will write $A = A(\alpha, \beta, m, N)$ and $y^A = y^A(y_c, m)$ for the sake of brevity. Notice that setting $m = 3, N = 2$, $\beta = 0$, and $y_c = 1$ yields the matrix (3.2) used in our MATLAB experiment.

The following lemma gives the exact solution to the problem 3.1.

Lemma 3.3.1. *Let $m, N \in \mathbb{N}$ with $m > N \geq 2$, and let $A = A(\alpha, \beta, m, N)$ and $y = y^A(y_c, m)$ be as in (3.5) and (3.6), respectively. Let $\lambda \in (0, y_c/\sqrt{m}]$, and suppose that at least one of α, β is equal to zero. Then if*

$$(w^0, w^1) \in \underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|\tilde{A}x^1 - x^0\mathbf{1}_m - y\|_2^2 + \lambda\|x^1\|_1, \tag{3.7}$$

where $\tilde{A}$ is the standardized matrix such that its columns satisfy $\tilde{\mathbf{a}}_i = (\mathbf{a}_i - \bar{\mathbf{a}_i}\mathbf{1}_m)d_i$ and $D = \operatorname{diag}(d_1, \cdots, d_N)$ is the unique diagonal matrix such that the $\tilde{\mathbf{a}}_i$ have norm $\sqrt{m}$, we have that $w^0 = 0$ and

$$Dw^1 = \begin{cases} \left\{(y_c - \lambda\sqrt{m})e_1\right\} & \text{if } \alpha = 0 < \beta \\ \left\{(y_c - \lambda\sqrt{m})e_2\right\} & \text{if } \beta = 0 < \alpha \\ \left\{(y_c - \lambda\sqrt{m})(te_1 + (1-t)e_2) : t \in [0,1]\right\} & \text{if } \alpha = \beta = 0 \end{cases}$$

Proof. Notice first that $\bar{\mathbf{a}_1} = \bar{\mathbf{a}_2} = 0$ and $\bar{a_3} = \cdots = \bar{\mathbf{a}_N} = \frac{1}{m}$ by (3.5). By the definition of the diagonal matrix D, we have that

$$\begin{aligned} d_1 &= \sqrt{m}/\|\mathbf{a}_1 - \bar{\mathbf{a}_1}\mathbf{1}_m\|_2 = \sqrt{m}\left[\left(\frac{1}{\sqrt{2}} - \alpha\right)^2 + \left(-\frac{1}{\sqrt{2}} - \alpha\right)^2 + 4\alpha^2\right]^{-1/2} \\ &= \sqrt{m}(6\alpha^2 + 1)^{-1/2} \end{aligned}$$

and

$$\begin{aligned} d_2 &= \sqrt{m}/\|\mathbf{a}_2 - \bar{\mathbf{a}_2}\mathbf{1}_m\|_2 = \sqrt{m}\left[\left(\frac{1}{\sqrt{2}} - \beta\right)^2 + \left(-\frac{1}{\sqrt{2}} - \beta\right)^2 + 4\beta^2\right]^{-1/2} \\ &= \sqrt{m}(6\beta^2 + 1)^{-1/2}. \end{aligned}$$

and

$$\begin{aligned} d_i &= \sqrt{m}/\|\mathbf{a}_i - \bar{\mathbf{a}_i}\mathbf{1}_m\|_2 = \sqrt{m}\big[(1-1/m)^2 + (m-1)(-1/m)^2\big]^{-1/2} \\ &= m(m-1)^{-1/2} \end{aligned}$$

for $3 \le i \le N$. In particular, we have that $d_1 \le \sqrt{m}$ with equality if and only if $\alpha = 0$, and similarly $d_2 \le \sqrt{m}$ with equality if and only if $\beta = 0$.

Next, write $\tilde{A} = (A - \bar{A})D$ by defining $\bar{A} = \begin{pmatrix}\bar{\mathbf{a}_1}\mathbf{1}_m & \cdots & \bar{\mathbf{a}_N}\mathbf{1}_m\end{pmatrix}$. Then we have that (3.7) is equivalent to

$$(w^0, w^1) \in \underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|(A - \bar{A})Dx^1 - x^0\mathbf{1}_m - y\|_2^2 + \lambda\|x^1\|_1,$$

which holds if and only if

$$(\hat{w}^0, \hat{w}^1) \in \underset{(x^0,x^1)\in\mathbb{R}\times\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|(A - \bar{A})x^1 - x^0\mathbf{1}_m - y\|_2^2 + \lambda\|D^{-1}x^1\|_1,$$

for $\hat{w}^1 = Dw^1$. Furthermore, if we let $a = (\bar{\mathbf{a}_1} \cdots \bar{\mathbf{a}_N})$, then

$$\begin{aligned} \|(A - \bar{A})x^1 - x^0\mathbf{1}_m - y\|_2^2 &= \|Ax^1 - \bar{A}x^1 - x^0\mathbf{1}_m - y\|_2^2 \\ &= \left[\left(\frac{1}{\sqrt{2}} - \alpha\right)x_1^1 + \left(\frac{1}{\sqrt{2}} - \beta\right)x_2^1 - \langle a, x^1\rangle - x^0 - y_c\frac{1}{\sqrt{2}}\right]^2 \\ &\quad + \left[\left(-\frac{1}{\sqrt{2}} - \alpha\right)x_1^1 + \left(-\frac{1}{\sqrt{2}} - \beta\right)x_2^1 - \langle a, x^1\rangle - x^0 + y_c\frac{1}{\sqrt{2}}\right]^2 \\ &\quad + (2\alpha x_1^1 + 2\beta x_2^1 - \langle a, x^1\rangle - x^0)^2 \\ &\quad + (x_3^1 - \langle a, x^1\rangle - x^0)^2 + \cdots + (x_N^1 - \langle a, x^1\rangle - x^0)^2 \\ &= \left[\frac{1}{\sqrt{2}}\big(x_1^1 + x_2^1 - y_c\big) - \big(\alpha x_1^1 + \beta x_2^1 + \langle a, x^1\rangle + x^0\big)\right]^2 \\ &\quad + \left[\frac{1}{\sqrt{2}}\big(x_1^1 + x_2^1 - y_c\big) + \big(\alpha x_1^1 + \beta x_2^1 + \langle a, x^1\rangle + x^0\big)\right]^2 \\ &\quad + (2\alpha x_1^1 + 2\beta x_2^1 - \langle a, x^1\rangle - x^0)^2 \\ &\quad + (x_3^1 - \langle a, x^1\rangle - x^0)^2 + \cdots + (x_N^1 - \langle a, x^1\rangle - x^0)^2 \end{aligned}$$

$$
\begin{aligned}
&= (x_1^1 + x_2^1 - y_c)^2 + 2(\alpha x_1^1 + \beta x_2^1 + \langle a, x^1\rangle + x^0)^2 \\
&\quad + (2\alpha x_1^1 + 2\beta x_2^1 - \langle a, x^1\rangle - x^0)^2 \\
&\quad + (x_3^1 - \langle a, x^1\rangle - x^0)^2 + \cdots + (x_N^1 - \langle a, x^1\rangle - x^0)^2 \\
&= (x_1^1 + x_2^1 - y_c)^2 + 6(\alpha x_1^1 + \beta x_2^1)^2 + 3(\langle a, x^1\rangle + x^0)^2 \\
&\quad + (x_3^1 - \langle a, x^1\rangle - x^0)^2 + \cdots + (x_N^1 - \langle a, x^1\rangle - x^0)^2
\end{aligned}
$$

Thus, we have that

$$
\begin{aligned}
&\frac{1}{2m}\|(A-\bar{A})x^1 - x^0\mathbf{1}_m - y\|_2^2 + \lambda\|D^{-1}x^1\|_1 \\
&\qquad \geq \frac{1}{2m}\|(A-\bar{A})x^1 - x^0\mathbf{1}_m - y\|_2^2 + \lambda\sum_{j=1}^{2}\left|\frac{x_j^1}{d_j}\right| \\
&\qquad \geq \frac{1}{2m}\|(A-\bar{A})x^1 - x^0\mathbf{1}_m - y\|_2^2 + \lambda\sum_{j=1}^{2}\frac{x_j^1}{d_j} \\
&\qquad \geq \frac{1}{2m}\|(A-\bar{A})x^1 - x^0\mathbf{1}_m - y\|_2^2 + \frac{\lambda}{\sqrt{m}}(x_1^1 + x_2^1) \\
&\qquad = \frac{1}{2m}\Big((x_1^1 + x_2^1 - y_c)^2 + 6(\alpha x_1^1 + \beta x_2^1)^2 + 3(\langle a, x^1\rangle + x^0)^2 \\
&\qquad\quad + (x_3^1 - \langle a, x^1\rangle - x^0)^2 + \cdots + (x_N^1 - \langle a, x^1\rangle - x^0)^2\Big) \\
&\qquad\quad + \frac{\lambda}{\sqrt{m}}(x_1^1 + x_2^1) \\
&\qquad \geq \frac{1}{2m}(x_1^1 + x_2^1 - y_c)^2 + \frac{\lambda}{\sqrt{m}}(x_1^1 + x_2^1) \\
&\qquad = \frac{1}{2m}\Big((x_1^1 + x_2^1 - y_c)^2 + 2\lambda\sqrt{m}(x_1^1 + x_2^1)\Big) \\
&\qquad = \frac{1}{2m}\Big((x_1^1 + x_2^1)^2 - 2\big(y_c - \lambda\sqrt{m}\big)(x_1^1 + x_2^1) + y_c^2\Big) \\
&\qquad = \frac{1}{2m}\Big(\big(x_1^1 + x_2^1 - \big(y_c - \lambda\sqrt{m}\big)\big)^2 - \big(y_c - \lambda\sqrt{m}\big)^2 + y_c^2\Big) \\
&\qquad \geq \frac{1}{2m}\Big(y_c^2 - \big(y_c - \lambda\sqrt{m}\big)^2\Big)
\end{aligned}
$$

Examining the inequalities in this calculation, we see that the first inequality is an equality if and only if $x_3^1 = \cdots = x_N^1 = 0$. The second inequality is an equality if both x_1^1 and x_2^1 are non-negative. The third inequality is an equality if and only if $x_1^1 = 0$ whenever $\beta = 0 < \alpha$, and $x_2^1 = 0$ whenever $\alpha = 0 < \beta$. If $\alpha = \beta = 0$, it is always an equality. The fourth inequality is an equality only if, in addition to the above, $\langle a, x^1\rangle + x^0 = 0$. However, since $\langle a, x^1\rangle = 0$ whenever $x_3^1 = \cdots = x_N^1 = 0$, it suffices to require that $x^0 = 0$. The last inequality is an equality if and only if $x_1^1 + x_2^1 = y_c - \lambda\sqrt{m}$. Notice that $y_c - \lambda\sqrt{m} \geq 0$ since $\lambda \in (0, y_c/\sqrt{m}]$, so this does not contradict the requirement that x_1^1 and x_2^1 should be non-negative.

Any minimizer $(\hat{w}^0, \hat{w}^1)$ must satisfy these conditions, and thus we have that $\hat{w}^0 = 0$, and $\hat{w}^1 = \big(y_c - \lambda\sqrt{m}\big)e_1$ if $\alpha = 0 < \beta$ and $\hat{w}^1 = \big(y_c - \lambda\sqrt{m}\big)e_2$ if $\beta = 0 < \alpha$. In the case where $\alpha = \beta = 0$, we get

$$
\hat{w}^1 \in \Big\{t\Big(y_c - \lambda\sqrt{m}\Big)e_1 + (1-t)\Big(y_c - \lambda\sqrt{m}\Big)e_2 : t \in [0,1]\Big\}.
$$

This proves the lemma. ■

We are now ready to construct the input set that will be used to prove theorem 3.2.1. For a given $\lambda \in (0, 3/5]$ and natural numbers $K \geq 1$, and $m > N \geq 2$, let

$$y^K(m) \coloneqq y^A(4 \cdot 10^{-K} + \lambda\sqrt{m}, m), \text{ and} \tag{3.8}$$

$$\Omega^s_{m,N,K} \coloneqq \{(y^K(m), A(\alpha, \beta, m, N)) : (\alpha, \beta) \in \mathcal{L}\} \tag{3.9}$$

where

$$\mathcal{L} \coloneqq \{[0, 1/4] \times \{0\} \cup \{0\} \times [0, 1/4]\}, \tag{3.10}$$

Similarly, if $K \geq 2$, let

$$\Omega^w_{m,N,K} \coloneqq \{(y^{K-1}(m), A(\alpha, \beta, m, N)) : (\alpha, \beta) \in \mathcal{L} \setminus \{z\}\}. \tag{3.11}$$

where $z = (0, 0)$. If $K = 1$, we let $\Omega^w_{m,N,K} \coloneqq \emptyset$. The superscripts s and w are used to note that the input sets will be used to show results connected to the strong breakdown epsilons and weak breakdown epsilons, respectively. For convenience, we call the inputs in $\Omega^w_{m,N,K}$ "weak" inputs, and the inputs in $\Omega^s_{m,N,K}$ "strong" inputs. Now, for each $N \geq 2$ and $m > N$, we define the fixed-dimension input set for a given $K \in \mathbb{N}$ as

$$\Omega_{m,N} = \Omega^s_{m,N,K} \cup \Omega^w_{m,N,K} \tag{3.12}$$

Lastly, we define the combined input set as the union of the fixed-dimension input sets:

$$\Omega = \bigcup_{\substack{m,N \in \mathbb{N} \\ m > N \geq 2}} \Omega_{m,N} \tag{3.13}$$

We now consider the lasso problem (3.3) for the inputs $\iota = (y, A) \in \Omega$. By lemma 3.3.1 we have that $\Xi(\iota) \subset \{4 \cdot 10^{-K+1} e_1, 4 \cdot 10^{-K+1} e_2\}$ if $\iota \in \Omega^w_{m,N,K}$ for any $N < m \leq 2$. Likewise, we have that $\Xi(\iota) \subset \{4 \cdot 10^{-K}(te_1 + (1-t)e_2) : t \in [0, 1]\}$ if $\iota \in \Omega^s_{m,N,K}$ for any $N < m \leq 2$. Figures 3.1 and 3.2 visualize the solutions.

Notice that for weak inputs, it is easy to construct an algorithm that outputs a solution that has less than 10^{-K+2} error. Indeed, simply outputting an x^N corresponding to any point within the intersection of the blue circles in figure 3.1 will do. However, for a solution that has less than 10^{-K+1} error, the algorithm needs to determine which of α or β is smaller. Since α and β may have infinite representations, determining which of them is smaller will in the worst case scenario take an infinite amount of input information. This is what induces a lower bound on the weak breakdown epsilon.

Similarly, it is easy to construct an algorithm that outputs a solution with less than 10^{-K+1} error for strong inputs. As before it suffices to output any x^N corresponding to a point within the intersection of the blue circles in figure 3.2. However, for a solution with less than 10^{-K} error, the algorithm needs to determine if one of α or β is smaller. But this is an impossible task, since the problem of determining if $\alpha = \beta$ or not is undecidable, even for general algorithms. This is what induces a lower bound on the strong breakdown epsilon.

The following lemmas give definite proof of the lower bounds on the breakdown epsilons for the strong input sets and weak inputs sets.

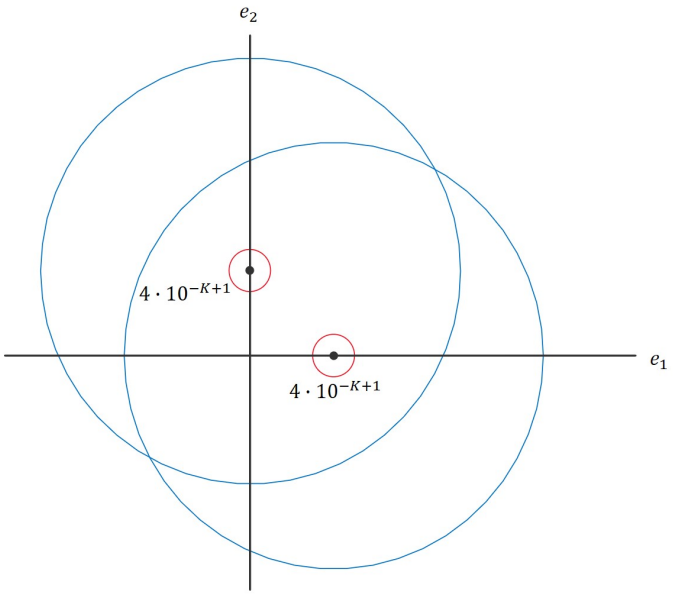


Figure 3.1: Graph of the solutions to the "weak" inputs along the first two dimensions. $\Xi(\iota)$ is $4 \cdot 10^{-K+1} e_1$ if $\alpha < \beta$, and $4 \cdot 10^{-K+1} e_2$ if $\beta < \alpha$. The red circles show the area within which the distance to the true solutions is less than 10^{-K+1}. The blue circles show the area within which the distance to the true solutions is less than 10^{-K+2}. In this example, distances are measured in the $\|\cdot\|_2$-norm, and we assume that $K \geq 2$.

Lemma 3.3.2. *Let $k, m, N \in \mathbb{N}$ with $m > N \geq 2$ and $k \geq 1$. Consider the computational problem $\{\Xi, \Omega^s_{m,N,k}, \mathcal{M}_N, \Lambda_{m,N}\}$, where Ξ is the solution map to the lasso problem (3.3), the input set $\Omega^s_{m,N,k}$ is as defined in (3.9), and the metric on $\mathcal{M}_N$ is induced by $\|\cdot\|_q$ for some $q \in [1, \infty]$. Then there exists a $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that, for the computational problem $\{\Xi, \Omega^s_{m,N,k}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, we have $\epsilon^s_B \geq 2^{1+1/q} \cdot 10^{-k}$, and in particular, $\epsilon^s_B > 10^{-k}$.*

Proof. The proof consists of constructing two input sequences $\{\iota^1_n\}_{n=1}^\infty, \{\iota^2_n\}_{n=1}^\infty$ and one input ι^0 in $\Omega^s_{m,N,k}$, as well as two sets $S^1, S^2 \subset \mathbb{R}^N$, that satisfy the conditions (a) - (c) of proposition 2.2.1. By the proposition, we will get the desired lower bound on the strong breakdown epsilon.

For the given $k \geq 1$ and $m > N \geq 2$, let $\iota^0 = (y^k(m), A(0, 0, m, N))$ and

$$\iota^1_n = (y^k(m), A(0, 2^{-1} \cdot 4^{-n}, m, N))$$
$$\iota^2_n = (y^k(m), A(2^{-1} \cdot 4^{-n}, 0, m, N)).$$

For ease of reading, let $A^0 = A(0, 0, m, N)$, $A^{1,n} = A(0, 2^{-1} \cdot 4^{-n}, m, N)$ and $A^{2,n} = A(2^{-1} \cdot 4^{-n}, 0, m, N)$. Let $S^1 = \{4 \cdot 10^{-k} e_1\}$ and $S^2 = \{4 \cdot 10^{-k} e_2\}$. We

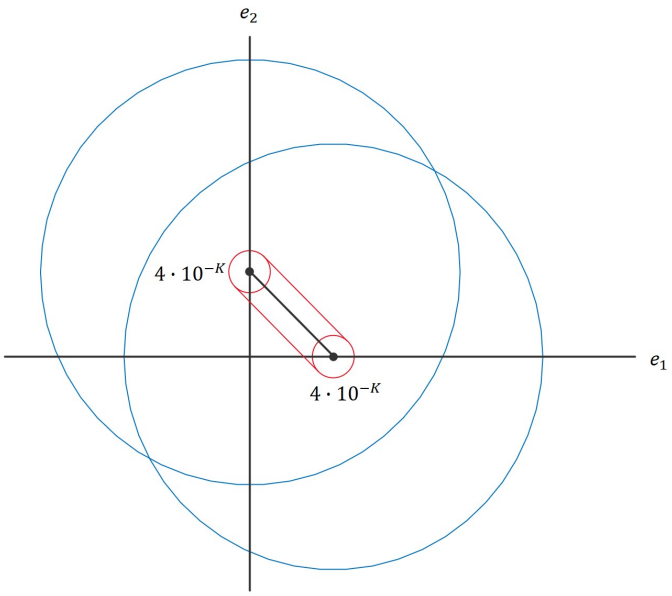


Figure 3.2: Graph of the solutions to the "strong" inputs along the first two dimensions. $\Xi(\iota)$ is $4 \cdot 10^{-K} e_1$ if $\alpha < \beta$, and $4 \cdot 10^{-K} e_2$ if $\beta < \alpha$. If $\alpha = \beta$, $\Xi(\iota)$ consists of the points on the line between and including these two points. The red circles show the area within which the distance to the true solutions is less than 10^{-K}. The blue circles show the area within which the distance to the true solutions is less than 10^{-K+1}. In this example, distances are measured in the $\|\cdot\|_2$-norm, and we assume that $K \geq 1$.

have that

$$\begin{aligned}\inf_{x_1 \in S^1,\, x_2 \in S^2} \|x_1 - x_2\|_q &= \|4 \cdot 10^{-k} e_1 - 4 \cdot 10^{-k} e_2\|_q \\ &= (2(4 \cdot 10^{-k})q)^{1/q} \\ &= 2^{2+1/q} \cdot 10^{-k}\end{aligned}$$

and by lemma 3.3.1 we have

$$\begin{aligned}\Xi(\iota_n^1) &= \Xi((y^k(m), A(0, 2^{-1} \cdot 4^{-n}, m, N)) = 4 \cdot 10^{-k} e_1 \in S^1 \\ \Xi(\iota_n^2) &= \Xi((y^k(m), A(2^{-1} \cdot 4^{-n}, 0, m, N)) = 4 \cdot 10^{-k} e_2 \in S^2\end{aligned}$$

for all $n \in \mathbb{N}$. So condition (a) of proposition 2.2.1 is satisfied. Next, we have that for any $f \in \Lambda_{m,N}$

$$\begin{aligned}|f(\iota_n^1) - f(\iota^0)| &= |f((y^k(m), A^{1,n})) - f((y^k(m), A^0))| \\ &\leq \max\{\|A^{1,n} - A^0\|_{\max}, \|y^k(m) - y^k(m)\|_\infty\} \\ &= \|A^{1,n} - A^0\|_{\max} = 4^{-n}\end{aligned}$$

and by the same steps, we get $|f(\iota_n^2) - f(\iota^0)| \leq 4^{-n}$. So condition (b) of proposition 2.2.1 is satisfied with $f(\iota^0)$ for all $f \in \Lambda_{m,N}$. Since $\iota^0 \in \Omega^s_{m,N,k}$, condition (c) is satisfied as well. By proposition 2.2.1 (ii), we conclude that $\epsilon_B^s \geq 2^{1+1/q} \cdot 10^{-k}$. In particular, we have that $\epsilon_B^s > 10^{-k}$. ■

Lemma 3.3.3. *Let $k, m, N \in \mathbb{N}$ with $m > N \geq 2$ and $k \geq 2$. Consider the computational problem $\{\Xi, \Omega^w_{m,N,k}, \mathcal{M}_N, \Lambda_{m,N}\}$, where Ξ is the solution map to the lasso problem (3.3), the input set $\Omega^w_{m,N,k}$ is as defined in (3.11), and the metric on $\mathcal{M}_N$ is induced by $\|\cdot\|_q$ for some $q \in [1, \infty]$. Then there exists a $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that, for the computational problem $\{\Xi, \Omega^w_{m,N,k}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, we have $\epsilon^w_B \geq 2^{1+1/q} \cdot 10^{-k+1}$, and in particular, $\epsilon^w_B > 10^{-k+1}$.*

Proof. The proof is very similar to the proof of Lemma 3.3.2 above. For the given $k, m, N \in \mathbb{N}$, let $\iota^0 = (y^{k-1}(m), A(0, 0, m, N))$ and

$$\iota^1_n = (y^{k-1}(m), A(0, 2^{-1} \cdot 4^{-n}, m, N))$$
$$\iota^2_n = (y^{k-1}(m), A(2^{-1} \cdot 4^{-n}, 0, m, N)).$$

Let $A^0 = A(0, 0, m, N)$, $A^{1,n} = A(0, 2^{-1} \cdot 4^{-n}, m, N)$ and $A^{2,n} = A(2^{-1} \cdot 4^{-n}, 0, m, N)$ as before. Finally, let $S^1 = \{4 \cdot 10^{-k+1} e_1\}$ and $S^2 = \{4 \cdot 10^{-k+1} e_2\}$. We have that

$$\begin{aligned} \inf_{x_1 \in S^1,\, x_2 \in S^2} \|x_1 - x_2\|_q &= \|4 \cdot 10^{-k+1} e_1 - 4 \cdot 10^{-k+1} e_2\|_q \\ &= (2(4 \cdot 10^{-k+1})^q)^{1/q} \\ &= 2^{2+1/q} \cdot 10^{-k+1} \end{aligned}$$

and by lemma 3.3.1 we have

$$\Xi(\iota^1_n) = \Xi((y^{k-1}(m), A(0, 2^{-1} \cdot 4^{-n}, m, N)) = 4 \cdot 10^{-k+1} e_1 \in S^1$$
$$\Xi(\iota^2_n) = \Xi((y^{k-1}(m), A(2^{-1} \cdot 4^{-n}, 0, m, N)) = 4 \cdot 10^{-k+1} e_2 \in S^2$$

for all $n \in \mathbb{N}$. Next, we have that for any $f \in \Lambda_{m,N}$

$$\begin{aligned} |f(\iota^1_n) - f(\iota^0)| &= |f((y^{k-1}(m), A^{1,n})) - f((y^{k-1}(m), A^0))| \\ &\leq \max\{\|A^{1,n} - A^0\|_{\max}, \|y^{k-1}(m) - y^{k-1}(m)\|_\infty\} \\ &= \|A^{1,n} - A^0\|_{\max} = 4^{-n} \end{aligned}$$

and by the same steps, we get $|f(\iota^2_n) - f(\iota^0)| \leq 4^{-n}$. So conditions (a) and (b) of proposition 2.2.1 are satisfied, and we conclude that $\epsilon^w_B \geq 2^{1+1/q} \cdot 10^{-k+1}$. In particular, $\epsilon^w_B > 10^{-k+1}$. ■

3.4 Proof of theorem 3.2.1

We now have the tools necessary to prove theorem 3.2.1. We separate the proof into three subsections, for each of the results (i) - (iii).

Proof of theorem 3.2.1 (i)

Consider the fixed dimensional problem $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$ where Ξ is the solution map to the lasso problem (3.3), and $\Omega_{m,N} = \Omega^s_{m,N,K} \cup \Omega^w_{m,N,K}$ as in (3.9) and (3.11). Lemma 3.3.2 establishes the existence of a

$$\hat{\Lambda}^s_{m,N} = \{f^s_{j,n} : j \leq n_{var}, n \in \mathbb{N}\} \in \mathcal{L}^1(\Lambda_{m,N})$$

such that for $\{\Xi, \Omega^s_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}^s_{m,N}\}$ we have $\epsilon^s_B > 10^{-K}$. Similarly, if $K \geq 2$, lemma 3.3.3 establishes the existence of a

$$\hat{\Lambda}^w_{m,N} = \{f^w_{j,n} : j \leq n_{var}, n \in \mathbb{N}\} \in \mathcal{L}^1(\Lambda_{m,N})$$

such that for $\{\Xi, \Omega^w_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}^w_{m,N}\}$ we have $\epsilon^w_B > 10^{-(K-1)}$.

Now, define $\hat{\Lambda}_{m,N} \coloneqq \{f_{j,n} : j \leq n_{var}, n \in \mathbb{N}\}$ where

$$f_{j,n} = \begin{cases} f^s_{j,n}(\iota) & \text{if} \quad \iota \in \Omega^s_{m,N,K} \\ f^w_{j,n}(\iota) & \text{if} \quad \iota \in \Omega^w_{m,N,K} \end{cases} \tag{3.14}$$

for $j \leq n_{var}$ and $n \in \mathbb{N}$. Since $\hat{\Lambda}^s_{m,N}$ and $\hat{\Lambda}^w_{m,N}$ provide Δ_1-information for $\{\Xi, \Omega^s_{m,N,K}, \mathcal{M}_N, \Lambda^s_{m,N}\}$ and $\{\Xi, \Omega^w_{m,N,K}, \mathcal{M}_N, \Lambda^w_{m,N}\}$, respectively, we have that $\hat{\Lambda}_{m,N}$ provides Δ_1-information for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$. Furthermore, $\Omega^s_{m,N,K}, \Omega^w_{m,N,K} \subset \Omega_{m,N}$, and

$$\hat{\Lambda}^s_{m,N} = \{f \restriction_{\Omega^s_{m,N,K}} : f \in \hat{\Lambda}_{m,N}\}$$
$$\hat{\Lambda}^w_{m,N} = \{f \restriction_{\Omega^w_{m,N,K}} : f \in \hat{\Lambda}_{m,N}\}.$$

By lemma 2.4.2, this implies that any breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ are at least as large as the corresponding breakdown epsilons for $\{\Xi, \Omega^s_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}^s_{m,N}\}$ and $\{\Xi, \Omega^w_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}^w_{m,N}\}$. Therefore, we have that

$$\epsilon^s_B > 10^{-K}, \text{ and}$$
$$\epsilon^w_B > 10^{-(K-1)} \text{ if } K \geq 2$$

for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, which establishes (i) as well as the first part of (ii) of theorem 3.2.1.

Proof of theorem 3.2.1 (ii)

Consider $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ where Ξ is the solution map to the lasso problem (3.3), and $\Omega_{m,N} = \Omega^s_{m,N,K} \cup \Omega^w_{m,N,K}$ as in (3.9) and (3.11). We already showed that $\epsilon^w_B > 10^{-(K-1)}$ for this computational problem in the proof of 3.2.1 (i) above. It remains to show the existence of a general algorithm that returns $K-1$ correct digits on all inputs of the problems $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$, for all $m, N \in N$ with $m > N \geq 2$. Specifically, we want to construct an algorithm Γ that, given m, N and any $\tilde{\iota} \in \tilde{\Omega}_{m,N}$, provides an output that satisfies $\text{dist}_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}), \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-1)}$.

To this end, we fix $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ and write $\tilde{\iota} = (\{y^{(n)}_j\}^\infty_{n=0}, \{A^{(n)}_{j,k}\}^\infty_{n=0})_{j,k}$ which corresponds to an $\iota = (y, A) \in \Omega_{m,N}$. Call the algorithm *Deterministic K-1 digit algorithm*. We need two so-called subroutines in this algorithm: One subroutine that can identify whether $\tilde{\iota}$ corresponds to an ι in $\Omega^s_{m,N,K}$ or $\Omega^w_{m,N,K}$, and one subroutine we can call upon if the latter is true (that is, if $\tilde{\iota}$ corresponds to an $\iota \in \Omega^w_{m,N,K}$). Call these subroutines *IdentifyStrongOrWeak* and *Weak*, respectively.

IdentifyStrongOrWeak
Inputs: The dimensions m, N.
Oracles: $\mathcal{O}_{vec}$ providing access to the components $y_j^{(n)}$ of an input $\tilde{\iota}$.
Output: Either 'InputStrong' or 'InputWeak'.
Procedure:

1. Compute w_1, w_2 such that $w_1 \approx \frac{1}{\sqrt{2}}$ and $w_2 \approx \frac{\sqrt{m}}{\sqrt{2}}$ to $4K$ bits of precision.
2. Compute $t_0 \approx 4 \cdot 10^{-K} w_1 + \lambda w_2$ and $t_1 \approx 4 \cdot 10^{-(K-1)} w_1 + \lambda w_2$ to $4K$ bits of precision.
3. Use $\mathcal{O}_{vec}$ to read $y_1' \coloneqq y_1^{(4K)}$.
4. If $y_1' \leq t_0 + 3 \cdot 2^{-4K}$, output 'InputStrong' and terminate. Else if $y_1' \geq t_1 - 3 \cdot 2^{-4K}$, output 'InputWeak' and terminate.

We need to prove that this subroutine correctly identifies whether $\tilde{\iota}$ corresponds to an ι in $\Omega^s_{m,N,K}$ ('InputStrong) or $\Omega^w_{m,N,K}$ ('InputWeak'). If $\iota \in \Omega^s_{m,N,K}$, then we have

$$\begin{aligned}
y_1 &= \Big(4 \cdot 10^{-K} + \lambda\sqrt{m}\Big)\frac{1}{\sqrt{2}} = 4 \cdot 10^{-K}\frac{1}{\sqrt{2}} + \lambda\frac{\sqrt{m}}{\sqrt{2}} \\
&= 4 \cdot 10^{-K}\Big(\frac{1}{\sqrt{2}} - w_1\Big) + \lambda\Big(\frac{\sqrt{m}}{\sqrt{2}} - w_2\Big) + 4 \cdot 10^{-K} w_1 + \lambda w_2 \\
&\leq 4 \cdot 10^{-K} \cdot 2^{-4K} + \lambda 2^{-4K} + t_0 + 2^{-4K} \\
&= 2^{-4K}(4 \cdot 10^{-K} + \lambda) + t_0 + 2^{-4K} \\
&\leq 2 \cdot 2^{-4K} + t_0,
\end{aligned}$$

where we have used that $|w_1 - 1/\sqrt{2}| \leq 2^{-4K}$, $|w_2 - \sqrt{m}/\sqrt{2}| \leq 2^{-4K}$, $|t_0 - (4 \cdot 10^{-K} w_1 + \lambda w_2)| \leq 2^{-4K}$, $\lambda \leq 3/5$, and $K \geq 1$. Thus $y_1' \leq y_1 + 2^{-4K} \leq t_0 + 3 \cdot 2^{-4K}$, and the subroutine outputs 'InputStrong' as it should.

If instead $\iota \in \Omega^w_{m,N,K}$, then we have that $K \geq 2$, since $\Omega^w_{m,N,1} = \emptyset$ by definition. Hence,

$$\begin{aligned}
y_1 &= \Big(4 \cdot 10^{-(K-1)} + \lambda\sqrt{m}\Big)\frac{1}{\sqrt{2}} = 4 \cdot 10^{-(K-1)}\frac{1}{\sqrt{2}} + \lambda\frac{\sqrt{m}}{\sqrt{2}} \\
&= 4 \cdot 10^{-(K-1)} w_1 + \lambda w_2 - 4 \cdot 10^{-(K-1)}\Big(w_1 - \frac{1}{\sqrt{2}}\Big) - \lambda\Big(w_2 - \frac{\sqrt{m}}{\sqrt{2}}\Big) \\
&\geq 4 \cdot 10^{-(K-1)} w_1 + \lambda w_2 - 4 \cdot 10^{-(K-1)} \cdot 2^{-4K} - \lambda \cdot 2^{-4K} \\
&\geq t_1 - 2^{-4K} - 2^{-4K}(4 \cdot 10^{-(K-1)} + \lambda) \\
&\geq t_1 - 3 \cdot 2^{-4K},
\end{aligned}$$

where we have again used that $|w_1 - 1/\sqrt{2}| \leq 2^{-4K}$, $|w_2 - \sqrt{m}/\sqrt{2}| \leq 2^{-4K}$, $|t_1 - (4 \cdot 10^{-(K-1)} w_1 + \lambda w_2)| \leq 2^{-4K}$, and $\lambda \leq 3/5$. Thus $y_1' \geq y_1 - 2^{-4K} \geq$

$t_1 - 3\cdot 2^{-4K}$, and the subroutine outputs 'InputWeak' as long as $y_1' \leq t_0 + 3\cdot 2^{-4K}$ is *not* satisfied. We see that this is indeed the case. Using the fact that $2^{-4K} < \frac{5}{8}\cdot 10^{-K}$, we get

$$\begin{aligned}
y_1' &\geq t_1 - 3\cdot 2^{-4K} \\
&\geq 4\cdot 10^{-K+1} w_1 + \lambda w_2 - 2^{-4K} - 3\cdot 2^{-4K} \\
&= 4\cdot 10^{-K} w_1 + \lambda w_2 + 36\cdot 10^{-K} w_1 - 4\cdot 2^{-4K} \\
&\geq t_0 - 2^{-4K} + 36\cdot 10^{-K} w_1 - 4\cdot 2^{-4K} \\
&\geq t_0 + 36\cdot 10^{-K}(1/\sqrt{2} - 2^{-4K}) - 5\cdot 2^{-4K} \\
&= t_0 + \frac{36}{\sqrt{2}} 10^{-K} - 2^{-4K}(36\cdot 10^{-K} + 5) \\
&> t_0 + \frac{36}{\sqrt{2}} 10^{-K} - \frac{5}{8}\cdot 10^{-K}(36\cdot 10^{-K} + 5) \\
&= t_0 + \left(\frac{36}{\sqrt{2}} - \frac{5}{8}(36\cdot 10^{-K} + 5)\right) 10^{-K} \\
&> t_0 + \left(\frac{36}{\sqrt{2}} - \frac{5}{8}\left(\frac{36}{10} + 5\right)\right) 10^{-K} \\
&> t_0 + 3\frac{5}{8}\cdot 10^{-K} \\
&> t_0 + 3\cdot 2^{-4K}.
\end{aligned}$$

We conclude that the subroutine *IdentifyStrongOrWeak* correctly identifies the cases 'InputStrong' and 'InputWeak'.

It is also worth noticing that the Turing runtime of this subroutine is polynomial in $\log(m)$. Let us look at the runtime of each of the steps:

1. In step 1 of the subroutine, we can use the Newton-Raphson iteration to compute $\sqrt{1/2}$ and $\sqrt{m/2}$ to $4K$ bits of precision as in [Mul05, p. 92-93]. Moreover, this can be done in polynomial time. Indeed, by using the Newton-Raphson iteration, division and square-root evaluation has the same complexity as multiplication; and for two b-bit numbers, multiplication can be done in $\mathcal{O}(b\log(b))$ time. Consequently, the runtime for calculating $w_1 \approx \sqrt{1/2}$ to $4K$ bits of precision is polynomial in $4K$, and the runtime for calculating $w_2 \approx \sqrt{m/2}$ to $4K$ bits of precision is polynomial in $4K$ and $\mathrm{Len}(m)$. However, $\mathcal{O}(4K) = \mathcal{O}(1)$ since K is assumed to be fixed. We conclude that step 1 of the subroutine has an overall runtime polynomial in $\log(m)$, as $\mathrm{Len}(m) = \mathcal{O}(\log(m))$.

2. In step 2 of the subroutine, t_0 and t_1 are computed by finitely many arithmetic operations on fractions whose bit encoding lengths are bounded by polynomials in $\mathrm{Len}(10^{-K})$, $\mathrm{Len}(\lambda)$, $\mathrm{Len}(w_1)$ and $\mathrm{Len}(w_2)$. As noted before, K and $\lambda \in \mathbb{Q}$ are fixed, and $\mathrm{Len}(w_1)$ is constant. However, $\mathrm{Len}(w_2)$ is bounded by its runtime, that is, a polynomial in $\log(m)$. So the overall complexity of step 2 is also $\mathcal{O}(\log(m))$.

3. The call to $\mathcal{O}_{vec}$ in step 3 has cost $\mathcal{O}(K) = \mathcal{O}(1)$, which is constant. However, the cost of reading $y_1' = y_1^{(4K)}$ corresponds to its bit size, which by definition 2.1.9 is bounded by $\mathcal{O}(\log(m))$.

4. Lastly, the bit sizes of t_0, t_1 and y_1' are all bounded by a polynomial in $\log(m)$, since they are computed in $\mathcal{O}(\log(m))$ time. This means that the comparisons in step 4 can be done in $\mathcal{O}(\log(m))$ time. The output 'InputStrong' or 'InputWeak' can be given as a boolean, and thus takes $\mathcal{O}(1)$ time.

We conclude that the overall Turing runtime of *IdentifyStrongOrWeak* is polynomial in $\log(m)$. Furthermore, there is only one call to the oracle $\mathcal{O}_{vec}$, and this call requires only $4K$ digits of the input. Since K is fixed, the number of digits requested from the oracle is constant. Next, we have the subroutine *Weak*.

Weak
Inputs: The dimensions m, N, and $k_\epsilon \in \mathbb{N}$.
Oracles: $\mathcal{O}_{mat}$ providing access to the components $A^{(n)}_{j,k}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ with $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-k_\epsilon}$
Procedure:

1. For $n = 1, 2, ...$:
   a. Use $\mathcal{O}_{mat}$ to read $A^{(n)}_{3,1}$ and $A^{(n)}_{3,2}$. Set $d = A^{(n)}_{3,1} - A^{(n)}_{3,2}$.
   b. If $d > 2^{-n+1}$, output $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K+1} e_1\|_q \leq 10^{-k_\epsilon}$, and terminate.
   c. If $d < -2^{-n+1}$, output $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K+1} e_2\|_q \leq 10^{-k_\epsilon}$, and terminate.
   d. Otherwise, continue loop.

We claim that if subroutine *Weak* is applied to an input $\tilde{\iota}$ corresponding to an $\iota \in \Omega^w_{m,N,K}$, the subroutine always terminates with an output $x \in \mathbb{D}^N$ satisfying $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-k_\epsilon}$. The proof goes as follows.

Fix the iteration n of the loop in step 1. Then $\mathcal{O}_{mat}$ is used to read $A_{3,1} = 2\alpha$ and $A_{3,2} = 2\beta$ to precision 2^{-n}. Thus $A^{(n)}_{3,1} \leq 2\alpha + 2^{-n}$ and $A^{(n)}_{3,2} \geq 2\beta - 2^{-n}$. Furthermore, by the definition of $\Omega^w_{m,N,K}$ we have that $\alpha \neq \beta$ but that one of α, β is equal to zero. Suppose that $\alpha = 0 < \beta$. Then $d = A^{(n)}_{3,1} - A^{(n)}_{3,2} \leq (2\alpha + 2^{-n}) - (2\beta - 2^{-n}) < 2^{-n+1}$. So the subroutine will never terminate at step b. However, if n is sufficiently large, then $d \leq (2\alpha + 2^{-n}) - (2\beta - 2^{-n}) = -2\beta + 2^{-n+1} < -2^{-n+1}$, so the subroutine will output an x such that $\|x - 4 \cdot 10^{-K+1} e_2\|_q \leq 10^{-k_\epsilon}$, and terminate at step c. By lemma 3.3.1, we get $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) = \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) = \|x - 4 \cdot 10^{-K+1} e_2\|_q \leq 10^{-k_\epsilon}$ as wanted. The case $\beta = 0 < \alpha$ is identical, except now d will never be smaller than -2^{-n+1}, and instead we will (after a sufficient number of iterations), have $d > 2^{-n+1}$. The subroutine thus outputs and terminates at step b, which also yields $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-k_\epsilon}$. We are now ready to construct the *Deterministic K-1 digit algorithm*.

Deterministic K-1 digit algorithm
Inputs: The dimensions m, N.
Oracles: $\mathcal{O}_{vec}$ and $\mathcal{O}_{mat}$ providing access to the components $y_j^{(n)}$ and $A_{j,k}^{(n)}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ with $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-1)}$.
Procedure:

1. Execute *IdentifyStrongOrWeak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N)$.
2. If it outputs 'InputWeak', execute *Weak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, k_\epsilon = K-1)$ and terminate.
3. If it outputs 'InputStrong', output an $x \in \mathbb{D}^N$ with $\|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \leq 10^{-K}$ and terminate.

Since both *IdentifyStrongOrWeak* and *Weak* always terminate, so will this algorithm. We need to prove that the output $x \in \mathbb{D}^N$ of the algorithm satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-1)}$. Consider the two possible cases:

Case 1: The input $\tilde{\iota}$ corresponds to an ι in $\Omega^s_{m,N,K}$. In this case *IdentifyStrongOrWeak* will output 'InputStrong', and the algorithm executes step 3. By lemma 3.3.1, we have that $\Xi(\iota) \cap \{4 \cdot 10^{-K} e_1, 4 \cdot 10^{-K} e_2\} \neq \emptyset$. Thus

$$\begin{aligned}
\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) &= \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) \\
&\leq \max\{\|x - 4 \cdot 10^{-K} e_1\|_q, \|x - 4 \cdot 10^{-K} e_2\|_q\} \\
&\leq \|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q + \|2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \\
&\leq 10^{-K} + 2^{1+1/q} \cdot 10^{-K} < 10^{-K+1}.
\end{aligned}$$

Case 2: The input $\tilde{\iota}$ corresponds to an ι in $\Omega^w_{m,N,K}$ In this case *IdentifyStrongOrWeak* will output 'InputWeak', and the algorithm will execute *Weak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, k_\epsilon = K-1)$ and then terminate. As was shown earlier, the output from *Weak* satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-k_\epsilon} = 10^{-(K-1)}$.

We conclude that the *Deterministic K-1 digit algorithm* always terminates with an output x that satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-1)}$. This establishes part (ii) of theorem 3.2.1.

Proof of theorem 3.2.1 (iii)

We want to show the existence of an algorithm that returns $K-2$ correct digits on all inputs of the problems $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$, for all $m, N \in N$ with $m > N \geq 2$. Specifically, we will construct an algorithm Γ that, given m, N and any $\tilde{\iota} \in \tilde{\Omega}_{m,N}$, provides an output that satisfies $\text{dist}_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}), \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-2)}$. Furthermore, the algorithm will have polynomial runtime and space complexity in the Turing model, and the number of digits it requires from the oracles will be bounded by a polynomial

in $\log(n_{var})$, where $n_{var} = mN + m$ is the number of variables in a given input. The algorithm is stated below.

Polynomial time K-2 digit algorithm
Inputs: The dimensions m, N.
Oracles: $\mathcal{O}_{vec}$ and $\mathcal{O}_{mat}$ providing access to the components $y_j^{(n)}$ and $A_{j,k}^{(n)}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ with $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-2)}$.
Procedure:

1. Execute *IdentifyStrongOrWeak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N)$.
2. If it outputs 'InputWeak', output an $x \in \mathbb{D}^N$ with $\|x - 2 \cdot 10^{-(K-1)}e_1 - 2 \cdot 10^{-(K-1)}e_2\|_q \leq 10^{-(K-1)}$, and terminate.
3. If it outputs 'InputStrong', output an $x \in \mathbb{D}^N$ with $\|x - 2 \cdot 10^{-K}e_1 - 2 \cdot 10^{-K}e_2\|_q \leq 10^{-(K-1)}$, and terminate.

Note that this algorithm will always terminate, since *IdentifyStrongOrWeak* always terminates with either 'InputStrong' or 'InputWeak' as output. We need to show that the output $x \in \mathbb{D}^N$ satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-2)}$. Consider the two possible cases:

Case 1: The input $\tilde{\iota}$ corresponds to an ι in $\Omega^w_{m,N,K}$. Then *IdentifyStrongOrWeak* will output 'InputWeak', so the algorithm executes step 2, and terminates. By lemma 3.3.1 we have that $\Xi(\iota) \cap \{4 \cdot 10^{-K+1}e_1, 4 \cdot 10^{-K+1}e_2\} \neq \emptyset$. Thus

$$\begin{aligned}
\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) &= \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) \\
&\leq \max\{\|x - 4 \cdot 10^{-K+1}e_1\|_q, \|x - 4 \cdot 10^{-K+1}e_2\|_q\} \\
&\leq \|x - 2 \cdot 10^{-K+1}e_1 - 2 \cdot 10^{-K+1}e_2\|_q \\
&\quad + \|2 \cdot 10^{-K+1}e_1 - 2 \cdot 10^{-K+1}e_2\|_q \\
&\leq 10^{-K+1} + 2^{1+1/q} \cdot 10^{-K+1} < 10^{-K+2}.
\end{aligned}$$

Case 2: The input $\tilde{\iota}$ corresponds to an ι in $\Omega^s_{m,N,K}$. Then *IdentifyStrongOrWeak* will output 'InputStrong', so the algorithm executes step 3, and terminates. By lemma 3.3.1 we have that $\Xi(\iota) \cap \{4 \cdot 10^{-K}e_1, 4 \cdot 10^{-K}e_2\} \neq \emptyset$. Thus

$$\begin{aligned}
\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) &= \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) \\
&\leq \max\{\|x - 4 \cdot 10^{-K}e_1\|_q, \|x - 4 \cdot 10^{-K}e_2\|_q\} \\
&\leq \|x - 2 \cdot 10^{-K}e_1 - 2 \cdot 10^{-K}e_2\|_q + \|2 \cdot 10^{-K}e_1 - 2 \cdot 10^{-K}e_2\|_q \\
&\leq 10^{-K+1} + 2^{1+1/q} \cdot 10^{-K} < 10^{-K+2}.
\end{aligned}$$

In both cases, we have that $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq 10^{-(K-2)}$. It remains to show that the algorithm has polynomial time and space complexity in the Turing model. We analyze the algorithm step by step:

1. Recall from the discussion in the proof of theorem 3.2.1 (ii) that *IdentifyStrongOrWeak* has runtime polynomial in $\log(m)$.

2. The runtime of computing $x \in \mathbb{D}^N$ if step 2 is executed, is bounded by a polynomial in K and N. It suffices to estimate $x_1 = x_2 \approx 2 \cdot 10^{-K+1}$ to $4K$ bits of precision. By setting $x_3 = \cdots = x_N = 0$ for the remaining components, x satisfies $\|x - 2 \cdot 10^{-(K-1)} e_1 - 2 \cdot 10^{-(K-1)} e_2\|_q \leq 10^{-(K-1)}$ as wanted. Estimating $2 \cdot 10^{-K+1}$ can be done in $\mathcal{O}(b \log(b))$ time by using the Newton-Raphson iteration to compute $b = 4K$ bits of the reciprocal of $10^K/2$ [Mul05, p. 92-93]. Setting $x_3 = \cdots = x_N = 0$ for the remaining components requires $\mathcal{O}(N)$ time. The overall runtime of computing x is therefore bounded by a polynomial in K and N.

3. The runtime of computing $x \in \mathbb{D}^N$ such that $\|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \leq 10^{-(K-1)}$ if step 3 is executed, is bounded by a polynomial in K and N. It suffices to estimate $x_1 = x_2 \approx 2 \cdot 10^{-K}$ to $4K$ bits of precision, and set the remaining components of x to zero. This has runtime bounded by a polynomial in K and N, by the same argument as before.

Since K is fixed, this means that the overall runtime is polynomial in $\log(m)$ and N, and thus bounded above by a polynomial in n_{var}. Note that this upper bound on the Turing runtime implies the same upper bound on the space complexity of the algorithm. We conclude that the *Polynomial time K-2 digit algorithm* has time and space complexity that is polynomial in n_{var}. Furthermore, the only call to the oracles occur in the execution of the subroutine $\mathit{IdentifyStrongOrWeak}^{\mathcal{O}_{vec}, \mathcal{O}_{mat}}(m, N)$ in step 1. The number of digits requested from the oracles in *IdentifyStrongOrWeak* is contant, thus the *Polynomial time $K-2$ digit algorithm* only requires a constant number of digits from the oracles as well. This establishes part (iii) of the theorem.

3.5 Computational barriers in terms of approximation error

So far, the results have been stated in terms of the number of correct digits achievable. These results can easily be generalized by stating them in terms of approximation error instead. This leads to fascinating similarities with the field of hardness of approximation, which will be explored in section 3.7. We start by rewriting theorem 3.2.1 in terms of approximation error, and with exact values for the breakdown epsilons. Their proofs are very similar, and only minor tweaks are needed.

Theorem 3.5.1. *Consider the solution map Ξ to the lasso problem (3.3) with $\lambda \in (0, 3/5]$ (and in the Turing case, let λ be rational as well). Let the metric on $\mathcal{M}_N$ be induced by the $\|\cdot\|_q$-norm, for an arbitrary $q \in [1, \infty]$. Let $K \geq 1$ be an integer. There exists a set of inputs*

$$\Omega = \bigcup_{\substack{m,N \in \mathbb{N} \\ m > N \geq 2}} \Omega_{m,N} \quad \textit{such that} \quad \Xi \colon \Omega_{m,N} \rightrightarrows \mathcal{M}_N \tag{3.15}$$

as well as Δ_1-information $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that

(i) *For* $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ *with* $m, N \in \mathbb{N}$, $m > N \geq 2$, *we have*

$$\epsilon_B^s = 2^{1+1/q} \cdot 10^{-K}. \tag{3.16}$$

(ii) *If* $K \geq 2$, *we additionally have that for the same* $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, *with* $m, N \in \mathbb{N}$ *and* $m > N \geq 2$,

$$\epsilon_B^w = 2^{1+1/q} \cdot 10^{-K+1}. \tag{3.17}$$

However, if $K = 1$ *then* $\epsilon_B^w = \epsilon_B^s$.

(iii) *When considering the computational problems* $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$, *there exists an algorithm* Γ *that takes the dimensions* m, N *and any* $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ *as well as an error threshold* $\epsilon_a > 0$ *as input, and satisfies*

$$dist_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}, \epsilon_a), \tilde{\Xi}(\tilde{\iota})) \leq \epsilon \text{ for all } \epsilon = 2^{1+1/q} \cdot 10^{-K} + \epsilon_a.$$

(iv) *There exists an algorithm* Γ *that takes the dimensions* m, N *and any* $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ *as well as an error threshold* $\epsilon_a > 0$ *as input, and satisfies*

$$dist_{\mathcal{M}}(\Gamma(m, N, \tilde{\iota}, \epsilon_a), \tilde{\Xi}(\tilde{\iota})) \leq \epsilon \text{ for all } \epsilon = 2^{1+1/q} \cdot 10^{-K+1} + \epsilon_a$$

such that, in the Turing model, the runtime and space complexity of the Turing machine Γ *are bounded by a polynomial in* $\log(1/\epsilon_a)$ *and* $n_{var} = mN + m$, *and the number of digits read from the oracle tape is bounded by a polynomial in* $\log(n_{var})$.

The proof of this theorem is given in the next section.

Remark 3.5.2 (The breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$). The various breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ are equivalent to the breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$. Note that by (i) and (ii) of the theorem above, and remark 2.3.6, the breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ satisfy $\epsilon_B^s \geq 2^{1+1/q} \cdot 10^{-K}$ and $\epsilon_B^w \geq 2^{1+1/q} \cdot 10^{-K+1}$ if $K \geq 2$, or $\epsilon_B^w \geq 2^{1+1/q} \cdot 10^{-K}$ if $K = 1$. However, by statement (iii) of the theorem, there is an algorithm for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ that computes ϵ-approximations for all inputs, for any $\epsilon > 2^{1+1/q} \cdot 10^{-K}$. This means that we can compute ϵ-approximations for ϵ arbitrarily close to $2^{1+1/q} \cdot 10^{-K}$. Thus we have that the exact value of the strong breakdown epsilon for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ is $\epsilon_B^s = 2^{1+1/q} \cdot 10^{-K}$. By the same argument, we have that statement (iv) of the theorem implies that the exact value of the weak breakdown epsilon for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ is $\epsilon_B^w = 2^{1+1/q} \cdot 10^{-K+1}$ if $K \geq 2$, and $\epsilon_B^w = 2^{1+1/q} \cdot 10^{-K}$ if $K = 1$.

3.6 Proof of theorem 3.5.1

Proof. Fix $K \geq 1$ and let $m, N \in \mathbb{N}$ be such that $m > N \geq 2$. Consider $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$ where $\Omega_{m,N} = \Omega^s_{m,N,K} \cup \Omega^w_{m,N,K}$ as in (3.9) and (3.11). By lemma 3.3.2 we have that there exists some $\hat{\Lambda}^s_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such

that $\epsilon_B^s \geq 2^{1+1/q} \cdot 10^{-K}$ for $\{\Xi, \Omega_{m,N,K}^s, \mathcal{M}_N, \hat{\Lambda}_{m,N}^s\}$. Similarly, if $K \geq 2$, then lemma 3.3.3 establishes the existence of a $\hat{\Lambda}_{m,N}^w \in \mathcal{L}^1(\Lambda_{m,N})$ such that $\epsilon_B^w \geq 2^{1+1/q} \cdot 10^{-K+1}$ for $\{\Xi, \Omega_{m,N,K}^w, \mathcal{M}_N, \hat{\Lambda}_{m,N}^w\}$. Let $\hat{\Lambda}_{m,N} \coloneqq \{f_{j,n} : j \leq n_{var}, n \in \mathbb{N}\}$, where the $f_{j,n}$ are defined as in (3.14). As in the proof of theorem 3.2.1(i), we have that $\hat{\Lambda}_{m,N}$ provides Δ_1 information for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$, and that the breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ are at least as large as the corresponding breakdown epsilons for $\{\Xi, \Omega_{m,N,K}^s, \mathcal{M}_N, \hat{\Lambda}_{m,N}^s\}$ and $\{\Xi, \Omega_{m,N,K}^w, \mathcal{M}_N, \hat{\Lambda}_{m,N}^w\}$. Thus we have that

$$\epsilon_B^s \geq 2^{1+1/q} \cdot 10^{-K} \tag{3.18}$$

and

$$\epsilon_B^w \geq 2^{1+1/q} \cdot 10^{-K+1} \text{ if } K \geq 2. \tag{3.19}$$

for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$.

Now consider the computational problem with Δ_1-information $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$. As before, we fix the notation for a $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ and write $\tilde{\iota} = (\{y_j^{(n)}\}_{n=0}^{\infty}, \{A_{j,k}^{(n)}\}_{n=0}^{\infty})_{j,k}$ which corresponds to an $\iota = (y, A) \in \Omega_{m,N}$. By making minor alterations to the subroutine *Weak*, we can construct an ϵ-approximation algorithm that works for all $\epsilon > 2^{1+1/q} \cdot 10^{-K}$.

Weak
Inputs: The dimensions m, N, and error threshold $\epsilon_a \in \mathbb{D}$.
Oracles: $\mathcal{O}_{mat}$ providing access to the components $A_{j,k}^{(n)}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ with $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$
Procedure:

1. For $n = 1, 2, ...$:
    a. Use $\mathcal{O}_{mat}$ to read $A_{3,1}^{(n)}$ and $A_{3,2}^{(n)}$. Set $d = A_{3,1}^{(n)} - A_{3,2}^{(n)}$.
    b. If $d > 2^{-n+1}$, output $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K+1} e_1\|_q \leq \epsilon_a$, and terminate.
    c. If $d < -2^{-n+1}$, output $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K+1} e_2\|_q \leq \epsilon_a$, and terminate.
    d. Otherwise, continue loop.

As before, *Weak* uses $\mathcal{O}_{mat}$ to read $A_{3,1}^{(n)}$ and $A_{3,2}^{(n)}$ to increasing precision n until it can determine whether $A_{3,1} < A_{3,2}$ or $A_{3,1} > A_{3,2}$ (i.e. which of α or β is larger for the input $\tilde{\iota}$). It then outputs $x \in \mathbb{D}^N$ sufficiently close to the true solution, which is $4 \cdot 10^{-K+1} e_1$ if $\alpha < \beta$ or $4 \cdot 10^{-K+1} e_2$ if $\alpha > \beta$. By the same argument as before, *Weak* always terminates with an output $x \in \mathbb{D}^N$ that satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$.

The following ϵ-approximation algorithm for $\epsilon > 2^{1+1/q} \cdot 10^{-K}$ is almost identical to the *Deterministic K - 1 digit algorithm* constructed previously, but

takes an error threshold ϵ_a as an additional input argument.

Deterministic $\epsilon > 2^{1+1/q} \cdot 10^{-K}$ algorithm
Inputs: The dimensions $m, N \in \mathbb{N}$ and positive error threshold $\epsilon_a \in \mathbb{D}$.
Oracles: $\mathcal{O}_{vec}$ and $\mathcal{O}_{mat}$ providing access to the components $y_j^{(n)}$ and $A_{j,k}^{(n)}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ with $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K}$.
Procedure:

1. Execute *IdentifyStrongOrWeak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N)$.
2. If it outputs 'InputWeak', execute *Weak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, \epsilon_a)$ and terminate.
3. If it outputs 'InputStrong', output an $x \in \mathbb{D}^N$ that satisfies $\|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$ and terminate.

This algorithm always terminates, since the subroutines *IdentifyStrongOrWeak* and *Weak* also always terminate. It remains to show that the output $x \in \mathbb{D}^N$ satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K}$. Consider the two possible cases for the input $\tilde{\iota}$:

Case 1: The input $\tilde{\iota}$ corresponds to an ι in $\Omega^w_{m,N,K}$. Then *IdentifyStrongOrWeak* will output 'InputWeak', and the algorithm executes *Weak*$^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, \epsilon_a)$ in step 2, which always outputs an $x \in \mathbb{D}^N$ such that $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a < \epsilon_a + 2^{1+1/q} \cdot 10^{-K}$.

Case 2: The input $\tilde{\iota}$ corresponds to an ι in $\Omega^s_{m,N,K}$. Then *IdentifyStrongOrWeak* will output 'InputStrong', and the algorithm skips step 2 and executes step 3. By lemma 3.3.1 we have that $\Xi(\iota) \cap \{4 \cdot 10^{-K} e_1, 4 \cdot 10^{-K} e_2\} \neq \emptyset$. Thus,

$$\begin{aligned}
\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) &= \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) \\
&\leq \max\{\|x - 4 \cdot 10^{-K} e_1\|_q, \|x - 4 \cdot 10^{-K} e_2\|_q\} \\
&\leq \|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q + \|2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \\
&\leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K}.
\end{aligned}$$

In both cases, the output $x \in \mathbb{D}^N$ of the algorithm always satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K}$. If the input ϵ_a is chosen such that $\epsilon_a + 2^{1+1/q} \cdot 10^{-K} \leq \epsilon$, then we have $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K} \leq \epsilon$. We conclude that the *Deterministic $\epsilon > 2^{1+1/q} \cdot 10^{-K}$ algorithm* provides ϵ-approximations for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ whenever $\epsilon > 2^{1+1/q} \cdot 10^{-K}$. Note that if $\epsilon \leq 2^{1+1/q} \cdot 10^{-K}$, the algorithm does not work, as no $\epsilon_a \in \mathbb{D}$ such that $0 < \epsilon_a + 2^{1+1/q} \cdot 10^{-K} \leq \epsilon$ exists. Indeed, since we proved that $\epsilon_B^s \geq 2^{1+1/q} \cdot 10^{-K}$ for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, we have that there does not exist an algorithm that can provide ϵ-approximations for $\epsilon < 2^{1+1/q} \cdot 10^{-K}$ for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ either. However, the existence of the *Deterministic $\epsilon > 2^{1+1/q} \cdot 10^{-K}$ algorithm* implies that we can compute ϵ-approximations arbitrarily close to $2^{1+1/q} \cdot 10^{-K}$

accuracy. Thus, we have that the exact value for the strong breakdown epsilon is $\epsilon_B^s = 2^{1+1/q} \cdot 10^{-K}$ for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, as stated in the theorem.

Consider now instead an $\epsilon > 2^{1+1/q} \cdot 10^{-K+1}$. By making small alterations to the *Polynomial time K - 2 digit algorithm*, we can construct an ϵ-approximation algorithm that has polynomial time and space complexity.

Polynomial $\epsilon > 2^{1+1/q} \cdot 10^{-K+1}$ algorithm
Inputs: The dimensions $m, N \in \mathbb{N}$, and positive error threshold $\epsilon_a \in \mathbb{D}$.
Oracles: $\mathcal{O}_{vec}$ and $\mathcal{O}_{mat}$ providing access to the components $y_j^{(n)}$ and $A_{j,k}^{(n)}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ with $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K+1}$.
Procedure:

1. Execute *IdentifyStrongOrWeak*$^{\mathcal{O}_{vec}, \mathcal{O}_{mat}}(m, N)$.
2. If it outputs 'InputWeak', output an $x \in \mathbb{D}^N$ with $\|x - 2 \cdot 10^{-K+1} e_1 - 2 \cdot 10^{-K+1} e_2\|_q \leq \epsilon_a$, and terminate.
3. If it outputs 'InputStrong', output an $x \in \mathbb{D}^N$ with $\|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$, and terminate.

Since *IdentifyStrongOrWeak* always terminates with either 'InputStrong' or 'InputWeak' as output, it is clear that this algorithm will always terminate as well. As before, we consider the two possible cases:

Case 1: The input $\tilde{\iota}$ corresponds to an ι in $\Omega_{m,N,K}^w$. Then *IdentifyStrongOrWeak* will output 'InputWeak', so the algorithm executes step 2 and terminates. By lemma 3.3.1 we have that $\Xi(\iota) \cap \{4 \cdot 10^{-K+1} e_1, 4 \cdot 10^{-K+1} e_2\} \neq \emptyset$. Then

$$\begin{aligned}
\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) &= \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) \\
&\leq \max\{\|x - 4 \cdot 10^{-K+1} e_1\|_q, \|x - 4 \cdot 10^{-K+1} e_2\|_q\} \\
&\leq \|x - 2 \cdot 10^{-K+1} e_1 - 2 \cdot 10^{-K+1} e_2\|_q \\
&\quad + \|2 \cdot 10^{-K+1} e_1 - 2 \cdot 10^{-K+1} e_2\|_q \\
&\leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K+1}
\end{aligned}$$

Case 2: The input $\tilde{\iota}$ corresponds to an ι in $\Omega_{m,N,K}^s$. Then *IdentifyStrongOrWeak* will output 'InputStrong', so the algorithm skips step 2, and executes step 3 before terminating. By lemma 3.3.1 we have that $\Xi(\iota) \cap \{4 \cdot 10^{-K} e_1, 4 \cdot 10^{-K} e_2\} \neq \emptyset$. Then

$$\begin{aligned}
\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) &= \text{dist}_{\mathcal{M}}(x, \Xi(\iota)) \\
&\leq \max\{\|x - 4 \cdot 10^{-K} e_1\|_q, \|x - 4 \cdot 10^{-K} e_2\|_q\} \\
&\leq \|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q + \|2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \\
&\leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K} \\
&< \epsilon_a + 2^{1+1/q} \cdot 10^{-K+1}
\end{aligned}$$

Thus the output $x \in \mathbb{D}^N$ satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a + 2^{1+1/q} \cdot 10^{-K+1}$. If $\epsilon_a \in \mathbb{D}$ is chosen to be sufficiently small such that $\epsilon_a + 2^{1+1/q} \cdot 10^{-K+1} \leq \epsilon$, then we have $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon$. As before, this only works for $\epsilon > 2^{1+1/q} \cdot 10^{-K+1}$, otherwise no such $\epsilon_a \in \mathbb{D}$ exists. It remains to show that the algorithm has time and space complexity bounded by some polynomial. Analyzing the algorithm step by step shows that:

1. The subroutine *IdentifyStrongOrWeak* is unchanged, so recall from the discussion in the proof of theorem 3.2.1 that it has runtime polynomial in $\log(m)$ and only requests $\mathcal{O}(1)$ digits from the oracles.

2. To calculate an $x \in \mathbb{D}^N$ such that $\|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$, it suffices to estimate $x_1 = x_2 \approx 2 \cdot 10^{-K}$ to b bits of precision, where b is some positive integer that satisfies $2^{-b} \leq \epsilon_a/2$. By setting $x_3 = \cdots = x_N = 0$ for the remaining components, x satisfies $\|x - 2 \cdot 10^{-K} e_1 - 2 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$. The runtime of computing x is then polynomial in N and $\log(1/\epsilon_a)$, and the argument for this claim goes as follows. Suppose $\epsilon_a \in \mathbb{D}$ is given by its (finite) binary representation (see definition 2.1.9) and that $\epsilon_a < 1$. This last assumption is natural since we want to investigate the time complexity as $\epsilon_a \to 0$, and if $\epsilon_a \geq 1$ then x_1 and x_2 can be computed in constant time. To find b, read ϵ_a and count the number d of 0s after the binary point, before the first 1 appears. Then $\epsilon_a < 2^{-d}$, but $\epsilon_a \geq 2^{-(d+1)}$. Setting $b = d+2$ ensures $2^{-b} \leq \epsilon_a/2$ as wanted. One can estimate $2 \cdot 10^{-K}$ to b bits of precision in $\mathcal{O}(b \log(b))$ time, by using the Newton-Raphson iteration to compute the reciprocal of $10^K/2$ (as in [Mul05, p. 92-93]). We have that $b = \mathcal{O}(d) < \mathcal{O}(\log(1/\epsilon_a))$, which means that the runtime of computing x_1 and x_2 is bounded by a polynomial in $\log(1/\epsilon_a)$. Setting $x_3 = \cdots = x_N = 0$ for the remaining components requires $\mathcal{O}(N)$ time. The overall runtime of computing x is therefore bounded by a polynomial in $\log(1/\epsilon_a)$ and N.

3. By the same argument as above, the runtime of calculating $x \in \mathbb{D}^N$ such that $\|x - 2 \cdot 10^{-K+1} e_1 - 2 \cdot 10^{-K+1} e_2\|_q \leq \epsilon_a$, is also bounded by a polynomial in $\log(1/\epsilon_a)$ and N.

We conclude that the overall runtime of the algorithm is polynomial in $\log(m)$, N, and $\log(1/\epsilon_a)$. Additionally, since the only call to the oracles occurs in the execution of the subroutine *IdentifyStrongOrWeak*, the number of digits needed from the oracles is constant.

The existence of the *Polynomial* $\epsilon > 2^{1+1/q} \cdot 10^{-K+1}$ *algorithm* means that we can compute ϵ-approximations for ϵ arbitrarily close to ϵ_B^w, with only a constant number of digits needed from the oracles. By definition 2.8 of the weak breakdown epsilon, this means that the inequality in (3.19) is an equality. Thus $\epsilon_B^w = 2^{1+1/q} \cdot 10^{-K+1}$ for $\{\Xi, \Omega, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ whenever $K \geq 2$.

Note that if $K = 1$, then $\Omega_{m,N,K}^w = \emptyset$ for all m, N. Consequently, Ω consists only of strong inputs, which means that step 2 of the *Deterministic* $\epsilon > 2^{1+1/q} \cdot 10^{-K}$ *algorithm* is never executed. But this means that it has runtime polynomial in $\log(m)$, N, and $\log(1/\epsilon_a)$ and only requires a constant number of digits from the oracles, by the same argument as was used above. So we can compute ϵ-approximations for ϵ arbitrarily close to $2^{1+1/q} \cdot 10^{-K}$, with only a constant number of digits needed from the oracles. By definition

2.8, we get $\epsilon_B^w \leq 2^{1+1/q} \cdot 10^{-K}$. However, $\epsilon_B^w \geq \epsilon_B^s = 2^{1+1/q} \cdot 10^{-K}$ by (2.16). Therefore, $\epsilon_B^w = \epsilon_B^s$ for $\{\Xi, \Omega, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ whenever $K = 1$. This concludes the proof of theorem 3.5.1.

■

For convenience, we will refer to the *Deterministic* $\epsilon > 2^{1+1/q} \cdot 10^{-K}$ *algorithm* as the *Deterministic* ϵ*-approximation algorithm* from now on, with the understanding that $\epsilon > \epsilon_B^s$. Similarly, we refer to the *Polynomial* $\epsilon > 2^{1+1/q} \cdot 10^{-K+1}$ *algorithm* as the *Polynomial* ϵ*-approximation algorithm* with the understanding that $\epsilon > \epsilon_B^w$.

3.7 Phase transitions and relation to hardness of approximation

The phenomenon described in theorem 3.5.1 leads to phase transitions reminiscent of the field of hardness of approximation. This field arose from the desire to find polynomial time approximation algorithms providing approximate solutions to NP-hard optimization problems. A problem L is NP-hard if every problem in the complexity class NP is polynomial time reducible to L. Recall that NP is the class of problems for which there exists polynomial time non-deterministic Turing machines that solves them. See [AB09; Pap94; Sip13] for a more in-depth explanation of NP and NP-hardness. It is widely believed that P $\neq$ NP, which would mean that for NP-hard problems, there does not exist a deterministic Turing machine that computes solutions in polynomial time. However, one may still be able to construct polynomial time Turing machines that compute approximate solutions instead. It turns out that finding approximate solutions to NP-hard problems is not always possible. A 1976 paper by T. Gonzalez and S. Sahni showed that some NP-hard optimization problems are NP-hard to approximate within a given error threshold as well [SG76].

Taking basis in the theory from [Pap94], suppose we have an optimization problem that depends on n variables, with domain $\Omega \subset \mathbb{R}^n$. For each $\iota \in \Omega$, let $F(\iota) \subset \mathbb{R}^d$ denote the set of feasible solutions, and let

$$\mathrm{OPT}(\iota) \coloneqq \min_{x \in F(\iota)} f_\iota(x) \tag{3.20}$$

where $f_\iota : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ is the cost function (often called the objective function) we want to minimize. There are several optimization problems that fit into this framework that are NP-hard; The Travelling Salesperson Problem, for example. This means that, unless P = NP, a range of optimization problems can not be solved by Turing machines in a realistic timeframe. However, there may exist polynomial time algorithms that can give approximate solutions instead. For an $\epsilon \geq 0$, an algorithm Γ is called an ϵ*-approximation algorithm* for the optimization problem given by (3.20) if, for all $\iota \in \Omega$ we have $\Gamma(\iota) \in F(\iota)$ and

$$f_\iota(\Gamma(\iota)) \leq (1 + \epsilon)\,\mathrm{OPT}(\iota). \tag{3.21}$$

An ϵ-approximation algorithm for which there exists a polynomial $\mathrm{pol} : \mathbb{R} \to \mathbb{R}$ such that $\mathbf{Runtime}_\Gamma(\iota) \leq \mathrm{pol}(n)$ for all $\iota \in \Omega$ is called a *polynomial time*

ϵ-approximation algorithm. An output $\Gamma(\iota)$ satisfying (3.21) is called an *ϵ-approximate solution*.

It turns out that for some NP-hard optimization problems, there exists a polynomial time ϵ-approximation algorithm for all $\epsilon > 0$. In other words, approximability has no limits. On the other hand, there also exists NP-hard optimization problems that have a threshold beyond which no polynomial time ϵ-approximation algorithm exists. We refer to this threshold as the *approximation threshold* of a problem, and define it as in [Pap94, p. 300] by

$$\begin{aligned}\epsilon_A \coloneqq \inf\{\epsilon \geq 0 : \text{there exists a polynomial time} \\ \epsilon\text{-approximation algorithm}\}.\end{aligned} \tag{3.22}$$

An example of an optimization problem for which $\epsilon_A = 0$ is Knapsack [p. 305][Pap94]. An even stronger result than $\epsilon_A = 0$, is if there exists an algorithm Γ that for each $\epsilon > 0$ and $\iota \in \Omega$, satisfies $\Gamma(\iota) \in F(\iota)$ and (3.21), and has runtime polynomial in n that depends on ϵ. Such an algorithm is referred to as a *polynomial time approximation scheme* (PTAS). If the runtime is polynomial in $1/\epsilon$ as well, it is called a *fully polynomial time ϵ-approximation scheme* (FPTAS).

On the other hand, if P $\neq$ NP, there are several optimization problems where $\epsilon_A > 0$. Some examples are the Travelling Salesperson Problem, the Cycle Cover Problem, and the Clique Problem; see [Pap94; SG76]. A non-zero approximation threshold implies that the problem of computing ϵ-approximate solutions is in P for $\epsilon > \epsilon_A$, but not in P for $\epsilon < \epsilon_A$. This sharp phase transition leads to the situation illustrated in figure 3.3.

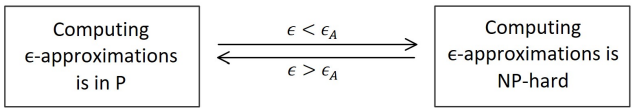


Figure 3.3: Phase transitions for NP-hard optimization problems with $\epsilon_A > 0$, if P $\neq$ NP.

Note that a key difference between the optimization problems (3.20) from hardness of approximation and lasso, is that the former seeks the minimum of the cost/objective function, while the latter seeks the minimizer itself. Furthermore, the algorithms we have constructed for lasso in this thesis do not access the inputs $\tilde{\iota} \in \tilde{\Omega}$ directly, but through oracles. Although the context is different, theorem 3.5.1 shows that the problem of computing ϵ-approximations to lasso minimizers has similar phase transitions, characterized by the strong and weak breakdown epsilons.

To be able to classify the problem of computing ϵ-approximations for lasso, we need to define some suitable complexity classes. Let $\mathrm{P}_\epsilon^{\mathcal{O}}$ denote the class of computational problems with Δ_1-information, for which ϵ-approximations can be computed by an oracle Turing machine with access to an oracle for each input $\tilde{\iota} \in \tilde{\Omega}$, with runtime polynomial in the number of variables n_{var} in the input. Similarly, let $\mathrm{EXPTIME}_\epsilon^{\mathcal{O}}$ denote the class of computational problems with Δ_1-information, for which ϵ-approximations can be computed by an oracle Turing machine, with access to an oracle for each input $\tilde{\iota} \in \tilde{\Omega}$, with $\mathcal{O}(2^{\mathrm{pol}(n_{var})})$ runtime for some polynomial function $\mathrm{pol}(n_{var})$.

Consider the input set Ω for which the result of theorem 3.5.1 holds. Then there are phase transitions for the computational problem $\{\Xi, \Omega\}^{\Delta_1}$, where Ξ is the solution map to the lasso problem (3.3). We have that for $\epsilon > \epsilon_B^w$, computing ϵ-approximations can be done in polynomial time, and the problem is therefore in $\mathrm{P}_\epsilon^{\mathcal{O}}$. On the other hand, if $\epsilon < \epsilon_B^s$, then ϵ-approximations are non-computable in general. However, if $\epsilon_B^s < \epsilon < \epsilon_B^w$, then ϵ-approximations are computable, but require an arbitrarily high amount of input information in the worst case. This clearly means that the problem is not in $\mathrm{P}_\epsilon^{\mathcal{O}}$; it is not even in $\mathrm{EXPTIME}_\epsilon^{\mathcal{O}}$. This is visualized in figure 3.4.

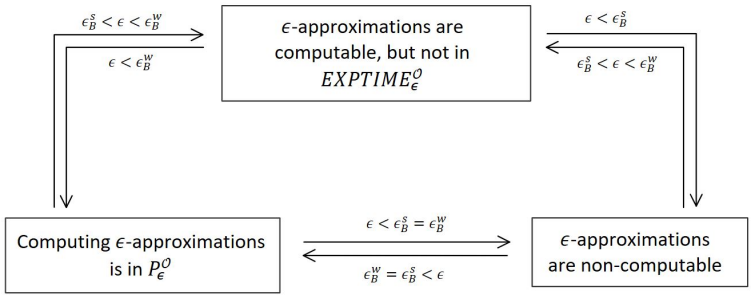


Figure 3.4: Phase transitions for lasso with domain Ω given by theorem 3.5.1.

The sharp phase transitions between ϵ-approximations being computable in polynomial time, and ϵ-approximations not being computable at all, occurs when $K = 1$. In this case, all inputs in Ω are "strong" inputs, and $\epsilon_B^w = \epsilon_B^s$. This means that the *Deterministic ϵ-approximation algorithm* is polynomial time, by the same argument used to show that the *Polynomial ϵ-approximation algorithm* is polynomial time. Subsequently, ϵ-approximations are computable in polynomial time for $\epsilon > \epsilon_B^w = \epsilon_B^s$, but non-computable for $\epsilon < \epsilon_B^w = \epsilon_B^s$. When $K \geq 2$, we have $\epsilon_B^w > \epsilon_B^s$, and therefore the problem does not transition directly between these two phases. Instead, we have a middle-ground; ϵ-approximations are computable, but not in $\mathrm{EXPTIME}_\epsilon^{\mathcal{O}}$, for $\epsilon_B^s < \epsilon < \epsilon_B^w$.

Unlike the phase transitions that occur in figure 3.3, the phase transitions in figure 3.4 are independent of whether P = NP or not. Furthermore, the result given by theorem 3.5.1 is stronger than a hardness of approximation result in the sense that the phase transitions are not between P and NP-hardness, but between $\mathrm{P}_\epsilon^{\mathcal{O}}$ and having completely unbounded runtime, or directly between $\mathrm{P}_\epsilon^{\mathcal{O}}$ and non-computability.

CHAPTER 4

Randomized computational barriers for lasso

In this chapter, the impossibility results for lasso will be strengthened by showing that even randomized algorithms are susceptible to the phenomenon outlined in theorem 3.2.1 and theorem 3.5.1. For the same input set that was constructed in chapter 3, we will show that randomized algorithms can not provide ϵ-approximations better than $\epsilon \geq 2^{1+1/q} \cdot 10^{-K}$ with probability greater than $1/2$ for all inputs, if we require that they always halt. However, if we relax the conditions by allowing algorithms that have a non-zero probability of not halting, we can construct an algorithm that produces arbitrarily good ϵ-approximations with probability at least $2/3$ for all inputs. While this is a positive result, it is also shown that this probability can not be improved. Indeed, for any $p < 1/3$ all algorithms will, for at least one input, fail to produce an ϵ-approximation better than $\epsilon \geq 2^{1+1/q} \cdot 10^{-K}$, with probability greater than p.

Recall that the lower bounds on the (deterministic) breakdown epsilons in theorem 3.2.1 and theorem 3.5.1 were given by lemmas 3.3.2 and 3.3.3. The proof of these lemmas were based on invoking proposition 2.2.1. However, by instead invoking proposition 2.4.5, we get lower bounds on the probabilistic breakdown epsilons as well.

4.1 Computational barriers for randomized algorithms

We start with generalizing lemmas 3.3.2 and 3.3.3 to hold for the probabilistic breakdown epsilons.

Lemma 4.1.1. *Let $k, m, N \in \mathbb{N}$ with $m > N \geq 2$ and $k \geq 1$. Consider the computational problem $\{\Xi, \Omega^s_{m,N,k}, \mathcal{M}_N, \Lambda_{m,N}\}$, where Ξ is the solution map to the lasso problem (3.3), the input set $\Omega^s_{m,N,k}$ is as defined in (3.9), and the metric on $\mathcal{M}_N$ is induced by $\|\cdot\|_q$ for some $q \in [1, \infty]$. Then there exists a $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that, for the computational problem $\{\Xi, \Omega^s_{m,N,k}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, we have $\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/2)$ and $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/3)$.*

Proof. The proof is identical to the proof of lemma 3.3.2, except that instead of invoking proposition 2.2.1 we invoke proposition 2.4.5 to get the lower bounds on the breakdown epsilons. Let $\{\iota^1_n\}_{n=1}^{\infty}$, $\{\iota^2_n\}_{n=1}^{\infty}$, ι^0 in $\Omega^s_{m,N,k}$ as well as S^1

and S^2 in $\mathbb{R}^N$, be the same as in the proof of lemma 3.3.2. Then conditions (a) - (c) of proposition 2.4.5 are satisfied. By result (ii) of the proposition, we get $\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/2)$ and $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/3)$. ■

Lemma 4.1.2. *Let $k, m, N \in \mathbb{N}$ with $m > N \geq 2$ and $k \geq 2$. Consider the computational problem $\{\Xi, \Omega^w_{m,N,k}, \mathcal{M}_N, \Lambda_{m,N}\}$, where Ξ is the solution map to the lasso problem (3.3), the input set $\Omega^w_{m,N,k}$ is as defined in (3.11), and the metric on $\mathcal{M}_N$ is induced by $\|\cdot\|_q$ for some $q \in [1, \infty]$. Then there exists a $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that, for the computational problem $\{\Xi, \Omega^w_{m,N,k}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, we have $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-k+1}$ for $p \in [0, 1/2)$.*

Proof. As above, the proof is identical to the proof of lemma 3.3.3, except that we invoke proposition 2.4.5 instead of proposition 2.2.1. Let $\{\iota^1_n\}_{n=1}^\infty$, $\{\iota^2_n\}_{n=1}^\infty$, ι^0 in $\Omega^w_{m,N,k}$ as well as S^1 and S^2 in $\mathbb{R}^N$, be the same as in the proof of lemma 3.3.3. Then conditions (a) and (b) of proposition 2.4.5 are satisfied. By result (i) of the proposition, we get $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-k+1}$ for $p \in [0, 1/2)$. ■

We can now state and prove the following randomized impossibility theorem, which can be considered an extension of theorem 3.5.1.

Theorem 4.1.3. *Consider the solution map Ξ to the lasso problem (3.3) with $\lambda \in (0, 3/5]$. Let the metric on $\mathcal{M}_N$ be induced by the $\|\cdot\|_q$-norm, for an arbitrary $q \in [1, \infty]$. Let $K \geq 1$ be an integer. For the same set of inputs*

$$\Omega = \bigcup_{\substack{m,N \in \mathbb{N} \\ m > N \geq 2}} \Omega_{m,N} \quad \text{such that} \quad \Xi \colon \Omega_{m,N} \rightrightarrows \mathcal{M}_N \tag{4.1}$$

and Δ_1-information $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ as in theorem 3.5.1, we additionally have that

(i) *For $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ with $m, N \in \mathbb{N}$, $m > N \geq 2$, we have*

$$\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) = 2^{1+1/q} \cdot 10^{-K} \text{ for all } p < 1/2, \text{ and} \tag{4.2}$$
$$\epsilon^s_{\mathbb{P}\mathrm{B}}(p) = 2^{1+1/q} \cdot 10^{-K} \text{ for all } p < 1/3. \tag{4.3}$$

(ii) *If $K \geq 2$, we additionally have that for the same $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, with $m, N \in \mathbb{N}$ and $m > N \geq 2$,*

$$\epsilon^w_{\mathbb{P}\mathrm{B}}(p) = 2^{1+1/q} \cdot 10^{-K+1} \text{ for all } p < 1/2. \tag{4.4}$$

However, if $K = 1$ then $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) = \epsilon^s_{\mathbb{P}\mathrm{B}}(p)$ for all $p < 1/2$.

(iii) *When considering the computational problems $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$, there exists a randomized algorithm Γ^{ran} with a non-zero probability of not halting, that takes the dimensions m, N and any $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ as input, as well as an error threshold $\epsilon_a \geq 0$, and satisfies*

$$\mathbb{P}_{\tilde{\iota}}(\mathrm{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}(m, N, \tilde{\iota}, \epsilon_a), \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a) \geq 2/3.$$

The proof of this theorem is given in the next section, however, we first give a short summary of the breakdown epsilons we have established for the lasso computational problem so far. The conjunction of this theorem and theorem 3.5.1 establishes the following relationships between the breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, where Ξ is the lasso problem (3.3) and $\Omega_{m,N}$ is as in (3.12), assuming $K \geq 2$:

$$\begin{aligned} \epsilon_B^s &= 2^{1+1/q} \cdot 10^{-K} && (4.5)\\ &= \epsilon_{\mathbb{P}\mathrm{hB}}^s(p) \text{ for } p \in [0, 1/2) && (4.6)\\ &= \epsilon_{\mathbb{P}\mathrm{B}}^s(p) \text{ for } p \in [0, 1/3), && (4.7) \end{aligned}$$

and

$$\begin{aligned} \epsilon_B^w &= 2^{1+1/q} \cdot 10^{-K+1} && (4.8)\\ &= \epsilon_{\mathbb{P}\mathrm{B}}^w(p) \text{ for } p \in [0, 1/2). && (4.9) \end{aligned}$$

Remark 4.1.4 (The breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$). We once again have that the various breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$ are equivalent to the breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$. Indeed, by statements (i) and (ii) in the theorem above, and remark 2.3.6, we have that $\epsilon_{\mathbb{P}\mathrm{hB}}^s(p) \geq 2^{1+1/q} \cdot 10^{-K}$ for all $p < 1/2$, $\epsilon_{\mathbb{P}\mathrm{B}}^s(p) \geq 2^{1+1/q} \cdot 10^{-K}$ for all $p < 1/3$, and $\epsilon_{\mathbb{P}\mathrm{B}}^w(p) \geq 2^{1+1/q} \cdot 10^{-K+1}$ for all $p < 1/2$, for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$, if $K \geq 2$. Recall from remark 3.5.2 that we also have $\epsilon_B^s = 2^{1+1/q} \cdot 10^{-K}$ and $\epsilon_B^w = 2^{1+1/q} \cdot 10^{-K+1}$ as well. However, by proposition 2.4.1, we know that $\epsilon_{\mathbb{P}\mathrm{hB}}^s(p) \leq \epsilon_B^s$, $\epsilon_{\mathbb{P}\mathrm{B}}^s(p) \leq \epsilon_B^s$, and $\epsilon_{\mathbb{P}\mathrm{B}}^w(p) \leq \epsilon_B^w$ for all $p \in [0, 1)$. Hence, we have that $\epsilon_{\mathbb{P}\mathrm{hB}}^s(p) = 2^{1+1/q} \cdot 10^{-K}$ for all $p < 1/2$, $\epsilon_{\mathbb{P}\mathrm{B}}^s(p) = 2^{1+1/q} \cdot 10^{-K}$ for all $p < 1/3$, and $\epsilon_{\mathbb{P}\mathrm{B}}^w(p) = 2^{1+1/q} \cdot 10^{-K+1}$ for all $p < 1/2$, for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1}$. The argument for the case when $K = 1$ is identical.

4.2 Proof of theorem 4.1.3

Proof. Fix $K \geq 1$ and let $m, N \in \mathbb{N}$ be such that $m > N \geq 2$, and consider $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$ where $\Omega_{m,N} = \Omega_{m,N,K}^s \cup \Omega_{m,N,K}^s$ is as in (3.9) and (3.11). By lemma 4.1.1, there exists some $\hat{\Lambda}_{m,N}^s \in \mathcal{L}^1(\Lambda_{m,N})$ such that

$$\begin{aligned} &\epsilon_{\mathbb{P}\mathrm{hB}}^s(p) \geq 2^{1+1/q} \cdot 10^{-K} \text{ for } p \in [0, 1/2), \text{ and}\\ &\epsilon_{\mathbb{P}\mathrm{B}}^s(p) \geq 2^{1+1/q} \cdot 10^{-K} \text{ for } p \in [0, 1/3) \end{aligned}$$

for $\{\Xi, \Omega_{m,N,K}^s, \mathcal{M}_N, \hat{\Lambda}_{m,N}^s\}$. By lemma 4.1.2 we similarly have that if $K \geq 2$, there exists some $\hat{\Lambda}_{m,N}^w \in \mathcal{L}^1(\Lambda_{m,N})$ such that

$$\epsilon_{\mathbb{P}\mathrm{B}}^w(p) \geq 2^{1+1/q} \cdot 10^{-K+1} \text{ for } p \in [0, 1/2)$$

for $\{\Xi, \Omega_{m,N,K}^w, \mathcal{M}_N, \hat{\Lambda}_{m,N}^w\}$. Let $\hat{\Lambda}_{m,N} \coloneqq \{f_{j,n} : j \leq n_{var}, n \in \mathbb{N}\}$, where the $f_{j,n}$ are defined as in (3.14). As in the proof of theorem 3.2.1 (i), we then have that $\hat{\Lambda}_{m,N}$ provides Δ_1-information for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}$ Furthermore, $\hat{\Lambda}_{m,N}^s = \{f \restriction_{\Omega_{m,N,K}^s} : f \in \hat{\Lambda}_{m,N}\}$ and $\hat{\Lambda}_{m,N}^w = \{f \restriction_{\Omega_{m,N,K}^w}:$

$f \in \hat{\Lambda}_{m,N}\}$. By lemma 2.4.2 we have that the breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ are at least as large as the corresponding breakdown epsilons for $\{\Xi, \Omega^s_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}^s_{m,N}\}$ and $\{\Xi, \Omega^w_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}^w_{m,N}\}$. We conclude that

$$\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq 2^{1+1/q} \cdot 10^{-K} \text{ for } p \in [0, 1/2),$$
$$\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-K} \text{ for } p \in [0, 1/3), \text{ and}$$
$$\epsilon^w_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-K+1} \text{ for } p \in [0, 1/2) \text{ if } K \geq 2.$$

for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$.

Notice that both $\Omega_{m,N}$ and $\hat{\Lambda}_{m,N}$ are the same sets used to prove theorem 3.5.1. Therefore, we have the same deterministic breakdown epsilons for $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$ as in theorem 3.5.1: $\epsilon^s_B = 2^{1+1/q} \cdot 10^{-K}$ and $\epsilon^w_B = 2^{1+1/q} \cdot 10^{-K+1}$ if $K \geq 2$, but $\epsilon^w_B = \epsilon^s_B$ if $K = 1$. However, we know that $\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \leq \epsilon^s_B$, $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^s_B$ and $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) \leq \epsilon^w_B$ for all $p \in [0, 1)$ by proposition 2.4.1. Hence

$$\epsilon^s_{\mathbb{P}\mathrm{hB}}(p) = 2^{1+1/q} \cdot 10^{-K} \text{ for } p \in [0, 1/2),$$
$$\epsilon^s_{\mathbb{P}\mathrm{B}}(p) = 2^{1+1/q} \cdot 10^{-K} \text{ for } p \in [0, 1/3), \text{ and}$$
$$\epsilon^w_{\mathbb{P}\mathrm{B}}(p) = 2^{1+1/q} \cdot 10^{-K+1} \text{ for } p \in [0, 1/2) \text{ if } K \geq 2.$$

If $K = 1$, we have $\epsilon^w_B \geq \epsilon^w_{\mathbb{P}\mathrm{B}}(p) \geq \epsilon^s_{\mathbb{P}\mathrm{B}}(p) = 2^{1+1/q} \cdot 10^{-K}$ for $p \in [0, 1/2)$ by proposition 2.4.1. But $\epsilon^w_B = 2^{1+1/q} \cdot 10^{-K}$ when $K = 1$, so this implies $\epsilon^w_{\mathbb{P}\mathrm{B}}(p) = 2^{1+1/q} \cdot 10^{-K}$ for $p \in [0, 1/2)$. This establishes (i) and (ii) of the theorem.

To prove part (iii), consider the computational problem $\{\Xi, \Omega_{m,N}, \mathcal{M}_N, \Lambda_{m,N}\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}_{m,N}, \mathcal{M}_N, \tilde{\Lambda}_{m,N}\}$. As before, we fix the notation of the algorithms for a single $\tilde{\iota} \in \tilde{\Omega}_{m,N}$, and write $\tilde{\iota} = (\{y^{(n)}_j\}^\infty_{n=0}, \{A^{(n)}_{j,k}\}^\infty_{n=0})_{j,k}$ which corresponds to an $\iota = (y, A) \in \Omega_{m,N}$. We need a new subroutine which, when called upon, will guess the solution for strong inputs. As in [BHV], we shall call this subroutine *Guess*.

Guess
Inputs: The dimensions m, N, and an error threshold $\epsilon_a \in \mathbb{D}$
Oracles: $\mathcal{O}_{vec}$ and $\mathcal{O}_{mat}$ providing access to the components $y^{(n)}_j$ and $A^{(n)}_{j,k}$ of an input $\tilde{\iota}$.
Output: $x \in \mathbb{D}^N$ which potentially satisfies $\mathrm{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$.
Procedure:

1. Make a random coin flip which returns *True* or *False*, each with probability 1/2.
2. If it outputs *True*, output an $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K} e_1\|_q \leq \epsilon_a$, and terminate.
3. If it outputs *False*, output an $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$, and terminate.

We also need a subroutine that acts as a biased coin flip, which returns *True* with probability $1/n$ for a given $n \in \mathbb{N}$. Note that such a subroutine can be constructed in any model that has access to a normal coin flip that returns *True* or *False* each with probability 1/2. In particular, a probabilistic Turing machine can run such a subroutine. Call this subroutine *BiasedCoinFlip*. Note that both *Guess* and *BiasedCoinFlip* halt with probability 1. Using these subroutines, as well as the subroutine *Weak* from the proof of theorem 3.5.1, we can construct a randomized ϵ-approximation algorithm that provides arbitrarily accurate solutions with probability greater than or equal to 2/3, for all inputs in $\tilde{\iota} \in \tilde{\Omega}$. Note that if this algorithm is to be executed by a probabilistic oracle Turing machine, the regression parameter λ must be rational.

Randomized ϵ-approximation algorithm
Inputs: The dimensions m, N, and an error threshold $\epsilon_a \in \mathbb{D}$
Oracles: $\mathcal{O}_{vec}$ and $\mathcal{O}_{mat}$ providing access to the components $y_j^{(n)}$ and $A_{j,k}^{(n)}$ of an input $\tilde{\iota}$.
Output: With probability at least 2/3, the algorithm outputs an $x \in \mathbb{D}^N$ which satisfies $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$.
Procedure:

1. Execute $\mathit{IdentifyStrongOrWeak}^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N)$.
2. If it outputs 'InputWeak', execute $\mathit{Weak}^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, \epsilon_a)$ and terminate.
3. If it outputs 'InputStrong', initiate the following loop.
   For $n = 1, 2, \ldots$:
   Execute *BiasedCoinFlip*$(2^{n-1} + 2)$. If it returns *True*, go to step 4. Otherwise, use $\mathcal{O}_{mat}$ to read $A_{3,1}^{(n)}$ and $A_{3,2}^{(n)}$. Set $d = A_{3,1}^{(n)} - A_{3,2}^{(n)}$. The next action depends on d:
   3a. If $d > 2^{-n+1}$, output an $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K} e_1\|_q \leq \epsilon_a$, and terminate.
   3b. If it $d < -2^{-n+1}$, output an $x \in \mathbb{D}^N$ with $\|x - 4 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$, and terminate.
   If neither 3a or 3b were executed, increment n and continue the loop.
4. Execute $\mathit{Guess}^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, \epsilon_a)$ and terminate.

For convenience we say that the algorithm is correct for an $\tilde{\iota} \in \tilde{\Omega}_{m,N}$ if it outputs an $x \in \mathbb{D}^N$ such that $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$. Otherwise, we say that the algorithm fails. Note that failure includes the possibility of the algorithm not halting. Indeed, it will be made clear that there is a non-zero probability that the algorithm does not halt at all. It remains to show that for all $\tilde{\iota} \in \tilde{\Omega}_{m,N}$, the *Randomized ϵ-approximation algorithm* is correct with probability at least

2/3. This can be proved by looking at the four possible cases for the input $\tilde{\iota}$:

Case 1: $\tilde{\iota}$ corresponds to a "weak" input $\iota \in \Omega^w_{m,N,K}$. Since *IdentifyStrongOrWeak* will correctly output 'InputWeak', the algorithm executes $\text{Weak}^{\mathcal{O}_{vec},\mathcal{O}_{mat}}(m, N, \epsilon_a)$ and terminates. Furthermore, *Weak* will always output an $x \in \mathbb{D}^N$ satisfying $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$, so the algorithm is correct for all $\tilde{\iota}$ corresponding to some $\iota \in \Omega^w_{m,N,K}$ with probability 1.

Case 2: $\tilde{\iota}$ corresponds to a "strong" input $\iota \in \Omega^s_{m,N,K}$ such that $\iota = (y^K(m), A(0, \beta, m, N))$ and $\beta \in (0, 1/2]$. Then $\tilde{\Xi}(\tilde{\iota}) = \Xi(\iota) = 4 \cdot 10^{-K} e_1$ by lemma 3.3.1. Since *IdentifyStrongOrWeak* will correctly output 'InputStrong', the algorithm skips step 2 and executes step 3 instead. The outcome of the algorithm from here on out depends on random events, so we want to calculate the probability that the algorithm outputs an $x \in \mathbb{D}^N$ satisfying $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$. Notice first however, that $d = A^{(n)}_{3,1} - A^{(n)}_{3,2} \leq 2^{-n} - (2\beta - 2^{-n}) < 2^{-n+1}$ for all $n \in \mathbb{N}$. Thus the algorithm will never terminate at step 3a. On the other hand, $d \leq 2^{-n+1} - 2\beta$ will be smaller than -2^{-n+1} for sufficiently large n. Then $n_0 = \inf\{n \in \mathbb{N} : d < -2^{-n+1}\}$ is finite. This means that the algorithm may terminate at step 3b, but this depends on the outcome of the execution of *BiasedCoinFlip* in each iteration of the loop in step 3. Let F_n be the event that the algorithm exits the loop by going to step 4 at the n-th iteration. Then $\mathbb{P}(F_n) = 0$ for $n > n_0$ because the algorithm will have terminated either by some earlier event F_j, $j \leq n_0$, or at step 3b on iteration n_0. For $n \leq n_0$, $\mathbb{P}(F_n)$ is equal to the probability that the algorithm has executed $n-1$ iterations of the loop without terminating, and then *BiasedCoinFlip*$(2^{n-1}+2)$ returns *True* at the n-th iteration. As the outcome of *BiasedCoinFlip* is independent of all previous outcomes, these two events are independent of each other. Furthermore, all the events F_j for $j = 1, \cdots, n-1$ prior to F_n are disjoint. Thus we have that

$$\begin{aligned}\mathbb{P}(F_n) &= \mathbb{P}\Big(\Big(\bigcup_{j=1}^{n-1} F_j\Big)^c\Big) \cdot \mathbb{P}(\textit{BiasedCoinFlip}(2^{n-1}+2) \text{ returns } \textit{True}) \\ &= \Big(1 - \mathbb{P}\big(\bigcup_{j=1}^{n-1} F_j\big)\Big)\frac{1}{(2^{n-1}+2)} = \Big(1 - \sum_{j=1}^{n-1}\mathbb{P}(F_j)\Big)\frac{1}{(2^{n-1}+2)}\end{aligned}$$

for $n \leq n_0$. By strong induction, we get $\mathbb{P}(F_n) = 3^{-1} \cdot 2^{-n+1}$ for $n \leq n_0$. It is clear that since n_0 is finite, the algorithm terminates with probability 1. However, the output $x \in \mathbb{D}^N$ may be incorrect. The only possible way for this to happen is if the subroutine *Guess* is executed, and it outputs $x \in \mathbb{D}^N$ such that $\|x - 4 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$ instead of $\|x - 4 \cdot 10^{-K} e_1\|_q \leq \epsilon_a$. Since the probability of *Guess* outputting $x \in \mathbb{D}^N$ such that $\|x - 4 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$ is $1/2$, the probability that the *Randomized ϵ-approximation algorithm* produces an incorrect output is

$$\bigcup_{n=1}^{\infty} \mathbb{P}(F_n)\frac{1}{2} = \sum_{n=1}^{\infty}\mathbb{P}(F_n)/2 \leq \sum_{n=1}^{\infty} 3^{-1} \cdot 2^{-n} = 1/3.$$

Consequently, the probability that the output is correct is

$$1 - \bigcup_{n=1}^{\infty} \mathbb{P}(F_n)\frac{1}{2} \geq 1 - 1/3 = 2/3.$$

Case 3: $\tilde{\iota}$ corresponds to a "strong" input $\iota \in \Omega^s_{m,N,K}$ such that $\iota = (y^K(m), A(\alpha, 0, m, N))$ and $\alpha \in (0, 1/2]$. Then $\tilde{\Xi}(\tilde{\iota}) = \Xi(\iota) = 4 \cdot 10^{-K} e_2$ by lemma 3.3.1. Since *IdentifyStrongOrWeak* will correctly output 'InputStrong', the algorithm skips step 2 and executes step 3 instead. The argument that the *Randomized ϵ-approximation algorithm* is correct with probability at least 2/3 is the same as in case 2, with only minor differences. Since $\alpha > \beta = 0$ we have that the algorithm will never terminate at step 3b, however it may terminate at step 3a after a sufficiently high number of iterations. As before, the algorithm halts with probability 1, and will only produce an incorrect output if *Guess* is executed and outputs an $x \in \mathbb{D}^N$ such that $\|x - 4 \cdot 10^{-K} e_1\|_q \leq \epsilon_a$ instead of $\|x - 4 \cdot 10^{-K} e_2\|_q \leq \epsilon_a$. This occurs with a probability $\sum_{n=1}^{\infty} \mathbb{P}(F_n)\frac{1}{2} \leq \sum_{n=1}^{\infty} 3^{-1} \cdot 2^{-n} = 1/3$, thus the probability that the output is correct is $1 - \sum_{n=1}^{\infty} \mathbb{P}(F_n)\frac{1}{2} \geq 1 - 1/3 = 2/3$ as before.

Case 4: $\tilde{\iota}$ corresponds to a "strong" input $\iota \in \Omega^s_{m,N,K}$ such that $\iota = (y^K(m), A(0, 0, m, N))$. Then $\tilde{\Xi}(\tilde{\iota}) = \Xi(\iota) \supset \{4 \cdot 10^{-K} e_1, 4 \cdot 10^{-K} e_2\}$ by lemma 3.3.1. As before *IdentifyStrongOrWeak* will correctly output 'InputStrong', so the algorithm skips step 2 and executes step 3 instead. However, since $\alpha = \beta = 0$, then $d = A^{(n)}_{3,1} - A^{(n)}_{3,2} = 0$ for all $n \in \mathbb{N}$. This means that the loop in step 3 will never terminate at step 3a or 3b. The only way for the algorithm to terminate is if *Guess* is executed, that is, if *BiasedCoinFlip*$(2^{n-1} + 2)$ returns true at the n-th iteration of the loop for some $n \in \mathbb{N}$. This corresponds to the event F_n; that the algorithm exits the loop by going to step 4 at the n-th iteration. As before,

$$\mathbb{P}(F_n) = \big(1 - \mathbb{P}(\bigcup_{j=1}^{n-1} F_j)\big)(2^{n-1} + 2)^{-1} = \mathbb{P}(F_n) = 3^{-1} \cdot 2^{-n+1}$$

by strong induction, only in this case it holds for all $n \in \mathbb{N}$. The probability that the algorithm terminates is thus

$$\mathbb{P}(\bigcup_{j=1}^{\infty} F_j) = \sum_{j=1}^{\infty} \mathbb{P}(F_j) = \sum_{j=1}^{\infty} 3^{-1} \cdot 2^{-j+1} = 2/3 \cdot \sum_{j=1}^{\infty} 2^{-j} = 2/3.$$

Furthermore, no matter what *Guess* outputs, it always satisfies

$$\mathrm{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) = \mathrm{dist}_{\mathcal{M}}(x, \Xi(\iota)) = \inf_{\xi \in \Xi(\iota)} \|x - \xi\|_q \leq \epsilon_a$$

because $\{4 \cdot 10^{-K} e_1, 4 \cdot 10^{-K} e_2\} \subset \Xi(\iota)$. Consequently, the algorithm is correct with probability 2/3.

These four cases cover all possible forms of the input $\tilde{\iota}$. Furthermore, the algorithm was correct with probability at least 2/3 in all cases. Therefore, the *Randomized ϵ-approximation algorithm* provides a solution $x \in \mathbb{D}^N$ such

that $\text{dist}_{\mathcal{M}}(x, \tilde{\Xi}(\tilde{\iota})) \leq \epsilon_a$ with probability at least $2/3$, for all $\tilde{\iota} \in \Omega_{m,N}$. This concludes the proof of theorem 4.1.3 (iii).

■

4.3 Phase transitions for randomized algorithms

Theorem 4.1.3 leads to phase transitions similar to what was shown in figure 3.4. As in chapter 3, we define a new complexity class $\text{BPP}_{\epsilon}^{\mathcal{O}}$, inspired by the complexity class BPP from classical complexity theory. BPP (Bounded-error Probabilistic Polynomial-time) is the class of problems that can be solved by a probabilistic Turing machine in polynomial time, with an error probability no larger than $1/3$ [Sip13]. Let $\text{BPP}_{\epsilon}^{\mathcal{O}}$ denote the class of computational problems with Δ_1-information, for which ϵ-approximations can be computed by a probabilistic oracle Turing machine with access to an oracle for each input $\tilde{\iota} \in \tilde{\Omega}$, with runtime polynomial in the number of variables n_{var} in the input, and with error probability no larger than $1/3$ for all inputs.

Let us consider the computational problem $\{\Xi, \Omega\}^{\Delta_1}$, where Ξ is the lasso solution map given in (3.3), and Ω is the input set defined in (3.13) for which theorem 4.1.3 holds. Note that the *Randomized ϵ-approximation algorithm* does not make the problem of computing ϵ-approximations for $\{\Xi, \Omega\}^{\Delta_1}$ in $\text{BPP}_{\epsilon}^{\mathcal{O}}$ for all $\epsilon \geq 0$. While the algorithm has error probability no larger than $1/3$ for all inputs, it does not run in polynomial time. In fact, it may not even halt on some inputs.

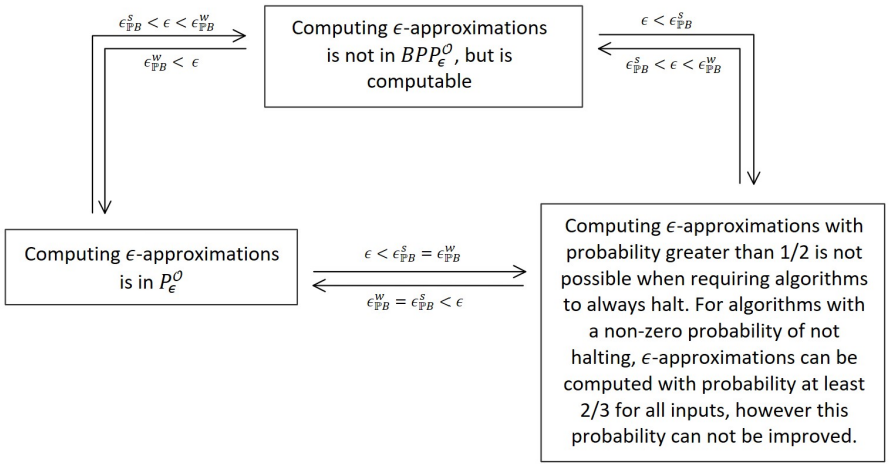


Figure 4.1: Phase transitions in the randomized case for lasso with domain Ω given by theorem 4.1.3. Here, we have fixed $\epsilon^w_{\mathbb{P}\text{B}} = \epsilon^w_{\mathbb{P}\text{B}}(p_1)$ for $p_1 < 1/2$ and $\epsilon^s_{\mathbb{P}\text{B}} = \epsilon^s_{\mathbb{P}\text{B}}(p_2)$ for $p_2 < 1/3$, for ease of reading.

.

Consider instead an error $\epsilon \geq 0$ satisfying $\epsilon^w_{\mathbb{P}\text{B}}(p_1) > \epsilon > \epsilon^s_{\mathbb{P}\text{B}}(p_2)$ for $p_1 \in [0, 1/2)$ and $p_2 \in [0, 1/3)$. We have that for all randomized general algorithms $\Gamma^{\text{ran}} \in \text{RGA}$ and all $M \in \mathbb{N}$, there exists an $\tilde{\iota} \in \tilde{\Omega}$ such that $\mathbb{P}_{\tilde{\iota}}\big(\text{dist}_{\mathcal{M}}(\Gamma^{\text{ran}}_{\tilde{\iota}}, \tilde{\Xi}(\tilde{\iota})) > \epsilon \text{ or } T_{\Gamma^{\text{ran}}}(\tilde{\iota}) > M\big) > p_1$. This means that if a probabilistic oracle Turing machine Γ^{ran} has polynomial runtime for all inputs,

then $\mathbb{P}_{\tilde{\iota}}\big(\operatorname{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}_{\tilde{\iota}}, \tilde{\Xi}(\tilde{\iota})) > \epsilon \text{ or } T_{\Gamma^{\mathrm{ran}}}(\tilde{\iota}) > M\big) = \mathbb{P}_{\tilde{\iota}}\big(\operatorname{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}_{\tilde{\iota}}, \tilde{\Xi}(\tilde{\iota})) > \epsilon\big) > p_1$ for at least one input $\tilde{\iota} \in \tilde{\Omega}$. This shows that the problem is still not in $\mathrm{BPP}^{\mathcal{O}}_{\epsilon}$, because p_1 can be chosen larger than $1/3$.

However, if we consider an error $\epsilon \geq 0$ satisfying $\epsilon > \epsilon^w_{\mathbb{P}\mathrm{B}}(p_1)$ for $p_1 \in [0, 1/2)$, then $\epsilon > \epsilon^w_B$ by (4.8). This means that the problem of computing ϵ-approximations is in $\mathrm{P}^{\mathcal{O}}_{\epsilon}$, as already established in section 3.7. The situation is summarized in figure 4.1.

Remark 4.3.1 ($\mathrm{BPP}^{\mathcal{O}}_{\epsilon}$ vs. $\mathrm{EXPTIME}^{\mathcal{O}}_{\epsilon}$). By their original definitions in classical complexity theory, BPP $\subseteq$ EXPTIME. Thus it would seem that the result from section 3.7 (that the problem of computing ϵ-approximations is not in $\mathrm{EXPTIME}^{\mathcal{O}}_{\epsilon}$ for $\epsilon^s_B < \epsilon < \epsilon^s_B$) is stronger than the result shown in the upper box of figure 4.1, considering (4.5) and (4.8). If it was the case that $\mathrm{BPP}^{\mathcal{O}}_{\epsilon} \subseteq \mathrm{EXPTIME}^{\mathcal{O}}_{\epsilon}$, then this would indeed be true. However, the exact relationship between $\mathrm{BPP}^{\mathcal{O}}_{\epsilon}$ and $\mathrm{EXPTIME}^{\mathcal{O}}_{\epsilon}$ has yet to be determined.

4.4 The breakdown epsilons can be arbitrarily large

We conclude this thesis with a short section remarking upon the size of the breakdown epsilons for the problem of computing approximate solutions to the lasso problem. Recall the definitions of $A(\alpha, \beta, m, N)$ and $y^A(y_c, m)$ from (3.5) and (3.6). For a given $\lambda \geq 0$ and natural numbers $K \geq 1$, and $m > N \geq 2$, let

$$\Omega^s_{m,N,K} = \{(y^K(m), A(\alpha, \beta, m, N)) : (\alpha, \beta) \in \mathcal{L}\} \tag{4.10}$$

where $\mathcal{L} = \{[0, 1/4] \times \{0\} \cup \{0\} \times [0, 1/4]\}$ and $y^K(m) := y^A(4 \cdot 10^K + \lambda\sqrt{m}, m)$. By lemma 3.3.1, we have that $\Xi(\iota) \cap \{4 \cdot 10^K e_1, 4 \cdot 10^K e_2\} \neq \emptyset$ for all $\iota \in \Omega^s_{m,N,K}$. Then we have the following result:

Lemma 4.4.1. *Let $k, m, N \in \mathbb{N}$ with $m > N \geq 2$ and $k \geq 1$. Consider the computational problem $\{\Xi, \Omega^s_{m,N,k}, \mathcal{M}_N, \Lambda_{m,N}\}$, where Ξ is the solution map to the lasso problem, the input set $\Omega^s_{m,N,k}$ is as defined in (4.10), and the metric on $\mathcal{M}_N$ is induced by $\|\cdot\|_q$ for some $q \in [1, \infty]$. Then there exists a $\hat{\Lambda}_{m,N} \in \mathcal{L}^1(\Lambda_{m,N})$ such that, for the computational problem $\{\Xi, \Omega^s_{m,N,k}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, we have $\epsilon^s_B \geq \epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/2)$ and $\epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/3)$.*

Proof. The proof is identical to the proof of lemma 3.3.2, except all exponents are changed from $-k$ to k. For the given $k \geq 1$ and $m > N \geq 2$, let $\iota^0 = (y^k(m), A(0, 0, m, N))$ and

$$\iota^1_n = (y^k(m), A(0, 2^{-1} \cdot 4^{-n}, m, N))$$
$$\iota^2_n = (y^k(m), A(2^{-1} \cdot 4^{-n}, 0, m, N)).$$

For ease of reading, let $A^0 = A(0, 0, m, N)$, $A^{1,n} = A(0, 2^{-1} \cdot 4^{-n}, m, N)$ and $A^{2,n} = A(2^{-1} \cdot 4^{-n}, 0, m, N)$. Let $S^1 = \{4 \cdot 10^k e_1\}$ and $S^2 = \{4 \cdot 10^k e_2\}$. We have that

$$\begin{aligned}\inf_{x_1 \in S^1,\, x_2 \in S^2} \|x_1 - x_2\|_q &= \|4 \cdot 10^k e_1 - 4 \cdot 10^k e_2\|_q \\ &= (2(4 \cdot 10^k)q)^{1/q} \\ &= 2^{2+1/q} \cdot 10^k\end{aligned}$$

and by lemma 3.3.1 we have

$$\Xi(\iota_n^1) = \Xi((y^k(m), A(0, 2^{-1} \cdot 4^{-n}, m, N)) = 4 \cdot 10^k e_1 \in S^1$$
$$\Xi(\iota_n^2) = \Xi((y^k(m), A(2^{-1} \cdot 4^{-n}, 0, m, N)) = 4 \cdot 10^k e_2 \in S^2$$

for all $n \in \mathcal{N}$. Next, we have that for any $f \in \Lambda_{m,N}$

$$\begin{aligned} |f(\iota_n^1) - f(\iota^0)| &= |f((y^k(m), A^{1,n})) - f((y^k(m), A^0))| \\ &\leq \max\{\|A^{1,n} - A^0\|_{\max}, \|y^k(m) - y^k(m)\|_\infty\} \\ &= \|A^{1,n} - A^0\|_{\max} = 4^{-n} \end{aligned}$$

and by the same steps, we get $|f(\iota_n^2) - f(\iota^0)| \leq 4^{-n}$. Then conditions (a) - (c) of proposition 2.4.5 are satisfied. By result (ii) of the proposition, we get $\epsilon^s_{\mathbb{P}\text{hB}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/2)$ and $\epsilon^s_{\mathbb{P}\text{B}}(p) \geq 2^{1+1/q} \cdot 10^{-k}$ for $p \in [0, 1/3)$. ■

This means that we can construct input classes for which the strong breakdown epsilons are arbitrarily large. By choosing any large $K \geq 1$, we have that for the computational problem $\{\Xi, \Omega^s_{m,N,K}, \mathcal{M}_N, \hat{\Lambda}_{m,N}\}$, where $\Omega^s_{m,N,K}$ is as defined in (4.10), the various strong breakdown epsilons satisfy

$$\epsilon^s_B > 10^k,$$
$$\epsilon^s_{\mathbb{P}\text{B}}(p) > 10^k \text{ for } p \in [0, 1/3), \text{ and}$$
$$\epsilon^s_{\mathbb{P}\text{hB}}(p) > 10^k \text{ for } p \in [0, 1/2).$$

This means that for any deterministic lasso algorithm, there are cases where the algorithm will output an estimate x for which $\text{dist}_{\mathcal{M}}(x, \Xi(\iota)) > 10^K$. Furthermore, for any halting randomized algorithm, there are cases where the probability that the algorithm outputs an x for which $\text{dist}_{\mathcal{M}}(x, \Xi(\iota)) > 10^K$ is greater than any $p < 1/2$. A similar result can be established for the weak breakdown epsilons.

CHAPTER 5

Conclusions

The main results of this thesis are theorem 3.5.1 and theorem 4.1.3. We summarize these two theorems in a more reader friendly manner below.

Summary of theorems 3.5.1 and 4.1.3. *Consider the lasso problem*

$$\underset{x\in\mathbb{R}^N}{\operatorname{argmin}} \quad \frac{1}{2m}\|\tilde{A}x - y\|_2^2 + \lambda\|x\|_1 \tag{5.1}$$

and some ℓ_q-norm to measure error, $q \in [1,\infty]$. For all $K \geq 2$, there exists a class of inputs Ω such that the following holds.

(i) *It is impossible for any algorithm to compute solutions to (5.1) for which the error is less than $2^{1+1/q}\cdot 10^{-K}$ for all inputs in Ω. Even randomized algorithms are unable to do this with probability greater than $1/2$, if they are required to always halt.*

(ii) *If we consider randomized algorithms with a non-zero probability of not halting, then there exists an algorithm that can compute solutions to (5.1) to arbitrary accuracy for all inputs in Ω, with success probability $2/3$. However, there does not exist any such algorithm that can compute solutions to arbitrary accuracy for all inputs in Ω with probability greater than $2/3$.*

(iii) *Any algorithm that computes solutions to (5.1) for which the error is less than $2^{1+1/q}\cdot 10^{-K+1}$ for all inputs in Ω will, in the worst case, need an arbitrarily long runtime. Furthermore, such an algorithm exists. This holds for randomized algorithms as well: There does not exist any randomized algorithm that can compute solutions for which the error is less than $2^{1+1/q}\cdot 10^{-K+1}$ and the runtime is bounded for all inputs in Ω, with probability greater than $1/2$.*

(iv) *There is an algorithm that can compute solutions to (5.1) with error less than or equal to any $\epsilon \geq 0$ satisfying $\epsilon > 2^{1+1/q}\cdot 10^{-K+1}$ for all inputs in Ω. This algorithm has runtime polynomial in $\log(1/\epsilon)$ and the number of variables in the input. In particular, the problem of computing ϵ-approximations to (5.1) is in $\mathrm{P}_\epsilon^{\mathcal{O}}$ for $\epsilon > 2^{1+1/q}\cdot 10^{-K+1}$.*

(v) *To establish the results* (i) - (iii)*, the input class Ω can be chosen with fixed dimensions $m > N \geq 2$ for its entries $(y, A) \in \mathbb{R}^m \times \mathbb{R}^{m\times N}$. Furthermore, if we only consider the results* (i) *and* (ii)*, then we can choose $K = 1$.*

In order to make clear what the consequences of this list of results are, some comments are in order. An immediate consequence of (i) is that the lasso problem is not computable in the traditional Turing sense. However, this does not mean that computing solutions to the lasso problem is completely hopeless. While solutions can not always be computed to arbitrary accuracy, result (iv) shows that if we are willing to sacrifice some accuracy, then approximate solutions may still be computed. In other words, even if lasso is non-computable for a particular input class Ω, solutions may sometimes be computable to the accuracy needed in practice. This can explain why lasso has had great success in practice.

As mentioned in the introduction to this thesis, the phenomenon characterized by theorems 3.5.1 and 4.1.3 is not unique to lasso. Similar results have been established for linear programming, basis pursuit, and constrained lasso in [BHV] (for the case where $N > m$ in the inputs) for the K, $K-1$ and $K-2$ correct digits cases. What has been developed in [BHV], and restated in chapter 2, is the beginnings of a new complexity theory for non-computable problems. Furthermore, related computational barriers have been established for neural networks in [ACH21]. It seems that we have only just begun to scratch the surface of the computational barriers inherent in our idea of what an algorithm is, and therefore the limitations of what a computer can solve.

Future work and areas of inquiry

The computational model developed in [BHV] is new, and there are many directions of inquiry that can be taken from here. Some suggestions for future work are listed below.

Similar impossibility results to theorems 3.5.1 and 4.1.3 can be established for other optimization problems. Some potential candidates are the matrix completion problem, and other forms of lasso such as the square-root lasso or group lasso.

Analysis into how prevalent the phenomenon characterized by theorem 3.5.1 and theorem 4.1.3 is would be useful, especially in the situations typically met in practice. The impossibility result stems from cases where the true solution "flips" between two disjoint subsets $S^1, S^2 \subset \mathcal{M}$ with a non-zero "gap" between them. Are there necessary and sufficient conditions under which lasso is always computable?

One could investigate the relationship between the complexity classes $\mathrm{BPP}_\epsilon^{\mathcal{O}}$ and $\mathrm{EXPTIME}_\epsilon^{\mathcal{O}}$. Other complexity classes can be defined, based on already established complexity classes in traditional complexity theory; like $\mathrm{PP}_\epsilon^{\mathcal{O}}$ for PP[1]. What are the relationships between the different complexity classes, and which problems do they contain?

[1]The complexity class PP consists of the decision problems that can be solved by a probabilistic Turing machine in polynomial time, with an error probability less than 1/2 for all inputs [Pap94].

Appendices

APPENDIX A

MATLAB code

A.1 MATLAB code for the lasso experiment in section 3.1

```
format shortG

% Machine epsilon for double precision floats
machine_eps = 2^(-52);
max_iter = 1/machine_eps;

% LASSO example that will lead to failure
lambda = 0.1;
y = [1/sqrt(2); -1/sqrt(2); 0];
Dx_sol = [0; 1-sqrt(3)*lambda]; % True solution
deltas = 2.^[-1, -7, -15, -20, -24, -26, -28, -30];

Dx_error = [];
warning = [];
runtime = [];

for i = 1:length(deltas)
    % Define the problem matrix A
    A = [1/sqrt(2) - deltas(i), 1/sqrt(2);
        -1/sqrt(2) - deltas(i), -1/sqrt(2);
        2*deltas(i), 0];

    % Use lasso to estimate Dx with default settings
    lastwarn('','); % Reset last warning
    tic;            % Start timer
    [Dx_star1, FitInfo1Dx] = lasso(A,y,'Lambda',lambda);
    runtime(i,1) = toc; % Save lasso runtime
    if (isempty(lastwarn())); warning(i,1) = 0;
    else; warning(i,1) = 1; end

    % Use lasso with 'RelTol' set to machine epsilon
    lastwarn('','); % Reset last warning
    tic;            % Start timer
    [Dx_star2, FitInfo2Dx] = lasso(A,y,'Lambda',lambda,...
```

```
        'RelTol',machine_eps);
    runtime(i,2) = toc; % Save lasso runtime
    if (isempty(lastwarn())); warning(i,2) = 0;
    else; warning(i,2) = 1; end

    % Use lasso with with 'RelTol' set to machine epsilon
    % and 'MaxIter' set to 1/(machine epsilon)
    lastwarn('','') % Reset last waning
    tic;             % Start timer
    [Dx_star3, FitInfo3Dx] = lasso(A,y,'Lambda',lambda,...
        'RelTol',machine_eps,'MaxIter',1e8);
    runtime(i,3) = toc; % Save lasso runtime
    if (isempty(lastwarn())); warning(i,3) = 0;
    else; warning(i,3) = 1; end

    % Calculate error
    Dx_error(i,1) = norm(Dx_star1 - Dx_sol);
    Dx_error(i,2) = norm(Dx_star2 - Dx_sol);
    Dx_error(i,3) = norm(Dx_star3 - Dx_sol);
end
```

Bibliography

[AB09] Arora, S. and Boaz, B. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ACH21] Antun, V., Colbrook, M. J. and Hansen, A. C. *Can stable and accurate neural networks be computed? – On the barriers of deep learning and Smale's 18th problem*. Jan. 2021. arXiv: 2101.08286.

[AH21] Adcock, B. and Hansen, A. C. *Compressive Imaging: Structure, sampling, learning*. In press. Cambridge University Press, 2021.

[Ben+15a] Ben-Artzi, J. et al. *Can everything be computed? – On the Solvability Complexity Index and Towers of Algorithms*. Aug. 2015. arXiv: 1508.03280.

[Ben+15b] Ben-Artzi, J. et al. 'New barriers in complexity theory: On the solvability complexity index and the towers of algorithms'. In: *Comptes Rendus Mathematique* vol. 353, no. 10 (2015), pp. 931–936.

[Ben+20] Ben-Artzi, J. et al. *Computing Spectra – On the Solvability Complexity Index Hierarchy and Towers of Algorithms*. 2020. arXiv: 1508.03280.

[BHV] Bastounis, A., Hansen, A. C. and Vlacic, V. 'The extended Smale's 9th problem - On computational barriers and paradoxes in estimation, regularization, learning and computed-assisted proofs'. Unpublished.

[Blu+98] Blum, L. et al. *Complexity and real computation*. Springer-Verlag New York, Inc., 1998.

[FR13] Foucart, S. and Rauhut, H. *A mathematical introduction to compressive sensing*. Birkhäuser, 2013.

[Han11] Hansen, A. C. 'On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators'. In: *Journal of the American Mathematical Society* vol. 24, no. 01 (2011), pp. 81–124.

[HTJ07] Hastie, T., Tibshirani, R. and J., F. *The Elements of Statistical Learning: Data mining, inference, and prediction*. 2nd ed. Springer, 2007.

[Kar84] Karmarkar, N. 'A New Polynomial-Time Algorithm for Linear Programming'. In: *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. STOC '84. New York, USA: Association for Computing Machinery, 1984, pp. 302–311.

[Kha80] Khachiyan, L. G. 'Polynomial Algorithms in Linear Programming'. Russian. In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* vol. 20, no. 1 (1980), pp. 51–68.

[Ko91] Ko, K.-I. *Complexity theory of real functions*. Birkhäuser Boston Inc., 1991.

[Lin18] Lindstrøm, T. *Spaces: An introduction to real analysis*. American Mathematical Society, 2018.

[Mul05] Muller, J.-M. *Elementary Functions: Algorithms and Implementation*. 2nd ed. Birkhäuser, 2005.

[Pap94] Papadimitriou, C. H. *Computational Complexity*. Addison-Wesley, 1994.

[SG76] Sahni, S. and Gonzalez, T. 'P-Complete Approximation Problems'. In: *Journal of the ACM* vol. 23, no. 3 (July 1976), pp. 555–565.

[Sip13] Sipser, M. *Introduction to the theory of computation*. 3rd ed. Cengage learning, 2013.

[Sma98] Smale, S. 'Mathematical Problems for the Next Century'. In: *The Mathematical Intelligencer* vol. 20, no. 2 (1998), pp. 7–15.

[Tib96] Tibshirani, R. 'Regression Shrinkage and Selection via the Lasso'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* vol. 58, no. 1 (1996), pp. 267–288.