The Dissertation Committee for Siddhesh Prashant Chaubal
certifies that this is the approved version of the following dissertation:

Complexity Measures of Boolean Functions and their Applications

Committee:

Anna Gál, Supervisor

Scott J Aaronson

Patrick K Nicholson

C Greg Plaxton

Complexity Measures of Boolean Functions and their Applications

by

Siddhesh Prashant Chaubal, B.Tech.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2020

Dedicated to my mom and dad.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Anna Gál, for her guidance, encouragement and support throughout my grad school journey. I learned a lot from Anna about research, as well as about good technical writing, presentation skills, and the art of effectively communicating my research to diverse audiences. My graduate school experience was greatly enriched thanks to all our stimulating discussions on challenging problems - it has been a delight working with her.

Next I would like to thank all the other members on my dissertation committee, Scott Aaronson, Patrick Nicholson, and Greg Plaxton, for their valuable suggestions and feedback.

I would like to thank Patrick Nicholson for being a wonderful mentor and collaborator during my visit to Nokia Bell Labs. I am also grateful to my other collaborators at Nokia Bell Labs, Alessandra and Guangyuan, and all my other colleagues there for numerous interesting discussions.

I am grateful to my labmates and friends for all the fun activities as well as valuable support during different phases of my PhD.

I am lucky to have a doting family: I am grateful to my grandparents, aunts, uncles and cousins for motivating me throughout grad school.

Finally, and most importantly, I would like to thank my parents who

have played a pivotal role in this endeavor, and also in my entire educational journey as a whole. I am forever indebted to them for their unwavering support and loving encouragement, without which this journey would have been unthinkable.

Complexity Measures of Boolean Functions and their Applications

Publication No. _____

Siddhesh Prashant Chaubal, Ph.D.
The University of Texas at Austin, 2020

Supervisor: Anna Gál

The central focus of computational complexity theory is to measure the "hardness" of computing different functions. Towards this end, several measures of complexity for Boolean functions have been studied over the past few decades. Examples of important complexity measures include sensitivity, block sensitivity, certificate complexity, decision tree complexity and degree. Studying the relationships between these different measures has been an active area of research.

In the first part of this dissertation, we prove several results towards tightening the relationships between some of these measures - particularly, the sensitivity and block sensitivity. Specifically, we prove better (cubic and sometimes quadratic) upper bounds on block sensitivity in terms of sensitivity for certain classes of transitive functions. We also prove tight lower bounds on the block sensitivity for the classes we consider.

In the other direction, we give various new constructions of families of Boolean functions that exhibit quadratic separation between sensitivity and block sensitivity. Our constructions have several novel aspects. For example, we give the first direct constructions of families of Boolean functions that have both 0-block sensitivity and 1-block sensitivity quadratically larger than sensitivity.

In the next part of this dissertation, we introduce a new complexity measure of Boolean functions we call *diameter*, that captures the relationship between certificate complexity and several other measures of Boolean functions. We argue that estimating diameter may help to get improved bounds on certificate complexity in terms of sensitivity, and other measures. Further, we prove lower bounds on the sensitivity and block sensitivity of transitive functions with constant diameter. We also prove some implications for the log-rank conjecture in communication complexity for XOR functions with bounded diameter.

In the last part of this dissertation, we study applications of the relationship between some of these measures to machine learning. Specifically we contribute towards two problems concerning decision trees as a machine learning model. First, we focus on the problem of modifying a given decision tree classifier to optimize a specific evaluation measure (the *recall* of the classifier) under some constraints. Finally, we present some algorithms for decision trees for regression problems in the context of transfer learning.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Complexity Measures

A complexity measure is a way of characterizing the "hardness" of computing a Boolean function. Some measures directly express the complexity of a function with respect to some computational model. On the other hand, there are measures that are not based on any computational model, but simply express some combinatorial or algebraic property of the function and may be relevant for the complexity in various computational models. We study relationships between different measures with the goal of getting a better understanding of the computational complexity of functions.

1.1.1 Computational models

In developing the theory of computation, several models of computation have been studied such as Turing machines, Boolean circuits, Boolean formulas etc. For any Boolean function, each model of computation has a complexity measure associated with it, which quantifies the size of the "smallest" instance of the model that can compute the given Boolean function.

The main focus of computational complexity theory is to establish tight

upper and lower bounds on the computational complexity of different functions. For most models of computation, our understanding of these complexity measures is very poor. For example, Shannon [88] proved that most Boolean functions require large circuits to compute them:

Theorem 1.1 (Shannon, 1949 [88])**.** *Most Boolean functions on n variables require circuits of size $\Omega(\frac{2^n}{n})$.*

However, the best known circuit lower bound for any explicit Boolean function is still only linear [42].

To get a better understanding of the relative complexity of different Boolean functions, simpler models of computation have been studied. The hope is that understanding the complexity of a Boolean function for one of these measures may lead us towards better understanding their complexity for the harder computational models. Some of these simpler models find applications in other disciplines as well and are therefore interesting in their own right. One such relatively simple model of computation is the decision tree model.

1.1.1.1 Decision trees

We first define the decision tree complexity of a Boolean function.

Definition 1. Decision tree complexity *A decision tree evaluating a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is a rooted binary tree with internal nodes labelled*

by variables and leaves labelled by $\{0,1\}$*. The depth of a decision tree is defined as the length of a longest root to leaf path in the tree.*

For any input $x \in \{0,1\}^n$*, the label at the leaf reached by following the decision tree queries is the evaluation of the decision tree on* x*. The decision tree is said to compute the function* $f\colon \{0,1\}^n \to \{0,1\}$ *if it evaluates to* $f(x)$ *on every input* $x \in \{0,1\}^n$*. The decision tree complexity of a Boolean function* f*, denoted* $D(f)$*, is defined as the smallest possible depth of any decision tree computing* f*.*

Decision trees have a long history in various areas of computer science and statistics. They were first used as a classification model by Fischer in 1936 [82]. Since then they have been used extensively in machine learning as we discuss in section 1.4. In the area of complexity theory, decision trees have been studied in several contexts, for example in the context of Aanderaa-Rosenberg conjecture for graph properties [83, 84, 54, 58]. Decision tree complexity is also sometimes called query complexity in certain lines of work [23, 93, 52, 6, 73]. Decision trees have been studied in the context of communication complexity where proving lower bounds on decision tree complexity imply communication complexity lower bounds due to lifting theorems [46, 45].

Several variants of this model have also been studied such as the *non-deterministic decision tree complexity*, *randomized decision tree complexity* and *quantum decision tree complexity*. Decision tree complexity is also related to several other measures as we discuss in the next section.

1.1.2 Combinatorial and Algebraic Complexity measures

It is also sometimes useful to define complexity measures simply based on the combinatorial or algebraic properties of a Boolean function, without reference to any computational model. Some examples of such complexity measures are sensitivity, block sensitivity, certificate complexity, degree etc.

We use the following notation: For any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, input $x \in \{0,1\}^n$ and index $i \in [n]$, we denote by x^i the input obtained by flipping the i-th bit of x. More generally, for $S \subseteq [n]$, we denote by x^S the input obtained by flipping the bits of x in all coordinates in the subset S.

Definition 2. Sensitivity *The sensitivity $s(f,x)$ of a Boolean function f on input x is the number of coordinates $i \in [n]$ such that $f(x) \neq f(x^i)$. The 0-sensitivity and 1-sensitivity of f are defined as $s_0(f) = \max\{s(f,x) : f(x) = 0\}$ and $s_1(f) = \max\{s(f,x) : f(x) = 1\}$, respectively. The sensitivity of f is defined as $s(f) = \max\{s(f,x) : x \in \{0,1\}^n\} = \max\{s_0(f), s_1(f)\}$.*

Definition 3. Block Sensitivity *The block sensitivity $bs(f,x)$ of a Boolean function f on input x is the maximum number of pairwise disjoint subsets $S_1, \ldots, S_k$ of $[n]$ such that for each $i \in [k]$ $f(x) \neq f(x^{S_i})$. The 0-block sensitivity and 1-block sensitivity of f are defined as $bs_0(f) = \max\{bs(f,x) : f(x) = 0\}$ and $bs_1(f) = \max\{bs(f,x) : f(x) = 1\}$, respectively. The block sensitivity of f is defined as $bs(f) = \max\{bs(f,x) : x \in \{0,1\}^n\} = \max\{bs_0(f), bs_1(f)\}$.*

It is convenient to refer to coordinates $i \in [n]$ such that $f(x) \neq f(x^i)$ as *sensitive bits* for f on x. Similarly, a subset $S \subseteq [n]$ is called a *sensitive*

block for f on x if $f(x) \neq f(x^S)$.

We shall now proceed to define the certificate complexity of a function. We will first need some additional definitions prior to that.

Definition 4. Partial assignment *Given an integer $n > 0$, a partial assignment α is a function $\alpha\colon [n] \to \{0, 1, \star\}$. A partial assignment α corresponds naturally to a setting of n variables $(x_1, x_2, \ldots x_n)$ to $\{0, 1, \star\}$ where x_i is set to $\alpha(i)$. The variables set to $\star$ are called unassigned or free, and we say that the variables set to 0 or 1 are fixed. We say that input $x \in \{0,1\}^n$ agrees with α if $x_i = \alpha(i)$ for all i such that $\alpha(i) \neq \star$. The size of a partial assignment α is defined as the number of fixed variables of α.*

Definition 5. Certificate *For a function $f\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$, a partial assignment α is a certificate of f on x if x agrees with α and any input y agreeing with α satisfies $f(y) = f(x)$. A certificate α is a 1-certificate (resp. 0-certificate) if $f(x) = 1$ (resp. $f(x) = 0$), on inputs x that agree with α.*

The size of a certificate α, denoted by $size(\alpha)$, is defined as the size of the partial assignment α. The weight of a certificate α, denoted by $wt(\alpha)$, is the number of bits fixed to 1 by α.

Definition 6. Certificate Complexity *The certificate complexity $C(f, x)$ of a Boolean function f on input x is the size of the smallest certificate of f on x. The 0-certificate complexity and 1-certificate complexity of f are defined as $C_0(f) = \max\{C(f, x) : f(x) = 0\}$ and $C_1(f) = \max\{C(f, x) : f(x) = 1\}$,*

respectively. The certificate complexity of f is defined as $C(f) = \max\{C(f,x) : x \in \{0,1\}^n\} = \max\{C_0(f), C_1(f)\}$.

It is also useful to consider the following definition of the smallest certificate size over all inputs. Note that this can be rephrased as the co-dimension of the largest subcube of the Boolean cube $\{0,1\}^n$ where f is constant.

Definition 7. Minimum Certificate Size *The minimum certificate size of a Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$ *is defined as* $C_{min}(f) = \min\{C(f,x) : x \in \{0,1\}^n\}$.

Note that the certificate complexity of a function can also be interpreted as its non-deterministic decision tree complexity.

We now proceed to define two important algebraic complexity measures: the real degree and the $\mathbb{F}_2$ degree of a Boolean function.

Definition 8. Real degree of a function

A polynomial p over the reals with n variables is said to represent a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ *if* $p(x) = f(x)$ *for all* $x \in \{0,1\}^n$. *The degree of the unique multilinear polynomial over the reals representing f is defined to be the* degree *of f, and is denoted as $deg(f)$.*

Definition 9. $\mathbb{F}_2$ degree of a function

For any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, *the degree of the unique multilinear polynomial over $\mathbb{F}_2$ representing f is called as the $\mathbb{F}_2$ degree of f, denoted $deg_2(f)$.*

All these measures are related to several computational models: here we mention the relationship with decision trees. The following relationships between these measures for any Boolean function f are not hard to see:

$$s(f) \leq bs(f) \leq C(f) \leq D(f)$$

In the other direction, the following bounds are known:

Theorem 1.2 ([23, 93, 2])**.** *For any function* $f\colon \{0,1\}^n \to \{0,1\}$*,*

$$D(f) \leq C_0(f)C_1(f)$$

Nisan also proved the following result relating $bs(f)$ and $C(f)$:

Theorem 1.3. *[74] For any Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$*,*

$$C(f) \leq bs(f)^2$$

This implies that $D(f)$ is also polynomially related to $bs(f)$ for any Boolean function f.

It is also easy to see that $deg(f) \leq D(f)$ for any Boolean function, and this is tight for the AND function on n bits since $deg(AND) = D(AND) = n$. Along with Theorem 1.2, this implies that $deg(f) \leq C_0(f)C_1(f) \leq C(f)^2$. In the other direction, it is known that $D(f) \leq 2deg(f)^3$ due to [68]. This bound is not known to be tight, and the best separating example is a function f such that $D(f) = deg(f)^{\log_3 6}$ (due to Kushilevitz, as mentioned in [77]).

Nisan and Szegedy [76] proved that for any Boolean function f, $bs(f) \leq 2deg(f)^2$. This bound was later improved by Tal [92] to $bs(f) \leq deg(f)^2$. This

too is not known to be tight and the best separating example is the same as the one for $D(f)$ versus $deg(f)$, which gives $bs(f) = deg(f)^{\log_3 6}$ ([77]). The relationship of $deg(f)$ with $s(f)$ plays an important role in answering some long-standing questions as we shall see in the next section.

For more details about the relationships between these different measures, see [25, 49] for surveys.

In particular, we explore the relationship between sensitivity of a function and its block sensitivity in the next section.

1.2 The sensitivity versus block sensitivity problem

1.2.1 History

Sensitivity was introduced by Cook, Dwork and Reischuk [35] as a measure to prove lower bounds on the parallel complexity of Boolean functions in the CREW PRAM model. Nisan [75] defined the more general block sensitivity measure, and showed that the CREW PRAM complexity of any Boolean function f is characterized by its block sensitivity up to constant factors as $\Theta(\log bs(f))$.

We mentioned in section 1.1.2 that the sensitivity of any Boolean function is upper bounded by its block sensitivity. In the other direction, Nisan and Szegedy [76] conjectured that the block sensitivity of a Boolean function is upper bounded by a polynomial in its sensitivity. This conjecture, famously called as the *Sensitivity Conjecture* remained open for about 25 years, until the recent breakthrough of Huang [51] who proved the following theorem:

Theorem 1.4. *[Huang, [51]] For every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$*,*

$$deg(f) \leq s(f)^2$$

This theorem is tight for the AND-of-ORs function as pointed out by [51, 49].

Theorem 1.4 along with the bound $bs(f) \leq deg(f)^2$ mentioned in the previous section implies the following corollary:

Corollary 1.1. *[51] For every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$*,*

$$bs(f) \leq s(f)^4$$

An improvement to Theorem 1.4 was given by Laplante et al. [60] using alternative techniques. They prove the following theorem:

Theorem 1.5. *[60] For every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$*,*

$$deg(f) \leq s_0(f)s_1(f)$$

Aaronson et al. [1] further strengthened Theorem 1.4 and showed that this implies optimal bounds on the deterministic query complexity in terms of quantum query complexity.

Two natural questions remain open about the relationship between sensitivity and block sensitivity:

Question 1. ***[Upper bound]*** *What is the best upper bound on block sensitivity of a Boolean function in terms of its sensitivity?*

Question 2. ***[Separation]*** *What is the best possible separation between sensitivity and block sensitivity of any Boolean function?*

We now briefly discuss the previous work related to these questions.

1.2.1.1 Upper bounds on block sensitivity in terms of sensitivity

Until recently, the best upper bounds on any of these measures were exponential in terms of sensitivity. For example, the previous best upper bound on block sensitivity in terms of sensitivity by Ambainis et al. [7, 9] gave $bs(f) \leq s(f)2^{s(f)-1}$. A constant factor improvement to this bound was given by He et al. [50], who proved the bound $bs(f) \leq (\frac{8}{9}+o(1))s(f)2^{s(f)-1}$. Theorem 1.4 due to Huang [51] improved these bounds drastically to $bs(f) \leq s(f)^4$ for any Boolean function f.

Despite a lot of attention to the problem, until Huang's result, the conjecture was verified only for a few special classes of Boolean functions. We give a brief overview of these classes. It is known that $s(f) = bs(f)$ for *monotone* Boolean functions [75], as well as for *unate* functions [72], and $bs(f) = O(s(f))$ for functions with constant *alternating number* [61]. The conjecture was previously verified for functions that can be represented by *read-k DNF* for $k \leq n^{\frac{1}{3}-\epsilon}$ for constant $\epsilon > 0$ [17]. A read-k DNF is a DNF formula where each variable appears in at most k terms. The paper [17] also verifies the conjecture for read-k formulas that are not necessarily in DNF form, but under some other restrictions and only for smaller values of k. Karthik

and Tavenas [55] proved the conjecture for functions represented by DNFs of a very special structure.

In the next section we discuss a special class of functions called *transitive functions*, which has received a lot of attention in this context.

1.2.1.2 Sensitivity and block sensitivity of transitive functions

Definition 10. Transitive functions *A Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$ *is invariant under a permutation* $\sigma\colon [n] \to [n]$*, if for any* $x \in \{0,1\}^n$*, it holds that* $f(x_1,\ldots,x_n) = f(x_{\sigma(1)},\ldots,x_{\sigma(n)})$*. The set of all permutations under which f is invariant forms a group, called the invariance group of f. A Boolean function is transitive if its invariance group* Γ *is transitive, that is, for each* $i,j \in [n]$*, there is a* $\sigma \in \Gamma$ *such that* $\sigma(i) = j$*.*

For example, the set of all permutations on n bits, denoted by S_n is a transitive group of permutations. Another example of a transitive group of permutations is the set of all *cyclic shifts* on n bits, denoted by $Shift_n = \{\xi_0, \xi_1, \ldots, \xi_{n-1}\}$, where the permutation ξ_j cyclically shifts the string by j positions.

An intriguing aspect of transitive functions is that so far no examples of transitive functions are known on n input bits with $o(n^{1/3})$ sensitivity. Chakraborty [28] constructed a transitive function on n variables with sensitivity $\Theta(n^{1/3})$ (and block sensitivity $\Theta(n^{2/3})$).

The following questions regarding the sensitivity of transitive functions

have been raised:

Question 3. *[28] If $f\colon \{0,1\}^n \to \{0,1\}$ is a transitive function, is then $s(f) \geq \Omega(n^\beta)$ for some constant $\beta > 0$?*

Question 4. *[49] If $f\colon \{0,1\}^n \to \{0,1\}$ is transitive and $f(0^n) \neq f(1^n)$, is then $s(f) = \Omega(\sqrt{n})$?*

These questions are partially answered by Huang's result as follows.

A remark in the survey [49] in combination with Huang's result [51] implies that any transitive function f on n variables where n is a prime power and $f(0^n) \neq f(1^n)$ has sensitivity $s(f) \geq \Omega(\sqrt{n})$. However, this does not seem to directly imply a similar consequence for transitive functions with $f(0^n) \neq f(1^n)$ when n is not a prime power, because for a transitive function, a subfunction obtained by fixing a subset of its bits is no longer necessarily transitive.

It is implicit in a paper by Sun [90] that for a transitive function f on n variables, $bs(f) \cdot s(f)^2 \geq n$. Together with Huang's result this gives

Corollary 1.2. *For any transitive function f on n variables $s(f) \geq \Omega(n^{1/6})$.*

We now define a subclass of transitive functions called *minterm-transitive functions*, which has received much attention in this respect. We shall first need to define minterms and maxterms:

Definition 11. Minterms and Maxterms

A certificate α is called minimal, if after changing any of its fixed variables to

a free variable, the resulting partial assignment α' is not a certificate, that is the function is not constant on inputs agreeing with α'. A minimal 1*-certificate is called a minterm, and a minimal* 0*-certificate is called a maxterm.*

For a partial assignment $\alpha : [n] \to \{0, 1, *\}$ and a permutation σ on n bits we denote by $\sigma(\alpha)$ the partial assignment obtained by applying the partial assignment α to the bits permuted according to σ, that is $\sigma(\alpha)(\ell) = \alpha(\sigma^{-1}(\ell))$ for $\ell \in [n]$.

Definition 12. Minterm-Transitive Function

Let Γ be a transitive group of permutations. A function $f\colon \{0,1\}^n \to \{0,1\}$ is minterm-transitive under Γ *if there exists a minterm α of f such that $f(x) = 1$ if and only if x agrees with $\sigma(\alpha)$ for some $\sigma \in \Gamma$. A function is called minterm-transitive if it is minterm-transitive under some transitive group of permutations.*

Previously, Chakraborty [28] proved that every *minterm-transitive* function f on n variables has sensitivity at least $\Omega(n^{1/3})$. It remains open if the statement of Question 3 holds with $\beta = 1/3$. Since $bs(f) \leq n$ for any function f on n variables, this would imply that $bs(f) \leq O(s(f)^3)$ for any transitive function f on n variables.

While it is still open if every transitive function has sensitivity $\Omega(n^{1/3})$, Sun [90] proved that every transitive function has *block sensitivity* at least $n^{1/3}$. This resulted in further studies of what is the smallest possible block sensitivity of transitive functions. Drucker [40] showed that *minterm-transitive* functions

have block sensitivity at least $\Omega(n^{3/7})$. This bound is tight for the class of minterm-transitive functions: Amano [4] constructed minterm-transitive functions with block sensitivity $O(n^{3/7})$, improving constructions of Sun [90] and Drucker [40] by logarithmic factors. It remains open if transitive functions with block sensitivity $o(n^{3/7})$ exist, and the following remains an interesting open question in this context:

Question 5. *If $f\colon \{0,1\}^n \to \{0,1\}$ is a transitive function, is $bs(f) \geq \Omega(n^{3/7})$?*

An analogous statement to questions 3 and 5 is known to hold for certificate complexity as mentioned in [65].

Theorem 1.6. *(Theorem 5.2 in [65]) For any transitive function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$C_0(f)C_1(f) \geq n.$$

This implies that if f is transitive, then $C(f) \geq \sqrt{n}$.

This is tight for the AND-of-ORs function on n bits, for which $C_0(f) = C_1(f) = \sqrt{n}$.

1.2.1.3 Separations

The best separation between sensitivity and block sensitivity remains quadratic. The first example of such a function with quadratic separation between its sensitivity and block sensitivity was given by Rubinstein [85] who constructed a function f with $bs(f) = \frac{1}{2}s(f)^2$. Other constructions with

quadratic separation were given in [97, 28, 47, 10]. The largest separation so far is achieved by the construction of Ambainis and Sun [10] who gave a function f with $bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f)$.

Improving the constant $\frac{2}{3}$ in the separation would be interesting, since a function f with $bs(f) > cs(f)^2$ for a constant $c > 1$ would imply a construction with superquadratic separation by iterated composition of the function f [8].

Ambainis and Prusis [8] (improving the constant of a statement in [57]) proved that $bs_0(f) \leq \frac{2}{3}s_0(f)C_1(f)$ where $C_1(f)$ denotes the 1-certificate complexity of f. On the other hand, [75] proved that $C_1(f) \leq bs_1(f)s_0(f)$. The analogous statements also hold for upper bounding bs_1 and C_0, respectively. Combining these results implies that in order to obtain much stronger separation between sensitivity and block sensitivity it is necessary to construct functions f such that both $bs_0(f)$ and $bs_1(f)$ are significantly larger than $s(f)$.

By direct constructions, the largest simultaneous separation has been $\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^{\log_2 3})$ in [5]. On the other hand, all previous direct constructions with quadratic separation between $bs(f)$ and $s(f)$ had $\min\{bs_0(f), bs_1(f)\} = O(s(f))$.

1.2.2 Our results

We make partial progress towards question 1: we prove better (cubic and sometimes quadratic) bounds for certain interesting subclasses of transitive functions. Note that these bounds are stronger than the bounds that follow for transitive functions due to Huang's work [51]. For the classes we con-

sider, we answer question 3 (with $\beta = 1/3$) and question 4. We also prove tight lower bounds on the block sensitivity of these subclasses, answering question 5. (See the paper [32].)

We make partial progress towards question 2 as well: we give new constructions of functions that match the current best separation between sensitivity and block sensitivity. Additionally, we give the first non-trivial direct constructions of Boolean functions with simultaneous quadratic separation between $bs_0(f), bs_1(f)$ and $s(f)$, i.e., we construct functions f such that $\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^2)$. (See the paper [30].)

1.2.2.1 Lower bound on sensitivity of transitive functions

In this part, we show that the statement of Chakraborty's question (Question 3) holds with $\beta = 1/3$ for some interesting subclasses of transitive functions. That is, for functions f on n variables in these classes, $s(f) \geq \Omega(n^{1/3})$, which implies $bs(f) \leq O(s(f)^3)$. We also prove the statement of Question 4 for arbitrary values of n for one of the classes we consider. That is, $s(f) \geq \Omega(\sqrt{n})$ and therefore $bs(f) \leq O(s(f)^2)$ when $f(0^n) \neq f(1^n)$ for functions f on n variables in this class.

We consider the following subclasses of transitive functions.

Transitive Functions with Sparse DNF (or CNF)

We consider transitive functions that can be represented by DNFs with up to $2^{n^{\frac{1}{2}-\epsilon}}$ terms, or by CNFs with up to $2^{n^{\frac{1}{2}-\epsilon}}$ clauses, for constant

$\epsilon > 0$. For any non-constant function f of this form we prove that $s(f) \geq \Omega(\min\{n^{1/3}, n^{2\epsilon}\})$. In particular, setting $\epsilon = 1/6$ gives the bound $s(f) \geq \Omega(n^{1/3})$ for transitive functions represented by DNFs (or CNFs) of size up to $2^{n^{1/3}}$.

Comparing with previous results, we note that any DNF with at most t terms is also a read-t DNF. Thus, the results of [17] imply that non-constant functions represented by DNFs with at most $n^{\frac{1}{3}-\epsilon}$ terms have sensitivity $\Omega(n^{\epsilon})$. Our results significantly improve this to DNFs with up to an exponential $2^{n^{\frac{1}{2}-\epsilon}}$ number of terms, in the case of transitive functions.

Transitive Functions Represented by DNF (or CNF) with a Not-Too-Frequent Variable

We further extend these results to transitive functions represented by DNFs (or CNFs) of arbitrary sizes, as long as there exists a variable that appears in at most $2^{n^{\frac{1}{2}-\epsilon}}$ terms (resp. clauses) of the formula, for constant $\epsilon > 0$. We prove the bound $s(f) \geq \Omega(\min\{n^{1/3}, n^{2\epsilon}\})$ for these functions. As above, setting $\epsilon = 1/6$ gives $s(f) \geq \Omega(n^{1/3})$.

Transitive Functions Represented by DNF (or CNF) with Approximately the Same Number of Positive Literals per Term

Next we consider transitive functions represented by DNF (or CNF) where the number of terms as well as the size of the terms (i.e. the width of the DNF) are arbitrary, but the number of positive literals in each term is the

same up to constant factors. We prove for transitive functions f on n variables with this property that $s(f) \geq \Omega(n^{1/3})$.

This class significantly extends the previously studied class of *minterm-transitive* functions. Roughly speaking, minterm-transitive functions have the property that all their 1-inputs are consistent with minterms that are equivalent to just one minterm, under permutations from the invariance group of the function. As we saw in Section 1.2.1.2, Chakraborty [28] proved that minterm-transitive functions f on n variables have $s(f) \geq \Omega(n^{1/3})$, and he noted that his argument extends to the case when the number of positive literals as well as the sizes of each term are the same up to constant factors. Our contribution is to further extend the argument without making any assumptions about the sizes of the terms.

Quadratic bounds for functions with $f(0^n) \neq f(1^n)$ We now answer question 4 for this class, under the additional assumption that $f(0^n) \neq f(1^n)$. We show that $s(f) \geq \Omega(\sqrt{n})$ for transitive functions f represented by DNF (or CNF) such that the number of positive literals per term is the same up to constant factors, and $f(0^n) \neq f(1^n)$. Previously this was not known to hold for arbitrary values of n, even for the special case of minterm-transitive functions.

1.2.2.2 Lower bounds on block sensitivity of transitive functions

We now answer question 5 for the above classes (with slightly different tradeoffs in parameters), that is, we prove tight, $\Omega(n^{3/7})$ lower bounds on the

block sensitivity of functions on n variables in the corresponding classes.

In particular, we first prove that for any transitive function f that can be represented by a DNF (or CNF) with at most $2^{\frac{n^{3/7}}{2}}$ terms (resp. clauses), it holds that $bs(f) \geq \Omega(n^{3/7})$.

We extend this bound as before, to the class of transitive functions that can be represented by a DNF (or CNF) such that its i-th variable appears in at most $2^{\frac{n^{3/7}}{3}}$ terms (resp. clauses) of the formula, for some $i \in [n]$, and prove that $bs(f) \geq \Omega(n^{3/7})$ for such functions.

Finally, we also prove that $bs(f) \geq \Omega(n^{3/7})$ for any transitive function f that can be represented by a DNF (or CNF) such that the number of positive literals per term of the DNF (resp. clauses of the CNF) is the same up to constant factors.

1.2.2.3 Tightness of our bounds

As noted before, Chakraborty [28] gave an example of a transitive function on n variables with sensitivity $\Theta(n^{1/3})$, and Amano [4] gave an example of a transitive function on n variables with block sensitivity $\Theta(n^{3/7})$. Both functions are minterm-transitive, thus they can be represented by DNFs where each term has the same number of positive literals. On the other hand, both functions can be represented by DNFs with n terms, thus they also belong to the other two classes of transitive functions that we consider. This shows that our bounds $s(f) \geq \Omega(n^{1/3})$ and $bs(f) \geq \Omega(n^{3/7})$ are the best possible for these classes, up to constant factors.

We present a simple example of a minterm-transitive function f on n variables, with sensitivity $\Theta(\sqrt{n})$ such that $f(0^n) \neq f(1^n)$. This shows that our $\Omega(\sqrt{n})$ lower bound on sensitivity is tight up to constant factors for the corresponding class.

1.2.2.4 Our Techniques

Our results are based on new upper bounds on the *minimum certificate size*, that hold for *arbitrary* Boolean functions, not just transitive functions. We give two such bounds: one upper bounds the minimum certificate size by the sensitivity of the function and by the logarithm of the number of terms of the DNF (Lemma 2.6 in Chapter 2), the other relates the minimum certificate size to the number of occurrences of any given variable and the influence of that variable (Lemma 2.7 in Chapter 2). We note that relating the minimum certificate size to influence has been also used in [13] in a different context. These upper bounds allow us to take advantage of a result of Chakraborty [28] (see Corollary 2.1 in Chapter 2) which shows that for transitive functions, upper bounds on the minimum certificate size imply lower bounds on the sensitivity of the function.

We emphasize that our upper bounds on minimum certificate size hold for arbitrary Boolean functions, not just transitive functions. The part of our arguments that is specific to transitive functions, is using the fact that for transitive functions, upper bounds on minimum certificate size imply lower bounds on sensitivity, and the relationship between the influences of different

variables of transitive functions.

We also provide a new, stronger tradeoff between sensitivity and the certificate size on two special inputs, (0^n and 1^n), that holds for arbitrary transitive functions (Lemma 2.9 in Chapter 2). This allows us to obtain tight, $\Omega(\sqrt{n})$ lower bounds on the sensitivity of functions f on n variables in our third class, when $f(0^n) \neq f(1^n)$.

Finally, we observe that upper bounds on the minimum certificate size also provide lower bounds on block sensitivity of arbitrary transitive functions, (Lemma 2.12 in Chapter 2), with a stronger tradeoff than what follows from tradeoffs between sensitivity and minimum certificate size. This allows us to obtain tight, $\Omega(n^{3/7})$ lower bounds on the block sensitivity of functions in all the classes we consider.

1.2.2.5 Towards stronger separations between $s(f)$ and $bs(f)$

In this part of the dissertation, we construct new families of Boolean functions that exhibit quadratic separation between sensitivity and block sensitivity. Our constructions have several novel aspects.

We first note that all previous constructions - with the exception of Chakraborty's functions [28] - were of the form $f = OR_m \circ g_k$ that is $f : \{0,1\}^{mk} \to \{0,1\}$ was obtained by composing the m-bit OR function with an appropriately chosen inner function g on k bits. Chakraborty [28] did not use function composition at all. As for the choice of the inner function, Gopalan, Servedio, Tal and Wigderson [47] defined the inner function g based on code-

words of a Hamming code. All other constructions (including Chakraborty [28]) used the presence of certain patterns in the input x to set the function value $g(x)$ to 1.

We observe that other function compositions instead of OR-composition can also yield quadratic separations. We define new functions, that could be used as inner or outer functions, based on algebraic criterions related to multiplication in finite fields or polynomial multiplication.

We give two versions of our "building block" functions based on finite field multiplication, denoted g_{FF} and g^*_{FF}. and two functions g_{poly} and g^*_{poly} based on polynomial multiplication. Using OR-composition, g_{FF} yields constant $\frac{1}{4}$, g^*_{FF} and g_{poly} give constant $\frac{1}{2}$ in quadratic separations, while g^*_{poly} gives constant $\frac{2}{3}$, matching the current largest separation between sensitivity and block sensitivity by Ambainis and Sun [10].

We present a general framework based on certificates that captures most previous constructions, and also highlights their limitations. We also give a general condition for achieving quadratic separations with the $\frac{2}{3}$ constant for functions defined by families of certificates. The function by Ambainis and Sun [5] fits into this framework.

In addition, we provide the first direct constructions of families of Boolean functions f with $\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^2)$. Our simultaneous quadratic separation of both 0-block sensitivity and 1-block sensitivity from sensitivity is based on a more refined study of the effects of function compo-

sition on these measures. We also present sufficient conditions for achieving such simultaneous separations and give several examples of functions satisfying these conditions.

1.3 A new complexity measure: diameter of Boolean functions

In this part of our work, we introduce a new complexity measure of Boolean functions we call *diameter*, that captures the relationship between certificate complexity and several other measures of Boolean functions. Our measure can be viewed as a variation on *alternating number*, but while alternating number can be exponentially larger than certificate complexity, we show that diameter is always upper bounded by certificate complexity. We argue that estimating diameter may help to get improved bounds on certificate complexity in terms of sensitivity, and other measures. (See the paper [31].)

1.3.1 Motivation

The alternating number of a Boolean function f, denoted $alt(f)$, is a well-studied complexity measure, with a wide range of applications in different areas of theoretical computer science. It is defined as the maximum number of times the function changes value along any monotone path from the input 0^n to the input 1^n (where a path is said to be monotone if the set of 1 bits of any input x on the path is a subset of the 1 bits of any input that appears later in the path). It follows from the definition that the alternating number is 1 for

monotone functions, and in general, the alternating number can be considered to be a measure of how close the function is to being monotone. It was first studied by Markov [66], who showed that the minimum number of negation gates to compute f by any Boolean circuit is exactly $\lceil \log_2(alt(f)+1) \rceil$. This led to further studies of alternating number in connection to understanding the effect of negation gates in various contexts such as circuit complexity [86, 91, 70, 71], learning theory [22], and cryptography [48].

Our new measure is motivated by an interesting paper of Lin and Zhang [61], who studied functions with small alternating number in the context of the sensitivity conjecture, and the log-rank conjecture for XOR functions. Both conjectures have been open for several decades, and only verified for a few special classes of Boolean functions (until Huang's result). Both conjectures can be easily verified to hold for monotone Boolean functions. Lin and Zhang [61] showed that both conjectures remain true when considering functions that are close to monotone, that is for functions with small alternating number. More precisely, they showed that the conjectures hold for functions with constant alternating number, as well as for functions with alternating number bounded above by some relevant complexity measures of the functions, such as sensitivity in the case of the sensitivity conjecture, and Fourier sparsity in the case of the log-rank conjecture for XOR functions. Thus, their work extended the class of functions where the conjectures can be verified. On the other hand, Dinesh and Sarma [39] presented a function f such that $alt(f)$ is exponentially larger than the certificate complexity $C(f)$. This means that the sensitivity

conjecture and the log-rank conjecture for XOR functions cannot be proved in the general case by providing upper bounds on alternating number.

Our new measure, called *diameter*, captures the relationship between certificate complexity and several other measures of Boolean functions. It is motivated by alternating number, but it is quite different from it. We shall now see some similarities as well as differences between the two measures at a high level.

Comparison of diameter with alternating number: First, the similarity is that both measures involve considering paths in the Boolean cube, where one step of the path involves flipping a block of input bits of the function, thus each step specifies a subcube. For alternating number, the requirement on the function values on the subcubes is that the function takes different values on the two opposite (all 0 and all 1) points of the subcubes. For our measure, we consider different classes of Boolean functions that can appear as subfunctions on the subcubes associated with a path.

We note that the definition of alternating number also requires that a path is monotone. We do not impose such requirement, and in fact as we will see, diameter does not measure closeness to monotonicity.

Furthermore, alternating number considers the longest "legal" path between just two specific points, the all 0 input and the all 1 input. For our measure, we consider the shortest "legal" path between inputs x and subcubes corresponding to certificates of the function on the complementary input $\bar{x}$.

Then, similarly to standard complexity measures like certificate complexity, we take the maximum over all inputs. Depending on the class $\mathcal{H}$ of functions we allow to appear as subfunctions along the steps of a path, we obtain variants of our measure, denoted $dia_{\mathcal{H}}$. We note that in general, diameter and alternating number are incomparable, and we provide examples that illustrate this. However, an important distinction is that while alternating number can be exponentially larger than certificate complexity [39], each variant of our measure is upper bounded by certificate complexity, up to constant factors.

1.3.2 Our results

1.3.2.1 Different types of diameters

We define the measures $dia_{\mathcal{H}}$ for the following classes $\mathcal{H}$: dia_{and} where $\mathcal{H}$ consists of the functions AND, OR, NAND and NOR (we refer to this class as "AND-OR"), dia_s where $\mathcal{H}$ includes all functions with full sensitivity, dia_{deg} where $\mathcal{H}$ includes all functions with full real degree, dia_{deg_2} where $\mathcal{H}$ includes all functions with full $\mathbb{F}_2$-degree.

Note that each of the classes we consider contains the functions AND, OR, NAND and NOR, since each of these functions has full sensitivity, full $\mathbb{F}_2$-degree and full real degree. Thus, each of the measures $dia_s(f)$, $dia_{deg}(f)$ and $dia_{deg_2}(f)$ is upper bounded by $dia_{and}(f)$, for every Boolean function f. Furthermore, since $deg_2(f) \leq deg(f)$, we have that $dia_{deg}(f) \leq dia_{deg_2}(f)$.

Other variants One could consider more versions of our measure, for various other classes $\mathcal{H}$. Another class that is natural to consider in connection

to the log-rank conjecture for XOR functions is taking $\mathcal{H}$ to be the class of Boolean functions with full Fourier sparsity, that is functions such that all their Fourier coefficients are nonzero. We remark that the results are analogous to our results on dia_{deg_2} with similar applications.

We would like to mention another variant of our definitions, that turns out to be helpful in proving some of our results for special classes of Boolean functions. As we saw before, it can often be quite useful to consider minimum certificate complexity, defined as $C_{min}(f) = min_x C(f,x)$. Similarly, while we define diameter as $dia_{\mathcal{H}}(f) = max_x dia_{\mathcal{H}}(f,x)$, we also consider minimum diameter defined as $dia_{min,\mathcal{H}}(f) = min_x dia_{\mathcal{H}}(f,x)$. We note that while we illustrate with examples that the alternating number of a function and its $\mathcal{H}$-diameter are incomparable for the different classes $\mathcal{H}$ that we consider, we prove that the min $\mathcal{H}$-diameters are upper bounded by the alternating number, for all our different classes $\mathcal{H}$.

1.3.2.2 Diameters and certificate complexity

We prove that for all the classes $\mathcal{H}$ considered in this dissertation,

$$dia_{\mathcal{H}}(f) \leq 2C(f).$$

Depending on the class $\mathcal{H}$, we can lower bound $dia_{\mathcal{H}}$ by certificate complexity divided by specific complexity measures, such as sensitivity. For the classes "AND-OR", and "full sensitivity" we get that for any Boolean function f,

$$C(f)/s(f) \leq dia_{\mathcal{H}}(f)$$

and thus for these classes, we can upper bound certificate complexity as follows:

$$C(f) \leq dia_s(f)s(f) \leq dia_{and}(f)s(f).$$

Similarly, considering the classes $\mathcal{H}$ consisting of Boolean functions with full real degree and full $\mathbb{F}_2$-degree, respectively, we get the following bounds relating diameter and certificate complexity. For any Boolean function f, $C(f) \leq dia_{deg}(f)deg(f) \leq dia_{and}(f)deg(f)$ and $C(f) \leq dia_{deg_2}(f)deg_2(f) \leq dia_{and}(f)deg_2(f)$.

1.3.2.3 Sensitivity and block sensitivity of transitive functions with small diameters

Previous results due to Lin and Zhang [61] imply that $s(f) \geq \Omega(n^{1/3})$ for transitive functions with constant alternating number. We improve and extend this bound and prove that $s(f) \geq \sqrt{n}$ for transitive functions with constant alternating number, as well as for transitive functions with constant diameter, considering dia_s or dia_{and}.

Since block sensitivity is at least sensitivity, our result above also implies that for transitive functions with constant alternating number or constant diameter (dia_s or dia_{and}), $bs(f) \geq \Omega(\sqrt{n})$. We prove that the conjectured $\Omega(n^{3/7})$ lower bound holds under a weaker condition, for all transitive functions with constant *minimum* diameter ($dia_{min,s}$ or $dia_{min,and}$). This answers Question 5 in the affirmative for these classes of transitive functions. We note that the previous best known lower bound on block sensitivity for such func-

tions was $n^{1/3}$, which followed from Sun's [90] bound on block sensitivity for all transitive functions mentioned in Section 1.2.1.2.

1.3.2.4 Log-rank conjecture for XOR functions with small diameters

The log-rank conjecture for functions of the form $f(x \oplus y)$ has been proved when f belongs to certain special classes such as monotone or linear threshold functions [69], symmetric functions [101], functions with low $\mathbb{F}_2$-degree or small spectral norm [94], AC^0 functions [59], read-k functions [29], and functions with constant alternating number by Lin and Zhang [61]. We prove that the log-rank conjecture holds for functions of the form $f(x \oplus y)$ for functions f with dia_{deg_2} or dia_{and} bounded above by a polynomial of the logarithm of the Fourier sparsity of the function f.

1.3.2.5 Implications of proving stronger bounds on diameters

As we noted before, $dia_{\mathcal{H}}(f) \leq 2C(f)$ for any f and any class $\mathcal{H}$ that contains the functions AND, OR, NAND and NOR. Huang's result implies that $C(f) = O(s(f)^5)$, which in turn implies that for any f, and any class $\mathcal{H}$ that contains the functions AND, OR, NAND and NOR (which means for every class considered in this work), we have

$$dia_{\mathcal{H}}(f) = O(s(f)^5).$$

Obtaining new upper bounds (of various form) on $dia_{\mathcal{H}}$ could lead to the following interesting consequences:

- An independent proof of the upper bound $dia_{\mathcal{H}}(f) \leq poly(s(f))$ for dia_s or dia_{and} could lead to an independent, purely combinatorial proof of the sensitivity conjecture.

- Improving the upper bound on dia_s or dia_{and} in terms of sensitivity to $dia_{\mathcal{H}}(f) \leq O(s(f)^2)$, would improve the current best upper bounds on block sensitivity and certificate complexity to $bs(f) \leq C(f) \leq O(s(f)^3)$.

  In connection to this question, we note that there are Boolean functions with $dia_s(f)$ (and thus $dia_{and}(f)$) at least $\Omega(s(f)^{1.22})$, since Ben-David et al. [20] exhibited a function with $C(f) = \Omega(s(f)^{2.22})$.

- Proving that $dia_{deg}(f) \leq deg(f)$ would imply the bound $C(f) \leq O(s(f)^4)$, using Huang's result, but improving its current implication which gives only $C(f) \leq O(s(f)^5)$. It would also imply that $C(f) \leq O(deg(f)^2)$, improving the current best bound giving $C(f) \leq deg(f)^3$ by [68].

  We note that there are Boolean functions with $dia_{deg}(f)$ (and therefore, $dia_{deg_2}(f)$ and $dia_{and}(f)$) at least $\Omega(deg(f)^{0.63})$, since there are Boolean functions with $C(f) = \Omega(deg(f)^{1.63})$ [77].

- Upper bounds on $dia_{\mathcal{H}}$ (considering an appropriate $\mathcal{H}$) for specific classes of Boolean functions could give stronger upper bounds on block sensitivity or certificate complexity in terms of sensitivity than currently known for these classes, and could verify the log-rank conjecture for XOR functions for new classes.

1.4 Decision trees in machine learning

In this part of the dissertation, we present algorithms for decision trees in the context of some machine learning problems. First, we focus on decision trees as a binary classification model in the supervised learning setting, and present algorithms to optimize certain evaluation metrics of a given decision tree. These algorithms are also quite relevant to the transfer learning setting as we shall see. Next, we present some algorithms for decision trees as a regression model in the context of the transfer learning regime.

1.4.1 History

Decision trees were first used for the *classification problem* in supervised learning by Fisher in 1936 [82]. They have been a popular supervised learning model since they were re-introduced in this context in 1984 by Breiman et al [24]. They remain popular mainly due to their easy interpretability, efficiency and the fact that they still exhibit competitive performance for most learning tasks. Please refer to the survey [63] about the history of the use of decision trees for classification as well as regression tasks. We shall now introduce the supervised learning setting, and the use of decision trees as a prediction model.

1.4.1.1 Supervised learning

The supervised learning setting consists of a data set of m points $\{(x_i, y_i) \mid i \in [m]\}$ such that each $x_i \in \mathcal{V}_1 \times \mathcal{V}_2 \ldots \times \mathcal{V}_n$ is an n-dimensional vector for some integer $n > 0$, where $\mathcal{V}_j$ is a discrete or continuous domain.

The components of the data points $(x_i)_j$ are called as *input variables* or *features* of the data, and the space $\mathcal{V}_1 \times \mathcal{V}_2 \ldots \times \mathcal{V}_n$ is called as the feature space. The component (y_i) of the data points is called as the *target variable* or the output variable. A feature belonging to a finite discrete domain is said to be a categorical feature, whereas one where the domain is an infinite set (usually continuous) is said to be a numeric feature. The data set is partitioned into a training set and a testing set. A *learning algorithm* takes as input the training set, and outputs a *prediction model* $\mathcal{M}$. A prediction model $\mathcal{M}$ takes the feature values x of a data point (x, y) as input, and outputs a prediction $\mathcal{M}(x)$. A prediction model is usually evaluated based on how closely the prediction $\mathcal{M}(x)$ matches the actual value of the target variable y, aggregated over the testing data set. We shall see some evaluation metrics in greater detail later.

Supervised learning can be further classified into two types of tasks depending on the domain of the target variable. The regime where the domain of the target variable y is a finite set is called as the *classification problem*, whereas the setting in which the target variable y belongs to an infinite set (typically, the set of reals $\mathbb{R}$) is called as the *regression problem*. More specifically, if the target variable y is Boolean, that is, $y \in \{0, 1\}$, the regime is called as the binary classification problem. A prediction model for a classification task is also called as a *classification model*, and that for a regression task is sometimes called a *regression model*.

Evaluation metrics for binary classification models: Different evaluation metrics are used to gauge the performance of a binary classification model.

We shall now define some of the commonly used metrics that shall be of interest to us. In these definitions, we denote the data set over which the model is evaluated (which is usually the testing set) as $\mathcal{D} = \{(x_i, y_i) \mid i \in [m]\}$.

Definition 13. Recall of a model *The recall of a binary classification model $\mathcal{M}$ for the data set $\mathcal{D}$ with respect to class 1 is defined as*

$$\begin{aligned} Recall_1(\mathcal{M}) &= \frac{Number\ of\ points\ from\ \mathcal{D}\ in\ class\ 1\ correctly\ classified\ by\ \mathcal{M}}{Number\ of\ data\ points\ in\ class\ 1\ in\ \mathcal{D}} \\ &= \frac{\sum_{y_i=1}(\mathcal{M}(x_i) = 1)}{\sum_{i\in[m]}(y_i = 1)} \end{aligned}$$

The recall of $\mathcal{M}$ with respect to class 0, $Recall_0(\mathcal{M})$, is defined analogously.

Definition 14. Precision of a model *The precision of a binary classification model $\mathcal{M}$ for the data set $\mathcal{D}$ with respect to class 1 is defined as*

$$\begin{aligned} Precision_1(\mathcal{M}) &= \frac{Number\ of\ points\ from\ \mathcal{D}\ in\ class\ 1\ correctly\ classified\ by\ \mathcal{M}}{Number\ of\ data\ points\ in\ \mathcal{D}\ classified\ as\ 1\ by\ \mathcal{M}} \\ &= \frac{\sum_{\mathcal{M}(x_i)=1} y_i}{\sum_{i\in[m]}(\mathcal{M}(x_i) = 1)} \end{aligned}$$

The precision of $\mathcal{M}$ with respect to class 0, $Precision_0(\mathcal{M})$, is defined analogously.

Definition 15. Error rate of a model *The error rate (or classification error) of a classification model $\mathcal{M}$ for the data set $\mathcal{D}$ is defined as:*

$$\begin{aligned} Error(\mathcal{M}) &= \frac{Number\ of\ wrong\ predictions\ on\ \mathcal{D}\ by\ \mathcal{M}}{Size\ of\ \mathcal{D}} \\ &= \frac{\sum_{i\in[m]} \mathcal{M}(x_i) \neq y_i}{m} \end{aligned}$$

Evaluation metrics for regression models: Since regression tasks involves predicting real values for training examples, different metrics are used to evaluate regression models. The following is a commonly used metric that will be of interest to us:

Definition 16. Root Mean Squared (RMS) Error *The root mean squared error for a model* $\mathcal{M}$ *and data set* $\mathcal{D} = \{(x_i, y_i) \mid i \in [m]\}$ *is defined as:*

$$RMS(\mathcal{M}) = \sqrt{\frac{\sum_{i\in[m]}(\mathcal{M}(x_i) - y_i)^2}{m}}$$

1.4.1.2 Decision trees in supervised learning

As mentioned before, decision trees are among some of the most frequently used prediction models for classification and regression. Decision trees used in the context of classification are a natural generalization of the computational model of decision trees defined in section 1.1.1.1. A decision tree used for binary classification is a rooted binary tree with internal nodes labelled by queries on the features and leaves labelled by $\{0, 1\}$. These queries are of two types: if the queried feature i is categorical, that is, if the domain $\mathcal{V}_i$ is a finite set, then the query is of the form $x_i = a$ for some constant $a \in \mathcal{V}_i$. On the other hand, if the queried feature i is numeric, that is, the domain $\mathcal{V}_i$ is not a finite set (e.g. $\mathbb{R}$), then the query is a threshold-query of type $x_i < a$, for some constant $a \in \mathcal{V}_i$. Given a data point x, a decision tree classifier follows the path starting from the root of the binary tree as per the queries on the feature-values of x and outputs the label at the leaf it reaches.

Notice that if the data points $\{(x_i, y_i) \mid i \in [m]\}$ of the binary classification problem come from the Boolean domain, that is, if $x_i \in \{0,1\}^n$, then a decision tree classifier for such a problem would correspond to the computational model of decision trees introduced in section 1.1.1.1.

Decision trees used in the context of regression tasks are similar except that the leaf labels come from the set of reals $\mathbb{R}$ instead of $\{0,1\}$.

1.4.1.3 Transfer learning

The regime of transfer learning encompasses problems similar to supervised learning, and is based on the high level idea of transferring the prediction skills learned for one data set to another related data set. We focus on the type of transfer learning called as *model transfer*, and from this point in the dissertation, whenever we say transfer learning, we shall be referring to this regime of model transfer, which we now describe. Please refer to the surveys by Pan and Yang [79] and Weiss et al. [98] for more details on the different types of transfer learning.

A (model) transfer learning problem consists of two correlated data sets - the *source data set* (usually large in size) and a *target data set* (which typically contains few data points) which have the same feature space. A transfer learning algorithm receives as input a prediction model trained on the source data set (referred to as the source model) and training data from the target data set, and aims to output a prediction model which will be evaluated on testing data from the target data set. Note that the algorithm has access

to the target data set, and a model trained on the source data set, but not to the source data set itself.

Following are some naïve transfer learning algorithms:

SourceOnly: This algorithm simply returns the source model without making use of the target data in any way.

TargetOnly: This algorithm trains a model only with the target data without making use of the given source model.

These two algorithms are commonly used as baselines against which any new transfer learning algorithm is compared.

Previous work on transfer learning in decision trees: We now mention some previous results on transfer learning with decision trees which are closely related to our work.

Most relevant to our setting is the work by Segev et al. [87] for transfer learning on decision trees. They present two algorithms: 1) *STRUT*, or structural transfer, which preserves the structure of the source model but uses a bicriteria optimization of two information theoretic measures – *information gain* and *divergence gain* – to adjust node thresholds, and; 2) *SER*, or structured expansion and reduction, that performs certain expansion and reduction steps on the source decision tree based on the target data. Another approach in a similar setting was given by Al-Otaibi et al. [3] based on the concept of *versatile decision trees*. Their algorithm first checks if there is data shift that can be corrected using a linear transformation. If so, they make an adjustment

to the data, and, if not, they perform an operation that keeps the structure of decision tree intact, changing only the thresholds of each node appropriately, much like STRUT.

There is also work not restricted to a specific classification model. Kauschke and Fürnkranz [56] propose a paradigm called *patching*, which generalizes SER to other models. Patching identifies regions in the input feature space that are most error prone (called patches), and retrains the underlying classifier on the target data in the patch region. Frustratingly Easy Domain Adaptation by Daumé [37], or *FEDA*, utilizes a simple yet powerful method of *feature augmentation*, that can be applied to any classification model.

1.4.2 Our results

1.4.2.1 Algorithms for decision trees in binary classification tasks

In this part of the dissertation, we present algorithms for decision trees for the binary classification problem focusing on specific evaluation metrics. More specifically, our algorithms start with a pre-trained decision tree model for some binary classification task, and aim to maximize the recall of the model with respect to class 1, subject to the constraint that its precision for class 1 not decrease by more than a specified value (see the paper [34]). This post-processing technique has particular relevance to transfer learning in the case of sparse data in the target domain.

Our approach is based on interpreting a decision tree as a union of axis-aligned hyper-rectangles in the feature space. We then focus on the geometric

problem of expanding each hyper-rectangle corresponding to class 1, while ensuring that the relative fraction of data points belonging to class 0 inside each such rectangle remains low. We define a geometric optimization problem based on this approach, which we call the *optimal expansion problem*. We prove that this problem is NP-complete, and therefore, we instead present heuristic methods to try to optimize the objective of this problem. We also give a mixed integer linear programming formulation of our optimal expansion problem which yields exact solutions to it. We then experimentally evaluate our heuristics in the context of transfer learning, and compare them with various existing transfer learning algorithms for decision trees as well as with the exact solutions mentioned above.

1.4.2.2 Transfer learning algorithms for decision trees in regression tasks

This part of our work focuses on algorithms for decision trees for regression tasks in the context of transfer learning (see the paper [33]). Although there has been much work on transfer learning for decision trees as mentioned before, all the previous algorithms have focused on classification tasks. We note that some of these algorithms need modifications to work in the regression setting, and notably no experimental evaluation has been performed in this setting. We modify existing transfer learning algorithms that were designed for decision trees for classification problems, so as to make them work for the regression setting. We then propose a new and simpler algorithm for this problem, and experimentally evaluate these different algorithms.

Chapter 2

Bounds on sensitivity and block sensitivity of some classes of transitive functions

In this chapter[1], we prove tight bounds on the sensitivity and block sensitivity of some special classes of transitive functions, significantly extending previous results about subclasses of transitive functions, and making partial progress towards Questions 1, 3, 4 and 5 mentioned in Chapter 1 for some of these classes. For the classes we consider, we show that for functions f on n variables $s(f) \geq \Omega(n^{1/3})$ which implies $bs(f) \leq O(s(f)^3)$. In addition, we prove that the block sensitivity of functions on n variables in all the classes we consider is at least $\Omega(n^{3/7})$. Furthermore, under the additional assumption that $f(0^n) \neq f(1^n)$, we show that $s(f) \geq \Omega(\sqrt{n})$ for transitive functions f represented by DNF (or CNF) such that the number of positive literals per term is the same up to constant factors. Previously this was not known to hold for arbitrary values of n, even for the special case of minterm-transitive functions. Our lower bounds on both sensitivity and block sensitivity are tight up to constant factors for the corresponding classes.

[1] The results presented in this chapter were published in [32].

2.1 Preliminaries

Recall the definitions of sensitivity, block sensitivity, partial assignments, certificate complexity, minterms and maxterms etc. from Chapter 1. We shall now state some additional definitions and results that we shall require for this chapter.

We state a lemma due to Simon [89] that we shall use in our proofs:

Lemma 2.1. *[89](see also [11]) Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant Boolean function. Then $|f^{-1}(1)| \geq 2^{n-s_1(f)}$ and $|f^{-1}(0)| \geq 2^{n-s_0(f)}$.*

We will use the following observations of Chakraborty about transitive functions. Recall that S_n denotes the group of all permutations on n bits.

We use the following notation: for a set $S \subseteq [n]$ and a permutation $\sigma \in S_n$ we denote by $\sigma(S)$ the set $\{\sigma(i) | i \in S\}$.

Lemma 2.2 (4.3 in [28])**.** *Let $\Gamma \subseteq S_n$ be a transitive group of permutations on n bits. Then, for any $\emptyset \neq S \subseteq [n]$ with $|S| = k$, there exists $\hat{\Gamma} \subseteq \Gamma$ with $|\hat{\Gamma}| \geq \frac{n}{k^2}$ such that for any two permutations $\sigma_1, \sigma_2 \in \hat{\Gamma}$ their images on S are disjoint, that is $\sigma_1(S) \cap \sigma_2(S) = \emptyset$.*

Lemma 2.3 (4.4 in [28])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a non-constant transitive function. Let α be a 1-certificate (resp. 0-certificate) for some $x \in \{0,1\}^n$, with size$(\alpha) = k > 0$. Then, $s_0(f) \geq \frac{n}{k^2}$ (resp. $s_1(f) \geq \frac{n}{k^2}$).*

Since this lemma is crucial for our arguments, we include its proof.

Proof. [28] Let Γ be the invariance group of f. Let S be the set of bits fixed by α, and let $\hat{\Gamma} \subseteq \Gamma$ with $|\hat{\Gamma}| = t \geq \frac{n}{k^2}$ be the set of permutations guaranteed by Lemma 2.2. For $\sigma_i \in \hat{\Gamma}$ let $\alpha_i : [n] \to \{0, 1, *\}$ denote the partial assignment $\sigma_i(\alpha)$ obtained by applying the partial assignment α to the bits permuted according to σ_i, that is $\alpha_i(\ell) = \alpha(\sigma_i(\ell))$ for $\ell \in [n]$. Then, for $1 \leq i \neq j \leq t$, we have that the set of bits fixed by α_i is disjoint from the set of bits fixed by α_j.

Assume that α is a 1-certificate for f (the proof for a 0-certificate is analogous). In this case note that for each $i \in [t]$, α_i is a 1-certificate of f, since each σ_i belongs to the invariance group of f.

Now, consider any 0-input $z \in f^{-1}(0)$. z must disagree with each 1-certificate α_i in at least one bit. Let T be the set of all bits j such that z disagrees in the bit j with some certificate α_i for $i \in [t]$.

Now, let $P \subset T$ be a maximal subset of T such that $f(z^P) = 0$. Since P is maximal, if we flip any other bit in $T \backslash P$, the value of the function will change to 1. Therefore, $s_0(f) \geq |T \backslash P|$. Also, z^P must still disagree with each 1-certificate α_i for $i \in [t]$, since $f(z^P) = 0$. Therefore, $|T \backslash P| \geq t \geq \frac{n}{k^2}$, and we have $s_0(f) \geq \frac{n}{k^2}$. □

Lemma 2.3 immediately gives the following corollary:

Corollary 2.1. *For a non-constant transitive function $f\colon \{0,1\}^n \to \{0,1\}$ we have:*

$$s(f)(C_{min}(f))^2 \geq n \ .$$

We will also use the following observation of Sun [90].

Lemma 2.4. *[90] Let $\Gamma \subseteq S_n$ be a transitive group of permutations on n bits. For any $x, y \in \{0,1\}^n$, if $wt(x) \cdot wt(y) < n$, then there exists some $\sigma \in \Gamma$, such that $\sigma(x)$ and y do not have any 1-s in the same position.*

2.1.1 Influence of Variables

We will take advantage of some Fourier analytic tools. The influence of the variables of a Boolean function can be expressed in terms of Fourier coefficients, but it is also closely related to the average sensitivity of the function. We focus on the connection between influence and average sensitivity.

Definition 17. ***Influence of a Variable*** *For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, the influence of the variable i, denoted by $Inf_i(f)$ is defined as:*

$$Inf_i(f) = \Pr_x[f(x) \neq f(x^i)]$$

where the probability is taken over the uniform distribution on $\{0,1\}^n$.

In other words, if we denote the total number of inputs that are sensitive to the i-th bit by $N_i(f)$, then the influence of the variable i is:

$$Inf_i(f) = \frac{N_i(f)}{2^n}$$

Definition 18. Total Influence *For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, its total influence $Inf(f)$ is defined as:*

$$Inf(f) = \sum_{i \in [n]} Inf_i(f) \ .$$

Alternatively, we have $Inf(f) = \sum_{i\in[n]} \frac{N_i(f)}{2^n}$. *So* $Inf(f)$ *is the average sensitivity of* f, *i.e.,* $Inf(f) = \sum_{x\in\{0,1\}^n} \frac{s(f,x)}{2^n}$.

We will use the following theorem of Kahn, Kalai and Linial [53].

Theorem 2.1. *[53] Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be a Boolean function that equals* 1 *with probability* p. *Then*

$$\max_{i\in[n]} Inf_i(f) \geq \Omega(p(1-p)\log n/n) \ .$$

We also use the following property of transitive functions.

Lemma 2.5. *(see also [78]) If* $f\colon \{0,1\}^n \to \{0,1\}$ *is a transitive function, then* $Inf_i(f) = Inf_j(f)$ *for any* $i, j \in [n]$.

Proof. Let Γ be the set of permutations under which the function f is invariant. For any fixed $i, j \in [n]$ there exists a permutation $\sigma \in \Gamma$ such that $\sigma(i) = j$. Also, $f(x) = f(\sigma(x))$ since f is invariant under σ. Therefore, we have:

$$\begin{aligned} Inf_i(f) &= \Pr_x[f(x) \neq f(x^i)] \\ &= \Pr_x[f(\sigma(x)) \neq f(\sigma(x^i))] \\ &= \Pr_y[f(y) \neq f(y^j)] \\ &= Inf_j(f) \ . \end{aligned}$$

Here we used the fact that a uniform distribution on x gives a uniform distribution on $\sigma(x) = y$. Moreover, $f(\sigma(x^i)) = f(y^j)$ since $\sigma(i) = j$. □

2.2 Lower Bounds on Sensitivity of Transitive Functions

2.2.1 Sparse DNF (or CNF)

In this section we prove lower bounds on the sensitivity of transitive functions that can be represented by DNFs with up to $2^{n^{\frac{1}{2}-\epsilon}}$ terms, or by CNFs with up to $2^{n^{\frac{1}{2}-\epsilon}}$ clauses, for constant $\epsilon > 0$.

We start with a lemma that holds for any Boolean function, transitivity is not required.

Lemma 2.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant Boolean function. If f can be represented by a DNF with t terms, then $C_{min}(f) \leq s_1(f) + \log t$, and if f can be represented by a CNF with t clauses, then $C_{min}(f) \leq s_0(f) + \log t$.*

Proof. We prove the statement about DNFs, the proof for CNFs is analogous. Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant Boolean function that can be represented by a DNF with t terms. Notice that for each term of the DNF, we get a 1-certificate by fixing the variables that appear in the given term, to a value so that the term is satisfied, and leaving the remaining variables free. This means that the number of variables that participate in any given term must be at least $C_{min}(f)$. Thus, the number of different inputs that satisfy a given term is at most $2^{n-C_{min}(f)}$. This means that $|f^{-1}(1)| \leq t2^{n-C_{min}(f)}$. On the other hand, by Simon's Lemma (see Lemma 2.1 in Section 2.1) $|f^{-1}(1)| \geq 2^{n-s_1(f)}$. Combining these two inequalities implies the statement of the lemma.

□

We obtain the following theorem.

Theorem 2.2. *Let $\epsilon > 0$ and let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant transitive function that can be represented by a DNF with up to $2^{n^{\frac{1}{2}-\epsilon}}$ terms, or by a CNF with up to $2^{n^{\frac{1}{2}-\epsilon}}$ clauses. Then $s(f) \geq \Omega(\min\{n^{1/3}, n^{2\epsilon}\})$.*

Proof. We prove the statement about DNFs, the proof for CNFs is analogous.

First note that it is enough to prove the statement for $0 < \epsilon \leq 1/6$, since this will imply that $s(f) \geq \Omega(n^{1/3})$ whenever $\epsilon \geq 1/6$.

Next, notice that if $s_1(f) \geq n^{\frac{1}{2}-\epsilon}$, and $\epsilon \leq 1/6$, then the statement obviously holds. Assume that $s_1(f) < n^{\frac{1}{2}-\epsilon}$. Then, by Lemma 2.6 $C_{min}(f) \leq 2n^{\frac{1}{2}-\epsilon}$, and Corollary 2.1 implies that $s(f) \geq \Omega(n^{2\epsilon})$. □

Remark 2.1. *Setting $\epsilon = 1/6$ gives $s(f) \geq \Omega(n^{1/3})$ for transitive functions represented by DNFs (or CNFs) of size up to $2^{n^{1/3}}$.*

2.2.2 DNF (or CNF) with a Not-Too-Frequent Variable

In this section we further extend the results of the previous section. We show that the same lower bounds for sensitivity hold for transitive functions represented by DNFs with an arbitrary number of terms, as long as there exists a variable that appears in no more than $2^{n^{\frac{1}{2}-\epsilon}}$ terms, for constant $\epsilon > 0$. An analogous result holds considering CNFs.

We once again start with an observation that holds for arbitrary Boolean functions, not just transitive functions.

Lemma 2.7. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function that can be represented by a DNF (or CNF) such that its i-th variable appears in at most k terms (resp. clauses) of the formula, for some $i \in [n]$. Then we have: $C_{min}(f) \leq \log k + 1 - \log Inf_i(f)$.*

Proof. We prove the statement about DNFs, the proof for CNFs is analogous.

As we noted in the proof of Lemma 2.6, for each term of the DNF, we get a 1-certificate by fixing the variables that appear in the given term, to a value so that the term is satisfied, and leaving the remaining variables free. This means that the number of variables that participate in any given term must be at least $C_{min}(f)$. Thus, the number of different inputs that satisfy a given term is at most $2^{n-C_{min}(f)}$.

Consider only those k terms that include the variable x_i. The number of inputs satisfying at least one of these terms is at most $k2^{n-C_{min}(f)}$. Also, notice that each of the 1-inputs that are sensitive to the i-th bit must satisfy one of the terms that include the variable x_i. (Each 1-input must satisfy at least one term, and an input that is sensitive to x_i cannot satisfy a term that does not depend on x_i.) Therefore, the number of 1-inputs that are sensitive to the i-th bit is at most $k2^{n-C_{min}(f)}$. On the other hand, the number of 1-inputs that are sensitive to the i-th bit equals $Inf_i(f) \cdot 2^{n-1}$. Thus, we get $Inf_i(f) \cdot 2^{n-1} \leq k2^{n-C_{min}(f)}$, and this gives the statement of the lemma.

□

We are ready to prove the following theorem for transitive functions.

Theorem 2.3. *Let $\epsilon > 0$ and let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant transitive function that can be represented by a DNF (or CNF) such that one of its variables appears in at most $2^{n^{\frac{1}{2}-\epsilon}}$ terms (resp. clauses) of the formula. Then $s(f) \geq \Omega(\min\{n^{1/3}, n^{2\epsilon}\})$.*

Proof. We prove the statement about DNFs, the proof for CNFs is analogous. As before, it is enough to prove the statement for $0 < \epsilon \leq 1/6$, since this will imply that $s(f) \geq \Omega(n^{1/3})$ whenever $\epsilon \geq 1/6$.

Let x_i be a variable that appears in at most $k = 2^{n^{\frac{1}{2}-\epsilon}}$ terms of the DNF. Recall from Lemma 2.5 that for transitive f, $Inf_i(f) = Inf_j(f)$ for any $j \in [n]$, and thus $Inf_i(f) = \max_{j\in[n]} Inf_j(f)$. By Theorem 2.1 due to Kahn, Kalai and Linial [53], $\max_{j\in[n]} Inf_j(f) \geq \Omega(p(1-p)\log n/n)$, where p is the probability that the function f equals 1. Then, by Lemma 2.7 we get $C_{min}(f) \leq \log k + 1 - \log Inf_i(f) \leq O(\log k + 1 + \log n + \log \frac{1}{p(1-p)})$.

Notice that if $\frac{1}{p(1-p)} \geq 2^{n^{\frac{1}{2}-\epsilon}}$, then it holds that $\min\{|f^{-1}(1)|, |f^{-1}(0)|\} \leq 2^{n+1-n^{\frac{1}{2}-\epsilon}}$. Then by Lemma 2.1, $s(f) \geq \Omega(n^{\frac{1}{2}-\epsilon})$, which is at least $\Omega(n^{2\epsilon})$ when $\epsilon \leq 1/6$.

Otherwise, $\frac{1}{p(1-p)} < 2^{n^{\frac{1}{2}-\epsilon}}$, and we get $C_{min}(f) \leq O(n^{\frac{1}{2}-\epsilon})$. Then, Corollary 2.1 implies that $s(f) \geq \Omega(n^{2\epsilon})$.

□

Remark 2.2. *Setting $\epsilon = 1/6$ gives $s(f) \geq \Omega(n^{1/3})$ for transitive functions represented by DNFs (or CNFs) such that one of the variables appears no more than $2^{n^{1/3}}$ times.*

2.2.3 DNF (or CNF) with Approximately the Same Number of Positive Literals per Term

In this section we consider transitive functions represented by DNFs where the number of terms as well as the size of the terms (i.e. the width of the DNF) are arbitrary, but the number of positive literals in each term is approximately the same. In other words, we consider transitive functions f such that the 1-inputs of f can be covered by subcubes that correspond to minterms with approximately equal weights.

Note that minterm-transitive functions have this property, since all their minterms have exactly the same weight. However, a minterm-transitive function f must have a single minterm α such that every 1-input of f agrees with either α or $\sigma(\alpha)$ for some σ in the invariance group of f. Our condition allows f to have a set $\Lambda = \{\alpha_1, \alpha_2, \ldots\}$ of an arbitrary number of different minterms, as long as they have approximately the same weight, and every 1-input of f agrees with some $\alpha_i \in \Lambda$.

Note also that we allow the different minterms in Λ to have different sizes, we only require that they have approximately equal weight. That is, we require that they each set approximately the same number of bits to 1 but they can set different numbers of bits to 0.

Remark 2.3. *Our arguments would also work if we require the number of bits fixed to* 0 *to be approximately the same in each minterm. Analogous results hold for maxterms and CNFs as well.*

First we prove a simple lemma that holds for arbitrary Boolean functions, not just for transitive functions.

Lemma 2.8. *Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant Boolean function. Let $\Lambda = \{\alpha_1, \alpha_2, \ldots\}$ be a set of minterms of f such that every 1-input of f agrees with some $\alpha_i \in \Lambda$. Let λ_1 denote the smallest number of 1-s fixed by any $\alpha_i \in \Lambda$, and let λ_0 denote the smallest number of 0-s fixed by any $\alpha_i \in \Lambda$. Then, $s_1(f) \geq \max\{\lambda_1, \lambda_0\}$.*

Proof. First note that although f may have minterms that are not included in the set Λ, every minterm of f must fix at least λ_1 bits to 1, and at least λ_0 bits to 0. This follows because every 1-input of f must agree with some minterm from Λ. This also means that every 1-input $x \in \{0,1\}^n$ must have at least λ_1 of its bits equal to 1 and at least λ_0 of its bits equal to 0. Let $\beta_1 \in \Lambda$ be the minterm that fixes exactly λ_1 bits to 1. Let $y \in \{0,1\}^n$ be the input that agrees with β_1 on the bits fixed by β_1, and is 0 on every free bit of β_1. Then, $f(y) = 1$, but changing any 1 bit of y we obtain a 0-input of f. Thus, $s(f, y) \geq \lambda_1$. Similarly, let $\beta_0 \in \Lambda$ be the minterm that fixes exactly λ_0 bits to 0. Let $z \in \{0,1\}^n$ be the input that agrees with β_0 on the bits fixed by β_0, and is 1 on every free bit of β_0. Then, $f(z) = 1$, but changing any 0 bit of z, we obtain a 0-input of f. Thus, $s(f, z) \geq \lambda_0$. This implies the statement. □

Note that the number of minterms in the set Λ can be arbitrarily large. An analogous statement considering sets of maxterms covering the 0-inputs of f gives a lower bound on $s_0(f)$.

We obtain the following theorem.

Theorem 2.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant transitive function. Assume that there is a set $\Lambda = \{\alpha_1, \alpha_2, \ldots\}$ of minterms of f such that every 1-input of f agrees with some $\alpha_i \in \Lambda$. Let w be the weight of the smallest weight minterm in Λ, and assume that for some constant c, $wt(\alpha_i) \leq c \cdot w$ for all $\alpha_i \in \Lambda$. Then $s(f) = \Omega(n^{1/3})$.*

Proof. Let $\alpha \in \Lambda$ be a minterm of f that fixes the smallest number of bits to 0 among the minterms in Λ. Note that if $size(\alpha) \leq n^{1/3}$ then by Corollary 2.1, we have: $s(f) \geq \frac{n}{n^{2/3}} \geq n^{1/3}$. Thus, we can assume that $size(\alpha) > n^{1/3}$.

Let u denote the number of bits fixed to 0 by α. We consider two cases.

Case 1. $u \geq \frac{size(\alpha)}{2}$. Then, by Lemma 2.8, $s_1(f) \geq u \geq \frac{size(\alpha)}{2} > \frac{n^{1/3}}{2}$ and we are done.

Case 2. $u < \frac{size(\alpha)}{2}$, and thus $wt(\alpha) \geq \frac{size(\alpha)}{2}$. Then we have $n^{1/3} < size(\alpha) \leq 2 \cdot wt(\alpha) \leq 2cw$. By Lemma 2.8 this gives $s(f) = \Omega(n^{1/3})$. □

A Stronger Tradeoff between Certificate Size and Sensitivity

Our bounds in the previous sections are based on using the tradeoff between minimum certificate size and sensitivity proved by Chakraborty (see Corollary 2.1). Next we observe that considering the certificate size of a transitive function on either the all 0 or all 1 string, one can obtain a stronger tradeoff between certificate size and sensitivity. More precisely, we prove the following lemma.

Lemma 2.9. *For a non-constant transitive function* $f : \{0,1\}^n \to \{0,1\}$,

$$C(f, 0^n) \cdot s(f) \geq n \quad \text{and} \quad C(f, 1^n) \cdot s(f) \geq n \; .$$

Proof. We prove the first statement, the proof of the second statement is analogous. Let B be a minimal block such that $f(0^n) \neq f((0^n)^B)$. Since B is minimal, $s(f) \geq |B|$.

Let α be a certificate of f on the all zero input 0^n. If $size(\alpha) \cdot |B| < n$, then, we apply Lemma 2.4 to the characteristic vectors of the set of bits that α fixes and the set B. This gives that there is a $\sigma \in \Gamma$ where Γ is the invariance group of f, such that $\sigma(\alpha)$ and B do not have any indices in common. But this gives a contradiction, since every certificate of f on 0^n must intersect B. Thus, $size(\alpha) \cdot |B| \geq n$, which implies the statement.

□

Lower Bound on Sensitivity when $f(0^n) \neq f(1^n)$

We use Lemma 2.9 to obtain stronger lower bounds on the sensitivity of transitive functions with approximately equal weight minterms in their DNF representation, under the additional condition that $f(0^n) \neq f(1^n)$.

Theorem 2.5. *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a non-constant transitive function, such that* $f(0^n) \neq f(1^n)$. *Assume that there is a set* $\Lambda = \{\alpha_1, \alpha_2, \ldots\}$ *of minterms of* f *such that every* 1*-input of* f *agrees with some* $\alpha_i \in \Lambda$. *Let* w *be the weight of the smallest weight minterm in* Λ, *and assume that for some constant* c, $wt(\alpha_i) \leq c \cdot w$ *for all* $\alpha_i \in \Lambda$. *Then* $s(f) = \Omega(\sqrt{n})$.

Proof. First we consider the case when $f(0^n) = 0$. Then, $f(1^n) = 1$, and any DNF representing f must include a term with only positive literals. For a given DNF representing f, let w denote the smallest number of positive literals in any term. Then, by the condition of the Theorem, $C(f, 1^n) \leq c \cdot w$ for some constant c, and combining Lemma 2.9 with Lemma 2.8 we get that $s(f) \geq \Omega(\sqrt{n})$.

In the case when $f(0^n) = 1$, any DNF for f must include a term with only negative literals, and then our condition implies that the DNF uses only negative literals. That is, in this case the function must be anti-monotone, which implies that $s(f, x) = bs(f, x) = C(f, x)$ for every input x. Thus, $s(f) \geq C(f, 0^n)$. Since $f(1^n) \neq f(0^n)$, f is not constant, and we can apply Lemma 2.9, which now directly gives $s(f) \geq \sqrt{n}$.

□

An example with sensitivity $\Theta(\sqrt{n})$ when $f(0^n) \neq f\ (1^n)$

We give a simple example of a minterm-transitive function f on n variables, with sensitivity $\Theta(\sqrt{n})$ such that $f(0^n) \neq f(1^n)$. This shows that our $\Omega(\sqrt{n})$ lower bound on sensitivity in Theorem 2.5 is tight up to constant factors for the corresponding class.

Define $f : \{0,1\}^n \to \{0,1\}$ to be a monotone function, with n minterms $\alpha_1, \ldots, \alpha_n$, each fixing $\sqrt{n}$ bits to 1 as follows. The minterm α_1 fixes the first consecutive $\sqrt{n}$ bits to 1, and α_i for $i \in [n]$ is obtained by cyclically shifting α_1 by $i - 1$ positions.

Note that $f(0^n) = 0$ and $f(1^n) = 1$.

We now show that $s(f) = \Theta(\sqrt{n})$.

1. $s_1(f) = \sqrt{n}$: Any 1-input is consistent with a minterm of size $\sqrt{n}$, fixing a set of $\sqrt{n}$ bits to 1. Therefore, $C_1(f) = \sqrt{n} = s_1(f)$, since f is monotone.

2. $2\sqrt{n} - 4 \leq s_0(f) \leq 2\sqrt{n}$: Note that for any 0-input x, the sensitive bits of x must be 0-bits (because f is monotone).

   Any sensitive 0-bit must be surrounded by a block of at least $(\sqrt{n} - 1)$ consecutive 1s, that is, for any sensitive 0, if the number of consecutive 1's on its left is a and the number of consecutive 1s on its right is b, then $a + b \geq \sqrt{n} - 1$. So for every sensitive 0-bit, there are $\sqrt{n} - 1$ 1-bits around it that ar e not sensitive.

   Moreover, any 1-bit can only contribute to surrounding at most 2 sensitive 0-bits. Thus, denoting the number of sensitive 0-bits by n_0 we have $n_0(\sqrt{n} - 1) \leq 2(n - n_0)$, which gives that $s_0(f)$ is at most $2\sqrt{n}$.

   On the other hand, let $x = (1^{(\frac{\sqrt{n}}{2})}01^{(\frac{\sqrt{n}}{2})}0\ldots)$. Then, $f(x) = 0$ and $s_0(f) \geq s(f, x) = 2\sqrt{n} - 4$.

2.3 Lower Bounds on Block Sensitivity of Transitive Functions

In this section, we prove lower bounds on block sensitivity for all the classes of transitive functions we consider, with suitable tradeoffs in parame-

ters. We start with two Lemmas that follow from Drucker's work [40].

Lemma 2.10. *(Implicit in Lemma 4 in [40]) Let $\Gamma \subseteq S_n$ be a transitive group of permutations on n bits and let $5 \leq r \leq 15$ be an integer.*
Then, for any $\emptyset \neq S \subseteq [n]$ with $|S| \leq n^{3/r}$, there exists $\hat{\Gamma} \subseteq \Gamma$ with $|\hat{\Gamma}| \geq \frac{(16-r)}{12} \cdot n^{(1-\frac{4}{r})}$ such that for each $i \in [n]$, there are at most 3 permutations $\sigma_j \in \hat{\Gamma}$ for which $i \in \sigma_j(S)$.

Lemma 2.11. *(Implicit in the proof of Theorem 2 in [40]) For any non-constant function $f : \{0,1\}^n \to \{0,1\}$, if f has a set of 1-certificates $\Lambda = \{\alpha_1, \ldots \alpha_t\}$ such that for any index $i \in [n]$, there exist at most three 1-certificates from Λ fixing i, then $bs(f) \geq \frac{t}{4}$.*

Combining the above two lemmas, we obtain the following tradeoff between the minimum certificate size and the block sensitivity of transitive functions.

Lemma 2.12. *For any non-constant transitive function $f : \{0,1\}^n \to \{0,1\}$, and an integer $5 \leq r \leq 15$, if $C_{min}(f) \leq O(n^{3/r})$, then $bs(f) \geq \Omega(n^{1-\frac{4}{r}})$.*

Proof. Let α be a 1-certificate of f of size at most $n^{3/r}$, and let S_α be the set of bits fixed by α. By Lemma 2.10, there exists $\hat{\Gamma} \subset \Gamma$ with $|\hat{\Gamma}| \geq \Omega(n^{1-\frac{4}{r}})$, such that for each $i \in [n]$, there are at most 3 permutations $\sigma_j \in \hat{\Gamma}$ for which $i \in \sigma_j(S_\alpha)$. Also notice that for every permutation $\sigma_j \in \hat{\Gamma} \subset \Gamma$, $\sigma_j(\alpha)$ is a 1-certificate of f. The set of 1-certificates $\{\sigma_j(\alpha) | \sigma_j \in \hat{\Gamma}\}$ satisfies the condition of Lemma 2.11 and thus we have that $bs(f) \geq \Omega(n^{1-\frac{4}{r}})$. □

Combining Lemma 2.12 with our arguments in the previous sections, we prove $\Omega(n^{3/7})$ lower bounds on the block sensitivity of functions on n bits in each of the classes we considered.

Theorem 2.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant transitive function. If f can be represented by a DNF with at most $2^{n^{3/7}}$ terms, or with a CNF with at most $2^{n^{3/7}}$ clauses, then $bs(f) \geq \Omega(n^{3/7})$.*

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be represented by a DNF with at most $2^{\frac{n^{3/7}}{2}}$ terms.

We now have two cases:

Case 1: $C_{min}(f) < n^{3/7}$. In this case, we use Lemma 2.12 with $r = 7$ to give $bs(f) \geq \Omega(n^{3/7})$.

Case 2: $C_{min}(f) \geq n^{3/7}$. By Lemma 2.6, we have: $C_{min}(f) \leq s_1(f) + \frac{n^{3/7}}{2}$. Thus we have $bs(f) \geq s_1(f) \geq \frac{n^{3/7}}{2}$ and we are done.

□

As before, we can extend this theorem to DNFs (or CNFs) with an arbitrary number of terms (resp. clauses) as long as there is at least one variable that is not used too many times.

Theorem 2.7. *Let $f : \{0,1\}^n \to \{0,1\}$ be a transitive function that can be represented by a DNF (or CNF) such that its i-th variable appears in at most $2^{n^{3/7}}$ terms (resp. clauses) of the formula, for some $i \in [n]$. Then we have: $bs(f) \geq \Omega(n^{3/7})$*

Proof. We prove the statement about DNFs, the proof for CNFs is analogous. Let x_i be a variable that appears in at most $k = 2^{\frac{n^{3/7}}{3}}$ terms of the DNF.

As noted in Lemma 2.5, for transitive f, $Inf_i(f) = Inf_j(f)$ for any $j \in [n]$, and thus $Inf_i(f) = \max_{j\in[n]} Inf_j(f)$.

Recall from Theorem 2.1 due to Kahn, Kalai and Linial [53], that

$$\max_{j\in[n]} Inf_j(f) \geq \Omega(p(1-p)\log n/n) \ ,$$

where p is the probability that the function f equals 1.

Then, by Lemma 2.7, for large enough n we get

$$C_{min}(f) \leq \log k + 1 - \log Inf_i(f) \leq \log k + 1 + \log n + \log \frac{1}{p(1-p)} \ .$$

Notice that if $\frac{1}{p(1-p)} \geq 2^{\frac{n^{3/7}}{3}}$, that implies that $\min\{|f^{-1}(1)|, |f^{-1}(0)|\} \leq 2^{n+1-\frac{n^{3/7}}{3}}$. Then by Lemma 2.1, $bs(f) \geq s(f) \geq \Omega(n^{3/7})$.

Otherwise, $\frac{1}{p(1-p)} < 2^{\frac{n^{3/7}}{3}}$, and, for large enough n, we get $C_{min}(f) \leq n^{3/7}$. Then, using Lemma 2.12 with $r = 7$ gives $bs(f) \geq \Omega(n^{3/7})$. □

Finally, we consider the class of transitive functions represented by DNFs (or CNFs) where the number of positive literals in each term (resp. clause) is approximately equal.

Theorem 2.8. *Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant transitive function. Assume that there is a set $\Lambda = \{\alpha_1, \alpha_2, \ldots\}$ of minterms of f such that every 1-input of f agrees with some $\alpha_i \in \Lambda$. Let w be the weight of the smallest*

weight minterm in Λ, and assume that for some constant c, $wt(\alpha_i) \leq c \cdot w$ for all $\alpha_i \in \Lambda$. Then $bs(f) = \Omega(n^{3/7})$.

Proof. Let $\alpha \in \Lambda$ be a minterm of f that fixes the smallest number of bits to 0 among the minterms in Λ. Note that if $size(\alpha) \leq n^{3/7}$ then by Lemma 2.12, we have: $bs(f) \geq \Omega(n^{3/7})$. Thus, we can assume that $size(\alpha) > n^{3/7}$.

Let u denote the number of bits fixed to 0 by α. We consider two cases.

Case 1. $u \geq \frac{size(\alpha)}{2}$. Then, by Lemma 2.8, $bs(f) \geq s_1(f) \geq u \geq \frac{size(\alpha)}{2} > \frac{n^{3/7}}{2}$ and we are done.

Case 2. $u < \frac{size(\alpha)}{2}$, and thus $wt(\alpha) \geq \frac{size(\alpha)}{2}$. Then we have $n^{3/7} < size(\alpha) \leq 2 \cdot wt(\alpha) \leq 2cw$. By Lemma 2.8 this gives $bs(f) \geq s(f) = \Omega(n^{3/7})$. □

Chapter 3

Towards stronger separations between sensitivity and block sensitivity of Boolean functions

In this chapter[1], we give various new constructions of families of Boolean functions that exhibit quadratic separation between sensitivity and block sensitivity. Our functions are novel in several ways: we use more general function compositions instead of just OR-composition, and give constructions based on algebraic operations.

Some of our constructions match the current largest separation between sensitivity and block sensitivity by Ambainis and Sun [10]. We also present a general condition for achieving quadratic separations with the constant $\frac{2}{3}$, which is also satisfied by the function by Ambainis and Sun [10]. We further generalize this condition to a framework based on certificates that captures most previous constructions, and also highlights their limitations.

In addition, we give the first direct constructions of families of Boolean functions that have both 0-block sensitivity and 1-block sensitivity quadratically larger than sensitivity. Towards that end, we also develop tools for a

[1] The results presented in this chapter were published in [30].

finer analysis of sensitivity and block sensitivity under function composition, which may be of independent interest.

3.1 Preliminaries

In this section, we first state some more definitions and results that we shall require in addition to those from Chapter 1. We then proceed to review several previous constructions achieving quadratic separation between sensitivity and block sensitivity.

Definition 19. Function defined by a set of partial assignments *Let* $C = \{\gamma_1, \ldots \gamma_t\}$ *be a set of partial assignments where* $\gamma_i \colon [n] \to \{0, 1, \star\}$. *Then* C *naturally defines a function* $g_C \colon \{0,1\}^n \to \{0,1\}$ *as* $g_C(x) = 1$ *if and only if* x *agrees with some partial assignment* $\gamma_i \in C$.

Definition 20. Distances *The distance between two inputs* $x, y \in \{0,1\}^n$ *is defined as the number of bits in which they differ.*

The distance between an input $x \in \{0,1\}^n$ *and a partial assignment* $\alpha \colon [n] \to \{0, 1, \star\}$ *is defined as the minimum distance between* x *and any input* y *agreeing with* α.

The distance between two partial assignments $\alpha, \beta \colon [n] \to \{0, 1, \star\}$ *is defined as the minimum distance between any input* x *agreeing with* α *and any input* y *agreeing with* β.

Definition 21. Function Composition *For Boolean functions* $f : \{0,1\}^m \to \{0,1\}$ *and* $g : \{0,1\}^k \to \{0,1\}$ *the function* $f \circ g : \{0,1\}^{mk} \to \{0,1\}$ *is defined*

on $z \in \{0,1\}^{mk}$ *as:*

$$f \circ g(z) = f(g(z_1, \ldots z_k), g(z_{k+1}, \ldots, z_{2k}), \ldots, g(z_{(m-1)k+1}, \ldots, z_{mk})).$$

Properties of function composition were formally studied with respect to sensitivity and block sensitivity (as well as other related measures) by Tal and Gilmer et al. [92, 44]. We note the following two properties, relevant for us.

Lemma 3.1. *[92, 44] For any Boolean functions* f *and* g *we have* $s(f \circ g) \leq s(f)s(g)$.

Definition 22. *[92] For* $z \in \{0,1\}$ *we say that* $f : \{0,1\}^n \to \{0,1\}$ *is in* z*-good form, if:*
(1) $f(z^n) = z$ *and (2)* $bs(f) = bs(f, z^n)$

Lemma 3.2. *[92] If both* f *and* g *are in* 0*-good form, or if both* f *and* g *are in* 1*-good form, then* $bs(f \circ g) \geq bs(f)bs(g)$.

We now restate a more detailed version of a result by Nisan [75] that we had partially mentioned in Chapter 1.

Theorem 3.1. *[75] For any Boolean function* f *we have*

$$s(f) \leq bs(f) \leq C(f) \, .$$

Moreover, for $z \in \{0,1\}$ *we have*

$$s_z(f) \leq bs_z(f) \leq C_z(f) \, .$$

3.1.1 Previous Constructions with Quadratic Separation

All previous constructions that achieve quadratic separation between sensitivity and block sensitivity - with the exception of Chakraborty's functions [28] - were based on the following "OR-composition Lemma" first used by Rubinstein [85].

Lemma 3.3. *[85] For any function* $g\colon \{0,1\}^m \to \{0,1\}$*, we have*

- $s_0(OR_n \circ g) = ns_0(g)$
- $s_1(OR_n \circ g) = s_1(g)$
- $bs_0(OR_n \circ g) = nbs_0(g)$
- $bs_1(OR_n \circ g) = bs_1(g)$

The quadratic separations of [85, 97, 10, 47] are based on using this lemma and considering functions of the form $f = OR_n \circ g$ for appropriately chosen inner functions g.

Next we briefly describe the previous constructions of functions with quadratic separation.

1. Rubinstein's function [85] Define $g\colon \{0,1\}^{2m} \to \{0,1\}$ as
$g(x) = 1$ iff $x_{2j\ -1} = x_{2j} = 1$ for some $j \in [m]$ and $x_i = 0$ for $i \neq 2j-1, 2j$.
This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = 2m$, $bs_0(g) = m$.
Let $f = OR_{2m} \circ g$. Then $s_0(f) = s_1(f) = bs_1(f) = 2m$, $bs_0(f) = 2m^2$, giving $bs(f) = \frac{1}{2}\ s(f)^2$.

2. Virza's function [97] Define $g\colon \{0,1\}^{2m+1} \to \{0,1\}$ as

$g(x) = 1$ iff one of the following holds:

1) $\exists j \in [m]$ such that $(x_{2j-1} = x_{2j} = 1)$ and $(x_i = 0 \ \ \forall\, i \neq 2j-1, 2j)$.

2) $(x_{2m+1} = 1)$ and $(x_i = 0\ \forall\, i \neq 2m+1)$.

This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = 2m+1$, $bs_0(g) = m+1$.

Let $f = OR_{2m+1} \circ g$. Then $s_0(f) = s_1(f) = bs_1(f) = 2m+1$, $bs_0(f) = (m+1)(2m+1)$. Therefore, $bs(f) = \frac{1}{2}s(f)^2 + \frac{1}{2}s(f)$.

3. Ambainis and Sun's function [10] Define $g\colon \{0,1\}^{2(2m+1)} \to \{0,1\}$ as

$g(x) = 1$ iff $\exists j \in [2m+1]$ such that

1) $x_{2j-1} = x_{2j} = 1$, and

2) For all $i \in [m]$, $x_{2j+2i} = x_{2j-2i} = x_{2j-2i-1} = 0$.

Here, the index of x is taken modulo $(2(2m+1))$ i.e. we index x as if it were laid around a circle.

This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = 3m+2$, $bs_0(g) = 2m+1$.

Let $f = OR_{3m+2} \circ g$. Then $s_0(f) = s_1(f) = bs_1(f) = 3m+2$, $bs_0(f) = (3m+2)(2m+1)$. Therefore, $bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f)$.

4. Function based on Hamming code by Gopalan, Servedio, Tal and Wigderson (Section 7 in [47]).

Consider the Hamming code on $m = 2^r - 1$ bits.

Define $g\colon \{0,1\}^m \to \{0,1\}$ as

$g(x) = 1$ iff x is a codeword of the Hamming code on m bits.

This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = m$, $bs_0(g) = \frac{m+1}{2}$.
Let $f = OR_m \circ g$. Then, $s_0(f) = s_1(f) = bs_1(f) = m$, $bs_0(f) = \frac{m(m+1)}{2}$.
Thus, $bs(f) \ = \frac{1}{2}s(f)^2 + \frac{1}{2}s(f)$.

Finally, we describe a construction by Chakraborty[28] that does not involve function composition. He also constructed another function in that paper, which is similar to the one we describe.

5. Chakraborty's function [28] For integers k, m such that $2 < k < m$ and $2k \mid m$, the function $g_k \colon \{0,1\}^m \to \{0,1\}$ is defined as follows.
For $x = (x_0, \ldots x_{m-1})$, $g_k(x) = 1$ iff $\exists\ i \in \{0, \ldots m-1\}$ such that $x_i = x_{i+1(\mod m)} = 1$ and $x_j = 0$ for all $j \in \{i + 2(\mod m), \ldots, i + k - 1(\mod m)\}$.
Then, $s_0(g_k) = \frac{2m}{k}$, $s_1(g_k) = k$, $bs_0(g_k) = \frac{m}{2}$ and $bs_1(g_k) = k$.
Therefore, setting $k = \sqrt{2m}$ gives $s(g_{\sqrt{2m}}) = \sqrt{2m}$ and $bs(g_{\sqrt{2m}}) = \frac{m}{2}$. So we have $bs(g_{\sqrt{2m}}) = \frac{1}{4}s(g_{\sqrt{2m}})^2$.

3.2 New Building Blocks for Quadratic Separation

In this section, we define several new functions that we will use as inner or outer functions in various function compositions to obtain quadratic separations.

3.2.1 A General Framework Based on Certificates

In this subsection we observe that several previous constructions fit into a common framework, that also explains the limitations of these constructions. Previously, Karthik and Tavenas [55] studied limitations of separations in a certificate framework under some special conditions.

We start with two lemmas that are also helpful for analyzing our new constructions presented in later subsections.

Lemma 3.4. *Let C be a set of partial assignments with $|C| \geq 2$ and consider the function g_C defined by C.*

1. *If the distance between any two partial assignments $\gamma_i, \gamma_j \in C$ for $i \neq j$ is at least 2 then $s_1(g_C) = bs_1(g_C) = C_1(g_C)$.*

2. *If the distance between any two partial assignments $\gamma_i, \gamma_j \in C$ for $i \neq j$ is at least 3 then $s_0(g_C) = 1$.*

Proof. We first note that since the distance between any two partial assignments is at least 2, the set of partial assignments C also forms a set of 1-certificates for g_C, such that every 1-input agrees with exactly one partial assignment from C.

To see that $s_1(g_C) = bs_1(g_C) = C_1(g_C)$, for any 1-input x consider the unique $\gamma_i \in C$ agreeing with x. The bits fixed by γ_i form exactly the set of sensitive bits for f on x, since any two partial assignments in C are at distance

at least 2 from each other. Note also that $C_1(f, x)$ is at most the number of bits fixed by γ_i. Thus, the statement follows by Theorem 3.1.

When the pairwise distance between the partial assignments in C is at least 3, then $s_0(g_C) = 1$ since for any 0-input x, there is at most one certificate $\gamma_i \in C$ such that x is at distance 1 from it.

□

Lemma 3.5. *Let C be a set of partial assignments such that the function g_C defined by C is not constant. Then $bs_0(g_C) \leq |C|$.*

Proof. Note that for a given function g there may be several different sets C such that $g = g_C$. The statement of the lemma holds for any such set of partial assignments C.

The proof is based on the following claim.

Claim 3.1. *Let $f : \{0,1\}^n \to \{0,1\}$ be any non-constant Boolean function. For $x \in f^{-1}(0)$ let A and B be disjoint subsets of $[n]$ such that $f(x^A) = f(x^B) = 1$. Then there is no partial assignment $\alpha : [n] \to \{0,1,*\}$ that is a certificate of f on both x^A and x^B.*

Proof of Claim: Suppose that some partial assignment $\alpha : [n] \to \{0,1,*\}$ is a certificate of f on both x^A and x^B. This means that both x^A and x^B agree with α in the bits fixed by α. On the other hand, since $f(x) = 0$, x must differ from α in at least one bit fixed by α. Since x^A agrees with α, the set A must contain all the coordinates where x and α disagree. Similarly, since x^B

agrees with α, the set B also must contain all the coordinates where x and α disagree. Therefore the sets A and B cannot be disjoint. ■

To prove the lemma, let x be any 0-input of g_C. Let $B_1, \ldots, B_k$ be disjoint sensitive blocks for g_C on x. Note that for every $i \in [k]$, x^{B_i} must agree with some partial assignment in C. By the above claim, for $i \neq j$, x^{B_i} and x^{B_j} must agree with different partial assignments from C. Therefore, the number of disjoint sensitive blocks is at most $|C|$. □

Next we give a sufficient condition for achieving quadratic separations with constant $\frac{2}{3}$, and prove a lemma that we will use in some of our later constructions in Subsection 3.2.3.

Definition 23. *For an odd integer $m > 2$, consider a set of partial assignments $C = \{\gamma_1, \ldots \gamma_m\}$, with $\gamma_i \colon [2m] \to \{0, 1, *\}$. We say that the set of partial assignments C is good if it satisfies the following two properties:*
(a) The distance between any two partial assignments $\gamma_i, \gamma_j \in C$ for $i \neq j$ is at least 3
(b) Each partial assignment $\gamma_i \in C$ has exactly two bits set to 1, $\frac{3}{2}(m-1)$ bits set to 0 and the remaining bits free.

We prove the following lemma for functions g_C defined by a good set of partial assignments C.

Lemma 3.6. *For an odd integer $m > 2$, and any function $g_C \colon \{0,1\}^{2m} \to \{0,1\}$ defined by a good set of partial assignments C, we have*

1. $s_0(g_C) = 1$

2. $bs_0(g_C) = m$

3. $s_1(g_C) = bs_1(g_C) = C_1(g_C) = \frac{3m+1}{2}$

Proof. Lemma 3.4 directly implies the first and third statements, and Lemma 3.5 implies that $bs_0(g_C) \leq m$. It remains to prove that $bs_0(g_C) \geq m$.

Recall that any two certificates γ_i, γ_j must be at a distance at least 3 from each other. But since each certificate only sets exactly 2 bits to 1, this implies that the bits set to 1 by γ_i must be disjoint from the bits set to 1 by γ_j, for any $\gamma_i, \gamma_j \in C$ and $i \neq j$.

Thus, for the 0-input 0^{2m}, the pair of bits set to 1 by the certificate γ_i gives a sensitive block for every $i \in [m]$. All these blocks are pairwise disjoint and therefore $bs_0(g_C) \geq m$.

□

Using OR-composition (Lemma 3.3) yields the following.

Theorem 3.2. *Consider any function $g_C \colon \{0,1\}^{2m} \to \{0,1\}$ defined by a good set of partial assignments C, for an odd integer $m > 2$. Then the function $f = OR_{\frac{3m+1}{2}} \circ g_C$ has*

$$bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f).$$

We note that the inner function defined by Ambainis and Sun [10] can be shown to fit into this framework. We will use Lemma 3.6 to analyze the functions defined in subsection 3.2.3.

Next we observe that Lemma 3.6 and Theorem 3.2 can be further generalized as follows. Recall that the size of a partial assignment is the number of variables fixed by it.

Theorem 3.3. *Let C be a set of partial assignments with $|C| \geq 2$ that satisfies the following two properties:*
(a) The distance between any two $\gamma_i, \gamma_j \in C$ for $i \neq j$ is at least 3
(b) The sets of bits set to 1 by the partial assignments in C are pairwise disjoint.
Let ℓ denote the size of the largest partial assignment in C, and let $m = |C|$ denote the number of partial assignments in C. Then for the function g_C defined by the set of partial assignments C we have

1. $s_0(g_C) = 1$
2. $bs_0(g_C) = m$
3. $s_1(g_C) = bs_1(g_C) = C_1(g_C) = \ell$

Hence, the function $f = OR_\ell \circ g_C$ has

$$bs(f) = \frac{m}{\ell} s(f)^2$$

The proof of this theorem follows directly from Lemma 3.4, Lemma 3.5 and the OR-composition Lemma (Lemma 3.3).

However, this approach has the following limitation.

Lemma 3.7. *Let C be a set of partial assignments with $|C| \geq 2$ such that the sets of bits set to 1 by the partial assignments in C are pairwise disjoint. Let ℓ denote the size of the largest partial assignment in C, and let $m = |C|$ denote the number of partial assignments in C. Let D denote the average pairwise distance between the partial assignments in C. Then*

$$\ell \geq \frac{D}{2}(m-1)\,.$$

The proof of this Lemma is a straightforward generalization of an argument of Ambainis and Sun [10].

Proof. (Implicit in the proof of Theorem 2 in [10]) For $\gamma_i \in C$, let A_i denote the set of bits fixed to 0 by γ_i, and let B_i denote the set of bits fixed to 1 by γ_i. Then, for $i \neq j$ the distance between γ_i and γ_j is equal to $|A_i \cap B_j| + |A_j \cap B_i|$. Thus,

$$\sum_{i,j:i\neq j} |A_i \cap B_j| = D\frac{m(m-1)}{2}$$

This implies that for some i,

$$\sum_{j:i\neq j} |A_i \cap B_j| \geq D\frac{(m-1)}{2}$$

which in turn implies that $|A_i| \geq D\frac{(m-1)}{2}$, since the sets B_j are pairwise disjoint. The statement of the lemma follows since $\ell \geq \max_i |A_i|$. □

Lemma 3.7 implies that using a construction satisfying the conditions of Theorem 3.3 with average pairwise distance D cannot give a constant greater than $\frac{2}{D}$ in quadratic separations. The construction of Ambainis and Sun fits into this framework and matches the bound of Lemma 3.7 with $D = 3$ yielding the current best quadratic separation with constant $\frac{2}{3}$. The constructions of Rubinstein and Virza also fit into this framework and match the bound of Lemma 3.7 with $D = 4$, yielding quadratic separations with constant $\frac{1}{2}$. Note that $D \geq 3$ is part of the conditions in Theorem 3.3, thus this framework cannot give constants greater than $\frac{2}{3}$ in quadratic separations.

3.2.2 Using Finite Field Multiplication

In this subsection, we give constructions of families of functions based on Finite Field Multiplication, which achieve quadratic separation between block sensitivity and sensitivity. We will define two versions, g_{FF} and g^*_{FF} yielding quadratic separations with constants $\frac{1}{4}$ and $\frac{1}{2}$, respectively.

Fix an irreducible polynomial p of degree m in $\mathbb{F}_2[z]$ and consider the representation of the elements of $\mathbb{F}_{2^m}$ as univariate polynomials modulo p. For $a \in \{0,1\}^m$, we interpret $a = (a_0, \ldots a_{m-1})$ as an element of $\mathbb{F}_{2^m}$ under this representation.

Definition 24. Function based on Finite Field Multiplication
The function $g_{FF} \colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ *is defined as follows:*
$g_{FF}(a,b) = 1$ *if and only if* $a \cdot b = c$, *where* $c \in \mathbb{F}_{2^m}$ *is the element represented as* $(0, \ldots, 0, 1)$ *and multiplication is over the field* $\mathbb{F}_{2^m}$.

We prove the following lemma listing the values of sensitivity and block sensitivity for the function g_{FF}.

Lemma 3.8. *For the function* $g_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$*, we have*

- $s_0(g_{FF}) \leq 2$
- $s_1(g_{FF}) = 2m$
- $m+1 \geq bs_0(g_{FF}) \geq m$
- $bs_1(g_{FF}) = 2m$

Proof.

- $s_0(g_{FF}) \leq 2$

  For any non-zero $a \in \mathbb{F}_{2^m}$, there exists a unique $b \in \mathbb{F}_{2^m}$ such that $a \cdot b = (0, \ldots, 0, 1)$ i.e. $g_{FF}(a,b) = 1$. Therefore, for any input $(a,b) \in g_{FF}^{-1}(0)$, at most 1 bit j of a may be flipped to get $a^j \cdot b = (0, 0 \ldots 1)$ i.e. a has at most 1 sensitive bit. Similarly, at most 1 bit of b may be sensitive.

- $s_1(g_{FF}) = 2m$

  Consider any input $(a,b) \in g_{FF}^{-1}(1)$. Flipping any bit of a or b changes the value of the product $a \cdot b$. Therefore every bit of (a,b) is sensitive, giving $s_1(g_{FF}) = 2m$.

- $m+1 \geq bs_0(g_{FF}) \geq m$

  Consider the 0-input $a = (0, \ldots 0), b = (0, \ldots 0)$.

  For each $j \in \{0, \ldots m-1\}$, we can flip the pair of bits (a_j, b_{m-1-j}),

so that their product becomes $c = (0, \ldots, 0, 1)$. This gives m disjoint sensitive blocks.

To see that $m+1 \geq bs_0(g_{FF})$, note that since $s_0(g_{FF}) \leq 2$, on any 0-input there are at most two sensitive blocks of size 1, and all other sensitive blocks must have size at least 2. Thus, $bs_0(g_{FF}) \leq 2 + \frac{2m-2}{2} = m + 1$.

- $bs_1(g_{FF}) = 2m$

  This follows since $2m \geq bs_1(g_{FF}) \geq s_1(g_{FF}) = 2m$

□

The following theorem follows from Lemma 3.8 and the OR-composition Lemma.

Theorem 3.4. *The function $f = OR_m \circ g_{FF}$ has*

$$bs(f) \geq \frac{1}{4} s(f)^2$$

We now modify the function g_{FF} to improve the constant of separation from $\frac{1}{4}$ to $\frac{1}{2}$.

Definition 25. *The function $g^*_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ is defined as follows:*
$g^*_{FF}(a, b) = 1$ if and only if the following two conditions hold.

1. *$a \cdot b = c$, where $c \in \mathbb{F}_{2^m}$ is the element represented as $(0, \ldots, 0, 1)$ and multiplication is over the field $\mathbb{F}_{2^m}$*

2. $a_0 \oplus a_1 \ldots \oplus a_{m-1} = 1$

Lemma 3.9. *For the function* $g^*_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$*, we have*

- $s_0(g^*_{FF}) = 1$
- $s_1(g^*_{FF}) = 2m$
- $bs_0(g^*_{FF}) = m$
- $bs_1(g^*_{FF}) = 2m$

Proof. Note that Conditions 1 and 2 of Definition 25 both have to hold for 1 inputs, and at least one is violated for 0 inputs.

- $s_0(g^*_{FF}) = 1$

  For a 0-input (a, b) which satisfies condition 1, flipping any bit of a or b changes the product $a \cdot b$ and condition 1 is no longer satisfied. Therefore, such a 0-input has no sensitive bit.

  For any 0-input (a, b) which leaves condition 1 unsatisfied, both a and b can have at most one sensitive bit each as observed in the proof of Lemma 3.8.

  We further note that for any given 0-input (a, b), only one of a or b can have a sensitive bit because condition 2 has to hold for 1-inputs. Therefore, $s_0(g^*_{FF}) = 1$.

- $s_1(g^*_{FF}) = 2m$

  Consider any 1-input (a, b). Flipping any bit of a or b changes the value of the product $a \cdot b$ and condition 1 is no longer satisfied. Therefore every bit of (a, b) is sensitive, giving $s_1(g^*_{FF}) = 2m$.

- $bs_0(g^*_{FF}) = m$

  Consider the 0-input $a = (0, \ldots 0), b = (0, \ldots 0)$.

  For each $j \in \{0, \ldots m-1\}$, we can flip the pair of bits (a_j, b_{m-1-j}), so that their product becomes $c = (0, 0 \ldots 1)$ to satisfy the first condition. Since a^j has exactly one 1, the second condition is satisfied as well, and $g^*_{FF}(a^j, b^{m-1-j}) = 1$. This gives m disjoint sensitive blocks and therefore, $bs_0(g^*_{FF}) \geq m$.

  To see that $bs_0(g^*_{FF}) \leq m$, note that since $s_0(g^*_{FF}) = 1$, on any 0-input there is at most one sensitive block of size 1, and all other sensitive blocks must have size at least 2. Thus, $bs_0(g^*_{FF}) \leq \lfloor 1 + \frac{2m-1}{2} \rfloor = m$.

- $bs_1(g^*_{FF}) = 2m$

  This follows since $2m \geq bs_1(g^*_{FF}) \geq s_1(g^*_{FF}) = 2m$

□

Using Lemma 3.9 and the OR-composition Lemma gives the following theorem.

Theorem 3.5. *The function $f = OR_{2m} \circ g^*_{FF}$ has*

$$bs(f) = \frac{1}{2} s(f)^2$$

Remark 3.1. *We could replace $c = (0, \ldots, 0, 1)$ in the above definitions by other field elements and still achieve quadratic separations. In fact using any $c \in \mathbb{F}_{2^m}$, we would get $s_0 \leq 2$ and $s_1 = 2m$ for the inner function. However,*

we need to choose c carefully to guarantee that bs_0 of the inner function is large enough.

3.2.3 Using Polynomial Multiplication

We now describe another family of functions similar in essence to the one involving finite field multiplication, but easier to analyze. We will define two versions, g_{poly} and g^*_{poly} yielding quadratic separations with constants $\frac{1}{2}$ and $\frac{2}{3}$, respectively.

Here we consider polynomials over the Integers. For $a \in \{0,1\}^m$, we interpret the bits of $a = (a_0, \ldots a_{m-1})$ as the coefficients of a univariate polynomial p_a that is $p_a(z) = a_0 + a_1 z + \ldots a_{m-1} z^{m-1}$.

Definition 26. Function based on Polynomial Multiplication
Let $m \geq 2$ be an integer. The function $g_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ is defined as follows:
$g_{poly}(a,b) = 1$ if and only if $p_a(z) \cdot p_b(z)$ has a non-zero coefficient for z^{m-1} and has coefficient 0 for z^j for all $j < m-1$.

It is convenient to use the following equivalent definition.

Definition 27 (Alternative definition)**.**
Let $m \geq 2$ be an integer. Consider the set of partial assignments $C = \{\gamma_0, \gamma_1 \ldots \gamma_{m-1}\}$ where $\gamma_i\colon \{0,1,\ldots,2m-1\} \to \{0,1,*\}$, defined as follows. For every $i \in \{0,\ldots m-1\}$,

$$\gamma_i(j) = \begin{cases} 1, & \text{if } \ j = i \\ 0, & \text{if } j < i \\ \star, & \text{if } m-1 \geq j > i \end{cases} \qquad \gamma_i(j) = \begin{cases} 1, & \text{if } \ j = 2m-1-i \\ 0, & \text{if } m \leq j < 2m-1-i \\ \star, & \text{if } j > 2m-1-i \end{cases}$$

Now the function g_{poly} is the function defined by the set of partial assignments C i.e. g_C.

We now analyze this function for its sensitivity and block sensitivity.

Lemma 3.10. *For g_{poly}: $\{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, we have*

- $s_0(g_{poly}) = 2$
- $s_1(g_{poly}) = m+1$
- $bs_0(g_{poly}) = m$
- $bs_1(g_{poly}) = m+1$

Proof.

- $s_1(g_{poly}) = bs_1(g_{poly}) = m+1$

  First, we observe from the alternative definition of g_{poly} that every 1-input of g_{poly} agrees with a certificate γ_i from the set C. Each $\gamma_i \in C$ fixes $m+1$ bits. Also, note that any two certificates of C are at a distance at least 2 from each other. Therefore, by Lemma 3.4 $s_1(g_{poly}) = bs_1(g_{poly}) = C_1(g_{poly}) = (m+1)$.

- $s_0(g_{poly}) = 2$

  First we prove $s_0(g_{poly}) \leq 2$.

  Observe that if either a or b is an all 0 vector, then the sensitivity on such input (a, b) is at most 1. Let (a, b) be any 0-input of g_{poly} such that neither a nor b is all 0. Let $i \in \{0, \ldots m-1\}$ be the smallest index such that $a_i = 1$ and j be the smallest index such that $b_j = 1$. Then, we have two cases.

  Case 1: $i + j > m - 1$. In this case, in the product $p_a(z) \cdot p_b(z)$ the coefficients are 0 for z^j for all $j \leq m - 1$. Thus, the only bits which can be flipped to change the value of g_{poly} from 0 to 1 are a_{m-1-j} and b_{m-1-i}.

  Case 2: $i + j < m - 1$. Now, the only way to flip a bit and possibly change the value of g_{poly} to 1 is by flipping the bits a_i or b_j.

  Next we argue that $s_0(g_{poly}) \geq 2$.

  The following 0-input (a, b) achieves $s_0(g_{poly}, (a, b)) = 2$.

  Let $a_{m-1} = 1$, $a_i = 0 \; \forall i < (m-1)$.

  Similarly, $b_{m-1} = 1$, $b_i = 0 \; \forall i < (m-1)$.

  Notice that (a, b) has 2 sensitive bits: a_0 and b_0.

- $bs_0(g_{poly}) = m$

  Lemma 3.5 implies that $bs_0(g_{poly}) \leq m$.

  It remains to prove that $bs_0(g_{poly}) \geq m$. Consider the 0-input with $a_i = b_i = 0$ for all $i \in \{0, \ldots m-1\}$. We can flip the pair of bits a_i, b_{m-1-i} for $i \in \{0, \ldots m-1\}$ so that the function changes value from

0 to 1. Therefore, $bs_0(g_{poly}) \geq m$.

□

Lemma 3.10 and the OR-composition Lemma imply the following theorem.

Theorem 3.6. *Consider* $g_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ *for any odd integer* $m > 2$.
Then the function $f = OR_{\frac{m+1}{2}} \circ g_{poly}$ *has*

$$bs(f) = \frac{1}{2}s(f)^2 - \frac{1}{2}s(f)$$

We modify the above function to improve the constant of separation from $\frac{1}{2}$ to $\frac{2}{3}$.

Definition 28. *Let* $m \geq 2$ *be an integer. The function* $g^*_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, *is defined as*
$g^*_{poly}(a,b) = 1$ *if and only if all the following conditions are met.*

1. $p_a(z) \cdot p_b(z)$ *has a non-zero coefficient for* z^{m-1} *and has coefficient 0 for* z^j *for all* $j < m-1$

2. *If* j *is the smallest index such that* $a_j = 1$, *then*
   $a_i = 0$ *for all* i *such that* $i > j$ *and* $i + j$ *is odd*

3. *If k is the smallest index such that $b_k = 1$, then*

   $b_i = 0$ *for all i such that $i > k$ and $i + k$ is even*

It is again helpful to consider an equivalent definition based on certificates.

Definition 29 (Alternative definition)**.**

Let $m \geq 2$ be an integer. Consider the set of partial assignments $C' = \{\gamma'_0, \gamma'_1, \ldots \gamma'_{m-1}\}$, where $\gamma'_i \colon \{0, 1, \ldots, 2m-1\} \to \{0, 1, *\}$ defined as follows. For every $i \in \{0, \ldots m-1\}$,

$$\gamma'_i(j) = \begin{cases} 1, & \text{if } \ j = i \\ 0, & \text{if } j < i \\ 0, & \text{if } m-1 \geq j > i \text{ and } i+j \text{ is odd} \\ \star, & \text{if } m-1 \geq j > i \text{ and } i+j \text{ is even} \end{cases}$$

$$\gamma'_i(j) = \begin{cases} 1, & \text{if } \ j = 2m-1-i \\ 0, & \text{if } m \leq j < 2m-1-i \\ 0, & \text{if } j > 2m-1-i \text{ and } (2m-1-i)+j \ \text{ is even} \\ \star, & \text{if } j > 2m-1-i \text{ and } (2m-1-i)+j \text{ is odd} \end{cases}$$

Now the function g^*_{poly} is the function defined by the set of partial assignments C' i.e. $g_{C'}$.

Lemma 3.11. *Consider $g^*_{poly} \colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ for any odd integer $m > 2$. Then*

- $s_0(g^*_{poly}) = 1$
- $s_1(g^*_{poly}) = \frac{3m+1}{2}$
- $bs_0(g^*_{poly}) = m$
- $bs_1(g^*_{poly}) = \frac{3m+1}{2}$

Proof. It is clear from the alternative definition of g^*_{poly} that it is defined by a set of good partial assignments as introduced in definition 23. The result then follows from Lemma 3.6. □

The following theorem follows from Lemma 3.11 and the OR-composition Lemma.

Theorem 3.7. *Consider* $g^*_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ *for any odd integer* $m > 2$.

Then the function $f = OR_{\frac{3m+1}{2}} \circ g^*_{poly}$ *has*

$$bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f)$$

Note that this bound matches the current best quadratic separation of Ambainis and Sun [10].

3.3 Additional Properties of Function Composition

As we noted in Section 3.1 of this chapter, properties of function composition have been formally studied by Tal and Gilmer et al. [92, 44] in the context of separating sensitivity and block sensitivity. Here we take a closer look at the effect of function composition on the measures 0-sensitivity, 1-sensitivity, 0-block sensitivity and 1-block sensitivity. These properties provide the tools we need to obtain quadratic separation of both 0-block sensitivity and 1-block sensitivity from sensitivity.

First we define measures to quantify the number of sensitive bits for f on x which are equal to 0 and those that are equal to 1 in x.

Definition 30. *For a function $f\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$, we define*

$\sigma_1(f,x) = |\{i | x_i = 1 \;\; AND \;\; f(x) \neq f(x^i)\}|$

$\sigma_0(f,x) = |\{i | x_i = 0 \;\; AND \;\; f(x) \neq f(x^i)\}|$.

We will use the following notation. We index the bits of the input $y \in \{0,1\}^{mn}$ to $f \circ g$ as $y = (y_{11}, y_{12}, \ldots y_{1m}, y_{21}, \ldots y_{2m}, \ldots y_{n1}, \ldots y_{nm})$. We denote by y_i the i-th group of m bits of y, that is $y_i = (y_{i1}, y_{i2}, \ldots y_{im})$.

Lemma 3.12. *For any functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$, we have*

$$s_0(f \circ g) = \max_{x \in f^{-1}(0)} \{\sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g)\}$$

$$s_1(f \circ g) = \max_{x \in f^{-1}(1)} \{\sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g)\}$$

Proof. We prove the first equation. The second equation has an analogous proof.

First we prove $s_0(f \circ g) \geq \max_{x \in f^{-1}(0)} \{\sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g)\}$. Consider the input $a \in f^{-1}(0)$ for which $(\sigma_0(f,a)s_0(g) + \sigma_1(f,a)s_1(g))$ is maximized. Note that $s_0(g), s_1(g)$ don't change for different choices of $a \in f^{-1}(0)$. Now, consider an input $y \in \{0,1\}^{mn}$ such that, $a = (g(y_1), \ldots g(y_n))$ and for each $i \in [n]$, if $a_i = g(y_i) = 0$, then $s(g, y_i) = s_0(g)$ and if $a_i = g(y_i) = 1$, then $s(g, y_i) = s_1(g)$. So if $a_i = 0$, we choose as y_i a 0-input of g which achieves the

0-sensitivity of g, and similarly, if $a_i = 1$, we choose as y_i a 1-input of g which achieves the 1-sensitivity of g.

Since $a \in f^{-1}(0)$, y must be a 0-input of $f \circ g$. Therefore, we have

$$s_0(f \circ g) \geq s(f \circ g, y) \geq \sigma_0(f, a)s_0(g) + \sigma_1(f, a)s_1(g)$$

Next we prove $s_0(f \circ g) \leq \max_{x \in f^{-1}(0)} \{\sigma_0(f, x)s_0(g) + \sigma_1(f, x)s_1(g)\}$. Consider an input $y \in \{0, 1\}^{mn}$ which achieves the 0-sensitivity of $f \circ g$ i.e. $s_0(f \circ g) = s(f \circ g, y)$. Let $g(y_1) = x_1$, $g(y_2) = x_2$ and so on, and let $x = (x_1, x_2 \ldots x_n)$.

Consider the expression $(\sigma_0(f, x)s_0(g) + \sigma_1(f, x)s_1(g))$. Now, if a bit y_{ij} of y is sensitive for $f \circ g$, then the bit $x_i = g(y_i)$ must be a sensitive bit for f on x. Now, consider the set $\mathcal{X}_0$ of indices $i \in [n]$ constructed the following way: i is included in $\mathcal{X}_0$ if and only if $x_i = g(y_i) = 0$ and there is a bit y_{ij} sensitive for $f \circ g$ on y.

Similarly, we define the set $\mathcal{X}_1$ of indices $i \in [n]$ constructed the following way: i is included in $\mathcal{X}_1$ if and only if $x_i = g(y_i) = 1$ and there is a bit y_{ij} sensitive for $f \circ g$ on y.

Note that for every $i \in \mathcal{X}_0$, the bit x_i is a 0-bit of x and f is sensitive to the i-th bit on x. So $|\mathcal{X}_0| \leq \sigma_0(f, x)$.

Similarly $|\mathcal{X}_1| \leq \sigma_1(f, x)$.

Now,

$$\begin{aligned} s(f \circ g, y) &= \sum_{i \in \mathcal{X}_0} s(g, y_i) + \sum_{j \in \mathcal{X}_1} s(g, y_j) \\ &\leq \sum_{i \in \mathcal{X}_0} s_0(g) + \sum_{j \in \mathcal{X}_1} s_1(g) \\ &\leq \sigma_0(f, x) s_0(g) + \sigma_1(f, x) s_1(g) \end{aligned}$$

Therefore,

$$s_0(f \circ g) = s(f \circ g, y) \leq \sigma_0(f, x) s_0(g) + \sigma_1(f, x) s_1(g)$$

Since $x \in f^{-1}(0)$, this concludes the proof. □

To simplify the equations of Lemma 3.12 (at the cost of being less precise), we define

- $\sigma_0^0(f) \coloneqq \max_{x \in f^{-1}(0)} \sigma_0(f, x)$
- $\sigma_1^0(f) \coloneqq \max_{x \in f^{-1}(0)} \sigma_1(f, x)$
- $\sigma_0^1(f) \coloneqq \max_{x \in f^{-1}(1)} \sigma_0(f, x)$
- $\sigma_1^1(f) \coloneqq \max_{x \in f^{-1}(1)} \sigma_1(f, x)$

Finally, we define

$$\sigma_0(f) \coloneqq \max\{\sigma_0^0(f), \sigma_0^1(f)\}$$

$$\sigma_1(f) \coloneqq \max\{\sigma_1^0(f), \sigma_1^1(f)\}$$

We can now use Lemma 3.12 to get the following bounds.

Corollary 3.1. *For any functions* $f\colon \{0,1\}^n \to \{0,1\}$ *and* $g\colon \{0,1\}^m \to \{0,1\}$

$$s_0(f \circ g) \leq \sigma_0^0(f)s_0(g) + \sigma_1^0(f)s_1(g)$$

$$s_1(f \circ g) \leq \sigma_0^1(f)s_0(g) + \sigma_1^1(f)s_1(g)$$

Note that the equalities in Lemma 3.12 change to inequalities in Corollary 3.1, since the max of $\sigma_0(f, x)$ and $\sigma_1(f, x)$ may be achieved on different inputs among $x \in f^{-1}(0)$ (or among $x \in f^{-1}(1)$, respectively).
We now state some simple observations for these measures.

Lemma 3.13. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, and any input x, we have*

1. $s(f, x) = \sigma_0(f, x) + \sigma_1(f, x)$
2. $\sigma_0^0(f) \leq s_0(f)$
3. $\sigma_1^0(f) \leq s_0(f)$
4. $\sigma_0^0(f) + \sigma_1^0(f) \geq s_0(f)$
5. $\sigma_0^1(f) \leq s_1(f)$
6. $\sigma_1^1(f) \leq s_1(f)$
7. $\sigma_0^1(f) + \sigma_1^1(f) \geq s_1(f)$

The proof of Lemma 3.13 is straightforward from the definitions.

Now we present an observation about these measures for monotone functions.

Lemma 3.14. *For any monotone function $f\colon \{0,1\}^n \to \{0,1\}$, we have*

- $\sigma_0^1(f) = 0$

- $\sigma_1^0(f) = 0$

The proof follows from the definition of monotone functions.

We now consider the effects of function composition on 0- block sensitivity and 1-block sensitivity.

Lemma 3.15. *For any functions* $f\colon \{0,1\}^n \to \{0,1\}$ *and* $g\colon \{0,1\}^m \to \{0,1\}$, *we have*

$$bs_0(f \circ g) \geq bs_0(f) \cdot \min\{bs_0(g), bs_1(g)\}$$

$$bs_1(f \circ g) \geq bs_1(f) \cdot \min\{bs_0(g), bs_1(g)\}$$

Proof. Consider $x \in \{0,1\}^n$ such that $f(x) = 0$ and $bs_0(f) = bs(f,x)$.
Now, consider input $y \in \{0,1\}^{mn}$ such that, $x = (g(y_1), \ldots g(y_n))$ and for each $i \in [n]$, if $x_i = g(y_i) = 0$, then $bs(g, y_i) = bs_0(g)$ and if $x_i = g(y_i) = 1$, then $bs(g, y_i) = bs_1(g)$. So if $x_i = 0$, we choose as y_i a 0-input of g on which its 0-block sensitivity is achieved, and similarly, if $x_i = 1$, we choose as y_i a 1-input of g on which its 1-block sensitivity is achieved.
Now, we claim that $bs_0(f \circ g, y) \geq bs_0(f) \min\{bs_0(g), bs_1(g)\}$. To see this, let $\rho_1, \rho_2, \ldots, \rho_k$ be the disjoint sensitive blocks for f on x where $k = bs(f,x)$. For each of these sensitive blocks, there are at least $\min\{bs_0(g), bs_1(g)\}$ disjoint blocks of y such that flipping any of them changes the value of $f \circ g$. This gives at least $bs_0(f) \cdot \min\{bs_0(g), bs_1(g)\}$ disjoint sensitive blocks for $f \circ g$ on the input y, and the first equation follows.
The second equation can be proved in an analogous way. □

We get a stronger form of Lemma 3.15 if f satisfies some additional conditions.

Lemma 3.16. *For any functions* $f\colon \{0,1\}^n \to \{0,1\}$ *and* $g\colon \{0,1\}^m \to \{0,1\}$ *if* f *satisfies (1)* $f(0^n) = 0$*, (2)* $bs_0(f) = bs(f, 0^n)$*, then* $bs_0(f \circ g) \geq bs_0(f) \cdot bs_0(g)$
and if f *satisfies (1)* $f(1^n) = 1$*, (2)* $bs_1(f) = bs(f, 1^n)$*, then* $bs_1(f \circ g) \geq bs_1(f) \cdot bs_1(g)$

Proof. Consider input $y \in \{0,1\}^{mn}$ such that, $0^n = (g(y_1), \ldots g(y_n))$ and $bs(g, y_i) = bs_0(g)$ for each $i \in [n]$.
Now, we claim that $bs(f \circ g, y) \geq bs_0(f)bs_0(g)$. To see this, let $\rho_1, \rho_2, \ldots, \rho_k$ be the disjoint sensitive blocks for f on input 0^n, where $k = bs(f, 0^n)$. For each of these $k = bs(f, 0^n)$ sensitive blocks, there are $bs_0(g)$ disjoint blocks of y that we can flip and change the value of $f \circ g$.
This gives $bs_0(f) \cdot bs_0(g)$ disjoint sensitive blocks for $f \circ g$ on the input y.
The second inequality can be proved analogously. □

Comparing the statement of Lemma 3.16 with Lemma 3.2 of Tal [92] we note that in the context of 0-block sensitivity and 1-block sensitivity it is enough to require an additional condition for the outer function. On the other hand the condition on the inner function in Lemma 3.2 of Tal [92] is necessary as illustrated by considering $f = OR_n$ and $g = AND_n$.

Note that the conditions we require are similar to, but slightly different from Definition 22 by Tal [92] of being in z-good form. It follows from Defi-

nition 22, that if f is in z-good form, then $bs(f) = bs_z(f)$. Our conditions do not require that $bs(f) = bs_z(f)$ for a specific z.

3.4 Quadratic Separation of both $bs_0(f)$ and $bs_1(f)$ from $s(f)$

We obtain constructions of functions with quadratic separation of both 0-block sensitivity and 1-block sensitivity from sensitivity by considering various compositions of our new building blocks as well as some of the inner functions used in previous quadratic separations.

Theorem 3.8. *Consider* $g_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$.
Let $f\colon \{0,1\}^{4m^2} \to \{0,1\}$ *be defined as* $f = g_{poly} \circ g_{poly}$.
Then, we have

- $s_0(f) = 2(m+1)$
- $s_1(f) = 4m$
- $bs_0(f) \geq m^2$
- $bs_1(f) \geq m(m+1)$

Therefore, we have

$$\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^2)$$

Proof. In this proof, we refer to g_{poly} by g, and we use the notation $bs_{min}(f) = \min\{bs_0(f), bs_1(f)\}$.

We first prove the following claims about σ-values for g.

Claim 3.2. *For any input $x \in g^{-1}(0)$ exactly one of the following must be true:*

- $\sigma_0(g,x) = s(g,x)$ *and* $\sigma_1(g,x) = 0$
- $\sigma_1(g,x) = s(g,x)$ *and* $\sigma_0(g,x) = 0$

Proof of Claim:

For any 0-input $x = (a,b)$ of g, (1) of Lemma 3.13 states that
$\sigma_0(g,x) + \sigma_1(g,x) = s(g,x)$.
As in the definition of g_{poly}, consider the polynomials $p_a(z), p_b(z)$. If the lowest degree monomial of $p_a(z)p_b(z)$ with a non-zero coefficient is z^t then, we have 2 cases:
Case 1: $t < m-1$. In this case, no 0-bit of a or b can be sensitive. Therefore, $\sigma_0(g,x) = 0$ and $\sigma_1(g,x) = s(g,x)$.
Case 2: $t > m-1$. In this case, no 1-bit of a or b can be sensitive. Therefore, $\sigma_1(g,x) = 0$ and $\sigma_0(g,x) = s(g,x)$. ■

Claim 3.3. *For an input $x \in g^{-1}(1)$,*

- $\sigma_0(g,x) = m-1$
- $\sigma_1(g,x) = 2$

Proof of Claim:

Recall the alternative definition based on certificates. Any 1-input x of g belongs to a unique subcube given by a certificate $\gamma_i \in C$. Since the subcubes corresponding to different certificates in C are disjoint and at distance at least

2 from each other, every bit of x that is fixed by γ_i is sensitive.

Since each certificate fixes exactly 2 bits to 1 and $(m-1)$ bits to 0, we have $\sigma_0(g,x) = (m-1)$ and $\sigma_1(g,x) = 2$. ■

We can now use Lemma 3.12 to compute the sensitivity of f:

$s_0(f) = 2s_1(g) = 2(m+1)$.

$s_1(f) = (m-1)\cdot 2 + 2\cdot(m+1) = 4m$

Since $g(0^{2m}) = 0$ and $bs_0(g) = bs(g, 0^{2m})$, we can use Lemma 3.16 to get $bs_0(f) \geq bs_0(g)^2 = m^2$.

We can use Lemma 3.15 to get

$bs_1(f) \geq bs_1(g)\cdot \min\{bs_0(g), bs_1(g)\} = m(m+1)$.

Therefore, we have $bs(f) \geq \frac{s(f)^2}{16}$ and $bs_{min}(f) = \Omega(s(f)^2)$. □

We prove the following general theorem.

Theorem 3.9. *For functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^n \to \{0,1\}$ such that the following conditions hold:*

1. *$\sigma_1(f) = c_1$, where c_1 is some fixed constant*
2. *$s_0(g) = c_2$, where c_2 is some fixed constant*
3. *$bs_0(f), bs_1(f), bs_0(g), bs_1(g) = \theta(n)$*

We have,

$$\min\{bs_0(f\circ g), bs_1(f\circ g)\} = \Omega(s(f\circ g)^2)$$

The proof is straightforward from Corollary 3.1 and Lemma 3.15.

This theorem allows us to use various compositions of our new building blocks and some of the inner functions of previous constructions to obtain other functions with both 0-block sensitivity and 1-block sensitivity quadratically larger than sensitivity.

In particular, let f, g be any two functions from the following list of functions: Rubinstein's inner function [85], Virza's inner function [97], Ambainis and Sun's inner function [10], g_{poly}, g^*_{poly}. In addition, we can also let g be g_{FF}, g^*_{FF}, or the inner function of the function based on Hamming Code [47]. Then, $bs_0(f \circ g)$ and $bs_1(f \circ g)$ are both quadratically larger than $s(f \circ g)$.

Chapter 4

A new complexity measure: diameter of Boolean functions

In this chapter [1], we define our new complexity measure called diameter and some variants of it, as promised in the introduction. We prove upper and lower bounds on these different diameters in terms of certificate complexity and other measures. These bounds show that analyzing diameters may help to obtain better bounds on certificate complexity and other measures as described in Section 1.3.2.5.

We then proceed to prove some implications for functions with small diameters. In particular, we prove that $s(f) \geq \sqrt{n}$ for transitive functions with constant alternating number, as well as for transitive functions with constant diameter, implying $bs(f) \leq s(f)^2$ for such functions. We also show that $bs(f) \geq \Omega(n^{3/7})$ for transitive functions under the weaker condition that the minimum diameter is constant. This answers Question 5 mentioned in Chapter 1 for these functions.

Furthermore, we prove that the log-rank conjecture holds for functions of the form $f(x \oplus y)$ for functions f with diameter bounded above by a poly-

[1] The results presented in this chapter appear in [31].

nomial of the logarithm of the Fourier sparsity of the function f.

4.1 Preliminaries

We now state some additional definitions and results that we shall need for this chapter.

We restate the definition of partial assignment to emphasize the notation we use in this chapter for the associated subcube and subfunction.

Definition 31. Partial assignment, subcube and subfunction
Given an integer $n > 0$, a partial assignment α is a function $\alpha\colon [n] \to \{0,1,\star\}$. A partial assignment α corresponds naturally to a setting of n variables $(x_1, x_2, \ldots, x_n)$ to $\{0,1,\star\}$ where x_i is set to $\alpha(i)$.

The variables set to $\star$ are called unassigned or free, and we say that the variables set to 0 or 1 are fixed. We say that $x \in \{0,1\}^n$ agrees with α if $x_i = \alpha(i)$ for all i such that $\alpha(i) \neq \star$. The set of all inputs $x \in \{0,1\}^n$ agreeing with α constitutes a subcube which we denote by $\mathcal{S}_\alpha$.

The size of a partial assignment α is defined as the number of fixed variables of α and denoted as $|\alpha|$.

For a function $f\colon \{0,1\}^n \to \{0,1\}$, we denote by f_α the subfunction obtained by restricting f to the subcube $\mathcal{S}_\alpha$.

We use the following notation. For any two inputs $x, y \in \{0,1\}^n$, we say $x \prec y$ if $x_i \leq y_i$ for all $i \in [n]$.

Definition 32. ***Alternating path***

For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$*, an alternating path is defined as any path* $\mathcal{Q} = x^{(0)}, x^{(1)}, x^{(2)}, \ldots x^{(t)}$ *that satisfies the following properties:*

- $x^{(0)} = 0^n$
- $x^{(i)} \prec x^{(i+1)}$ *for all* $i \in \{0, 1, \ldots, t-1\}$.
- $f(x^{(i)}) \neq f(x^{(i+1)})$ *for all* $i \in \{0, 1, \ldots, t-1\}$

Definition 33. ***Alternating Number of a function***

For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$*, the alternating number of* f*,* $alt(f)$*, is defined as the maximum length of any alternating path of* f*.*

We now state a result from the paper of Lin and Zhang [61] that we shall require:

Lemma 4.1 (Lemma 12 from [61])**.** *For any function* $f\colon \{0,1\}^n \to \{0,1\}$*, the following two statements hold:*

$$\max\{C(f, 0^n), C(f, 1^n)\} \leq alt(f) \cdot s(f)$$

$$\max\{C(f, 0^n), C(f, 1^n)\} \leq alt(f) \cdot deg_2(f)$$

Definition 34. Fourier sparsity of a function

Any function $f\colon \{0,1\}^n \to \{0,1\}$ *can be uniquely represented as*

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)(-1)^{\sum_{i \in S} x_i}$$

This is said to be the Fourier expansion of f and the coefficients $\hat{f}(S)$ are called the Fourier coefficients of f. The number of non-zero Fourier coefficients of f is defined to be the Fourier sparsity of f, denoted $||\hat{f}||_0$.

We note that usually the Fourier representation is considered for functions of the form $f\colon \{0,1\}^n \to \{+1,-1\}$. We use this version of the definition because it exactly captures the rank of the communication matrix for XOR functions, as we shall see later.

4.1.1 Communication Complexity

We consider a setting with two parties Alice and Bob and a fixed Boolean function $f\colon \{0,1\}^{2n} \to \{0,1\}$. For $x, y \in \{0,1\}^n$, input x is provided to Alice and input y to Bob. Their collective objective is to compute $f(x,y)$.

The communication complexity of f, denoted $CC(f)$, is the maximum value of the minimum number of bits exchanged by Alice and Bob in order to compute $f(x,y)$, where the maximum is taken over all input pairs $(x,y) \in \{0,1\}^{2n}$.

The communication matrix corresponding to f, denoted M_f, is a $2^n \times 2^n$ matrix with rows indexed by all possible values of $x \in \{0,1\}^n$ i.e. Alice's part of the input and columns indexed by all possible values of $y \in \{0,1\}^n$ i.e. Bob's part of the input. For each row x and column y, the corresponding entry of M_f contains $f(x,y)$. It can be shown that $CC(f) \geq \log rank(M_f)$ [67], where the rank is over the reals.

The log-rank conjecture proposed by Lovász and Saks [64] asks whether the communication complexity of a function can also be upper bounded by a polynomial in logarithm of the rank of its communication matrix as:

Conjecture 1. *For any function* $f\colon \{0,1\}^{2n} \to \{0,1\}$,

$$CC(f) \leq poly(\log rank(M_f))$$

For any $f\colon \{0,1\}^n \to \{0,1\}$, the corresponding XOR-composed function $f \circ \oplus\colon \{0,1\}^{2n} \to \{0,1\}$ is defined as:

$f \circ \oplus(x,y) = f(x \oplus y)$, where $x \oplus y$ is the bitwise XOR of $x, y \in \{0,1\}^n$.

4.2 Diameters of Boolean functions

We begin with some notation. For the Boolean cube $\mathcal{B}_n$, a path is any sequence of inputs $x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(t)}$ where $x^{(i)} \in \{0,1\}^n$ for $i \in \{0,1,\ldots,t\}$. We define the length of such a path to be the number of steps t.

For a path $x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, we define a sequence of partial assignments $\{\beta^{(0)}, \beta^{(1)}, \ldots \beta^{(t-1)}\}$ where $\beta^{(i)}\colon [n] \to \{0,1,\star\}$ is defined as follows:

$\beta^{(i)}_j = x^{(i)}_j$ for all $j \in [n]$ such that $x^{(i)}_j = x^{(i+1)}_j$ and $\beta^{(i)}_j = \star$ otherwise.

Note that we can view each step $x^{(i)} \to x^{(i+1)}$ on a path as flipping the bits where $x^{(i)}$ and $x^{(i+1)}$ differ. The free variables of the partial assignment $\beta^{(i)}$ are exactly these bits. Thus, for a Boolean function f, the subfunction $f_{\beta^{(i)}}$ depends on the bits where $x^{(i)}$ and $x^{(i+1)}$ differ.

Definition 35. $\mathcal{H}$***-distance between a point and a partial assignment***
Let $\mathcal{H}$ be a class of Boolean functions. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, input $x \in \{0,1\}^n$ and a partial assignment $\alpha\colon [n] \to \{0,1,\star\}$ corresponding to a maximal constant subcube, we define an $\mathcal{H}$-path from x to α as any path $x^{(0)}, x^{(1)}, \ldots x^{(t)}$ that satisfies the following properties:

- $x^{(0)} = x$
- $x^{(t)}$ *agrees with* α
- *The subfunction $f_{\beta^{(i)}}$ belongs to class $\mathcal{H}$ for $i \in \{0,1,\ldots,t-1\}$.*

We define the $\mathcal{H}$-distance between x and α with respect to f, denoted $dist_{f,\mathcal{H}}(x,\alpha)$, to be the length of a shortest $\mathcal{H}$-path from x to α.

Definition 36. $\mathcal{H}$***-diameter of a function***
For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, input $x \in \{0,1\}^n$, and a class of Boolean functions $\mathcal{H}$, we use the notation

$$dia_{\mathcal{H}}(f,x) = \min_{\alpha \in \Gamma_f(x)} dist_{f,\mathcal{H}}(\bar{x},\ \alpha)\ .$$

Recall that $\bar{x}$ denotes the complement of x and $\Gamma_f(x)$ denotes the set of all certificates of f on x.

We define the $\mathcal{H}$-diameter of f as:

$$dia_{\mathcal{H}}(f) = \max_{x \in \{0,1\}^n} dia_{\mathcal{H}}(f,x)\ .$$

In this work, we will be concerned with the $\mathcal{H}$-diameter of functions for the following classes $\mathcal{H}$:

- AND-OR diameter denoted $dia_{\text{and}}(f)$: Corresponds to the class $\mathcal{H}$ that includes the functions AND, OR and their respective negations i.e. the NAND and the NOR functions.
- Sensitivity diameter denoted $dia_s(f)$: Defined by the class $\mathcal{H}$ with functions that have sensitivity equal to the number of input variables.
- Real degree diameter denoted as $dia_{deg}(f)$: Corresponding to the class of functions $\mathcal{H}$ with real degree equal to the number of input variables.
- $\mathbb{F}_2$-degree diameter denoted as $dia_{deg_2}(f)$: Defined by the class $\mathcal{H}$ that include functions with $\mathbb{F}_2$-degree equal to the number of variables.

Note that since the functions AND, OR, NAND and NOR belong to each of the classes we consider in this work, and since $deg_2(f) \leq deg(f)$ for any Boolean function f, we have

$$dia_s(f) \leq dia_{and}(f) \tag{4.1}$$

and

$$dia_{deg}(f) \leq dia_{deg_2}(f) \leq dia_{and}(f) \tag{4.2}$$

Finally, similarly to minimum certificate complexity, we also define the minimum version of the diameter:

Definition 37. ***Min $\mathcal{H}$-diameter of a function***

For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and a class of Boolean functions $\mathcal{H}$, we define the min $\mathcal{H}$-diameter of f as:

$$dia_{min,\mathcal{H}}(f) = \min_{x\in\{0,1\}^n} dia_{\mathcal{H}}(f,x)\ .$$

Note that for any function f and class $\mathcal{H}$, we have $dia_{min,\mathcal{H}}(f) \leq dia_{\mathcal{H}}(f)$.

4.2.1 Upper bounds on diameters

Lemma 4.2. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$dia_{and}(f) \leq 2C(f)$$

Proof. For any input $x \in \{0,1\}^n$, we shall prove that $dia_{\text{and}}(f,x) \leq 2C(f,x)$.

Let α be a certificate of f on x achieving $|\alpha| = C(f,x)$. We shall construct an AND-OR path $\mathcal{Q} = x^{(0)}, x^{(1)}, \ldots x^{(t)}$ from $\bar{x}$ to α, with length $t \leq 2|\alpha|$.

We start with $x^{(0)} = \bar{x}$ as required. We now give an inductive description of our construction. Let us assume that we have found the first $i+1$ vertices of the path i.e. $x^{(0)}, x^{(1)}, \ldots x^{(i)}$. Then we get the next vertex $x^{(i+1)}$ in the following way based on the value of $f(x^{(i)})$:

1. Case 1: $f(x^{(i)}) = f(x)$

   If $x^{(i)}$ agrees with α, then we have successfully found the required $\mathcal{H}$-path from $\bar{x}$ to α and we can stop.

Otherwise, if $x^{(i)}$ does not agree with α, then we choose $x^{(i+1)} = (x^{(i)})^S$ where S is any minimal sensitive block of f on $x^{(i)}$ such that S does not contain any index where $x^{(i)}$ and α agree. Note that such a block S must exist because otherwise, the set of bits where $x^{(i)}$ and α agree would be a certificate of f, which would contradict the minimality of the certificate α.

We note that in this case, $x^{(i+1)}$ agrees with α on at least as many bits as $x^{(i)}$ agrees with α.

2. Case 2: $f(x^{(i)}) \neq f(x)$

   In this case, we first observe that the set T of all the bits where $x^{(i)}$ and α disagree constitutes a sensitive block for f on $x^{(i)}$. Therefore, there exists a subset $S \subset T$ which is a minimal sensitive block of f on $x^{(i)}$. We then choose $x^{(i+1)} = (x^{(i)})^S$.

   Note that the number of bits where $x^{(i+1)}$ agrees with α is strictly greater than the number of bits where $x^{(i)}$ agrees with α.

First we note that since each step consists of flipping a minimal sensitive block, each of the subfunctions $f_{\beta^{(i)}}$ is either AND or NAND.

Further, since an AND-OR path is also an alternating path, the value of $f(x^{(i)})$ alternates between 0 and 1. Therefore, the above described procedure to construct the AND-OR path alternates between case 1 and case 2.

Also, as noted before, the number of bits where $x^{(i+1)}$ agrees with α is strictly greater than the number of bits where $x^{(i)}$ agrees with α in case 2,

whereas in case 1, we can guarantee that this number does not decrease. Since the procedure alternates between the two cases, $x^{(i+2)}$ must agree with α on at least one more bit than $x^{(i)}$, for $i \in \{0, 1, \ldots t-2\}$. Therefore, the procedure must terminate in at most $2|\alpha|$ steps, implying that $t \leq 2|\alpha|$.

□

Next, note that since each of our measures is upper bounded by certificate complexity (as we proved above), known upper bounds on certificate complexity imply that for each class $\mathcal{H}$ considered in this work, we have

$$dia_{\mathcal{H}} \leq O(s(f)^5)$$

(using Huang's result [51]) and

$$dia_{\mathcal{H}} \leq O(deg(f)^3)$$

by [68].

Improving these upper bounds would have interesting consequences, as we described in Section 1.3.2.5.

4.2.2 Upper bounds on certificate complexity in terms of diameters

Lemma 4.3. *For any function $f\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$, we have:*

$$C(f,x) \leq dia_s(f,x) \cdot s(f) \leq dia_{and}(f,x) \cdot s(f) \tag{4.3}$$

$$C(f,x) \leq dia_{deg}(f,x) deg(f) \leq dia_{and}(f,x) \cdot deg(f) \tag{4.4}$$

$$C(f,x) \leq dia_{deg_2}(f,x) deg_2(f) \leq dia_{and}(f,x) \cdot deg_2(f) \tag{4.5}$$

Proof. We shall first prove inequality 4.3.

Let α be a certificate of f on x and $\mathcal{Q} = x^{(0)}, x^{(1)}, \ldots, x^{(t)}$ be a corresponding full sensitivity path from $\bar{x}$ to α that achieves the minimum value of $dist_{f,s}(\bar{x}, \alpha)$.

Then note that every fixed bit of the certificate α must be contained in $\beta^{(i)}$ for some $i \in \{0, 1, \ldots, t-1\}$ since $x^{(t)}$ agrees with α and $x^{(0)}$ (i.e. $\bar{x}$) disagrees with all the fixed bits of α.

Therefore, $|\beta^{(0)}| + |\beta^{(1)}| + \ldots + |\beta^{(t-1)}| \geq |\alpha|$. So there must exist an $i \in \{0, 1, \ldots, t-1\}$ such that $|\beta^{(i)}| \geq \frac{|\alpha|}{t}$. Now consider the subfunction $f_{\beta^{(i)}}$. Since we considered a full sensitivity path, this subfunction has sensitivity $|\beta^{(i)}|$. Therefore, $s(f) \geq |\beta^{(i)}| \geq \frac{|\alpha|}{t}$.

This implies that $s(f) \cdot dist_{f,s}(\bar{x}, \alpha) \geq |\alpha|$.

Taking the minimum over all the certificates for f on x gives the first part of inequality 4.3. The second part follows due to inequality 4.1. The other two inequalities follow by an analogous argument. □

Lemma 4.3 immediately implies the following two theorems.

Theorem 4.1. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

1. $C(f) \leq dia_s(f) \cdot s(f) \leq dia_{and}(f) \cdot s(f)$
2. $C(f) \leq dia_{deg}(f) \cdot deg(f) \leq dia_{and}(f) \cdot deg(f)$
3. $C(f) \leq dia_{deg_2}(f) \cdot deg_2(f) \leq dia_{and}(f) \cdot deg_2(f)$

Proof. Let x be the input achieving $C(f,x) = C(f)$. Then, equation 4.3 gives:

$$\begin{aligned} C(f) &= C(f,x) \\ &\leq dia_s(f,x) \cdot s(f) \\ &\leq dia_s(f) \cdot s(f). \end{aligned}$$

This gives the first part of the first statement of the theorem, the second part follows by equation 4.1. The other statements follow similarly from Lemma 4.3 and equation 4.2. □

Theorem 4.2. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

1. $C_{min}(f) \leq dia_{min,s}(f) \cdot s(f) \leq dia_{min,and}(f) \cdot s(f)$
2. $C_{min}(f) \leq dia_{min,deg}(f) \cdot deg(f) \leq dia_{min,and}(f) \cdot deg(f)$
3. $C_{min}(f) \leq dia_{min,deg_2}(f) \cdot deg_2(f) \leq dia_{min,and}(f) \cdot deg_2(f)$

Proof. Let x be the input for which the minimum value of $dia_s(f,x)$ is achieved. Then, equation 4.3 gives:

$$\begin{aligned} C_{min}(f) &\leq C(f,x) \\ &\leq dia_s(f,x) \cdot s(f) \\ &= dia_{min,s}(f) \cdot s(f). \end{aligned}$$

The first part of the first statement of the theorem follows. A similar argument with the second part of equation 4.3 implies the second part of the first statement.

The other statements of the theorem follow from similar arguments using equations 4.4 and 4.5.

□

4.3 Results for families of functions with small diameters

4.3.1 Transitive functions with small diameters

In this section, we improve the lower bounds on sensitivity of transitive functions with constant alternating number that follow due to the work of Lin and Zhang [61] and then also prove a similar result for transitive functions with constant AND-OR diameter. We then proceed to prove lower bounds on block sensitivity of transitive functions with constant min AND-OR diameter.

Lemma 4.4. *For any non-constant transitive function* $f\colon \{0,1\}^n \to \{0,1\}$ *the following two statements hold:*

$$s(f)^2 \cdot alt(f) \geq n$$

$$s(f) \cdot deg_2(f) \cdot alt(f) \geq n$$

Proof. Recall from Chapter 2 that Lemma 2.9 gives:

$$C(f, 0^n)s(f) \geq n.$$

Further, first part of Lemma 4.1 gives that:

$$C(f, 0^n) \leq alt(f) \cdot s(f).$$

Together they imply the first part of the lemma.

Using a similar argument with second part of Lemma 4.1 gives the second part of the result. □

We notice that the first part of Lemma 4.4 is tight for the TRIBES function on n variables: TRIBES is monotone and therefore has $alt(f) = 1$. Also, TRIBES is transitive as noted in [78]. It is easy to see that TRIBES has $s(f) = \sqrt{n}$.

Note that Lemma 4.4 implies $s(f) \geq \sqrt{n}$ for transitive functions with constant alternating number, giving the best possible bound for such functions.

Next, we prove that this bound of $s(f) \geq \sqrt{n}$ also holds for transitive functions with constant diameter, considering dia_s or dia_{and}. This is implied by the following lemma:

Lemma 4.5. *For any non-constant transitive function $f\colon \{0,1\}^n \to \{0,1\}$ the following two statements hold:*

$$s(f)^2 \cdot dia_{and}(f) \geq n$$

$$s(f) \cdot deg_2(f) \cdot dia_{and}(f) \geq n$$

Proof. Recall that Lemma 2.9 from Chapter 2 gives that:

$$C(f, 0^n)s(f) \geq n$$

Further, equation 4.3 gives that:

$$C(f, 0^n) \leq dia_{and}(f, 0^n) \cdot s(f)$$

Therefore, we get:

$$dia_{and}(f, 0^n) \cdot s(f)^2 \geq n$$

which implies the first part of the lemma.

Using a similar argument with equation 4.5 gives the second part of the lemma. □

Corollary 4.1. *For any non-constant transitive function* $f\colon \{0,1\}^n \to \{0,1\}$ *with constant alternating number or constant AND-OR diameter* $dia_{and}(f) = O(1)$*, we have:*

$$s(f) \geq \Omega(\sqrt{n}).$$

We now prove a lower bound on the block sensitivity of transitive functions under the weaker condition of having constant minimum AND-OR diameter.

Lemma 4.6. *For any non-constant transitive function* $f\colon \{0,1\}^n \to \{0,1\}$ *with* $dia_{min,and}(f) = O(1)$*, we have:*

$$bs(f) \geq \Omega(n^{3/7})$$

Proof. We have two cases:

Case 1: $C_{min}(f) \geq n^{3/7}$.

The first part of Theorem 4.2 implies that $C_{min}(f) \leq dia_{min,\text{and}}(f) \cdot s(f) \leq O(s(f))$, since $dia_{min,\text{and}}(f) = O(1)$.

Then $n^{3/7} \leq C_{min}(f) \leq O(s(f)) \leq O(bs(f))$ and we are done.

Case 2: $C_{min}(f) \leq n^{3/7}$.

Now we use Lemma 2.12 from Chapter 2, with $r = 7$. This implies that if any transitive function f has $C_{min}(f) \leq n^{3/7}$, then $bs(f) \geq n^{3/7}$, implying the statement of the lemma in this case.

□

4.3.2 Implications to the log-rank conjecture for XOR functions

In this section, we prove that the log-rank conjecture holds for functions of the form $f(x \oplus y)$ such that the $\mathbb{F}_2$-degree diameter of f is upper bounded by a polynomial in the logarithm of the Fourier sparsity of f.

4.3.2.1 The approach of Lin and Zhang

In this part, we shall review the approach of Lin and Zhang [61] which we build upon in the next section. We first go over some essential tools and techniques required for their approach.

We now define the closure of min certificate complexity which was implicit in the approach of [94], and defined in [61]:

Definition 38. *[94, 61]* ***Closure of min Certificate Complexity***
For a function $f\colon \{0,1\}^n \to \{0,1\}$, we define the closure of the min certificate complexity of f as:

$$C^{clo}_{min}(f) = \max_{\alpha} C_{min}(f_\alpha)$$

where the maximum is taken over all possible partial assignments α on n variables, or in other words, over all possible subfunctions f_α of f.

Note that $C^{clo}_{min}(f) \leq C(f)$.

They also note that it is possible to define the closure for any complexity measure $M(f)$ for function f as $M^{clo}(f) = \max_\alpha M(f_\alpha)$, where the maximum is taken over all subfunctions f_α of f. A measure M is said to be downward non-increasing if for any function f, it holds that $M(f_\alpha) \leq M(f)$ for any subfunction f_α of f. Note that the measures sensitivity, block sensitivity, certificate complexity, decision tree complexity, $\mathbb{F}_2$-degree, Fourier sparsity, alternating number are all downward non-increasing. It follows from the definition of downward non-increasing measures that whenever measure M is downward non-increasing, it holds that $M^{clo}(f) = M(f)$.

We first note the following result from [21] stating that the rank of the communication matrix $M_{f\circ\oplus}$ exactly equals the Fourier sparsity of f:

Lemma 4.7. *[21] For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$rank(M_{f\circ\oplus}) = ||\hat{f}||_0$$

Next, we note the following lemma about communication complexity of XOR functions:

Lemma 4.8. *[69] For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$CC(f \circ \oplus) \leq 2D(f)$$

The following lemma upper bounds the decision tree complexity in terms of the product of its certificate complexity and $\mathbb{F}_2$-degree. We note that the second inequality follows because $C^{clo}_{min}(f) \leq C(f)$.

Lemma 4.9. *[95, 96, 61] For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$D(f) \leq C^{clo}_{min}(f) \cdot deg_2(f) \leq C(f) \cdot deg_2(f)$$

The $\mathbb{F}_2$-degree of any function is upper bounded by the logarithm of its Fourier sparsity as proved in [21]:

Lemma 4.10. *[21] For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$deg_2(f) \leq \log ||\hat{f}||_0$$

Lemmas 4.7, 4.8, 4.9, 4.10 imply the following result as also noted in [61]:

Lemma 4.11. *[61] For any function $f\colon \{0,1\}^n \to \{0,1\}$:*

$$CC(f \circ \oplus) \leq 2C^{clo}_{min}(f) \cdot \log rank(M_{f\circ\oplus}) \leq 2C(f) \cdot \log rank(M_{f\circ\oplus})$$

It follows from Lemma 4.11 that proving an upper bound on the certificate complexity of a function f in terms of a polynomial in $\log rank(M_{f\circ\oplus})$ would imply the log-rank conjecture for the corresponding XOR-composed function $f \circ \oplus$.

Another approach towards proving the log-rank conjecture for XOR-functions would be to upper bound $C^{clo}_{min}(f)$ directly. One way to achieve this

for a class of functions would be to prove an upper bound of the form $C_{min}(f) \leq M(f)$ where $M(f)$ is a complexity measure that is downward non-increasing, since that would imply the bound $C^{clo}_{min}(f) \leq M(f)$, thereby proving the log-rank conjecture for functions $f \circ \oplus$ with bounded value of $M(f)$. Lin and Zhang [61] use this approach to prove that the log-rank conjecture holds for XOR-composed functions $f \circ \oplus$ which are such that $alt(f)$ is at most polynomial in $\log ||\hat{f}||_0$.

In particular, they achieve this by proving that for every Boolean function f, $C_{min}(f) \leq alt(f)deg_2(f)$. Since both $alt(f)$ and $deg_2(f)$ are downward non-increasing, they get the following result:

Theorem 4.3. *[Theorem 2 from [61]] For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$CC(f \circ \oplus) \leq 2alt(f) \cdot \log^2 rank(M_{f \circ \oplus})$$

4.3.2.2 Our results

We now prove an analogous result to Theorem 4.3 implying the log-rank conjecture for XOR-composed functions with bounded $\mathbb{F}_2$-degree diameter. However, in contrast to Theorem 4.3, we do not need to upper bound $C^{clo}_{min}(f)$, since Theorem 4.1 proves an upper bound directly on $C(f)$ in terms of the product of $dia_{deg_2}(f)$ and $deg_2(f)$ for any function f. This gives us the following result confirming the log-rank conjecture for functions $f \circ \oplus$ with $dia_{deg_2}(f)$ upper bounded by a polynomial in the logarithm of the Fourier sparsity of f:

Theorem 4.4. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$CC(f \circ \oplus) \leq 2dia_{deg_2}(f) \log^2 rank(M_{f\circ\oplus})$$

Proof. Recall the third statement of Theorem 4.1 which states that:

$$C(f) \leq dia_{deg_2}(f) \cdot deg_2(f)$$

Along with Lemma 4.11, we get that:

$$\begin{aligned} CC(f \circ \oplus) &\leq 2C(f) \cdot \log rank(M_{f\circ\oplus}) \\ &\leq 2dia_{deg_2}(f) \cdot deg_2(f) \cdot \log rank(M_{f\circ\oplus}) \\ &\leq 2dia_{deg_2}(f) \cdot \log^2 rank(M_{f\circ\oplus}) \end{aligned}$$

Here the last inequality follows from Lemmas 4.7 and 4.10.

□

We note that the statement of Theorem 4.4 also holds with $dia_{\text{and}}(f)$ replacing $dia_{deg_2}(f)$, since the $\mathbb{F}_2$-degree diameter is upper bounded by the AND-OR diameter for any function f as noted in equation 4.2. We state Theorem 4.4 with $dia_{deg_2}(f)$ instead of $dia_{\text{and}}(f)$ because it gives a stronger statement since there exist functions with dia_{deg_2} much smaller than dia_{and} as illustrated by example 2.

Next we prove a common strengthening of Theorem 4.3 and Theorem 4.4. We show that the communication complexity of a function of the form $f(x \oplus y)$ can also be upper bounded in terms of the closure of its min $\mathbb{F}_2$-degree diameter and the square of the log of rank of its communication matrix $M_{f\circ\oplus}$.

Theorem 4.5. *For any function* $f\colon \{0,1\}^n \to \{0,1\}$*:*

$$CC(f \circ \oplus) \leq 2dia^{clo}_{min,deg_2}(f) \cdot \log^2 rank(M_{f\circ\oplus})$$

Proof. From the third statement of Theorem 4.2, we have that:

$$C^{clo}_{min}(f) \leq dia^{clo}_{min,deg_2}(f) \cdot deg_2(f)$$

Combining this with Lemmas 4.7, 4.10 and 4.11 gives the result. □

As we shall note in Lemma 4.12, $dia_{min,deg_2}(f) \leq dia_{min,and}(f) \leq alt(f)$. Since alternating number is a downward non-increasing measure, this implies that $dia^{clo}_{min,deg_2}(f) \leq alt(f)$. Furthermore, example 5 illustrates a family of functions $f\colon \{0,1\}^n \to \{0,1\}$ with $dia^{clo}_{min,deg_2}(f) \leq dia_{deg_2}(f) \leq 3$ and $alt(f) = n$ (when n is even, for the choice $k = \frac{n}{2}$ in the example). This shows that Theorem 4.5 is a strictly stronger statement than Theorem 4.3.

By definition, we have that for any function f, $dia^{clo}_{min,deg_2}(f) \leq dia_{deg_2}(f)$. Therefore, Theorem 4.5 is a potentially stronger statement than Theorem 4.4. As of now, we are not aware of any example function f separating $dia^{clo}_{min,deg_2}(f)$ from $dia_{deg_2}(f)$. However, we remark that the TRIBES function separates $dia^{clo}_{min,and}(f)$ from $dia_{\text{and}}(f)$ as noted in section 4.4.5.

We illustrate with examples in the next section that, in general, the alternating number of a function and its $\mathcal{H}$-diameter are incomparable for the different classes $\mathcal{H}$ that we consider. However, in the following lemma, we show that the min AND-OR diameter, and consequently, the min $\mathcal{H}$-diameter for all our different classes $\mathcal{H}$, are upper bounded by the alternating number.

Lemma 4.12. *For any function* $f\colon \{0,1\}^n \to \{0,1\}$*, we have,*

$$dia_{min,and}(f) \leq alt(f).$$

Proof. We shall prove the following relationship, which would immediately imply the statement of the lemma:

$$dia_{and}(f, 1^n) \leq alt(f) \tag{4.6}$$

Let $alt(f) = t$ and let $\mathcal{Q} = x^{(0)}, x^{(1)}, \ldots x^{(t)}$ be an alternating path of length t.

Now, we construct an AND-OR path $\mathcal{Q}' = z^{(0)}, z^{(1)}, \ldots z^{(t)}$ from 0^n to a certificate α of 1^n in the following way:

Let $z^{(0)} = x^{(0)} = 0^n$.

Let $z^{(1)} \preceq x^{(1)}$ be the minimal element so that $f(z^{(1)}) = f(x^{(1)})$. In other words, $z^{(1)} \preceq x^{(1)}$ is such that $f(y) = f(z^{(0)})$ for all y such that $z^{(0)} \prec y \prec z^{(1)}$.

In general, let $z^{(i+1)} \preceq x^{(i+1)}$ be the minimal element such that $f(z^{(i+1)}) = f(x^{(i+1)})$, and $z^{(i)} \prec z^{(i+1)}$ for $i \in \{0, 1, \ldots, t-1\}$.

Note that for the path $\mathcal{Q}'$, each subfunction f_{β^i} is either an AND or a NAND.

We will thus get an AND-OR path $z^{(0)}, z^{(1)}, \ldots, z^{(t)}$ of length t. Note that $z^{(t)}$ must agree with some certificate of 1^n, since otherwise, we can get an alternating path for f of length greater than t in the following way: consider

the alternating path $\mathcal{Q}' = z^{(0)}, z^{(1)}, \ldots z^{(t)}$. Since $z^{(t)}$ does not agree with any certificate of 1^n, the partial assignment α defined as $\alpha_i = 1$ whenever $z_i^{(t)} = 1$ and $\alpha_i = \star$ otherwise, is not a certificate of f. This implies the existence of an input $z^{(t+1)}$ such that $z^{(t)} \prec z^{(t+1)}$ and $f(z^{(t)}) \neq f(z^{(t+1)})$. Therefore, the path $\mathcal{Q}'' = z^{(0)}, z^{(1)}, \ldots z^{(t)}, z^{(t+1)}$ is an alternating path of length $t+1$ contradicting the fact that $alt(f) = t$.

Therefore, the path $z^{(0)}, z^{(1)}, \ldots, z^{(t)}$ is an AND-OR path from 0^n to some certificate of f on 1^n and the statement follows. □

We note that equation 4.6 implies the stronger statement that the min AND-OR diameter is also upper bounded by the shift invariant alternating number ($salt(f)$) as defined in [39].

4.4 Examples separating various measures

In this section, we give several examples, separating various types of diameters from each other and from some pre-existing measures.

4.4.1 Separating different diameters from each other

In this subsection, we present some examples separating the different diameters we consider, i.e., $dia_s(f)$, $dia_{deg}(f)$, $dia_{deg_2}(f)$ and $dia_{\text{and}}(f)$ from each other.

Separating $dia_s(f)$ from $dia_{\text{and}}(f)$

The following example has constant sensitivity diameter, whereas its AND-OR diameter equals the number of input variables.

Example 1. *Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be the parity function on* n *bits i.e.* $f(x) = \bigoplus_{i\in[n]} x_i$.

It is easy to see that $dia_{and}(f) = n$*, since for any partial assignment* α*, the subfunction* f_α *belongs to the AND-OR class only if* α *fixes all but 1 variable.*

On the other hand, for any input $x \in \{0,1\}^n$*,* $dia_s(f,x) = 1$*. This is because we can consider* α *to be the certificate fixing all the bits of* x *and then the path* $\mathcal{Q} = (\bar{x}, x)$ *is a valid* $\mathcal{H}$*-path since* $s(f) = n$ *(i.e.* f *induced on the "entire cube" has full sensitivity).*

Separating $dia_{deg}(f)$ **(and also** $dia_{deg_2}(f)$**) from** $dia_{\mathbf{and}}(f)$

The next example has constant values for both its real degree diameter as well as $\mathbb{F}_2$-degree diameter, but has large ($\Theta(n)$) AND-OR diameter.

Example 2. *Define* $f\colon \{0,1\}^n \to \{0,1\}$ *as follows:*

$f(0^n) = 1$,

$f(x) = \bigoplus_{i\in[n]} x_i$ *otherwise.*

It is easy to show that $dia_{and}(f) = n - 1$ *by a similar argument as in example 1.*

Also, we note that $\deg_2(f) = n$. *This follows from a result of Beigel and Bernasconi [19], stating that for any Boolean function,* $deg_2(f) = n$ *iff* $|f^{-1}(1)|$ *is odd.*

Therefore, by an analogous argument as in example 1, for any input $x \in \{0,1\}^n$, $dia_{deg_2}(f,x) = 1$. *(We can again consider* α *to be the certificate fixing all the bits of* x *and the path* $\mathcal{Q} = (\bar{x}, x)$ *is a valid* $\mathcal{H}$*-path since* $deg_2(f) = n$.*)*

Therefore, we have $dia_{deg_2}(f) = 1$.

We also have that for any Boolean function f, $deg_2(f) \leq deg(f)$ *(see for example, proposition 6.23 in [78]).*

Therefore, $deg(f) = n$, *and by a similar argument as for the* $\mathbb{F}_2$*-degree,* $dia_{deg}(f) = 1$.

We note that the parity function from example 1 also separates $dia_{deg}(f)$ from $dia_{\text{and}}(f)$. This is because $dia_{deg}(f) = 1$ as we shall note in the next example, whereas $dia_{\text{and}}(f) = n$ as noted in example 1.

Separating $dia_{deg}(f)$ **from** $dia_{deg_2}(f)$

Recall from equation 4.2 that $dia_{deg}(f) \leq dia_{deg_2}(f)$ for any Boolean function f. We now illustrate a function separating these two measures.

To separate $dia_{deg}(f)$ from $dia_{deg_2}(f)$, we consider again the parity function discussed in example 1. It is easy to see that for the parity function, $deg_2(f) = 1$. Moreover, for any partial assignment $\alpha\colon [n] \to \{0,1,\star\}$,

the subfunction f_α is also the parity function on the free bits and therefore, $deg_2(f_\alpha) = 1$. So we have $dia_{deg_2}(f) = n$. However, $deg(f) = n$ due to the characterization due to Shi and Yao mentioned in the survey [25] which states that for any function $f\colon \{0,1\}^n \to \{0,1\}$, $deg(f) = n$ iff the number of 1-inputs with an even number of 1's does not equal the number of 1-inputs with an odd number of 1's. Therefore, $dia_{deg}(f) = 1$ by a similar argument as in example 1.

Separating $dia_s(f)$ and $dia_{deg_2}(f)$ from each other

The measures $dia_s(f)$ and $dia_{deg_2}(f)$ are incomparable, and we provide examples separating them in both directions.

We again revisit the parity function considered in example 1 to illustrate a function with small $dia_s(f)$ and large $dia_{deg_2}(f)$. As noted before, $dia_{deg_2}(f) = n$, whereas $dia_s(f) = 1$, achieving the required separation.

To illustrate a separation in the other direction, we consider the TRIBES function on n^2 bits that has $dia_s(f) = \Theta(n)$ as we shall see in example 3. However, $deg_2(f) = n^2$ and therefore, $dia_{deg_2}(f) = 1$.

Separating $dia_s(f)$ from $dia_{deg}(f)$

We now mention a separating example with $dia_{deg}(f)$ much smaller than $dia_s(f)$. As noted before, the TRIBES function on n^2 bits has $dia_s(f) = \Theta(n)$ as analysed in greater detail in example 3. However, $deg(f) = n^2$ and therefore, $dia_{deg}(f) = 1$.

However, we are not aware of any examples as yet, separating these measures in the other direction. We do believe these measures to be incomparable, and it would be an interesting exercise to find functions f with $dia_s(f)$ asymptotically smaller than $dia_{deg}(f)$.

4.4.2 Example with $alt(f)$ smaller than $dia_{\mathbf{and}}(f)$ (and also $dia_s(f)$)

We now present an example where the alternating number is much smaller than $dia_{\text{and}}(f)$ as well as $dia_s(f)$.

Example 3. *Consider the TRIBES function* $f\colon \{0,1\}^{n^2} \to \{0,1\}$ *defined as:*

$$f(x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, x_{n1}, \ldots x_{nn}) = \bigvee_{i\in[n]} \bigwedge_{j\in[n]} x_{ij}$$

We first note that since f *is monotone,* $alt(f) = 1$.

We shall show that $dia_{and}(f) \geq 2n - 1$.

Consider the 1-input $x = 1^n(0^2 1^{n-2})^{n-1}$. *In other words, the first block of* n *bits of* x *are set to 1, the remaining* $n-1$ *blocks of* n *bits each have the first* 2 *bits set to 0 and the rest set to 1.*

Note that $f(\bar{x}) = 0$.

Now, let $\mathcal{Q} = (x^{(0)}, x^{(1)}, \ldots, x^{(t)})$ *be any valid AND-OR path from* x *to* α *where* α *is a certificate of* $\bar{x}$. *(Note that we have switched the roles of* x *and* $\bar{x}$ *in this example for convenience.)*

Since $f(x^{(0)}) \neq f(x^{(1)}) = 0$, *the partial assignment* $\beta^{(0)}$ *must have free bits belonging to the first block of* n *bits, and moreover, it cannot contain any*

free bits from any of the other blocks, in order for the function $f_{\beta^{(0)}}$ to belong to the AND-OR class.

For the next step, since $f(x^{(2)}) = 1$, it is necessary that both the 0-bits must be flipped from one of the remaining blocks. In other words, the partial assignment $\beta^{(1)}$ must have exactly two free variable corresponding to the first two bits of some block (other than the first block).

Again, as before, the partial assignment $\beta^{(2)}$ must contain free variables from the block which now only contains 1-bits.

In this way, any shortest valid AND-OR path must alternate between changing some block to contain only 1-bits (thereby changing the function value to 1), and then flipping some 1-bit in that block to 0 (changing the function value to 0).

Eventually, every block will contain a bit that was flipped from a 1 to a 0, and the set of these bits shall constitute a certificate of $\bar{x}$.

Therefore, we have that $dia_{and}(f) \geq 2n - 1$. We note that this bound is tight up to constant factors, as can be seen from Theorem 4.6 which implies that $dia_{and}(f) \leq O(n)$.

A similar argument also works to show that $dia_s(f) \geq 2n - 1$, and therefore, $dia_s(f) = \Theta(n)$ due to Theorem 4.6.

However, since $deg(f) = n^2$, we have that $dia_{deg}(f) = 1$.

Similarly, $deg_2(f) = n^2$ and therefore, $dia_{deg_2}(f) = 1$.

4.4.3 Examples with $alt(f)$ larger than all our diameters

We first mention an example from [38] that separates $alt(f)$ from $C(f)$ (and consequently, from all types of diameters).

Example 4. *Let f be the function constructed in [38] as an example where $alt(f)$ is exponentially larger than $D(f)$ and therefore also $C(f)$ (since $D(f) \geq C(f)$). We note that since $dia_{and}(f) \leq 2C(f)$ due to Lemma 4.2, f also acts as a separating example where $alt(f)$ is exponentially larger than $dia_{and}(f)$, and therefore, also exponentially larger than $dia_s(f), dia_{deg_2}(f)$ and $dia_{deg}(f)$.*

Next, we present examples of functions with even stronger separations between $alt(f)$ and $dia_{\text{and}}(f)$ (and also other diameters) as compared to the exponential separations exhibited by example 4. We consider the example of shifted-threshold functions and show that they have a constant AND-OR diameter (and therefore also, constant values of all other diameters we consider), but large values for alternating number (i.e. $\Theta(k)$ for the threshold-k function):

Example 5. *For $k \geq 0$, the threshold-k function $Th_k \colon \{0,1\}^n \to \{0,1\}$ is defined as:*

$Th_k(x) = 1$ *iff* $|\{j \mid x_j = 1\}| \geq k$.

Since Th_k is monotone, $alt(Th_k) = 1$.

We now show that $dia_{and}(Th_k, x) \leq 3$ for any $x \in \{0,1\}^n$. Assume $k \geq \frac{n}{2}$ for now. A similar argument goes through when $k < \frac{n}{2}$.

Consider an input $x = 0^t1^{n-t}$ *for some* $t < k$. *Then,* $\bar{x} = 1^t0^{n-t}$ *is a* 0*-input of* f.

Case 1*:* $n - t \geq k$

Note that $f(x) = f(0^t1^{n-t}) = 1$.

Then the following is a valid AND-OR path:

$x^{(0)} = \bar{x} = 1^t0^{n-t},\ x^{(1)} = 1^k0^{n-k},\ x^{(2)} = 0^t1^{k-t}0^{n-k},\ x^{(3)} = 0^t1^k0^{n-k-t}$

In this case, $x^{(3)}$ *agrees with* $(\star)^t1^k(\star)^{n-k-t}$, *which is a certificate for* f *on the input* $x = 0^t1^{n-t}$.

Case 2*:* $n - t < k$

In this case, $f(x) = f(0^t1^{n-t}) = 0$.

Then it can be verified that the following is a valid AND-OR path:

$x^{(0)} = \bar{x} = 1^t0^{n-t},\ x^{(1)} = 1^k0^{n-k},\ x^{(2)} = 0^t1^{k-t}0^{n-k}$

This is because fixing the first t *input bits to 0 gives a certificate for* f *on the input* $x = 0^t1^{n-t}$.

This proves that $dia_{and}(Th_k, x) \leq 3$ *for input of the form* $x = 0^t1^{n-t}$ *when* $t < k$. *Since threshold-k is a symmetric function, this implies that* $dia_{and}(Th_k, x) \leq 3$ *for any input* x *with exactly* t *0's. A similar argument can be used to prove that* $dia_{and}(Th_k, x) \leq 3$ *for inputs with exactly* t *1's when* $t \geq k$. *Thus, we have that* $dia_{and}(Th_k) \leq 3$.

We now define the shifted-threshold-k function for any fixed input $a \in \{0,1\}^n$, *denoted* $Th_{k,a}\colon \{0,1\}^n \to \{0,1\}$ *as:*

$Th_{k,a}(x) = Th_k(x \oplus a)$ for all $x \in \{0,1\}^n$.

Observe that $dia_{and}(Th_{k,a},\ x) = dia_{and}(Th_k,\ x \oplus a) \leq 3$, for each $x \in \{0,1\}^n$.

Therefore, we have that $dia_{and}(Th_{k,a}) = dia_{and}(Th_k) \leq 3$ for any $a \in \{0,1\}^n$.

We shall now see that there exist inputs $a \in \{0,1\}^n$ such that $alt(Th_{k,a}) \geq 2k$:

Let $k \leq \frac{n}{2}$ and consider $a = 1^k 0^{n-k}$.

Now, consider the sequence of vertices:
$x^{(0)} = 1^k 0^{n-k},\ x^{(1)} = 1^{k-1}0^{n-k+1},\ x^{(2)} = 1^{k-1}0^{n-k}1, x^{(3)} = 1^{k-2}0^{n-k+1}1, \ldots, x^{(2k)} = 0^{n-k}1^k$.

It is easy to see that the corresponding path $\{x^{(i)} \oplus a | i \in \{0,1,\ldots,2k\}\}$ is an alternating path of length $2k$ for $Th_{k,a}$.

Therefore, for any $k \leq \frac{n}{2}$ and for $a = 1^k 0^{n-k}$, we have that $alt(Th_{k,a}) \geq 2k$, whereas $dia_{and}(Th_{k,a}) \leq 3$.

(A similar argument would work when $k > \frac{n}{2}$ to give a function with $alt(Th_{k,a}) \geq 2(n-k)$ and $dia_{and}(Th_{k,a}) \leq 3$.)

4.4.4 Separating min diameters from $alt(f)$

Next, we give an example of a transitive function $f\colon \{0,1\}^n \to \{0,1\}$ with constant $dia_{min,\text{and}}(f)$ (and therefore, constant $dia_{min,deg}(f), dia_{min,deg_2}(f)$ and $dia_{min,s}(f)$) and $alt(f) = \theta(n)$.

We note that a separation between $dia_{min,\text{and}}(f)$ and $alt(f)$ is also achieved by examples 4 and 5, since $dia_{min,\text{and}}(f) \leq dia_{\text{and}}(f)$. However, in constrast with those examples, the following example is also transitive.

Example 6. *Let $n > 0$ be any multiple of 4. Define $f\colon \{0,1\}^n \to \{0,1\}$ as:*

$f(x) = 1$ iff x contains $\frac{n}{4}$ consecutive 1's immediately followed by $\frac{n}{4}$ consecutive 0's when x is laid out around a circle, and $f(x) = 0$ otherwise.

We construct an AND-OR path Ω from the input 0^n to a certificate α of f on 1^n of length 6, thereby proving that $dia_{min,and}(f) \leq 6$ as follows:

$$x^{(0)} = 0^n, \qquad x^{(1)} = (1)^{\frac{n}{4}}(0)^{\frac{3n}{4}}, \qquad x^{(2)} = (1)^{\frac{n}{4}}(0)^{\frac{n}{4}-1}1(0)^{\frac{n}{2}},$$
$$x^{(3)} = (1)^{\frac{n}{2}}(0)^{\frac{n}{2}}, \qquad x^{(4)} = (1)^{\frac{n}{2}}(0)^{\frac{n}{4}-1}1(0)^{\frac{n}{4}}, \qquad x^{(5)} = (1)^{\frac{3n}{4}}(0)^{\frac{n}{4}},$$
$$x^{(6)} = (1)^{\frac{3n}{4}+1}(0)^{\frac{n}{4}-1}$$

Note that fixing the 1-bits of $x^{(6)}$ gives a certificate of f for the input 1^n, as required.

We now show that $alt(f) \geq \frac{n}{2} + 2$ by constructing an alternating path as follows:

$$x^{(0)} = 0^n$$
$$x^{(1)} = (1)^{\frac{n}{4}}(0)^{\frac{3n}{4}}$$
$$x^{(2)} = (1)^{\frac{n}{4}}010^{\frac{3n}{4}-2}$$
$$x^{(3)} = (1)^{\frac{n}{4}+2}(0)^{\frac{3n}{4}-2}$$
$$x^{(4)} = (1)^{\frac{n}{4}+2}01(0)^{\frac{3n}{4}-4}$$

$x^{(5)} = (1)^{\frac{n}{4}+4}(0)^{\frac{3n}{4}-4}$

$\vdots$

$x^{(\frac{n}{2}+1)} = (1)^{\frac{3n}{4}}(0)^{\frac{n}{4}}$

$x^{(\frac{n}{2}+2)} = (1)^{\frac{3n}{4}+1}(0)^{\frac{n}{4}-1}$

This gives an alternating path of length $(\frac{n}{2}+2)$.

4.4.5 Separating $dia^{clo}_{min,and}(f)$ from $dia_{\mathbf{and}}(f)$

For separating $dia^{clo}_{min,and}(f)$ from $dia_{\text{and}}(f)$, we revisit the TRIBES function considered in example 3. As noted in that example, $dia_{\text{and}}(f) = \Theta(n)$. On the other hand, we have that $dia^{clo}_{min,and}(f) \leq alt(f) = 1$, thereby achieving the required separation. The same separation is also achieved for the TRIBES function between $dia^{clo}_{min,s}(f)$ and $dia_s(f)$ by an analogous argument.

4.4.6 Diameter under OR-composition

In this section, we study the behavior of diameters under OR-composition. In particular, we prove that any diameter of the OR-composition of two functions is upper bounded by a sum of their corresponding individual diameters (plus a constant factor).

In what follows, for two functions $f, g\colon \{0,1\}^n \to \{0,1\}$, we define the OR-composed function $f \vee g\colon \{0,1\}^{2n} \to \{0,1\}$ as $f \vee g(x_1, x_2) = 1$ if $f(x_1) = 1$ or $g(x_2) = 1$, and $f \vee g(x_1, x_2) = 0$ otherwise, for $x_1, x_2 \in \{0,1\}^n$.

We now prove a result relating the diameter of OR-composition of two

functions with their individual diameters as mentioned before:

Theorem 4.6. *For any functions $f, g\colon \{0,1\}^n \to \{0,1\}$, we have:*

$$dia_{and}(f \vee g) \leq dia_{and}(f) + dia_{and}(g) + 3$$

Proof. Consider any input $(x_1, x_2) \in \{0,1\}^{2n}$. We have two cases as follows:

Case 1: $f \vee g(x_1, x_2) = 0$

Consider the AND-OR path $z_1^{(0)}, z_1^{(1)}, \ldots, z_1^{(t_1)}$ (where $\bar{x_1} = z_1^{(0)}$) achieving $dia_{\text{and}}(f, x_1)$ and the AND-OR path $z_2^{(0)}, z_2^{(1)}, \ldots, z_2^{(t_2)}$ (where $\bar{x_2} = z_2^{(0)}$) achieving $dia_{\text{and}}(g, x_2)$.

We first deal with the case when $f(\bar{x_1}) = 0$ and $g(\bar{x_2}) = 0$. Observe that the following is a valid AND-OR path for $f \vee g$:

$$(z_1^{(0)}, z_2^{(0)}), (z_1^{(0)}, z_2^{(1)}), \ldots (z_1^{(0)}, z_2^{(t_2)}), (z_1^{(1)}, z_2^{(t_2)}), (z_1^{(2)}, z_2^{(t_2)}), \ldots, (z_1^{(t_1)}, z_2^{(t_2)}).$$

This follows because of the simple observation, that for any input $(a, b) \in \{0,1\}^{2n}$, if $g(b) = 0$, then $f \vee g(a, b) = f(a)$. Due to this observation, in the first t_2 steps of the above path, the input to f is fixed to $z_1^{(0)}$ which is such that $f(z_1^{(0)}) = 0$. Therefore, throughout these steps, we have that $f \vee g(z_1^{(0)}, z_2^{(i)}) = g(z_2^{(i)})$ for any $i \in \{0, 1, \ldots t_2\}$. Since $z_2^{(0)}, z_2^{(1)}, \ldots, z_2^{(t_2)}$ is a valid AND-OR path of g, the first t_2 steps form valid steps of an AND-OR path of $f \vee g$.

Similarly, since $z_2^{(t_2)}$ agrees with a certificate for g on x_2, it holds that $g(z_2^{(t_2)}) = 0$. Therefore, for the next t_1 steps of the path, it holds that $f \vee$

$g(z_1^{(i)}, z_2^{(t_2)}) = f(z_1^{(i)})$ for any $i \in \{0, 1, \ldots t_1\}$, and a similar argument goes through as for the first t_2 steps of the path.

Finally, since $z_1^{(t_1)}$ agrees with a certificate for f on x_1 and $z_2^{(t_2)}$ agrees with a certificate for g on x_2, the input $(z_1^{(t_1)}, z_2^{(t_2)})$ agrees on a certificate for $f \vee g$ on the 0-input (x_1, x_2). We therefore get a valid AND-OR path for the function $f \vee g$ of length $t_1 + t_2$.

Now, we consider the case when $f(\bar{x_1}) = 1$ and $g(\bar{x_2}) = 0$. In this case, we first perform an additional step of flipping the bits of any minimal sensitive block B for f on $\bar{x_1}$ to get to a 0-input of f, i.e., $\bar{x_1}^B$. We then follow the argument of the previous case, and without changing the input to f, take t_2 steps corresponding to the AND-OR path for g, i.e., $z_2^{(0)}, z_2^{(1)}, \ldots, z_2^{(t_2)}$. We then "undo" the first step by flipping back the bits of B in the input corresponding to f to get to the input $(\bar{x_1}, z_2^{(t_2)})$. Finally, we take the t_1 steps of the AND-OR path for f starting from input $\bar{x_1}$ to get a valid AND-OR path of total length $t_1 + t_2 + 2$.

Similar argument works for the case when $f(\bar{x_1}) = 0$ and $g(\bar{x_2}) = 1$.

Finally, the case with $f(\bar{x_1}) = 1$ and $g(\bar{x_2}) = 1$ goes through with a similar argument, with the difference that the first step involves simultaneously flipping minimal sensitive blocks for both f and g, say B_1 and B_2 respectively, on the respective inputs $\bar{x_1}$ and $\bar{x_2}$, to get input $(\bar{x_1}^{B_1}, \bar{x_2}^{B_2})$. Note that $B_1 \cup B_2$ is a minimal sensitive block of $f \vee g$ for the input $(\bar{x_1}, \bar{x_2})$. The next step flips back these bits only for g to get the input $(\bar{x_1}^{B_1}, \bar{x_2})$. The argument then

proceeds as in the previous case, where we take the AND-OR path for g starting from input $\bar{x_2}$, followed by flipping back the bits of block B_1 for the input to f, followed by the AND-OR path for f starting from input $\bar{x_1}$.

This gives a valid AND-OR path of length $t_1 + t_2 + 3$ for $f \vee g$ starting from the input $(\bar{x_1}, \bar{x_2})$.

Case 2:$f \vee g(x_1, x_2) = 1$

Wlog, assume that $g(x_2) = 1$.

In this case, we follow a similar proof strategy as in Case 1, with the difference that in this case, we only need to follow the steps corresponding to a valid AND-OR path for g starting from $\bar{x_2}$, since in this case, a certificate for g on x_2 would also be a certificate for $f \vee g$ on (x_1, x_2). Apart from that, we follow a similar argument, where we first flip appropriate blocks of bits, if necessary, to ensure that we are at a 0-input of f. We then follow the AND-OR path for g to get a certificate for $f \vee g$ on (x_1, x_2). This gives an AND-OR path for $f \vee g$ of length at most $t_2 + 2$.

□

We remark that Theorem 4.6 also holds for the three other diameters we consider i.e. for $dia_{deg}(f)$, $dia_{deg_2}(f)$ and $dia_s(f)$ by an analogous argument.

Note that the TRIBES function $f\colon \{0,1\}^{n^2} \to \{0,1\}$ as discussed in example 3 is an OR-composition of n copies of the AND_n function (i.e. the AND function on n bits). Since $dia_{\text{and}}(AND_n) = 1$, Theorem 4.6 implies the bound:

$dia_{\text{and}}(TRIBES) \leq \Theta(n)$. As seen in example 3, $dia_{\text{and}}(TRIBES) \geq 2n - 1$ and therefore, this bound is asymptotically tight for the TRIBES function.

However, Theorem 4.6 does not always give a tight bound for OR-composed functions, as can be seen from the example of the OR function on n bits, $OR_n \colon \{0,1\}^n \to \{0,1\}$. OR_n is also an OR-composition of n copies of the function $g \colon \{0,1\} \to \{0,1\}$ with single-bit inputs, where $g(x) = x$. Since $dia_{\text{and}}(g) = 1$, Theorem 4.6 gives the bound $dia_{\text{and}}(OR_n) \leq \Theta(n)$. This bound is not tight since it holds that $dia_{\text{and}}(OR_n) = 1$.

Chapter 5

Decision trees in machine learning

In the first part of this chapter[1], we present a new post-processing method for binary classification decision trees, with the objective of improving the recall of the model with respect to class 1, while minimizing the loss of precision of the decision tree for class 1. Towards this end, we define a geometric optimization problem which we prove to be NP-complete. We propose some heuristic solutions to help optimize the objective for the above problem, and also come up with a *mixed integer linear programming* (MILP) formulation for our optimization problem. We experimentally evaluate these heuristics in the context of transfer learning, and validate that our heuristics increase the recall significantly compared to alternative approaches, and require several orders of magnitude *less* compute time than exact MILP solutions.

In the next part of this chapter, we adapt existing transfer learning algorithms for decision trees, originally designed for classification tasks, enabling them for the regression setting. We also propose a new and simpler algorithm for this problem. Finally, we provide a comparison of these different algorithms for performance and efficiency via experimental evaluation.

[1] The results presented in this chapter appear in [34] and [33].

5.1 Preliminaries

5.1.1 Learning algorithms for decision trees

We shall now present a brief review of a learning algorithm commonly used to train decision trees, while emphasizing on the definitions of information gain and variance gain which will be important for us.

We first describe the general structure of this learning algorithm which is common to the ID3 algorithm due to Quinlan [80] and the CART algorithm [24]. We then go into the differences based on the type of task being considered (i.e. classification versus regression). The learning algorithm is given as input a training data set, and starts by building the decision tree in a top-down manner. It first creates a root node for the tree along with selecting a suitable feature to query at the node, and an appropriate threshold value. During any step of this top-down algorithm, it considers a node v and data set $\mathcal{D}$ reaching it, and chooses a feature j and threshold value a for that node, so that the node would query any data point x as $x_j < a$ if the feature j is numeric, and as $x_j = a$ if it is categorical.

The technique used for selection of feature j and threshold a depends on the type of problem being considered:

Binary classification problem: For a binary classification problem, this selection of feature and threshold value is done based on a criterion of "purity" of the data set $\mathcal{D}$ reaching a node v called as *entropy*. This is based on the ID3 algorithm [80]. We shall now define entropy of a data set for a binary classification problem:

Definition 39. Entropy of a data set *Given a data set $\mathcal{D}$ corresponding to a binary classification task, let p_z be the proportion of data points in $\mathcal{D}$ belonging to the class z for $z \in \{0, 1\}$. Then, the entropy of $\mathcal{D}$ is defined as:*

$$\textit{Entropy}(\mathcal{D}) = \sum_{z \in \{0,1\}} -p_z \log_2 p_z.$$

Here, the term $p_z \log_2 p_z$ is defined to be 0 if $p_z = 0$.

The feature and threshold values of a node are chosen so as to maximize the expected reduction in entropy, as measured by the *information gain (IG)* which we now define:

Definition 40. Information gain *The information gain (IG) for data set $\mathcal{D}$, feature j and threshold a is defined as:*

$$IG(\mathcal{D}, j, a) = \textit{Entropy}(\mathcal{D}) - \sum_{t \in \{0,1\}} \frac{|\mathcal{D}_t|}{|\mathcal{D}|} \textit{Entropy}(\mathcal{D}_t)$$

where, $\mathcal{D}_0 \subseteq \mathcal{D}$ is the set of data points $x \in \mathcal{D}$ such that $x_j < a$, and $\mathcal{D}_1 \subseteq \mathcal{D}$ contains data points $x \in \mathcal{D}$ such that $x_j \geq a$, for a numeric feature j. If the feature j is categorical, $\mathcal{D}_0 \subseteq \mathcal{D}$ is the set of points $x \in \mathcal{D}$ such that $x_j = a$, and $\mathcal{D}_1 \subseteq \mathcal{D}$ is the set of points with $x_j \neq a$.

Regression problem: For a regression problem, the *variance gain* is used as a criterion to select a feature and threshold value for the node, as introduced in the CART algorithm [24]. We first define the variance of a regression data set $\mathcal{D}$:

Definition 41. Variance of a data set *Given a data set* $\mathcal{D} = \{(x_i, y_i) \mid i \in [m]\}$ *corresponding to a regression task, let its mean be denoted by* $\bar{y} = \sum_{i\in[m]} \frac{y_i}{m}$. *Then, the variance of* $\mathcal{D}$ *is defined as:* $Var(\mathcal{D}) = \sum_{i\in[m]} \frac{(y_i-\bar{y})^2}{m}$.

Definition 42. Variance gain *The variance gain (VG) for data set* $\mathcal{D}$, *feature* j *and threshold* a *is defined as:*

$$VG(\mathcal{D}, j, a) = Var(\mathcal{D}) - \sum_{t\in\{0,1\}} \frac{|\mathcal{D}_t|}{|\mathcal{D}|} Var(\mathcal{D}_t)$$

where, $\mathcal{D}_0 \subseteq \mathcal{D}$ *is the set of points belonging to the left child, and* $\mathcal{D}_1 \subseteq \mathcal{D}$ *is the set of points belonging to the right child, if the data* $\mathcal{D}$ *were split using feature* j *with the query* $x_j < a$ *if feature* j *is numeric, or the query* $x_j = a$ *in case* j *is a categorical feature.*

In case of regression tasks, the algorithm chooses the feature j and threshold a that gives the maximum variance gain for the data $\mathcal{D}$ at node v.

Finally, the algorithm stops expanding if certain stopping conditions are met e.g. if a node has sufficiently few data points reaching it, or if the node has data points belonging to a single class in case of classification problems etc. The leaf labels are set to be the majority class of the data points reaching the leaf for classification tasks and the mean of the data points for regression tasks.

Divergence measures for probability distributions: We shall now define some divergence measures for probability distributions commonly used in

statistics, which we shall require later. Let P and Q be discrete probability distributions defined on the same probability space $\mathcal{X}$.

The Kullback–Leibler (KL) divergence, also called as relative entropy, from Q to P is defined as:

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log_2 \frac{P(x)}{Q(x)}$$

The Jensen-Shannon (JS) divergence between P and Q is defined as:

$$JSD(P||Q) = \frac{D_{KL}(P||M) + D_{KL}(Q||M)}{2}$$

where $M = \frac{1}{2}(P + Q)$.

Note that the JS divergence is symmetric with respect to the two distributions, unlike the KL divergence. See [62] for more details about the different types of divergence measures.

5.2 Algorithms for decision trees in binary classification tasks

In this section, we present our approach to solving the problem of maximizing the recall of a given decision tree with respect to class 1, while limiting the drop in its precision for class 1 as described in the introduction.

5.2.1 The Optimal Expansion Problem

Let us denote the given decision tree by T, and the training data set with m samples and n features as $\{(x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}), y_i), i \in \{1, \ldots, m\}\}$.

In this part, we work with binary classification tasks and therefore, $y_i \in \{0, 1\}$. We shall ignore categorical features in this section, but we note that our methods operate in the presence of—without acting upon—such features.

Consider any leaf node u in a decision tree T. For an input $(x'_1, \ldots, x'_n)$ that is contained in u, the path from the root of T to u corresponds to a list of threshold conditions and indicators $(C_1, I_1), ..., (C_k, I_k)$, where condition C_i has the form "$x'_{j_i} < t_i$?" for some $j_i \in \{1, \ldots, n\}$ and I_i indicates whether C_i is true or false. Thus, $(C_1, I_1), ..., (C_k, I_k)$ induces an n-dimensional axis-aligned hyper-rectangle in the feature space. In what follows, for brevity, we refer to such axis-aligned hyper-rectangles simply as rectangles. Since a feature can appear more than once in the path to u, it may be the case that the rectangle is unbounded in $n - k'$ dimensions, where $1 \leq k' \leq k$. For a given decision tree T, let $\mathcal{R}_1, ..., \mathcal{R}_z$ denote the set of disjoint axis-aligned rectangles whose union corresponds to the set of points that are classified as 1 by T.

Our approach: We consider an approach where our algorithm proceeds by iteratively, and locally, expanding each rectangle $\mathcal{R}_i$ for $i \in [z]$, so that the *precision of* $\mathcal{R}_i$ does not drop by more than a pre-specified amount $\delta \geq 0$, on the training data. Explicitly, the precision of $\mathcal{R}_i$ is defined as $\text{Precision}_1(\mathcal{R}_i) = \frac{\sum_{(x_j, y_j) \in \mathcal{R}_i} y_j}{\sum_{(x_j, y_j) \in \mathcal{R}_i} 1}$. Since expanding $\mathcal{R}_i$ can only cause recall to increase, this method—much like lowering the prediction probability (i.e. classification threshold) for class 1—will result in a monotone increase in recall. This algorithmic approach of expanding each rectangle maximally, as long as the drop in its precision is limited, can be formalized as the following geometric

optimization problem for each of the rectangles $\mathcal{R}_i$:

Optimal Expansion Problem. Consider an m sample dataset $\{(x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}), y_i), i \in \{1, \ldots, m\}\}$ in n-dimensional feature space, where each $y_i \in \{0, 1\}$. Given a rectangle $\mathcal{R}$ containing at least one data point, and $\delta \geq 0$, return a rectangle $\mathcal{R}^\star$ such that $\mathcal{R} \subseteq \mathcal{R}^\star$, $\sum_{(x_i,y_i)\in\mathcal{R}} y_i$ is maximized, and $\text{Precision}_1(\mathcal{R}^\star) \geq (1 - \delta)\text{Precision}_1(\mathcal{R})$.

Theorem 5.1. *The optimal expansion problem is NP-complete.*

Proof. To prove NP-hardness of the optimal expansion problem, we present a reduction from the Bichromatic Rectangle (BR) problem. The decision version of the BR problem takes as input m red *and* blue points in n-dimensional space, and outputs "yes" if there exists a rectangle containing at least k blue points, but no red points. This was proved to be NP-hard if the number of dimensions n is part of the input [15, 41].

Suppose we are given an instance of Bichromatic Rectangle (BR) problem. We reduce from the BR problem to our optimal expansion problem as follows. During a pre-processing phase, all blue points that overlap with red ones are removed, as they are never part of a solution to the BR problem. We assume at least one blue point remains after this pruning, otherwise the solution to the BR instance is trivial. For each remaining blue point, we create a rectangle that surrounds only that point. For the created rectangle $\mathcal{R}$, assume we have an oracle that can answer the decision version of the optimal

expansion problem (i.e., determine whether there is an expansion of $\mathcal{R}$ containing at least k points in the positive class). We query this oracle, setting $\delta = 0$, and treating red points as those in the negative class, and blue points as those in the positive class. Since the input rectangle $\mathcal{R}$ has $\text{Precision}_1(\mathcal{R}) = 1$, and $\delta = 0$, a "yes" answer to the decision optimal expansion problem implies that there exists a rectangle $\mathcal{R}^\star$ with $\text{Precision}_1(\mathcal{R}^\star) = 1$ (i.e., no red points) and containing at least k positive (i.e., blue) points. Similarly, a "no" answer implies that there exists no such rectangle. Therefore, the answer given out by the oracle exactly answers the BR problem for the given instance.

Since, i) BR is NP-hard, ii) our reduction requires time polynomial in m and n, and, iii) the optimal expansion decision problem is clearly in NP via a linear-time verification step, we have shown that the optimal expansion problem is also NP-complete. □

We also remark there are strong connections to a host of related geometric problems [18, 16, 12].We note that these related problems admit a polynomial time solution when n is not part of the input (i.e., a fixed constant value), but this is not a reasonable assumption for our machine learning setting.

5.2.2 Heuristics for Geometric Expansion

Since we have shown the optimal expansion problem is NP-complete, we instead focus on heuristic methods to provide solutions quickly, albeit without guaranteeing solution optimality. We next describe our heuristic method for

expanding rectangles, and how to compute it in terms of a set of abstract data structure operations.

Suppose we are given an arbitrary rectangle $\mathcal{R}$, with "left" boundary $\text{lb}(\mathcal{R}, j)$ (i.e., the minimum), and "right" boundary $\text{rb}(\mathcal{R}, j)$ (i.e., the maximum), with respect to feature j. Define the width with respect to feature j as $w(\mathcal{R}, j) = \text{rb}(\mathcal{R}, j) - \text{lb}(\mathcal{R}, j)$. Note that $\text{lb}(\mathcal{R}, j)$ can be $-\infty$ and $\text{rb}(\mathcal{R}, j)$ can be ∞. Suppose $\text{lb}(\mathcal{R}, j) = -\infty$, but $\text{rb}(\mathcal{R}, j)$ is finite (the alternative case is symmetric). We set $w(\mathcal{R}, j) = \text{rb}(\mathcal{R}, j) - \gamma$, where γ is the minimum value of feature j among points in $\mathcal{R}$. If both boundaries are infinite, then $w(\mathcal{R}, j) = \infty$. Our heuristics will expand $\mathcal{R}$ by iteratively adjusting these boundaries of $\mathcal{R}$.

Our prototypical geometric heuristic for expanding a single rectangle is formalized in Algorithm 1. Input parameters to the algorithm are: i) DS, a data structure that supports search operations on the dataset; ii) a parameter $\delta \geq 0$, controlling the allowed drop in precision on the target data; iii) a value $\beta > 0$ controlling the rate of expansion; iv) a value $\text{iter}_{\max}$ indicating the maximum number of iterations, and; v) the rectangle $\mathcal{R}$ that we wish to expand.

We defer the technical details of our data structure until first walking through the execution of the algorithm. The algorithm first calls DS.Initialize (line 2), preparing the internal representation of the data structure to support efficient expansion of the input rectangle $\mathcal{R}$. The heuristic then iteratively selects the next feature, j, and attempts to expand rectangle $\mathcal{R}$ with respect to feature j. An expansion factor Δ is computed, which determines the rate

Algorithm 1 BasicAlgorithm($\mathrm{DS}, \delta, \beta, \mathrm{iter}_{\max}, \mathcal{R}$)

Require: Data structure DS, rectangle $\mathcal{R}$, parameters δ, β, $\mathrm{iter}_{\max}$

$\mathcal{R}_0 \leftarrow \mathcal{R}$
DS.Initialize($\mathcal{R}$)
for $j \in \{1, \ldots, n\}$ **do**
    $\Delta \leftarrow \beta \times w(\mathcal{R}, j)$
    if $\Delta = 0$ or $\Delta = \infty$ **then**
        continue
    for iter $\in \{1, \ldots, \mathrm{iter}_{\max}\}$ **do**
        if $\mathrm{lb}(\mathcal{R}, j) - \Delta \geq \min_i x_{i,j}$ **then**
            $S \leftarrow$ DS.collect($j, -1, \Delta$)
            if $(1-\delta) \cdot \mathrm{Precision}_1(\mathcal{R}_0) \leq \mathrm{Precision}_1(\mathcal{R} \cup S)$ **then**
                $\mathcal{R} \leftarrow$ DS.Commit()
            else
                DS.Rollback()
        if $\mathrm{rb}(\mathcal{R}, j) + \Delta \leq \max_i x_{i,j}$ **then**
            $S \leftarrow$ DS.Collect($j, 1, \Delta$)
            if $(1-\delta) \cdot \mathrm{Precision}_1(\mathcal{R}_0) \leq \mathrm{Precision}_1(\mathcal{R} \cup S)$ **then**
                $\mathcal{R} \leftarrow$ DS.Commit()
            else
                DS.Rollback()
return $\mathcal{R}$

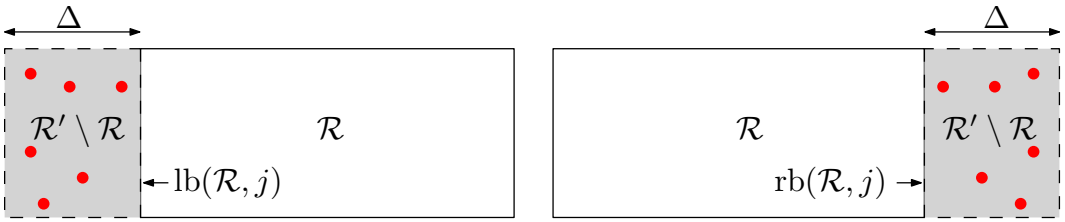

Figure 5.1: Left: extending the left boundary of $\mathcal{R}$ by Δ: see lines 8-13 of Algorithm 1. Red points are returned as S by the function DS.COLLECT. Right: symmetric case, extending the right boundary of $\mathcal{R}$ by Δ, performed on lines 14-19.

at which $\mathcal{R}$ will expand with respect to feature j, based on β and the width of $\mathcal{R}$. The algorithm iteratively expands both the left and right boundaries of $\mathcal{R}$ with respect to feature j by Δ, stopping only when either the precision of $\mathcal{R}$ drops by more than δ compared to the original rectangle $\mathcal{R}_0$, or $\text{iter}_{\max}$ is exceeded.

During the process, there are a few crucial operations executed on the DS. The first is called DS.COLLECT(j, sign, Δ) on line 9. Let $\mathcal{R}'$ be the rectangle induced by extending the $\text{lb}(\mathcal{R}, j)$ (resp. $\text{rb}(\mathcal{R}, j)$) to the left (resp. right) by Δ, if sign $= -1$ (resp. sign $= 1$). Then, DS.COLLECT(j, sign, Δ) returns the set of points in $\mathcal{R}' \setminus \mathcal{R}$ (i.e., the new points added by moving the boundary). This is illustrated for a case where $n = 2$ in Figure 5.1. Since adjusting left and right boundaries of $\mathcal{R}$ may result in too large a precision drop, DS provides the ability to reverse an expansion (called a ROLLBACK). Alternatively, after computing the precision of $\mathcal{R} \cup S$, another operation, called COMMIT, finalizes the boundary expansion. Moreover, DS has the ability to remember once ROLLBACK has been called on a boundary, and does not

attempt to expand the same boundary again in the next iteration.

Theorem 5.2. *There is a data structure* DS *that occupies* $\mathcal{O}(mn)$ *space, supports the operations* Collect, Commit, *and* Rollback, *and can be used to execute the for loop (lines 3-19) in Algorithm 1 in* $\mathcal{O}((m + iter_{max})n)$ *time after an initial pre-processing phase (i.e.,* Initialize *has completed on line 2). This pre-processing phase on line 2 requires* $\mathcal{O}(mn \lg m)$ *time and* $\mathcal{O}(mn)$ *space.*

Proof. The data structure DS maintains n sorted lists, as well as several additional structures. Let $\mathcal{L}_j$, for $j \in \{1, \ldots, n\}$, be the lists of feature values for the j-th dimension, sorted in ascending order. Each index $\mathcal{L}_j[k]$ will store more than just these feature values, so we denote the feature values as $\mathcal{L}_j[k]$.val. We re-index the data set into a rank index $\mathcal{I}$ such $\mathcal{I}[i]$ represents x_i, and stores the list of ranks $(k_1, ..., k_n)$ such that $(\mathcal{L}_1[k_1].\text{val}, ..., \mathcal{L}_n[k_n].\text{val}) = x_i$. Next we describe the additional fields associated with each list element $\mathcal{L}_j[k]$: i) a back pointer to its associated index in $\mathcal{I}$, such that $\mathcal{I}[\mathcal{L}_j[k].\text{back}][j] = k$, and ii) a flag $\mathcal{L}_j[k]$.active indicating whether $\mathcal{L}_j[k]$ is *active* in the expanding current rectangle. Overall, although there are several additional fields for each data point, the space occupied is linear, $\mathcal{O}(mn)$, and it can be constructed in the time required to sort each list of feature values, or $\mathcal{O}(mn \lg m)$ time.

Consider rectangle $\mathcal{R}$ input to Initialize. Let $\{a_j, a_j + 1, \ldots, b_j\}$ be the indices of the range of feature values spanned by $\mathcal{R}$ in the j-th dimension. Formally, we have $\text{lb}(\mathcal{R}, j) \leq \mathcal{L}_j[a_j].\text{val} \leq ... \leq \mathcal{L}_j[b_j].\text{val} \leq \text{rb}(\mathcal{R}, j)$. So, all

the data points contained in $\mathcal{R}$ are associated with feature values in $\mathcal{L}_j$ between indices a_j and b_j, but the converse is not true. Moreover, we can identify a_j, b_j for all j in $\mathcal{O}(n \lg m)$ time. We maintain pointers to a_j and b_j, to keep track of the current rectangle $\mathcal{R}$.

We maintain the following invariants on the active flags: all feature values are active in $\mathcal{L}_1$. All feature values that are active in $\mathcal{L}_j$, for $j > 1$, represent data points that are contained in the current rectangle $\mathcal{R}$ with respect to feature dimensions $\{1, ..., j-1\}$. So, by definition, the feature values that are active in $\mathcal{L}_n$ and are in the range $\{a_n, a_n + 1, \ldots, b_n\}$ represent points that are in rectangle $\mathcal{R}$.

We have the following recursive scheme to iteratively expand $\mathcal{R}$: this describes how the COLLECT function operates. Suppose we expand dimension j in the positive direction, without loss of generality. Now $\text{rb}(\mathcal{R}, j) = \gamma$. We increment b_j, stopping when $\mathcal{L}_j[b_j].\text{val} \leq \gamma$ but $\mathcal{L}_j[b_j+1].\text{val} > \gamma$. Let b'_j be this new value of b_j. For any feature values that we encounter in $\mathcal{L}_j[b_j], ..., \mathcal{L}_j[b'_j]$ that were active in $\mathcal{L}_j$, we find the position of their associated data point's feature values in $\mathcal{L}_{j+1}$ using the back pointers and index $\mathcal{I}$. We then make their feature values active in $\mathcal{L}_{j+1}$. If they are contained in $\{a_{j+1}, \ldots, b_{j+1}\}$, then we repeat the process for dimensions $j+2, ...$, etc. Finally, if $j = n$ and the active points are in the range $\{a_n, \ldots, b_n\}$, then we add these points to the set S.

Finally, we comment on the ROLLBACK and COMMIT operations. Conceptually, COLLECT is reversible, provided we had all the old values of the

pointers a_j and b_j. Thus, we can easily reverse the operation. Similarly, COMMIT need not actually do anything, other than erase the information necessary to perform a ROLLBACK.

It is clear that the COLLECT operations can only touch each data point at most $\mathcal{O}(n)$ times, since a data point can only be activated once in each dimension. Moreover, a point that is *de-activated* by a ROLLBACK will never become active later with respect to that dimension. This means we will never propagate the point to next active list, so overall each point requires $\mathcal{O}(n)$ time to process, since it was activated in each dimension only once. Thus, ROLLBACK is at most as expensive as `Collect`, so the total running time after INITIALIZE is $\mathcal{O}((m + \text{iter}_{\max})n)$. □

Initially, we record the precision of each rectangle in the tree T. As a pre-processing step, we then discard all points contained in all the rectangles $\mathcal{R}_1, \ldots, \mathcal{R}_z$. We apply Algorithm 1 to each rectangle $\mathcal{R}_1, \ldots, \mathcal{R}_z$ which satisfies $pr(\mathcal{R}_i) \geq pr(T)$. That is, we only mark rectangles that have precision higher than the overall precision (of class 1) of the tree T for expansion. After expanding $\mathcal{R}_i$, we discard all points contained in the expanded rectangle before processing $\mathcal{R}_{i+1}$. This is required since, after expanding the rectangles, they may no longer be mutually disjoint. Let us denote by $T^\star$, the rule-based classifier obtained after this process. It can be proved using simple calculations that $pr(T^\star) \geq (1 - \delta) \cdot pr(T)$, which guarantees that the precision does not decrease by more than a $(1 - \delta)$ factor on the available training data. It is

important to note that this does not ensure that the precision of the original tree will only drop by a $(1 - \delta)$ factor on any new (e.g., testing) data.

This process of applying Algorithm 1 to rectangles iteratively illustrates only one heuristic, and many choices are possible. We have experimented with the following alternatives: i) changing the order in which the input rectangles are processed - for example in decreasing order of number of target samples covered; ii) changing the order in which features are processed - such as in increasing order of depth of the feature in the decision tree, or ordering features based on their Gini importance, and; iii) expanding rectangles uniformly, i.e., expand each feature simultaneously during one iteration. We found all variants exhibit similar performance, with a slight edge given to the heuristic that orders rectangles in decreasing order of the number of target samples covered, and then expands them uniformly. Thus, in later evaluation we applied this heuristic.

Setting hyperparameters. Algorithm 1 depends on three hyperparameters - β, δ and $\text{iter}_{\max}$. We found after preliminary experimentation that the dependence of the recall and precision on parameters δ and $\text{iter}_{\max}$ was monotone: higher values typically yield higher recall and lower precision if all other parameters are fixed. We also found that smaller values of β are advantageous as they result in more gradual expansion of the rectangles, avoiding early termination of the algorithm. For all our experiments, we set $\beta = 0.01$ (corresponding to a 1% increase in the width of the rectangle) and then tune $\text{iter}_{\max}$ automatically using a grid-search approach that maximizes F_1-score

(defined later) on sampled subsets of the target data. Thus, from a user perspective, they need only specify δ, the multiplicative precision loss parameter, in order to use our algorithm. For our experiments we used $\delta = 0.1$.

5.2.3 Mixed Integer Linear Programming approach

We now describe a mixed integer linear programming (MILP) approach to find the exact solution to the optimal expansion problem. For a given decision tree T, recall that $\mathcal{R} = \{\mathcal{R}_1, ..., \mathcal{R}_z\}$ denotes the set of disjoint axis-aligned rectangles corresponding to the leaves labelled 1 in T which satisfy $\text{Precision}_1(\mathcal{R}_i) \geq \text{Precision}_1(T)$. We denote the corresponding output (i.e., expanded) rectangles by $\mathcal{R}'_1, \mathcal{R}'_2, \ldots, \mathcal{R}'_z$. Similar to our heuristics, we conceptually delete all the points contained in any rectangle in the set $\mathcal{R}$ from the dataset by marking them.

We now present our mixed integer linear program, denoted $\mathcal{B}$, which captures the optimal expansion problem.

Variables: For each rectangle $\mathcal{R}_i$ and each feature $j \in n$, we have continuous-valued variables $z^l_{i,j}$ representing the left boundary of $\mathcal{R}'_i$ (i.e. new left boundary of $\mathcal{R}_i$ after expansion), in the dimension corresponding to feature j and similarly, variables $z^r_{i,j}$ for the corresponding right boundaries of $\mathcal{R}'_i$.

To ensure that each boundary only expands and never contracts, we

add the following constraints:

$$\min\left\{\text{lb}(\mathcal{R}_i, j), \min_{i\in\{1,\ldots,m\}} x_{i,j}\right\} \leq z^l_{i,j} \leq \text{lb}(\mathcal{R}_i, j) \text{ and}$$
$$\max\left\{\text{rb}(\mathcal{R}_i, j), \max_{i\in\{1,\ldots,m\}} x_{i,j}\right\} \geq z^r_{i,j} \geq \text{rb}(\mathcal{R}_i, j) \ .$$

For each rectangle $\mathcal{R}_i$ and data point (x_j, y_j), we have variables $b_{i,j}$. Each variable $b_{i,j}$ is to be set to 1 if point j is inside rectangle $\mathcal{R}'_i$, and set to 0 otherwise. To achieve this, we add constraints: $b_{i,j} = \bigwedge_{k\in\{1,\ldots,n\}} d_{j,k}$, where variable $d_{j,k} = 1$ iff $x_{j,k} \in [\text{lb}(\mathcal{R}_i, k), \text{rb}(\mathcal{R}_i, k)]$: this can be encoded via a pair of indicator constraints. Further, for each data point (x_j, y_j), we have binary variables $c_j = \bigvee_{\mathcal{R}_i\in\mathcal{R}} b_{i,j}$ where the logical-OR is over all rectangles $\mathcal{R}_i$. In other words, variable c_j is 1 if point j is included in some rectangle from the set $\mathcal{R}$, and is 0 otherwise. For each rectangle $\mathcal{R}_i$ and each data point (x_j, y_j), we have binary variables $a_{i,j}$. Variable $a_{i,j}$ is set to 0 if data point x_j is outside rectangle $\mathcal{R}'_i$; it is set to 0 or 1 otherwise. Additionally, for each data point (x_j, y_j), $a_{i,j}$ is set to 1 for exactly one rectangle $\mathcal{R}'_i$ it is in unless it is not covered by any rectangle. Thus, we observe that: $c_j = \sum_{\mathcal{R}_i\in\mathcal{R}} a_{i,j}$ for $j \in \{1, \ldots, n\}$.

Constraints to limit precision drop: For each rectangle $\mathcal{R}_i$, we require: $\text{Precision}_1(\mathcal{R}'_i) \geq (1-\delta)\text{Precision}_1(\mathcal{R}_i)$. This is achieved by having the following constraints in $\mathcal{B}$:

$$\frac{\sum_{y_j=1} a_{i,j} + \sum_{(x_j,y_j)\in\mathcal{R}_i} y_j}{\sum_{j\in\{1,\ldots,n\}} a_{i,j} + \sum_{(x_j,y_j)\in\mathcal{R}_i} 1} \geq (1-\delta)\mathcal{C}(T) \ , \tag{5.1}$$

where $\mathcal{C}(T)$ is a constant that depends only on T. Thus, this is a set of linear constraints in terms of variables $a_{i,j}$.

Objective function: Our linear program $\mathcal{B}$ uses the following objective function which corresponds exactly to maximizing the recall:

Maximize $\sum_{y_j=1} c_j$, subject to the constraints described above.

5.2.4 Evaluating our heuristics

In this section, we describe the experimental evaluation of our heuristics in the transfer learning setting for **7** public datasets. We used the following datasets, which are publicly available at UCI machine learning repository [14].

Table 5.1: Dataset information: n is the number of features in the dataset, Src size is the number of samples in the source data, and Src imbalance ratio (Tgt imbalance ratio) gives the fraction of samples in source data (respectively target data) from Class 1.

Dataset	n	Src size	Src imbalance ratio	Tgt imbalance ratio
CreditAU [81]	14	468	0.44	0.45
Wine [36]	11	4898	0.66	0.53
Letter [43]	16	10822	0.055	0.099
Digits [87]	64	5620	0.099	0.1
Landmine	9	8535	0.05	0.07
CreditDE	19	690	0.27	0.35
Invert [87]	784	2000	0.1	0.1

Baseline and State-of-art Competition of Algorithms: Now we describe the transfer learning algorithms with which we compare our heuristics:

SrcOnly: The baseline approach that trains a model only with the source domain data without using any target domain data, but with hyperparameters tuned on the source data.

TgtOnly: Another baseline approach which trains a model only with

the target domain data without transfer learning, and the same hyperparameters as the source model.

STRUT [87]: The `STRUT` (structure transfer) algorithm transfers the structure of the decision tree learned from the source domain and chooses new thresholds for each node of the decision tree based on the target domain data.

SER: [87]: The `SER` (structure expansion/reduction) algorithm can refine the decision tree learned from the source domain by specializing the rules of decision tree with the expansion transformation or generalizing the rules with the reduction transformation. We use the implementations of `STRUT` and `SER` from the code used in [87] (can be obtained from `http://tiny.cc/kgz5dz`).

Our approach, denoted **GeoX**, is to apply our geometric heuristics to baseline **X**, where **X** can be any of the above-mentioned baselines.

We also compare our approach against the exact solution to the Optimal Expansion problem obtained via the mixed integer linear programming (**MILP**) approach described in section 5.2.3. In our experiments, we compare heuristics `GeoSER` to this MILP based expansion applied after running `SER`, which we denote as **MILPSER**.

Our goal is to increase the recall of the model for class 1, while minimizing the loss in terms of precision. To ensure that, we examine the recall of class 1, precision of class 1, and F_1-score of the model. The latter metric is the harmonic mean of precision and recall (i.e., $2(\text{precision} \times \text{recall})/(\text{precision} +$

recall)) which summarizes the performance of the binary classifier for class 1, giving equal importance to recall and precision.

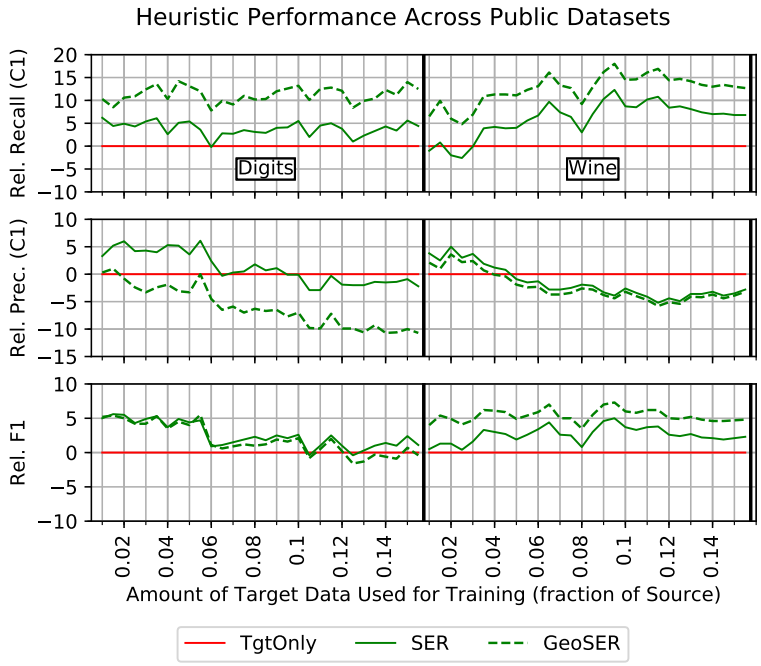

Figure 5.2: `Digits` and `Wine` datasets: The top chart shows the recall of class 1 (y-axis) for each heuristic, relative to the `TgtOnly` baseline, when the amount of target data available, as a fraction of the total source data, is varied (x-axis). Middle chart shows the relative precision of class 1 (y-axis), whereas the bottom chart shows the *relative* difference between the F_1-score of `TgtOnly` and the other approaches.

Varying the fraction of target data:

Figure 5.2 shows the results of the `TgtOnly`, `SER` and `GeoSER` algorithms on two of the public datasets (`Digits` and `Wine`). In the figure we show the results for each of the metrics (recall, precision, and F_1-score) with respect to `TgtOnly`, and for different amounts of data used from the target domain.

This is to give a sense of how much better or worse the approach is doing, compared to the naïve baseline. The figure shows that, for various amounts of target data, up to around a 0.1 fraction of the source data, that both `SER` and `GeoSER` have better F_1-scores than `TgtOnly`. This indicates firstly that there is some potential gain by using these transfer learning methods. Secondly, we see that there is a distinct trade-off in terms of recall and precision: as expected, `SER` tends to have higher precision, and lower recall than `GeoSER`. Thirdly, this trade-off is relatively stable as the fraction of target data varies.

Table 5.2: The recall, precision and F_1-scores for `MILPSER` and `GeoSER` when the target size is 5% of source size. The MILP solver did not terminate within 8 hours for the Wine data set.

	Recall			Precision			F_1-score		
Dataset	SER	MILP SER	Geo SER	SER	MILP SER	Geo SER	SER	MILP SER	Geo SER
CreditAU	83.1	88.2	86.4	87.1	85.5	87.0	85.0	86.8	86.6
Wine	70.6	?	77.7	72.3	?	71.3	71.0	?	74.0
Letter	74.8	83.2	84.3	68.5	60.5	59.6	71.4	69.9	69.7
Digits	69.1	74.5	76.8	68.9	60.5	62.0	68.7	66.5	68.3
Landmine	23.7	33.4	34.8	22.7	16.4	16.5	23.0	22.0	22.3
CreditDE	28.5	33.1	29.0	49.5	46	47.2	35.9	38.1	35.7
Invert	29.3	49.7	38.5	38.1	33.1	27.5	31.1	38.4	30.9

Comparison to MILP optimal solution:

Table 5.2 compares `MILPSER` and `GeoSER` with respect to recall, precision and F_1 for a target data set that is 5% of the source data size. We observe that both `GeoSER` and `MILPSER` provide similar recall-precision trade-offs with very similar F_1-scores for most datasets.

In Table 5.3 we show the average running times (over 20 runs) of these

two heuristics, along with the multiplicative speedup obtained by using GeoSER over MILPSER. In many cases, GeoSER offers a significant speedup over MILPSER, often greater than a factor of 100 when the target dataset contains a few hundred points. In the extreme case of the Wine dataset, the MILP solver did not terminate within 8 hours, so we can only estimate the speedup. Importantly, the Wine dataset is relatively balanced, meaning that there are several hundred class 1 samples in the target dataset. Wine is unique in this sense, as the other datasets are highly imbalanced. This shows an issue of scalability with the MILPSER solution. Finally, there are a few exceptional cases where MILPSER is faster than GeoSER. However, in these cases the running times of both approaches are between 50-200 milliseconds. Overall, based on these results, we conclude that GeoSER consistently outperforms MILPSER in terms of running time, while providing similar recall-precision trade-offs.

Table 5.3: Comparing the average running time of MILPSER with that of GeoSER in seconds. The Speedup column gives the multiplicative speedup of GeoSer over MILPSER. For example, for the Letter dataset, when the target size is 10% of source size, GeoSER is 178.94 times faster than MILPSER.

	Tgt size = 5% of Src size			Tgt size = 10% of Src size		
Dataset	MILP SER	Geo SER	Speedup	MILP SER	Geo SER	Speedup
CreditAU	0.062	0.054	1.14	0.13	0.10	1.3
Wine	$>$ 8 hrs	0.23	$>$125819	$>$ 8 hrs	0.32	$>$90851
Letter	24.27	0.49	49.53	216.52	1.21	178.94
Digits	0.45	0.10	4.5	2.81	0.32	8.78
Landmine	13.94	0.28	49.78	161.3	0.81	199.13
CreditDE	0.069	0.19	0.36	0.062	0.126	0.49
Invert	0.75	0.69	1.08	1.62	1.38	1.17

5.3 Transfer learning algorithms for decision trees in regression tasks

In this section, we present our modifications of existing transfer learning algorithms for decision trees, which were designed for classification problems, so as to make them work for regression problems. We then propose a new and simpler algorithm for the same problem, and follow up with some experimental comparison of these algorithms.

For this part, we denote by T_S the tree trained on the source dataset $\mathcal{D}_s$ and by $\mathcal{D}_t = \{(x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}), y_i), i \in \{1, \ldots, m\}\}$ the target dataset. Recall that our transfer learning algorithms get T_S and $\mathcal{D}_t$ as input, but not $\mathcal{D}_s$.

5.3.1 Existing algorithms for transfer learning with decision trees

In this part, we describe the existing transfer learning algorithms `STRUT` and `SER` due to Segev *et al.* [87]. These algorithms are designed for classification tasks.

STRUT is based on preserving the structure of the source-trained decision tree T_S and modifying only the threshold value at each node.

It traverses the tree T_S top-down, keeping track of the samples from the target data set reaching each node. So at any step, it focuses on a pair $(v, \mathcal{D})$ where v is the current node, and $\mathcal{D}$ is the data set reaching node v. It is initially called with the root node of T_S and the entire target dataset $\mathcal{D}_t$. If $\mathcal{D}$ is empty, the branch is pruned away by making v into a leaf. Otherwise,

the threshold value of v is set using a 2-step process (note that the feature compared by the node v is kept unchanged):

1) The first step shortlists candidates for the new threshold value of the node consisting of local maxima of information gain (IG) (see section 5.1 for definition). Concretely, for a fixed constant $\epsilon > 0$, a value a is shortlisted as a candidate threshold value if the information gain at a is larger than the information gain in its ϵ-neighborhood i.e. at any other point $b \in [a-\epsilon, a+\epsilon]$.

2) During the second step, among these candidate threshold values, it chooses the "best" value by comparing the Jensen-Shannon divergence of the distributions of the source and target data reaching the children of the node, and chooses the threshold value with the least JS divergence between the two distributions. It also considers swapping the two children, if the JS divergence of the swapped distributions is lower.

More concretely, it chooses the threshold value for which the *divergence gain (DG)* is maximized. If node v had data $\mathcal{D}_{s,v}$ from source data set reaching it, and data $\mathcal{D}_{t,v}$ from target data set reaching it, then for a candidate threshold value a, the divergence gain is defined as follows:

$$DG(\mathcal{D}_{s,v}, \mathcal{D}_{t,v}, v, a) = 1 - \frac{|\mathcal{D}_{t,L}|}{|\mathcal{D}_{t,v}|} JSD(\mathcal{D}_{s,L}, \mathcal{D}_{t,L}) - \frac{|\mathcal{D}_{t,R}|}{|\mathcal{D}_{t,v}|} JSD(\mathcal{D}_{s,R}, \mathcal{D}_{t,R}) \tag{5.2}$$

Here, $\mathcal{D}_{t,L} \subseteq \mathcal{D}_{t,v}$ is the subset of points from the target data set reaching the left child of v if a is used as the threshold value. Similarly,

$\mathcal{D}_{t,R} \subseteq \mathcal{D}_{t,v}$ is the subset reaching the right child if a is used as threshold. $\mathcal{D}_{s,L} \subseteq \mathcal{D}_{s,v}$ and $\mathcal{D}_{s,R} \subseteq \mathcal{D}_{s,v}$ are respectively, the subsets of data points from the source data reaching the left and right child respectively of node v. Also, we abuse notation, so that $JSD(\mathcal{D}_{s,L}, \mathcal{D}_{t,L})$ corresponds to the JS divergence between the empirical label distributions of data sets $\mathcal{D}_{s,L}$ and $\mathcal{D}_{t,L}$, and similarly for $JSD(\mathcal{D}_{s,R}, \mathcal{D}_{t,R})$.

SER, or structured expansion and reduction, due to Segev et al [87] performs certain expansion and reduction steps on the given decision tree $T_{\mathbb{S}}$ based on the target data $\mathcal{D}_t$.

In the expansion step, this algorithm traverses the tree $T_{\mathbb{S}}$ top-down keeping track of the samples from target data $\mathcal{D}_t$ reaching each node until it reaches the leaves. It then expands each leaf based on the target data reaching that leaf, similar to the learning algorithm described in Section 5.1. It uses information gain (IG) as the splitting criterion to choose the feature and threshold values for the query at each node.

In the reduction step, it then prunes away the tree in a bottom-up fashion. It does this by computing the subtree error for a node, which is the empirical classification error of the subtree rooted at that node, and the leaf-error, which is the empirical classification error if that node were to be converted into a leaf. The node is then converted into a leaf if the leaf-error is smaller than the subtree error, and it is kept intact otherwise.

5.3.2 Adapting transfer learning algorithms for regression

In this part, we describe our adaptations of the transfer learning algorithms `STRUT` and `SER` to the regression setting. We denote them for convenience by `STRUTReg` and `SERReg`, respectively.

`STRUTReg`: This algorithm is similar to `STRUT` in that it preserves the structure of the source decision tree $T_{\mathcal{S}}$ as well as the feature queried at each node, and modifies only the threshold value for the node.

It follows the same structure as `STRUT`, traversing the tree $T_{\mathcal{S}}$ in a top-down manner. At any step, it considers a node v and the data set $\mathcal{D}$ reaching v. If $\mathcal{D}$ is empty, v is converted into a leaf, pruning the branch away. If $\mathcal{D}$ is not empty, the threshold value of v is selected using the following 2-step process (which is where it differs from `STRUT`):

1) Candidate threshold values are shortlisted based on local maxima of variance gain (VG) (see Section 5.1 for definition) instead of information gain, as is common for regression tasks. `STRUTReg` shortlists a value a as a candidate threshold value if the variance gain at a is larger than the variance gain in its ϵ-neighborhoood, for some fixed constant $\epsilon > 0$.

2) During the second step, similar to `STRUT`, it chooses from among these candidate threshold values based on a measure of dissimilarity between the distributions of the source and target data reaching the children of the node if that value were to be used as threshold. However, it estimates the dissimilarity between the source and target empirical distributions reaching

the children of the node by the distance based on the Kolmogorov–Smirnov (KS) test. For two empirical distributions with cumulative density functions F, G, the KS distance is defined as $\sup_x |F(x) - G(x)|$. `STRUTReg` now chooses the threshold value for which the gain based on KS distance is maximized. This gain is calculated from equation 5.2 by replacing JSD by the KS distance between the two distributions. Similar to `STRUT`, it also considers exchanging the left and right children if the resultant gain is higher.

We note that computing the KS distance necessitates storing of cumulative distribution function (CDF) of the source samples reaching each node. This adds a significant overhead to the tree structure as well as the running time, making it proportional to the size of the source dataset. Moreover, it is not entirely fair to claim that such an algorithm does not require access to the source data, as these stored CDFs certainly contain a lot of information about the source dataset, which is a drawback of the algorithm.

`SERReg` is a slight modification of the structure expansion reduction (`SER`) algorithm, necessary for the regression setting.

Similar to `SER`, it goes through expansion and reduction steps on the given decision tree $T_{\mathbb{S}}$ based on the target data $\mathcal{D}_t$. The only modifications `SERReg` requires is that the expansion step uses the criterion of variance gain (VG) instead of information gain, as is typical for regression settings. Also, during the reduction step, the leaf-errors and subtree errors are now computed using *Root Mean Square (RMS) Error*, instead of classification error.

5.3.3 A simple transfer algorithm for decision trees in regression setting

We now describe a simple algorithm, that we call `PredShift`, which preserves the internal structure of the source tree, including the threshold values. The only adjustment it makes is to *shift* the *predicted values* of the leaves of the source tree according to the relevant target data. Recall that the predicted value of a leaf of a regression tree is the value output by the tree for any input data point that reaches that leaf. `PredShift` proceeds by traversing the nodes of the source tree top-down, keeping track of the target samples reaching each node. We present this procedure formally in Algorithm 2.

Algorithm 2 PredShift$(v, \mathcal{D})$

Require: Node v, dataset $\mathcal{D}$

if $|\mathcal{D}| \leq 1$ **then**
    Make v a leaf
if v is a Leaf **then**
    if $|\mathcal{D}| = 0$ **then**
        Predicted-value$(v) \leftarrow$ EmptyLeafPredict(v)
    else
        Predicted-value$(v) \leftarrow \left(\sum_{(x,y)\in\mathcal{D}} y\right) / |\mathcal{D}|$
else
    PredShift$(\text{Left}(v), \mathcal{D}_{\text{left}})$
    PredShift$(\text{Right}(v), \mathcal{D}_{\text{right}})$

Algorithm 2 is invoked with the root node of the tree T_{s} and the target dataset $\mathcal{D}_t$. If the data $\mathcal{D}$ reaching node v has at most one sample, one of the children of node v (if it has children) must have no target data reaching it, and

the algorithm therefore chooses to convert v into a leaf (if it is not already).

For internal nodes v, it simply partitions the data $\mathcal{D}$ into $\mathcal{D}_{\text{left}}$ and $\mathcal{D}_{\text{right}}$, where $\mathcal{D}_{\text{left}}$ ($\mathcal{D}_{\text{right}}$) is the subset of $\mathcal{D}$ reaching the left child of v (respectively, the right child of v). The left and right children of v are denoted by $\textsc{Left}(v)$ and $\textsc{Right}(v)$, respectively. The algorithm `PredShift` is then recursively called on these two children with the corresponding datasets reaching them.

For leaf nodes, lines 3 - 7 modify the predicted value for the node based on the target data reaching it. If the leaf is not empty, i.e., if it is reached by at least one target data point, then line 7 sets the predicted value of the leaf to be the mean of the y values for these target samples. On the other hand, if a leaf is empty, then line 5 invokes a method called `EmptyLeafPredict` to estimate its predicted value. During preliminary experimentation, we observed that the error rates for algorithm 2 are highly dependent on the the details of `EmptyLeafPredict`. Since the target data is sparse, quite a few leaves end up with no target data, meaning that the choice of `EmptyLeafPredict` is crucial. Different implementations for `EmptyLeafPredict` are possible, and we present the following two variants of our `PredShift` algorithm:

1. `PredShift-A`: `EmptyLeafPredict` returns the mean of the labels y of all the target samples reaching the parent of v in the tree.

2. `PredShift-B`: `EmptyLeafPredict` does not change the predicted value of the leaf, i.e., it is the same as in the source tree T_{s}. Thus, in the case

that a node v is converted into a leaf due to our pruning process (line 2), its predicted value is the mean of the labels y of all the source samples reaching node v.

5.3.4 Evaluation of our transfer learning algorithms

We now present an experimental evaluation of the different transfer learning algorithms mentioned above on some public datasets on random forests of regression trees. Table 5.4 presents the details of the datasets used. All the datasets are publicly available at UCI machine learning repository [14].

Table 5.4: Dataset information: n is the number of features, Src size is the number of samples in the source data, Src mean is the sample mean of the y values of source data and y-unit is the unit of measurement of y values in the dataset

Dataset	n	Src size	Src mean	y-unit
AutoMPG	6	266	20.5	MPG
ConcreteStrength [99]	7	824	32.8	MPa
Housing	12	338	24.5	1000 $s
Automobile	24	134	10.6	1000 $s
AquaticToxicity [27]	7	437	4.5	(-log(mol/ltr))
Wine [36]	11	4898	5.9	Rating
RealEstate [100]	5	332	37.9	10000 NT$/Ping
Energy [26]	25	1000	82.1	Wh

Table 5.5 records the RMS errors for all datasets when the target data available is fixed to be 5% of source size.

We observe that `SERReg` and `PredShift` perform quite consistently across the datasets, with a few exceptions. `SERReg` outperforms the baseline `TgtOnly` for most datasets, with the notable exceptions of `ConcreteStrength` and `AutoMPG`. It has the lowest error rates for 4 datasets when the target

Table 5.5: RMS Error for different algorithms when the target size is 5% of source size.

Dataset	SrcOnly	TgtOnly	STRUTReg	SERReg	Pred shift	
					A	B
AutoMPG	8.75	7.06	**4.82**	8.17	6.6	6.92
Concrete Strength	17.07	7.16	9.76	15.55	**6.38**	9.72
Housing	7.28	6.9	8.36	**6.12**	7.52	6.91
Automobile	6.38	7.16	5.95	5.82	6.01	**5.66**
Aquatic Toxicity	1.5	1.56	1.49	**1.38**	1.43	1.42
Wine	1.09	0.68	0.7	**0.67**	0.69	0.69
RealEstate	**9.2**	12.45	11.27	9.44	10.29	9.94
Energy	26.64	22.96	22.93	21.65	20.61	**19.79**

data size is fixed to 5% of source data. Similarly, `PredShift-A` gives consistently lower RMS error rates across datasets, outperforming `TgtOnly` for all except the `Housing` and `Wine` datasets. Note that `PredShift-A` is the only algorithm outperforming `TgtOnly` for the `ConcreteStrength` dataset. In contrast, `STRUTReg` performs poorly with the exception of the `AutoMPG` dataset.

`PredShift-B` exhibits the lowest error rates for two of the datasets, and we observe that its performance for most of the other datasets is also at par with the competitors.

Table 5.6 records the running times of the different transfer learning algorithms, along with the multiplicative speedup achieved by `PredShift-A` in comparison. As expected, `PredShift-A` is much faster: up to 80 times faster than SERReg. Moreover, this speedup of `PredShift-A` is more pronounced for larger datasets such as `Wine` and `Energy`. The running times of `PredShift-B` were similar to those of `PredShift-A`, and we excluded them from the table

to avoid crowding.

Table 5.6: Transfer time in milliseconds with the target size fixed to 5% of source size. The number in parenthesis describes the multiplicative speedup achieved by `PredShift-A` over that algorithm e.g. for the Wine dataset, `PredShift-A` runs 80 times faster than `SERReg`

Dataset	`TgtOnly`	`STRUTReg`	`SERReg`	`Pred Shift-A`
AutoMPG	6.2(12)	10.6(20)	3.4(6)	0.5
Concrete Strength	44.7(17)	212.2(80)	7.5(3)	2.7
Housing	13.2(21)	14.3(23)	6.1(10)	0.6
Automobile	4.4(22)	1.6(8)	3.2(16)	0.2
Aquatic Toxicity	20.2(22)	23.9(27)	5.9(7)	0.9
Wine	1432.1(333)	18150.1(4221)	345.3(80)	4.3
RealEstate	13.9(17)	18.1(23)	4.3(5)	0.8
Energy	256.4(117)	126.2(57)	19.0(9)	2.2

In conclusion, we remark that two of these algorithms, `PredShift` and `SERReg` provide positive transfer with respect to RMS error over a wide range of public datasets. Our `PredShift` algorithm has a distinct advantage of being extremely simple to implement. Of particular importance is how such algorithms handle leaves containing no target data, and it would be interesting to develop and compare other heuristics for this case.

Bibliography

[1] Scott Aaronson, Shalev Ben-David, Robin Kothari, and Avishay Tal. Quantum Implications of Huang's Sensitivity Theorem. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:66, 2020.

[2] Alfred V Aho, Jeffrey D Ullman, and Mihalis Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 133–139. ACM, 1983.

[3] Reem Al-Otaibi, Ricardo B. C. Prudêncio, Meelis Kull, and Peter Flach. Versatile decision trees for learning over multiple contexts. In *Machine Learning and Knowledge Discovery in Databases*, pages 184–199, 2015.

[4] Kazuyuki Amano. Minterm-transitive functions with asymptotically smallest block sensitivity. *Inf. Process. Lett.*, 111(23-24):1081–1084, 2011.

[5] Andris Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006.

[6] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *J. ACM*, 64(5), September 2017.

[7] Andris Ambainis, Mohammad Bavarian, Yihan Gao, Jieming Mao, Xiaoming Sun, and Song Zuo. Tighter Relations between Sensitivity and Other Complexity Measures. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 101–113, 2014.

[8] Andris Ambainis and Krišjānis Prūsis. A Tight Lower Bound on Certificate Complexity in Terms of Block Sensitivity and Sensitivity. In *Proceedings of the 39th International Symposium, Mathematical Foundations of Computer Science (MFCS)*, pages 33–44, 2014.

[9] Andris Ambainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Sensitivity versus Certificate Complexity of Boolean Functions. In *Proceedings of the 11th International Computer Science Symposium in Russia (CSR)*, pages 16–28, 2016.

[10] Andris Ambainis and Xiaoming Sun. New separation between *s(f)* and *bs(f)*. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:116, 2011.

[11] Andris Ambainis and Jevgēnijs Vihrovs. Size of Sets with Small Sensitivity: A Generalization of Simon's Lemma. In *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 122–133, 2015.

[12] Bogdan Armaselu and Ovidiu Daescu. Maximum area rectangle separating red and blue points. In *Proceedings of the 28th Canadian Conference*

on Computational Geometry, CCCG, pages 244–251, 2016.

[13] Srinivasan Arunachalam, Sourav Chakraborty, Michal Koucký, Nitin Saurabh, and Ronald de Wolf. Improved bounds on Fourier entropy and min-entropy. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020*, volume 154 of *LIPIcs*, pages 45:1–45:19, 2020.

[14] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[15] Jonathan Backer and J. Mark Keil. The mono- and bichromatic empty rectangle and square problems in all dimensions. In *LATIN 2010: Theoretical Informatics*, pages 14–25, 2010.

[16] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, pages 81:1–81:13, 2016.

[17] Mitali Bafna, Satyanarayana V. Lokam, Sébastien Tavenas, and Ameya Velingker. On the sensitivity conjecture for read-k formulas. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 16:1–16:14, 2016.

[18] Jérémy Barbay, Timothy M. Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in $O(n^2)$ time (and better).

Inf. Process. Lett., 114(8):437–445, 2014.

[19] Richard Beigel and Anna Bernasconi. A note on the polynomial representation of Boolean functions over GF(2). *International Journal of Foundations of Computer Science*, 10(04):535–542, 1999.

[20] Shalev Ben-David, Pooya Hatami, and Avishay Tal. Low-sensitivity functions from unambiguous certificates. In *Proceedings of Innovations in Theoretical Computer Science Conference (ITCS)*, pages 28:1–28:23, 2017.

[21] A. Bernasconi and B. Codenotti. Spectral analysis of Boolean functions as a graph eigenvalue problem. *IEEE Transactions on Computers*, 48(3):345–351, 1999.

[22] Eric Blais, Clément L Canonne, Igor C Oliveira, Rocco A Servedio, and Li-Yang Tan. Learning circuits with few negations. *arXiv preprint arXiv:1410.8420*, 2014.

[23] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes (extended abstract). In *28th Annual Symposium on Foundations of Computer Science*, pages 118–126, 1987.

[24] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[25] Harry Buhrman and Ronald De Wolf. Complexity Measures and Decision Tree Complexity: A Survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[26] Luis M. Candanedo, Veronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81 – 97, 2017.

[27] Matteo Cassotti, Davide Ballabio, Viviana Consonni, Andrea Mauri, Igor V Tetko, and Roberto Todeschini. Prediction of acute aquatic toxicity toward daphnia magna by using the GA-kNN method. *Alternatives to Laboratory Animals*, 42(1):31–41, 2014.

[28] Sourav Chakraborty. On the sensitivity of cyclically-invariant Boolean functions. *Discrete Mathematics & Theoretical Computer Science*, 13(4):51–60, 2011.

[29] J.-C Chang and H.-L Wu. The log-rank conjecture for read-k XOR functions. *Journal of Information Science and Engineering*, 34:391–399, 2018.

[30] Siddhesh Chaubal and Anna Gál. New constructions with quadratic separation between sensitivity and block sensitivity. In *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 13:1–13:16, 2018.

[31] Siddhesh Chaubal and Anna Gál. Diameter versus certificate complexity of Boolean functions. *Manuscript*, 2020.

[32] Siddhesh Chaubal and Anna Gál. Tight bounds on sensitivity and block sensitivity of some classes of transitive functions. In *LATIN 2020: Theoretical Informatics*, pages 323–335, 2020.

[33] Siddhesh Chaubal and Patrick K. Nicholson. Algorithms for transfer learning in regression forests. *Manuscript*, 2020.

[34] Siddhesh Chaubal, Mateusz Rzepecki, Patrick K. Nicholson, Guangyuan Piao, and Alessandra Sala. Geometric heuristics for transfer learning in decision trees. *Manuscript*, 2020.

[35] Stephen A. Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and lower time bounds for Parallel Random Access Machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, 1986.

[36] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[37] Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, June 2007.

[38] Krishnamoorthy Dinesh and Jayalal Sarma. Alternation, Sparsity and Sensitivity: Combinatorial Bounds and Exponential Gaps. In *Proceed-*

ings of the 4th International Conference on Algorithms and Discrete Applied Mathematics (CALDAM), pages 260–273, 2018.

[39] Krishnamoorthy Dinesh and Jayalal Sarma. Sensitivity, affine transforms and quantum communication complexity. In *Computing and Combinatorics - 25th International Conference, COCOON 2019, Proceedings*, volume 11653 of *Lecture Notes in Computer Science*, pages 140–152, 2019.

[40] Andrew Drucker. Block sensitivity of minterm-transitive functions. *Theor. Comput. Sci.*, 412(41):5796–5801, 2011.

[41] Jonathan Eckstein, Peter L. Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. The maximum box problem and its application to data analysis. *Comp. Opt. and Appl.*, 23(3):285–298, 2002.

[42] M. G. Find, A. Golovnev, E. A. Hirsch, and A. S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 89–98, Oct 2016.

[43] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

[44] Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some Boolean function complexity measures. *Combinatorica*, 36(3):265–311, Jun 2016.

[45] Mika Goos. Lower bounds for clique vs. independent set. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, page 1066–1076, 2015.

[46] Mika Goos, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016.

[47] Parikshit Gopalan, Rocco A. Servedio, Avishay Tal, and Avi Wigderson. Degree and Sensitivity: Tails of Two Distributions. *CoRR*, abs/1604.07432, 2016.

[48] Siyao Guo, Tal Malkin, Igor C. Oliveira, and Alon Rosen. The power of negations in cryptography. In *Theory of Cryptography*, pages 36–65, 2015.

[49] Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. Variations on the sensitivity conjecture. *Theory of Computing, Graduate Surveys*, 4:1–27, 2011.

[50] Kun He, Qian Li, and Xiaoming Sun. A tighter relation between sensitivity complexity and certificate complexity. *Theoretical Computer Science*, 762:1–12, 2019.

[51] Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

[52] Russell Impagliazzo and Moni Naor. Decision trees and downward closures. In *Proceedings: Third Annual Structure in Complexity Theory Conference*, pages 29–38, 1988.

[53] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions. In *29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988.

[54] Jeff Kahn, Michael Saks, and Dean Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4(4):297–306, 1984.

[55] Karthik C. S. and Sébastien Tavenas. On the sensitivity conjecture for disjunctive normal forms. In *Proceedings of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 15:1–15:15, 2016.

[56] Sebastian Kauschke and Johannes Fürnkranz. Batchwise patching of classifiers. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[57] Claire Kenyon and Samuel Kutin. Sensitivity, block sensitivity, and l-block sensitivity of Boolean functions. *Inf. Comput.*, 189(1):43–53, February 2004.

[58] Daniel J Kleitman and David J Kwiatkowski. Further results on the Aanderaa-Rosenberg conjecture. *Journal of Combinatorial Theory, Series B*, 28(1):85–95, 1980.

[59] Raghav Kulkarni and Miklos Santha. Query complexity of matroids. In *Algorithms and Complexity*, pages 300–311, 2013.

[60] Sophie Laplante, Reza Naserasr, and Anupa Sunny. Sensitivity lower bounds from linear dependencies. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:14, 2020.

[61] Chengyu Lin and Shengyu Zhang. Sensitivity conjecture and log-rank conjecture for functions with small alternating numbers. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 51:1–51:13, 2017.

[62] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[63] W.-Y. Loh. Fifty years of classification and regression trees (with discussion). *International Statistical Review*, 34:329–370, 2014.

[64] László Lovász and Michael Saks. Lattices, mobius functions and communications complexity. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 81–90. IEEE Computer Society, 1988.

[65] László Lovász and Neal E. Young. Lecture notes on evasiveness of graph properties. *CoRR*, cs.CC/0205031, 2002.

[66] A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, October 1958.

[67] Kurt Mehlhorn and Erik M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 330–337. Association for Computing Machinery, 1982.

[68] Gatis Midrijanis. Exact quantum query complexity for total Boolean functions. *arXiv preprint quant-ph/0403168*, 2004.

[69] Ashley Montanaro and Tobias Osborne. On the communication complexity of XOR functions. *CoRR*, abs/0909.3392, 2009.

[70] Hiroki Morizumi. Limiting negations in formulas. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, page 701–712, 2009.

[71] Hiroki Morizumi. Limiting negations in non-deterministic circuits. *Theoretical Computer Science*, 410(38):3988 – 3994, 2009.

[72] Hiroki Morizumi. Sensitivity, block sensitivity, and certificate complexity of unate functions and read-once functions. In *Proceedings of 8th IFIP International Conference on Theoretical Computer Science*, pages 104–110, 2014.

[73] Sagnik Mukhopadhyay and Swagato Sanyal. Towards better separation between deterministic and randomized query complexity. In *Proceedings of the 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 206–220, 2015.

[74] Noam Nisan. CREW PRAMs and Decision Trees. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 327–335, 1989.

[75] Noam Nisan. CREW PRAMs and Decision Trees. *SIAM J. Comput.*, 20(6):999–1007, 1991.

[76] Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, Dec 1994.

[77] Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Comb.*, 15(4):557–565, 1995.

[78] Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.

[79] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[80] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.

[81] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[82] Fisher R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[83] Ronald L. Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371 – 384, 1976.

[84] Arnold L. Rosenberg. On the time required to recognize properties of graphs: A problem. *SIGACT News*, 5(4):15–16, October 1973.

[85] David Rubinstein. Sensitivity vs. block sensitivity of Boolean functions. *Combinatorica*, 15(2):297–299, Jun 1995.

[86] Miklos Santha and Christopher Wilson. Limiting negations in constant depth circuits. *SIAM Journal on Computing*, 22(2):294–302, 1993.

[87] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1811–1824, Sep. 2017.

[88] C. E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, Jan 1949.

[89] Hans-Ulrich Simon. A tight $\Omega(\log \log n)$-bound on the time for parallel RAM's to compute nondegenerated Boolean functions. *Information and Control*, 55(1):102 – 107, 1982.

[90] Xiaoming Sun. Block sensitivity of weakly symmetric functions. In *Proceedings of the Third International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 339–344, 2006.

[91] Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of Markov's theorem. In *Algorithms and Computation*, pages 108–116. Springer, 2003.

[92] Avishay Tal. Properties and applications of Boolean function composition. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 441–454, 2013.

[93] G. Tardos. Query complexity, or why is it difficult to separate $NP^A \cap coNP^A$ from P^A by random oracles A ? *Combinatorica*, 9(4):385–392, 1989.

[94] H. Y. Tsang, C. H. Wong, N. Xie, and S. Zhang. Fourier sparsity, spectral norm, and the log-rank conjecture. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 658–667, 2013.

[95] Hing Yin Tsang. On Boolean functions with low sensitivity. *Manuscript*, 2014.

[96] Hing Yin Tsang, Chung Hoi Wong, Ning Xie, and Shengyu Zhang. Fourier sparsity, spectral norm, and the log-rank conjecture. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–667, 2013.

[97] Madars Virza. Sensitivity versus block sensitivity of Boolean functions. *Information Processing Letters*, 111(9):433 – 435, 2011.

[98] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[99] I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797 – 1808, 1998.

[100] I-Cheng Yeh and Tzu-Kuang Hsu. Building real estate valuation models with comparative approach through case-based reasoning. *Appl. Soft Comput.*, 65(C):260–271, April 2018.

[101] Zhiqiang Zhang and Yaoyun Shi. Communication complexities of symmetric XOR functions. *Quantum Info. Comput.*, 9(3):255–263, March 2009.

Vita

Siddhesh Prashant Chaubal was born in Mumbai, India. He completed his high school in Thane and joined Indian Institute of Technology, Bombay in 2009. After receiving B.Tech. in computer science from IIT Bombay, he joined the PhD program in the department of computer science at the University of Texas at Austin in 2013.

Permanent address: siddhesh@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.