Nearest Neighbor Complexity and Boolean Circuits

Mason DiCicco
Worcester Polytechnic Institute, MA, USA

Vladimir Podolskii
Tufts University, Medford, MA, USA

Daniel Reichman
Worcester Polytechnic Institute, MA, USA

Abstract

A nearest neighbor representation of a Boolean function f is a set of vectors (anchors) labeled by 0 or 1 such that $f(\boldsymbol{x}) = 1$ if and only if the closest anchor to $\boldsymbol{x}$ is labeled by 1. This model was introduced by Hajnal, Liu and Turán [2022], who studied bounds on the minimum number of anchors required to represent Boolean functions under different choices of anchors (real vs. Boolean vectors) as well as the analogous model of k-nearest neighbors representations.

We initiate a systematic study of the representational power of nearest and k-nearest neighbors through Boolean circuit complexity. To this end, we establish a close connection between Boolean functions with polynomial nearest neighbor complexity and those that can be efficiently represented by classes based on linear inequalities – *min-plus polynomial threshold functions* – previously studied in relation to threshold circuits. This extends an observation of Hajnal et al. [2022]. Next, we further extend the connection between nearest neighbor representations and circuits to the k-nearest neighbors case.

As an outcome of these connections we obtain exponential lower bounds on the k-nearest neighbors complexity of explicit n-variate functions, assuming $k \leq n^{1-\epsilon}$. Previously, no superlinear lower bound was known for any $k > 1$. At the same time, we show that proving superpolynomial lower bounds for the k-nearest neighbors complexity of an explicit function for arbitrary k would require a breakthrough in circuit complexity. In addition, we prove an exponential separation between the nearest neighbor and k-nearest neighbors complexity (for unrestricted k) of an explicit function. These results address questions raised by [17] of proving strong lower bounds for k-nearest neighbors and understanding the role of the parameter k. Finally, we devise new bounds on the nearest neighbor complexity for several families of Boolean functions.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Complexity, Nearest Neighbors, Circuits

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.42

Related Version *Previous Version*: `https://arxiv.org/pdf/2402.06740`

Acknowledgements We would like to thank Bill Martin for several insightful comments. The second and third author thank the Simons Institute for the Theory of Computing where collaboration on this project has began.

1 Introduction

A *nearest-neighbor representation* of a function $f : \{0,1\}^n \to \{0,1\}$ is a set of vectors, called "anchors," say $S = P \cup N$ such that $f(\boldsymbol{x}) = 1$ if and only if the nearest anchor to $\boldsymbol{x}$ (under the Euclidean distance metric) belongs to P. The set of anchors can be seen as a (disjoint) union of "positive" and "negative" examples. If $S \subseteq \{0,1\}^n$, we refer to the representation as *Boolean*, and if $S \subseteq \mathbb{R}^n$ we call it *real*. This model was pioneered by [17], who advocated the study of Boolean functions admitting efficient representations, focusing on the *number of anchors* as a measure of the complexity of the representation. They also consider the *k-nearest neighbors* variation, where the value of f on input x is computed as a majority vote of the k nearest anchors to x.

In particular, [17] observed a relationship between nearest-neighbor representations and threshold circuits. Motivated by their work, we initiate a systematic study of the connections to circuit complexity. Some of our results are related to the *weight complexity* (number of bits) needed when representing Boolean functions by real anchors: A topic receiving recent attention by [24].

There are numerous extensions and variations of nearest neighbor complexity. While studying all of these here is infeasible, one goal of ours is to encourage further exploration of this broad topic. We discuss some future directions in the conclusion section.

1.1 Motivation

We believe that nearest neighbor representations are a natural and interesting model of computation worthy of study. Furthermore, nearest neighbor complexity is relevant to several central research topics listed next.

1.1.1 Boolean function complexity

Nearest neighbor representations are related to depth-two threshold circuits and decision trees: [17] establish that lower bounds for these models are useful for establishing lower bounds for nearest neighbors. Polynomial threshold functions and their relation to geometric representations of Boolean functions also bear a strong connection to nearest neighbor representations. In fact, we will show that the expressivity of nearest neighbors is essentially equivalent to that of polynomial threshold functions *over the tropical semiring* – An interesting model of computation due to [18].

Our approach to understanding nearest neighbor complexity in terms of Boolean circuit complexity follows a long tradition in computational learning theory and computational complexity (e.g., [29, 22, 26, 27, 41, 39]). Uncovering new connections between nearest neighbors and well-studied computational models (such as linear decision lists and depth-two circuits) enables us to prove new unconditional lower bounds and upper bounds on the nearest neighbor complexity of explicit[1] functions. It also allows us to phrase some open problems in circuit complexity in terms of nearest neighbor complexity. For instance, the difficulty of proving super-polynomial lower bounds for k-nearest neighbors representations – for appropriate values of k – follows from representations (that we construct) of circuits with the same difficulty.

1.1.2 Machine learning and pattern recognition

Learning algorithms based on nearest neighbors have been the subject of extensive research for more than 50 years [8, 11]. For example, there is evidence that increasing k in the k-nearest neighbors rule can decrease its estimation error [10]. However, much less seems to be known about the *capacity* of the nearest and k-nearest neighbor rules to represent certain functions, while the capacity of other machine learning models such as Boolean circuits and neural networks has received considerable interest [16, 19, 30, 12, 38].

[1] By "explicit," we mean a function that can be computed in polynomial time by a Turing machine.

1.1.3 Algorithms for nearest neighbors classification and search

Efficient implementation of the nearest neighbor rule in high dimension has received considerable interest, leading to efficient algorithms and sophisticated data structures [7, 2, 1]. The study of nearest neighbor complexity leads naturally to the study of nonuniform algorithms (Boolean circuits) for this rule. Our work (along with previous findings) yields upper and lower bounds on the size of a circuit needed to implement nearest neighbor classification. We focus on exact representations, leaving the study of circuit complexity for *approximate* nearest neighbor search (e.g., [20, 28]) for future work.

1.2 Our results

The objects of our study are the *classes* of Boolean functions admitting *real* and *Boolean* nearest neighbor representation of polynomial complexity, NN and HNN respectively. Standard complexity classes based on circuit-like models of computation are closed under two natural "rewiring" operations: Substitution of variables by constants and duplication of variables (See Definition 11). However, it is not clear how to efficiently perform these operations in the nearest neighbor model. Thus, if we would like to give a precise characterization of NN and HNN in terms of circuits, we have to consider the *closure* of these classes under the same operations. As a result, we obtain classes of *subfunctions* of polynomial-size nearest neighbor representations, $\overline{\mathsf{NN}}$ and $\overline{\mathsf{HNN}}$ (see Defintion 12). We then give a precise characterization (equality) of $\overline{\mathsf{NN}}$ and $\overline{\mathsf{HNN}}$ in terms of the class of *min-plus polynomial threshold functions* of polynomial complexity (mpPTF)[2]. This adds to previous results of [17] showing the *containment* of NN and HNN in mpPTF. As a consequence, we prove (among other results) that HNN contains functions that cannot be computed by depth-two threshold circuits with polynomial size and weight. We also observe that the closure of HNN is closely connected to the class of functions with NN representations of logarithmic bit-complexity.

We study the k-nearest neighbors complexity of Boolean functions for $k > 1$. First, we extend the aforementioned characterization – the closure of NN in terms of mpPTF – to the closure of kNN. We use this characterization to prove that kNN for *constant* k is closely related to NN and that there exists an explicit function that requires exponential kNN complexity when $k \leq n^{1-\epsilon}$ (for an n-variate function). Next, we generalize the characterization of kNN to *arbitrary* k by introducing a new class, kSTAT – functions realizable by an inequality of the k-statistics of two sets of linear forms – which generalizes mpPTF. Consequently, we show that proving lower bounds for kNN for *arbitrary* k would result in lower bounds for the circuit class $\mathsf{SYM} \circ \mathsf{MAJ}$, which would be a major breakthrough in Boolean circuit complexity.

Finally, we present new bounds for nearest neighbor complexity of specific Boolean functions such as disjointness, CNFs, and majority. For example, we show that CNFs with polynomially many clauses have NN representations with polynomially-many anchors which also exhibit *constant bit-complexity*. In contrast, there exist CNFs of polynomial size with exponential HNN complexity. We also establish a new lower bound of $n/2 + 2$ on the HNN complexity of the majority function with an even number of inputs (n). This lower bound is tight, as it matches the upper bound proved in [17].

[2] An mpPTF is an expression of the form $\min\{L_1(\boldsymbol{x}), \cdots, L_\ell(\boldsymbol{x})\} \leq \min\{R_1(\boldsymbol{x}), \cdots, R_r(\boldsymbol{x})\}$ where L_i, R_i are linear forms.

1.3 Related work

Nearest neighbor complexity (under Euclidean distance) was formalized by [17]. They prove that the functions[3] $\mathsf{THR}^{\lfloor n/3 \rfloor}$ and XOR both require an exponential number of Boolean anchors but only 2 and $n+1$ real anchors, respectively. In fact, the same argument proves that THR^t requires at least $\binom{n}{t}/\binom{2t}{t}$ anchors for any t, which gives exponential lower bound for t bounded away from $n/2$, but is vacuous for $\mathsf{THR}^{n/2}$. It was subsequently shown in [24] that any symmetric Boolean function f has an NN representation with $I(f)$ anchors, where $I(f)$ denotes the number of *intervals* of f – the minimal number of contiguous intervals $[a, b]$ partitioning $[0, n]$ where $f(\boldsymbol{x})$ is constant for $a \leq \sum_i x_i \leq b$ – and this bound is optimal when all intervals have the same length. This extends the result of [17] that every symmetric function has nearest neighbor complexity at most $n+1$.

1.3.1 Connections to circuits

It was observed in [17] that functions with polynomial nearest neighbor complexity can be simulated by min-plus polynomial threshold functions, but it is an open question of whether or not the inclusion $\mathsf{NN} \subseteq \mathsf{mpPTF}$ is proper. Relations to the class mpPTF are of interest because it deepens the connection between nearest neighbors and circuit complexity. For instance, [18] establish that systems of mpPTFs compute exactly the class of $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{THR}$ circuits.

The expressive power of k-nearest neighbors rule was also studied by [17]. In particular, they prove that kNN can simulate *linear decision trees*, which yields a linear (in n) lower bound for the number of anchors in kNN representations of the $\mathsf{IP}\mod 2$ function. They also state the open problem of proving stronger lower bounds for the k-nearest neighbors complexity of an explicit function. In [25], it is shown that, under some regularity assumptions, polynomial-size nearest neighbor representations can simulate convex polytopes (i.e., $\mathsf{AND} \circ \mathsf{THR}$) as well as logarithmic fan-in $\mathsf{SYM} \circ \mathsf{THR}$ circuits and linear decision lists (LDL).

Constructions of Boolean circuits computing nearest neighbor classification are known. [31] constructs an $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{THR}$ circuit computing any function with an m-anchor NN representation in size $O(m^2)$. (See Appendix B). A very similar depth-three construction for kNN, also with size $O(m^2)$, was found by [6]. Note that the weights of the above circuits are bounded by a polynomial in n.

1.3.2 Bit complexity

It was shown in [24] that (the aforementioned) NN representations for symmetric functions have *logarithmic bit-complexity*, and that this is tight for some functions. It is left as an open problem to characterize NN representations of *threshold functions* in terms of bit-complexity. To this end, the same authors (in [25]) show that logarithmic bit complexity suffices to represent the comparison, equality, and odd-max bit Boolean functions, and conjecture that a logarithmic upper bound holds for any threshold function.

Other works have studied the role of bit-complexity in *approximate nearest neighbor search*; where we wish to find an anchor whose distance is minimal to a query point, up to a factor of $(1+\epsilon)$. For example, [21] provide tight bounds (in terms of bit-complexity) on the size of data structures performing approximate nearest neighbor search. This setting is quite different from our focus on exact classification of Boolean vectors.

---

[3] $\mathsf{THR}^t(\boldsymbol{x}) = 1 \iff \sum_i x_i \geq t$

The bit-complexity of the weights in polynomial-size threshold circuits has been studied extensively (see, e.g., surveys [34, 37]). For example, it was proved by [14, 15] that arbitrarily large weights can be reduced to have logarithmic bit-complexity by slightly increasing the depth of the circuit (along with a polynomial blow-up in size).

1.4 Organization

Section 2 outlines basic definitions required in subsequent sections. Section 3 establishes the equivalence between $\overline{\mathsf{HNN}}, \overline{\mathsf{NN}}$ and min-plus polynomial threshold functions, then discusses some of the consequences. Section 4 generalizes mpPTF to a new class, kSTAT, and proves that a similar equivalence holds with $\overline{\mathsf{kNN}}$. Here, we also derive several connections to circuit classes such as $\mathsf{SYM} \circ \mathsf{MAJ}$. Section 5 contains new results (upper and lower bounds) for the nearest neighbor complexity of explicit Boolean functions. Many proofs are relegated to Appendix A due to space constraints. Appendix B contains some direct constructions of threshold circuits computing HNN, one of which has depth two.

2 Preliminaries

We use the following notation throughout the paper:

Vectors are written in bold (i.e. $\boldsymbol{x} = (x_1, \cdots, x_n)$).
The k'th statistic of $\boldsymbol{x}$, denoted $\boldsymbol{x}_{(k)}$, is the k'th smallest element of $\boldsymbol{x}$.
– In particular, $\boldsymbol{x}_{(k)} = x_{\sigma(k)} \iff x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)}$ for some $\sigma \in S_n$
$\Delta(\boldsymbol{x}, \boldsymbol{y}) := ||\boldsymbol{x} - \boldsymbol{y}||_2^2$ denotes the squared Euclidean distance between $\boldsymbol{x}, \boldsymbol{y}$.
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes the real inner product (dot product), $x_1 y_1 + \cdots + x_n y_n$.
$\mathrm{poly}(n)$ refers to an arbitrary polynomial in the variable n.
$\mathbb{1}[P]$ denotes the Boolean function whose value is 1 if and only if P holds.

Note that the (squared) Euclidean distance between two Boolean vectors is equal to their Hamming distance, $\Delta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \leq n} \mathbb{1}[x_i \neq y_i]$, so the Hamming weight of a Boolean vector $\boldsymbol{p}$ is denoted $\Delta(\boldsymbol{p}) := \Delta(\boldsymbol{p}, \boldsymbol{0}) = ||\boldsymbol{p}||_2^2$.

2.1 Boolean functions

▶ **Definition 1.** *A* threshold gate *is a Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *defined by a weight vector* $\boldsymbol{w} \in \mathbb{R}^n$ *and a threshold* $\theta \in \mathbb{R}$ *such that*

$$f(\boldsymbol{x}) = 1 \iff \langle \boldsymbol{w}, \boldsymbol{x} \rangle \geq \theta. \tag{1}$$

A threshold circuit *is a sequence* $(f_1, \cdots, f_s)$ *of* $s \geq n$ *gates such that the first* n *gates are equal to the input variables (i.e.,* $f_i = x_i$ *for* $i \leq n$*) and subsequent gates are threshold gates whose inputs are some subset of the previous gates. The output of the circuit is equal to the output of the final gate. The size of the circuit is equal to* $s - n$.

A threshold circuit can be viewed as a directed acyclic graph. Nodes with fan-in 0 correspond to inputs, and other nodes correspond to threshold gates applied to the values of the preceding nodes. The node with fan-out 0 correspond to the output node. The *depth* of the circuit is the length of the longest path from an input node to the output node.

▶ Remark 2. It is well known that we may assume that the weights (and the threshold) are integers without loss of generality: Since the domain of a threshold gate is finite, we may approximate each weight by a rational number and multiply by a common denominator. See [23] for a comprehensive introduction to circuit complexity.

▶ **Definition 3.** *A* Nearest Neighbor (NN) *representation of a Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *is defined by two disjoint sets of positive and negative* anchors $P, N \subseteq \mathbb{R}^n$ *such that*

- $f(\boldsymbol{x}) = 1$ *if there exists a* $\boldsymbol{p} \in P$ *with* $\Delta(\boldsymbol{x}, \boldsymbol{p}) < \Delta(\boldsymbol{x}, \boldsymbol{q})$ *for all* $\boldsymbol{q} \in N$.
- $f(\boldsymbol{x}) = 0$ *if there exists a* $\boldsymbol{q} \in N$ *with* $\Delta(\boldsymbol{x}, \boldsymbol{q}) < \Delta(\boldsymbol{x}, \boldsymbol{p})$ *for all* $\boldsymbol{p} \in P$.

A Hamming Nearest Neighbor (HNN) *representation is defined identically for Boolean anchors in* $\{0,1\}^n$.

▶ **Definition 4.** *A* k-Nearest Neighbors (kNN) *representation of a function* $f : \{0,1\}^n \to \{0,1\}$ *is defined by two disjoint sets of positive and negative* anchors $P, N \subseteq \mathbb{R}^n$ *and an integer* k *such that*

$$f(\boldsymbol{x}) = 1 \iff \text{there exists an } A \subseteq P \cup N \text{ with the following properties:}$$

1. $|A| = k$
2. $|A \cap P| \geq |A \cap N|$
3. $\Delta(\boldsymbol{x}, \boldsymbol{a}) < \Delta(\boldsymbol{x}, \boldsymbol{b})$ *for all* $\boldsymbol{a} \in A$, $\boldsymbol{b} \notin A$.

A kHNN *representation is defined identically for Boolean anchors.*

▶ **Definition 5** ([18])**.** *A* min-plus polynomial threshold function (mpPTF) *is a Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *defined by two sets of linear forms with integer coefficients*[4] $\{L_1, \cdots, L_{\ell_1}\} \cup \{R_1, \cdots, R_{\ell_2}\}$ *satisfying*

$$f(\boldsymbol{x}) = 1 \iff \min_{i \leq \ell_1} L_i(\boldsymbol{x}) \leq \min_{j \leq \ell_2} R_j(\boldsymbol{x}) \tag{2}$$

The number of terms *in an* mpPTF *is equal to* $\ell_1 + \ell_2$*, and the* maximum weight *is equal to the largest absolute value of the coefficients of any form.*

▶ **Definition 6** ([36])**.** *A* linear decision list (LDL) *representation of a Boolean function* f *is a sequence of instructions "if* $f_i(\boldsymbol{x}) = 1$*, then output* c_i *(and halt)" for* $1 \leq i \leq m$*, followed by "output 0." Here,* $f_1, \cdots, f_m$ *are threshold gates and* $c_1, \cdots, c_m \in \{0,1\}$. Exact *linear decision lists (*ELDL*) are defined similarly using* exact *threshold functions – threshold gates where the inequality in* (1) *is replaced with equality. The* length *of an* LDL *or* ELDL *is the number of gates,* m*, and its* maximum weight *is equal to the largest coefficient of any* f_i.

▶ **Definition 7.** *We consider the following well-known Boolean functions.*

The majority *function,* $\mathsf{MAJ}(x_1, \cdots, x_n) = \mathbb{1}[x_1 + \cdots + x_n \geq n/2]$
The disjointness *function,* $\mathsf{DISJ}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{1}[\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0]$
The inner product mod 2 *function,* $\mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \mod 2$
The odd-max-bit *function,* $\mathsf{OMB}(x_1, \cdots, x_n) = \max\{i : x_i = 1\} \mod 2$

2.2 Function classes

First, we define classes of Boolean circuits whose inputs may be variables, their negations, or the constants 0 and 1. AND, OR, THR, and SYM are the classes of polynomial-size[5] depth-one circuits composed of AND, OR, threshold gates, and symmetric functions (i.e., Boolean functions which depend only on the Hamming weight of the input) respectively. $\mathsf{MAJ} \subset \mathsf{THR}$ is the set of threshold gates with polynomial weights[6]. AC^0 is the class of constant-depth circuits consisting of a polynomial number of AND, OR, and NOT gates.

[4] As for threshold gates, there is no loss of generality in the assumption that weights of mpPTFs are integers.
[5] "polynomial" in this context is always with respect to the input size, n.
[6] We abuse the notation denoting by MAJ both specific function and a class of function. The meaning of our notation will be also clear from the context.

For two circuit classes C_1, C_2, the class of circuits consisting of a circuit from C_1 whose inputs are (the outputs of) a polynomial number of circuits from C_2 is denoted by $C_1 \circ C_2$. (e.g., $\mathsf{THR} \circ \mathsf{THR}$ refers to depth two threshold circuits of polynomial size.)

▶ **Definition 8.** NN *is the class of Boolean functions that have nearest neighbor representations with polynomially-many anchors.* HNN *is the same class where anchors are Boolean.* kNN *and* kHNN *are defined in the same manner for a positive integer* k*.*

▶ **Definition 9.** $\mathsf{mpPTF}(\infty)$ *is the class of min-plus polynomial threshold functions with a polynomial number (in terms of the number of inputs) of terms and unbounded maximum weight.* $\mathsf{mpPTF}(\mathrm{poly}(n))$ *is the same class with polynomially-bounded maximum weight.*

▶ **Definition 10.** LDL *is the class of Boolean functions representable by linear decision lists with polynomial length.* $\widehat{\mathsf{LDL}}$ *is the same class with polynomially-bounded maximum weight.* ELDL *and* $\widehat{\mathsf{ELDL}}$ *are defined similarly for exact linear decision lists.*

3 Min-plus PTFs vs. nearest neighbors

In this section, we introduce the *closure* operation and derive an equivalence between (the closure of) NN, HNN and mpPTF.

▶ **Definition 11.** *Define a substitution of variables as a function* $v : \{0,1\}^n \rightsquigarrow \{0,1\}^{\widetilde{n}}$ *where* $\rightsquigarrow$ duplicates *variables or* adds constant *variables (e.g.,* $x_1x_2 \rightsquigarrow x_1x_1x_2x_2x_20$*). Then, a Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *is a* subfunction *of* $g : \{0,1\}^{\widetilde{n}} \to \{0,1\}$ *when* $\widetilde{n} = \mathrm{poly}(n)$ *and there exists a substitution of variables* v *such that* $f(\boldsymbol{x}) = g(v(\boldsymbol{x}))$ *for all* $\boldsymbol{x} \in \{0,1\}^n$*.*

Subfunctions may equivalently be obtained from $g : \{0,1\}^{\widetilde{n}} \to \{0,1\}$ by *identifying variables* (e.g., $x_1 = x_2$) and *assigning variables* to constants (e.g., $x_1 = 0$).

▶ **Definition 12.** *For any function class* C*, let* $\overline{C}$ *denote the* closure *of* C*: The set of subfunctions derived from the elements of* C*. In particular, we say that a Boolean function* f *has an "*$\overline{\mathsf{NN}}$ *representation" if it is a subfunction of some* $g \in \mathsf{NN}$*.*

▶ Note 13. Circuit classes are already closed under this operation. For example, $\overline{\mathsf{MAJ}} = \mathsf{MAJ}$: Subfunctions of the majority function simply add (polynomially-bounded) coefficients and constant terms.

▶ **Theorem 14.**

$$\overline{\mathsf{NN}} = \mathsf{mpPTF}(\infty), \quad \overline{\mathsf{HNN}} = \mathsf{mpPTF}(\mathrm{poly}(n))$$

Theorem 14 and some consequences are proved in Appendix A.1. Namely, we observe that any n-variate function in $\overline{\mathsf{NN}}$ is a sub-function of an $(n+1)$-variate NN representation, and that $\mathsf{mpPTF}(\mathrm{poly}(n))$ captures precisely the power of $\overline{\mathsf{NN}}$ representations with *bit-complexity* $O(\log n)$. Then, using the results of [17] and [18], we immediately establish the following two corollaries.

▶ **Corollary 15.** $\mathsf{HNN} \subsetneq \overline{\mathsf{HNN}}$

▶ **Corollary 16.** $\overline{\mathsf{NN}}$ *representations of* IP *and* $f_n(\boldsymbol{x}, \boldsymbol{y}) := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n^2} (x_{i,j} \wedge y_{i,j})$ *require* $2^{\Omega(n)}$ *anchors.*

(Proofs in A.2 and A.3.) Theorem 14 also yields lower bounds for the circuit complexity of functions belonging to HNN. (A direct construction in Appendix B shows that $\mathsf{HNN} \subseteq \mathsf{THR} \circ \mathsf{MAJ}$.)

▶ **Theorem 17.**

$$\mathsf{HNN} \not\subseteq \mathsf{MAJ} \circ \mathsf{MAJ}$$

More precisely, there is a Boolean function with an HNN *representation with* $n+1$ *anchors which cannot be computed by a depth-two majority circuit with* $\mathrm{poly}(n)$ *gates.*

Proof. First, we claim that $\mathsf{OMB} \circ \mathsf{AND}_2 \in \overline{\mathsf{HNN}}$. Indeed, f is computed by an mpPTF with $n+1$ terms:

$$\min\{L_1(\boldsymbol{x},\boldsymbol{y}), L_3(\boldsymbol{x},\boldsymbol{y}), \cdots\} \leq \min\{-1, L_2(\boldsymbol{x},\boldsymbol{y}), L_4(\boldsymbol{x},\boldsymbol{y}), \cdots\}$$

where $L_k(\boldsymbol{x},\boldsymbol{y}) = (k+1)\cdot(1-x_i-y_i)$. Note that if $x_i = y_i = 1$, then $L_i(\boldsymbol{x},\boldsymbol{y}) = -(i+1)$ and otherwise $L_i(\boldsymbol{x},\boldsymbol{y}) \geq 0$. Hence, the minimum is obtained at the maximum index j where $x_j = y_j = 1$. The claim follows from Theorem 14.

Second, it is known that $\mathsf{OMB} \circ \mathsf{AND}_2 \notin \mathsf{MAJ} \circ \mathsf{MAJ}$ by [4, 16]. Thus, if HNN was in $\mathsf{MAJ} \circ \mathsf{MAJ}$, then we could use the $\overline{\mathsf{HNN}}$ representation described above to get a $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuit computing $\mathsf{OMB} \circ \mathsf{AND}_2$, which is a contradiction. ◀

Finally, we observe a connection between mpPTFs and linear decision lists. This provides additional proof techniques for $\overline{\mathsf{HNN}}$ and helps to relate a question of separation of $\overline{\mathsf{HNN}}$ and $\overline{\mathsf{NN}}$ to the similar question for linear decision lists. The following lemma is proved in Appendix A.4.

▶ **Lemma 18.**

$$\mathsf{mpPTF}(\mathrm{poly}(n)) \subseteq \widehat{\mathsf{LDL}}.$$

More precisely, any mpPTF *with* m *terms and maximum weight* W *is equivalent to a linear decision list with length and maximum weight* $O(m^2 W)$.

▶ Remark 19. This lemma enables another technique to prove lower bounds for $\overline{\mathsf{HNN}}$ apart from sign-rank. More specifically, it is known that any function without large monochromatic rectangles must have a large linear decision list by [5].

▶ **Lemma 20.** $\mathsf{LDL} \subseteq \mathsf{mpPTF}(\infty)$.

Proof. It was shown in [18, Lemma 22] that $\mathsf{OMB} \circ \mathsf{THR} \subseteq \mathsf{mpPTF}(\infty)$. Our lemma follows since OMB is complete for the class of decision lists – See [18, Lemma 22]. ◀

It is open whether $\widehat{\mathsf{LDL}}$ and LDL are equal by [5]. Lemmas 18 and 20 immediately allow us to relate this problem to the problem of separating $\overline{\mathsf{HNN}}$ and $\overline{\mathsf{NN}}$.

▶ **Corollary 21.** *If* $\widehat{\mathsf{LDL}} \neq \mathsf{LDL}$, *then* $\overline{\mathsf{HNN}} \neq \overline{\mathsf{NN}}$.

Proof. From Theorem 14 and Lemmas 18, 20, we have the following sequence of inclusions.

$$\overline{\mathsf{HNN}} = \mathsf{mpPTF}(\mathrm{poly}) \subseteq \widehat{\mathsf{LDL}} \subseteq \mathsf{LDL} \subseteq \mathsf{mpPTF}(\infty) = \overline{\mathsf{NN}},$$

If $\overline{\mathsf{HNN}} = \overline{\mathsf{NN}}$, then the whole sequence of inclusions collapses and, in particular, $\widehat{\mathsf{LDL}} = \mathsf{LDL}$. ◀

4 kNN vs. Circuits

In this section, we give a circuit-style characterization of kNN and provide connections to known circuit classes. From these results, we obtain a separation between kNN and NN. Additionally, our results imply complexity theoretic barriers for proving superpolynomial lower bounds for kNN representations of explicit functions.

4.1 Characterization for small k

Here, we use the connection to mpPTF representations to get our first results on k-nearest neighbors complexity. In particular, we relate k-nearest neighbors representations for constant k to $\overline{\text{NN}}$ and prove a lower bound on k-nearest neighbors complexity for sublinear k.

▶ **Theorem 22.** *Any Boolean function with an m-anchor* kNN *representation is computed by an* mpPTF *with $\binom{m}{k}$ terms.*

Proof. We prove only the first statement as both arguments are identical. As noted in the proof of Theorem 14, the distances from anchors to a query point $\boldsymbol{x}$ are linear forms $L_1(\boldsymbol{x}), \cdots, L_m(\boldsymbol{x})$. Assign each linear form a label $\ell_1, \cdots, \ell_m \in \{1, -1\}$ where a positive label indicates placement on the left-hand side of the mpPTF and vice versa.

Then, consider the collection $A^+(\boldsymbol{x}) = \{L_{i_1}(\boldsymbol{x}) + \cdots + L_{i_k}(\boldsymbol{x}) \mid \ell_{i_1} + \cdots + \ell_{i_k} \geq 0\}$ and the compliment $A^-(\boldsymbol{x}) = \{L_{i_1}(\boldsymbol{x}) + \cdots + L_{i_k}(\boldsymbol{x}) \mid \ell_{i_1} + \cdots + \ell_{i_k} < 0\}$. The resulting mpPTF with $\binom{m}{k}$ terms, $\mathbb{1}[\min A^+(\boldsymbol{x}) \leq \min A^-(\boldsymbol{x})]$, realizes the original kNN representation: The minimum is attained by groups of k-nearest neighbors and if any such group has a positive majority then the inequality holds. ◀

It follows that Boolean functions with m-anchor kNN representations can be represented in $\overline{\text{NN}}$ with $\binom{m}{k}$ anchors. These results generalize to both weighted kNN and to non-Boolean inputs. See Appendix A.5 for a discussion.

As a consequence of Theorem 22, sign-rank lower bounds (e.g., Corollary 16) also apply to kNN. In particular, we get an exponential lower bound for kNN with $k = O(n^{1-\epsilon})$ for constant $\epsilon > 0$. This addresses an open question posed in [17] regarding k-nearest neighbors complexity.

▶ **Corollary 23.** *Any* $\overline{\text{kNN}}$ *representation of* IP *or $f_n(\boldsymbol{x}, \boldsymbol{y}) := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n^2} (x_{i,j} \wedge y_{i,j})$ requires $2^{\Omega(n/k)}$ anchors.*

Proof. Assume that IP (or f_n) has a $\overline{\text{kNN}}$ representation with m anchors. By Theorems 14 and 22 , IP has an $\overline{\text{NN}}$ representation with $\binom{m}{k} \leq m^k$ anchors. By Corollary 16, we have $m^k \geq 2^{\Omega(n)}$ and thus $m \geq 2^{\Omega(n/k)}$. ◀

4.2 Characterization for arbitrary k

In this section, we generalize the ideas of Theorem 14 to the closure of kNN, yielding further connections between nearest neighbors and circuit complexity.

▶ **Definition 24.** *Define by* kSTAT *the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ representable by an inequality between k-statistics of two sets consisting of a polynomial number of linear forms: Given $\{L_1, \cdots, L_{\ell_1}\} \cup \{R_1, \cdots, R_{\ell_2}\}$ and integers k_l and k_r,*

$$f(\boldsymbol{x}) = 1 \iff (L_1(\boldsymbol{x}), \cdots, L_{\ell_1}(\boldsymbol{x}))_{(k_l)} < (R_1(\boldsymbol{x}), \cdots, R_{\ell_2}(\boldsymbol{x}))_{(k_r)} \tag{3}$$

and $\ell_1 + \ell_2$ is bounded by a polynomial in n.

As usual, we can assume that all coefficients in the linear forms are integers. Define the subclass $\widehat{\mathsf{kSTAT}}$ *where all coefficients are bounded by a polynomial in* n[7].

Note that we can reduce Definition 24 to the case of $k_l = k_r$ with only a linear increase in the size. This can be done by adding "dummy" linear forms that are always smaller than all others.

▶ **Theorem 25.**

$\overline{\mathsf{kNN}} = \mathsf{kSTAT}$, $\overline{\mathsf{kHNN}} = \widehat{\mathsf{kSTAT}}$.

See Appendix A.6 for the proof. Next, we provide another equivalent form of kSTAT that is sometimes more convenient.

▶ **Theorem 26.** *The class* kSTAT *consists exactly of functions* $f\colon \{0,1\}^n \to \{0,1\}$ *for which there exist linear forms* $\{L_1, \cdots, L_p\}$ *with* $p = \mathrm{poly}(n)$*, a positive integer* k*, and a labelling function* $\mathsf{label}\colon \{1, \cdots, p\} \to \{0,1\}$*, such that for all* $\boldsymbol{x}$*,*

$$f(\boldsymbol{x}) = 1 \iff (L_1(\boldsymbol{x}), \cdots, L_p(\boldsymbol{x}))_{(k)} = L_i(\boldsymbol{x}) \text{ for some } i \text{ with } \mathsf{label}(i) = 1. \tag{4}$$

The class $\widehat{\mathsf{kSTAT}}$ *consists exactly of functions with the same representation with polynomial-size coefficients in the linear forms.*

See Appendix A.7 for the proof. Now we show that some well-known circuit classes, for which we do not have any known lower bounds, are computable by $\overline{\mathsf{kHNN}}$.

▶ **Theorem 27.**

$\mathsf{SYM} \circ \mathsf{MAJ} \subseteq \widehat{\mathsf{kSTAT}}$.

Any symmetric function of s *threshold functions has a* $\widehat{\mathsf{kSTAT}}$ *representation with* $k = s + 1$.

See Appendix A.8 for the proof. Using the same strategy, we can embed a large complexity class into kNN directly:

▶ **Theorem 28.**

$\mathsf{SYM} \circ \mathsf{AND} \subseteq \mathsf{kNN}$.

Any symmetric function of s *conjunctions has a* kNN *representation with* $k = 2s + 1$.

See Appendix A.9 for the proof.

▶ Remark 29. Note that $\mathsf{SYM} \circ \mathsf{AND} \subseteq \mathsf{SYM} \circ \mathsf{MAJ}$ and $\mathsf{SYM} \circ \mathsf{AND}$ is known to simulate the whole class of ACC^0 within quasi-polynomial size [3]. Related classes are of interest in the context of obtaining lower bounds through circuit satisfiability algorithms [40, Conjecture 1].

As a result of Theorem 28, if we prove for some explicit function f that $f \notin \mathsf{kNN}$, it will follow that $f \notin \mathsf{SYM} \circ \mathsf{AND}$, and this would be a major breakthrough in circuit complexity. Also note that $\mathsf{IP} \in \mathsf{SYM} \circ \mathsf{AND}$ and thus, by Theorem 28, $\mathsf{IP} \in \mathsf{kNN}$. Together with Corollary 16, this gives a separation between NN and kNN. This also shows that in Corollary 23 we cannot get rid of k in the lower bound.

▶ **Theorem 30.** $\mathsf{ELDL} \subseteq \mathsf{kSTAT}$, $\widehat{\mathsf{ELDL}} \subseteq \widehat{\mathsf{kSTAT}}$.

See Appendix A.10 for the proof.

▶ Remark 31. The class ELDL is known to be contained in $\mathsf{THR} \circ \mathsf{THR}$ and proving super-polynomial lower bounds for ELDL is an open problem (See [9]).

[7] mpPTF can be viewed as a special case of kSTAT in which $k_l = k_r = 1$.

5 New bounds for the nearest neighbor complexity of Boolean functions

In this section, we derive several bounds on the nearest neighbor complexity of Boolean functions.

5.1 Nearest neighbor complexity of CNFs

We first show that any CNF admits an efficient NN representation.

▶ **Theorem 32.** *Any CNF or DNF with m clauses has an* NN *representation with $m+1$ anchors and constant bit-complexity.*

Proof. It suffices to prove the statement for DNFs as any CNF can be converted to a DNF by negation.

Let $N = \{\boldsymbol{q} := (\frac{1}{2}, \cdots, \frac{1}{2})\}$ and note that $d(\boldsymbol{x}, \boldsymbol{q}) = n/4$ for every input $\boldsymbol{x} \in \{0,1\}^n$ (where d is the squared Euclidean distance). For each clause, say $C(\boldsymbol{x}) = (x_1 \wedge \cdots \wedge x_k)$, introduce a positive anchor

$$\boldsymbol{p_C} = \left(\underbrace{1, \frac{3}{2}, \cdots, \frac{3}{2}}_{k}, \underbrace{\frac{1}{2}, \cdots, \frac{1}{2}}_{n-k}\right)$$

If any variable is negated, replace the corresponding $\frac{3}{2}$ (or 1) with $-\frac{1}{2}$ (or 0).

If $C(\boldsymbol{x}) = 1$, then $d(\boldsymbol{x}, \boldsymbol{p_C}) = (n-1)/4 < d(\boldsymbol{x}, \boldsymbol{q})$. Otherwise, some literal in C is equal to zero, hence $d(\boldsymbol{x}, \boldsymbol{p_C}) \geq 1 + (n-1)/4 > d(\boldsymbol{x}, \boldsymbol{q})$. Therefore, the entire DNF, say $C_1 \vee \cdots \vee C_m$, is satisfied if and only if some $\boldsymbol{p_{C_i}}$ is a nearest neighbor of $\boldsymbol{x}$. ◀

The polynomial-size representation above does not generalize to deeper AC^0 circuits of depth larger than 2. For instance, Corollary 16 exhibits a function computable by a depth-three De Morgan circuit of polynomial size which does not belong to $\overline{\mathsf{NN}}$. For the well studied disjointness function (that admits a compact CNF representation) we can get an efficient HNN representation:

▶ **Theorem 33.**

$$\mathsf{DISJ} \in \mathsf{HNN}$$

The disjointness function (in $2n$ dimensions) has an HNN *representation with $3n$ anchors.*

Proof. Consider anchors $P = \{(\boldsymbol{e_1}, \boldsymbol{e_1}), \cdots, (\boldsymbol{e_n}, \boldsymbol{e_n})\}$ and $N = \{\boldsymbol{e_1}, \cdots, \boldsymbol{e_{2n}}\}$ where $\boldsymbol{e_i}$ denotes the i'th standard basis vector and $(\boldsymbol{e_i}, \boldsymbol{e_i})$ their concatenation.

Let $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ and suppose $x_i = y_i = 1$ for some i. Then, for all j it holds that $\Delta((\boldsymbol{x}, \boldsymbol{y}), (\boldsymbol{e_i}, \boldsymbol{e_i})) \leq \Delta((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{e_j}) - 1$ with equality when $i = j$. Otherwise, $\Delta((\boldsymbol{x}, \boldsymbol{y}), (\boldsymbol{e_i}, \boldsymbol{e_i})) \geq \Delta((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{e_j}) + 1$ for all i, j. ◀

▶ Remark 34. It can be shown that the number of anchors in Theorem 33 is nearly tight; based on the $\Omega(n)$ lower bound for DISJ of [33], a simple argument proves that NN representations of disjointness require $\Omega(n/\log n)$ anchors. We omit the details.

Proceeding, we show that some CNFs with polynomially many clauses have exponential *Boolean* nearest neighbor complexity.

▶ **Definition 35.** *The* Hamming cube graph *is an undirected graph with vertices* $V = \{0,1\}^n$ *and edges* $E = \{(\boldsymbol{u}, \boldsymbol{v}) \in V : \Delta(\boldsymbol{u}, \boldsymbol{v}) = 1\}$. *The* components *of a Boolean function* f *are the connected components of the subgraph of the Hamming cube graph induced by the vertex set* $f^{-1}(1)$.

▶ **Lemma 36.** *If a Boolean function* f *has* m *components then any* HNN *representation of* f *has at least* m *anchors.*

Proof. Consider some component C of f and let $\delta(C)$ denote the vertex boundary of C: Vertices in $\{0,1\}^n \setminus C$ with a neighbor in C. Note that $\delta(C) \subseteq f^{-1}(0)$.

Suppose f has HNN representation $P \cup N$ and let $\boldsymbol{p} \in P$ be the nearest anchor to some $\boldsymbol{x} \in C$. Assume for contradiction that $\boldsymbol{p} \notin C$. Note that $\Delta(\boldsymbol{x}, \boldsymbol{p})$ is equal to the length of the shortest path from $\boldsymbol{x}$ to $\boldsymbol{p}$ in the Hamming cube graph, which by assumption must contain some $\boldsymbol{y} \in \delta(C)$. (In particular, $\Delta(\boldsymbol{x}, \boldsymbol{p}) = \Delta(\boldsymbol{x}, \boldsymbol{y}) + \Delta(\boldsymbol{y}, \boldsymbol{p})$.) Thus, there must exist some negative anchor $\boldsymbol{q} \in N$ with $\Delta(\boldsymbol{y}, \boldsymbol{q}) < \Delta(\boldsymbol{y}, \boldsymbol{p})$. By the triangle inequality,

$$\Delta(\boldsymbol{x}, \boldsymbol{q}) \leq \Delta(\boldsymbol{x}, \boldsymbol{y}) + \Delta(\boldsymbol{y}, \boldsymbol{q}) < \Delta(\boldsymbol{x}, \boldsymbol{y}) + \Delta(\boldsymbol{y}, \boldsymbol{p}) = \Delta(\boldsymbol{x}, \boldsymbol{p})$$

which contradicts the minimality of $\boldsymbol{p}$. Thus, each component contains an anchor. ◀

Using the previous results, another separation between HNN and NN follows from the existence of a CNF (over n-variables) with $\text{poly}(n)$ clauses and exponentially (in n) many components. (See Appendix A.)

▶ **Theorem 37.** *For any* $k > 0$*, there exists a* k*-CNF over* n *variables with* $\text{poly}(n)$ *clauses for which any* HNN *representation has* $2^{\Omega(n)}$ *anchors.*

5.2 A new lower bound for majority

We now discuss the disparity between the HNN complexity of the majority function in [17, Theorem 4]: In particular, when n is even, the best upper bound is $\frac{n}{2} + 2$ anchors, whereas 2 anchors suffices when n is odd. Note that if ties were allowed (won by positive anchors) in Definition 3, then $P = \{1^n\}$ and $N = \{0^n\}$ would suffice as an HNN representation for MAJ for all n.

▶ **Theorem 38.** *For even* n*, any* HNN *representation of* MAJ *requires* $\frac{n}{2} + 2$ *anchors.*

Proof. Suppose $P \cup N$ is an HNN representation of MAJ for even n. We claim that for each $\boldsymbol{x} \in \{0,1\}^n$ satisfying $\Delta(\boldsymbol{x}) = n/2$, there is a positive anchor $\boldsymbol{p} \neq \mathbf{1}$ with $\boldsymbol{x} \leq \boldsymbol{p}$ in coordinate-wise order:

It follows from [17] that the nearest anchor $\boldsymbol{p}$ to $\boldsymbol{x}$ satisfies $\boldsymbol{x} \leq \boldsymbol{p}$. Indeed, for some i it holds that $x_i = 1$, so suppose for contradiction that $p_i = 0$. Then, construct $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{e_i}$ and let $\boldsymbol{q} \in N$ be the nearest anchor to $\boldsymbol{y}$. This yields $\Delta(\boldsymbol{x}, \boldsymbol{p}) = \Delta(\boldsymbol{y}, \boldsymbol{p}) + 1 > \Delta(\boldsymbol{y}, \boldsymbol{q}) + 1$, contradicting the fact that

$$\Delta(\boldsymbol{x}, \boldsymbol{p}) < \Delta(\boldsymbol{x}, \boldsymbol{q}) \leq \Delta(\boldsymbol{y}, \boldsymbol{q}) + 1. \tag{5}$$

A similar argument shows that $\boldsymbol{q} \leq \boldsymbol{y}$. Hence, $\Delta(\boldsymbol{y}, \boldsymbol{q}) \leq \frac{n}{2} - 1$, and (5) becomes $\Delta(\boldsymbol{x}, \boldsymbol{p}) < \frac{n}{2}$ which implies that $\Delta(\boldsymbol{p}) \leq n - 1$, proving the claim.

For contradiction, assume that $|P \cup N| \leq \frac{n}{2} + 1$. Since there must be at least one negative anchor, we have $|P| \leq \frac{n}{2}$. Then, we can construct $\boldsymbol{x} \in \{0,1\}^n$ with $\Delta(\boldsymbol{x}) = \frac{n}{2}$ for which there is no positive anchor $\boldsymbol{p} \neq \mathbf{1}$ with $\boldsymbol{x} \leq \boldsymbol{p}$, leading to a contradiction: For each $\boldsymbol{p} \in P \setminus \mathbf{1}$, arbitrarily select some i where $p_i = 0$ and set $x_i = 1$, ensuring $\boldsymbol{x} \not\leq \boldsymbol{p}$. After this process, $\Delta(\boldsymbol{x}) \leq |P| \leq \frac{n}{2}$. Arbitrarily fixing more coordinates of $\boldsymbol{x}$ to 1 so that $\Delta(\boldsymbol{x}) = \frac{n}{2}$ completes the construction. ◀

6 Conclusion

We have studied nearest neighbor representations of Boolean functions, proving new lower and upper bounds and devising connections to circuit complexity. There are many future questions and research directions:

- Studying representations of Boolean functions using ideas from *approximate* nearest neighbor search [20, 28] could be of interest. Such a study could potentially lead to new insights and more compact representations avoiding the curse of dimensionality.
- Studying nearest neighbor complexity with respect to additional discrete domains such as grids as well as more than two labels is an interesting future direction.
- Circuit complexity has been used to derive new algorithms for nearest neighbor problems [1]. Can ideas about nearest neighbor complexity such as connections to mpPTFs be used to obtain new algorithms for nearest neighbor classification and search?
- Finally, it remains open whether $\overline{\mathsf{NN}} = \overline{\mathsf{HNN}}$.

References

1 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 136–150. IEEE, 2015. doi:10.1109/FOCS.2015.18.

2 Alexandr Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

3 Richard Beigel and Jun Tarui. On ACC. *Comput. Complex.*, 4:350–366, 1994. doi:10.1007/BF01263423.

4 Harry Buhrman, Nikolay Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 24–32. IEEE, 2007. doi:10.1109/CCC.2007.18.

5 Arkadev Chattopadhyay, Meena Mahajan, Nikhil S. Mande, and Nitin Saurabh. Lower bounds for linear decision lists. *Chic. J. Theor. Comput. Sci.*, 2020, 2020. URL: http://cjtcs.cs.uchicago.edu/articles/2020/1/contents.html.

6 Yan Qiu Chen, Mark S Nixon, and Robert I Damper. Implementing the k-nearest neighbour rule via a neural network. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 1, pages 136–140. IEEE, 1995. doi:10.1109/ICNN.1995.488081.

7 Kenneth L Clarkson. Nearest neighbor queries in metric spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 609–617, 1997. doi:10.1145/258533.258655.

8 Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967. doi:10.1109/TIT.1967.1053964.

9 Yogesh Dahiya, K. Vignesh, Meena Mahajan, and Karteek Sreenivasaiah. Linear threshold functions in decision lists, decision trees, and depth-2 circuits. *Inf. Process. Lett.*, 183:106418, 2024. doi:10.1016/J.IPL.2023.106418.

10 Luc Devroye. On the asymptotic probability of error in nonparametric discrimination. *The Annals of Statistics*, 9(6):1320–1327, 1981.

11 Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31. Springer Science & Business Media, 2013.

12 Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940. PMLR, 2016. URL: http://proceedings.mlr.press/v49/eldan16.html.

13 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002. doi:10.1016/S0022-0000(02)00019-3.

14 Mikael Goldmann, Johan Håstad, and Alexander Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992. doi:10.1007/BF01200426.

15 Mikael Goldmann and Marek Karpinski. Simulating threshold circuits by majority circuits. *SIAM Journal on Computing*, 27(1):230–246, 1998. doi:10.1137/S0097539794274519.

16 András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993. doi:10.1016/0022-0000(93)90001-D.

17 Péter Hajnal, Zhihao Liu, and György Turán. Nearest neighbor representations of boolean functions. *Information and Computation*, 285:104879, 2022. doi:10.1016/J.IC.2022.104879.

18 Kristoffer Arnsfelt Hansen and Vladimir V Podolskii. Polynomial threshold functions and boolean threshold circuits. *Information and Computation*, 240:56–73, 2015. doi:10.1016/J.IC.2014.09.008.

19 Lisa Hellerstein and Rocco A Servedio. On PAC learning algorithms for rich boolean function classes. *Theoretical Computer Science*, 384(1):66–76, 2007. doi:10.1016/J.TCS.2007.05.018.

20 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998. doi:10.1145/276698.276876.

21 Piotr Indyk and Tal Wagner. Approximate nearest neighbors in limited space. In *Conference On Learning Theory*, pages 2012–2036. PMLR, 2018. URL: http://proceedings.mlr.press/v75/indyk18a.html.

22 Jeffrey C Jackson, Adam R Klivans, and Rocco A Servedio. Learnability beyond AC^0. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 776–784, 2002.

23 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 5. Springer, 2012. doi:10.1007/978-3-642-24508-4.

24 Kordag Mehmet Kilic, Jin Sima, and Jehoshua Bruck. On the information capacity of nearest neighbor representations. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 1663–1668, 2023. doi:10.1109/ISIT54713.2023.10206832.

25 Kordag Mehmet Kilic, Jin Sima, and Jehoshua Bruck. Nearest neighbor representations of neurons. In *2024 IEEE International Symposium on Information Theory (ISIT)*, 2024.

26 Adam R Klivans and Rocco A Servedio. Learning DNF in time $2^{O(n^{1/3})}$. *Journal of Computer and System Sciences*, 2(68):303–318, 2004.

27 Adam R Klivans and Rocco A Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7(4), 2006. URL: https://jmlr.org/papers/v7/klivans06a.html.

28 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 614–623, 1998. doi:10.1145/276698.276877.

29 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993. doi:10.1145/174130.174138.

30 James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted boltzmann machines. *Advances in Neural Information Processing Systems*, 26, 2013.

31 O Murphy. Nearest neighbor pattern classification perceptrons. *Neural Networks: Theoretical Foundations and Analysis*, pages 263–266, 1992.

32 Edward A Patrick and Frederic P Fischer III. A generalized k-nearest neighbor rule. *Information and control*, 16(2):128–152, 1970. doi:10.1016/S0019-9958(70)90081-1.

33 Alexander A Razborov. On the distributional complexity of disjointness. In *International Colloquium on Automata, Languages, and Programming*, pages 249–253. Springer, 1990. doi:10.1007/BFB0032036.

34 Alexander A Razborov. On small depth threshold circuits. In *Scandinavian Workshop on Algorithm Theory*, pages 42–52. Springer, 1992. doi:10.1007/3-540-55706-7_4.

35 Alexander A Razborov and Alexander A Sherstov. The sign-rank of AC^0. *SIAM Journal on Computing*, 39(5):1833–1855, 2010.

36 Ronald L Rivest. Learning decision lists. *Machine learning*, 2:229–246, 1987. doi:10.1007/BF00058680.

37 Michael E. Saks. *Slicing the hypercube*, pages 211–256. London Mathematical Society Lecture Note Series. Cambridge University Press, 1993.

38 Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016. URL: http://proceedings.mlr.press/v49/telgarsky16.html.

39 Gal Vardi, Daniel Reichman, Toniann Pitassi, and Ohad Shamir. Size and depth separation in approximating benign functions with neural networks. In *Conference on Learning Theory*, pages 4195–4223. PMLR, 2021. URL: http://proceedings.mlr.press/v134/vardi21a.html.

40 Nikhil Vyas and R. Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of #SAT algorithms. *Theory Comput. Syst.*, 67(1):149–177, 2023. doi:10.1007/S00224-022-10106-8.

41 R. Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *33rd Computational Complexity Conference (CCC 2018)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

A Omitted proofs

A.1 Proof of Theorem 14

We break the proof of this theorem into two separate lemmas.

▶ **Lemma 39.**

$$\overline{\mathsf{NN}} \subseteq \mathsf{mpPTF}(\infty), \quad \overline{\mathsf{HNN}} \subseteq \mathsf{mpPTF}(\mathrm{poly}(n))$$

More precisely, any $\overline{\mathsf{NN}}$ *representation with* m *anchors is equivalent to an* mpPTF *with* m *terms, and any* $\overline{\mathsf{HNN}}$ *representation with* m *anchors in* $\widetilde{n}$ *dimensions is equivalent to an* mpPTF *with* m *terms and maximum weight* $\widetilde{n}$.

Proof. The distance from $\boldsymbol{x} \in \{0,1\}^n$ to an anchor $\boldsymbol{p} \in \mathbb{R}^n$ is a linear form in variables $\boldsymbol{x}$:

$$\begin{aligned}\sum_i (x_i - p_i)^2 &= \sum_i \left[x_i^2 - 2p_i x_i + p_i^2\right] \\ &= \sum_i \left[(1-2p_i)x_i + p_i^2\right] \\ &= \langle \mathbf{1} - 2\boldsymbol{p}, \boldsymbol{x}\rangle + ||\boldsymbol{p}||_2^2.\end{aligned}$$

We can observe that NN representations essentially compute $\mathbb{1}[\min_{\boldsymbol{p}\in P} \Delta(\boldsymbol{x},\boldsymbol{p}) \leq \min_{\boldsymbol{q}\in N} \Delta(\boldsymbol{x},\boldsymbol{q})]$, which is an mpPTF. Subfunctions merely multiply coefficients and add constants to each linear form – For example, $d(x_1x_10, p_1p_2p_3) = 2\cdot(1-2p_1)x_1 + (p_3^2 + 2p_1^2)$.

In the case of HNN, we have for all anchors that $\boldsymbol{p} \in \{0,1\}^n$ and $\Delta(\boldsymbol{x},\boldsymbol{p})$ is a linear form with ± 1 coefficients and positive constants bounded (in absolute value) by n. As a result, the weights in mpPTF are bounded by n as well. ◀

▶ **Lemma 40.**

$$\mathsf{mpPTF}(\infty) \subseteq \overline{\mathsf{NN}}, \quad \mathsf{mpPTF}(\mathrm{poly}(n)) \subseteq \overline{\mathsf{HNN}}$$

More precisely, any mpPTF *with* m *terms has an* $\overline{\mathsf{NN}}$ *representation with* m *anchors in* $n+1$ *dimensions. Any* mpPTF *with* m *terms and maximum weight* W *has an* $\overline{\mathsf{HNN}}$ *representation with* m *anchors in* $\widetilde{n} = O(nW)$ *dimensions.*

Proof. We start with the $\mathsf{mpPTF}(\mathrm{poly}(n))$ case. Let $\mathbb{1}[\min_{i\le\ell_1} L_{1i}(\boldsymbol{x}) \le \min_{j\le\ell_2} L_{2j}(\boldsymbol{x})]$ be an arbitrary $\mathsf{mpPTF}(\mathrm{poly}(n))$. First make some pre-processing steps. First, multiply each linear form by 2 and add one to the right-hand side, so that ties are won by the left-hand side. Second, we would like to make all coefficients positive. For this, while there exists a negative term $-a_{ijk}x_k$ (or constant $-\theta_{ij}$), just add x_k (or 1) to every linear form until all negative terms are eliminated. No coefficient (or constant) will increase by more than W. Third, we make all coefficients even by multiplying all linear forms by two. Finally, we add the same constant Θ (to be decided later) to all linear forms. Then, every linear form is equal to $L_{ij}(\boldsymbol{x}) = a_{ij1}x_1 + \cdots + a_{ijn}x_n + \theta_{ij} + \Theta$, for positive, even constants $a_{ijk}, \theta_{ij} \le 8W$.

Define n *block sizes* $t_1, \cdots, t_n$ by $t_k := \max_{i,j} a_{ijk}$ (i.e., the maximum coefficient of x_k in any linear form). Also define $C = \Theta + \max_{i,j}\theta_{ij}$ and let $\widetilde{n} := t_1 + \cdots + t_n + C$. Inputs $\boldsymbol{x} \in \{0,1\}^n$ are mapped ($\rightsquigarrow$) to query points $\widetilde{\boldsymbol{x}} \in \{0,1\}^{\widetilde{n}}$ and linear forms L_{ij} are mapped to anchors $\widetilde{\boldsymbol{p}}_{ij} \in \{0,1\}^{\widetilde{n}}$ such that $\Delta(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{p}}_{ij}) = L_{ij}(\boldsymbol{x})$. In particular,

$$\widetilde{\boldsymbol{x}} := \underbrace{x_1 \cdots\cdots x_1}_{t_1 \text{ many}} \cdots \underbrace{x_n \cdots\cdots x_n}_{t_n \text{ many}} \cdot \underbrace{1 \cdots 1}_{C \text{ many}}$$

and

$$\widetilde{\boldsymbol{p}}_{ij} = \underbrace{0\cdots\cdots 0}_{(t_1+a_{ij1})/2}\,\underbrace{1\cdots\cdots 1}_{(t_1-a_{ij1})/2} \cdots \underbrace{0\cdots\cdots 0}_{(t_n+a_{ijn})/2}\,\underbrace{1\cdots\cdots 1}_{(t_n-a_{ijn})/2} \cdot \underbrace{0\cdots\cdots 0}_{z_{ij}}\underbrace{1\cdots\cdots 1}_{C-z_{ij}}$$

where z_i will be chosen momentarily. (Let $P = \{\widetilde{\boldsymbol{p}}_{1j}\}_{j\le\ell_1}$ and $N = \{\widetilde{\boldsymbol{p}}_{2j}\}_{j\le\ell_2}$.) The distance between $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{p}}_{ij}$ is equal to

$$\begin{aligned}\Delta(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{p}}_{ij}) &= z_{ij} + \sum_k \left(\frac{t_k + a_{ijk}}{2}\right) x_k + \left(\frac{t_k - a_{ijk}}{2}\right)(1 - x_k) \\ &= z_{ij} + \langle \boldsymbol{a_{ij}}, \boldsymbol{x}\rangle + \sum_k \left(\frac{t_k - a_{ijk}}{2}\right)\end{aligned}$$

Now let $z_{ij} = \Theta + \theta_{ij} - \sum_k \left(\frac{t_k - a_{ijk}}{2}\right)$ so that $\Delta(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{p}}_{ij}) = \langle \boldsymbol{a_{ij}}, \boldsymbol{x}\rangle + \Theta + \theta_{ij}$. This is valid (i.e., z_{ij} is a non-negative integer) if we choose a large enough value for Θ: The minimal value of Θ such that $z_{ij} \ge 0$ for all i, j is

$$\Theta = \max_{i,j}\left(\sum_k \left(\frac{t_k - a_{ijk}}{2}\right) - \theta_{ij}\right) \le \sum_k \frac{t_k}{2} \le 4nW.$$

Thus, for $\Theta = 4nW$, we may always choose $0 \le z_{ij} \le \Theta + \theta_{ij} \le C$. Observe that $\boldsymbol{x} \rightsquigarrow \widetilde{\boldsymbol{x}}$ by duplicating each x_i at most $8W$ times and introducing at most $4nW + 8W$ constant variables. Thus, the original mpPTF is equivalent to a subfunction of an HNN representation with m anchors at most $4nW + 8W$ dimensions.

For the $\mathsf{mpPTF}(\infty)$ case, the same method applies, only now we do not need to increase the dimension that much. All coefficients can be realized by choosing anchors $\boldsymbol{p}_{ij} = (1 - \boldsymbol{a}_{ij})/2$ and all constants θ_{ij} can be corrected using one additional dimension. ◀

From this we can also deduce the following:

▶ **Theorem 41.** *Any function with an m-anchor NN representation with bit-complexity $O(\log n)$ is equivalent to an $\mathsf{mpPTF}(\mathrm{poly}(n))$ with m terms. Any function of n inputs with an $\mathsf{mpPTF}(\mathrm{poly}(n))$ representation with m terms is equivalent to a subfunction of a function of $n+1$ inputs with an m-anchor NN representation with bit-complexity $O(\log n)$.*

Proof. Observe that in Lemmas 39 and 40 – for NN and $\mathsf{mpPTF}(\infty)$ – the bit-complexity of NN and the logarithms of weights of mpPTF are linearly related. ◀

A.2 Proof of Corollary 15

Proof. It is shown in [17] that XOR has a unique HNN representation with 2^n anchors. Furthermore, it is established in [18] that $\mathsf{XOR} \in \mathsf{mpPTF}(\mathrm{poly}(n))$: In particular, $\mathsf{XOR}(\boldsymbol{x}) = 1$ if and only if $\min\{L_0(\boldsymbol{x}), L_2(\boldsymbol{x}), \cdots\} \leq \min\{L_1(\boldsymbol{x}), L_3(\boldsymbol{x}), \cdots\}$ where $L_i(\boldsymbol{x}) = i^2 - 2i \cdot (x_1 + \cdots + x_n)$. ◀

A.3 Proof of Corollary 16

Proof. It was shown by [17] that the NN complexity of a Boolean function f is bounded below by the *sign-rank* of f, and this can be easily extended to $\overline{\mathsf{NN}}$ through Theorem 14: The number of terms in an mpPTF computing f is also bounded below by the sign-rank of f, by [18].

[13] and [35] respectively establish that the sign rank of IP is equal to $2^{n/2}$ and the sign rank of f_n is $2^{\Omega(n)}$. ◀

A.4 Proof of Lemma 18

Proof. Consider a function $f \in \mathsf{mpPTF}(\mathrm{poly}(n))$ and let $\mathbb{1}[\min_{i \leq \ell_1} L_{1i}(\boldsymbol{x}) \leq \min_{j \leq \ell_2} L_{2j}(\boldsymbol{x})]$ be its representation. We can assume that all possible values of all linear forms are distinct. For this it is enough to multiply all forms by $\ell_1 + \ell_2$ and to add to each form it's own unique remainder modulo $\ell_1 + \ell_2$.

Observe that all linear forms obtain only polynomially many variables (since there output is polynomially bounded in absolute value). Denote possible values of the form L_{ij} by $a_{ij1}, \cdots, a_{ijt}$ for some t polynomially bounded in n. Note that, for different linear forms, the number of the values obtained might be not the same. To simplify the notation we assume that we add several equal values to the list to make them all of equal size t.

Now we are ready to produce the decision list. Let $c_1 = 1$ and $c_2 = 0$. We consider each a_{ijk} in increasing order and query if $L_{ij}(\boldsymbol{x}) \leq a_{ijk}$. If so, we output c_i. If not, we proceed to the next a_{ijk}.

This decision list computes f since we are just looking for the minimal value of a linear form among all possible values of the forms. ◀

A.5 Consequences of Theorem 22

▶ **Corollary 42.** *Any Boolean function with a* $\overline{\mathsf{kNN}}$ *representation with* m *anchors has an* $\overline{\mathsf{NN}}$ *representation with* $\binom{m}{k}$ *anchors. (Similarly, Boolean function with a* $\overline{\mathsf{kHNN}}$ *representation with* m *anchors has an* $\overline{\mathsf{HNN}}$ *representation with* $\binom{m}{k}$ *anchors.)*

▶ Remark 43. Theorem 22 and Corollary 42 can be extended to non-Boolean inputs. More precisely, the same statements are true over any finite domain $D \subseteq \mathbb{R}^n$. For this we can express (squared) distances to anchors as quadratic forms, for each subset of distances of size k consider the average of these distances and represent them as a distance to a new anchor. We still need to add an extra dimension to absorb constant terms.

▶ Remark 44. Theorem 22 and Corollaries 42 and 23 can be extended to the case of weighted kNN. Indeed, in Theorem 22, instead of sums of linear forms we will have weighted sums. This will require $\binom{m}{k} \cdot k! = \frac{m!}{(m-k)!}$ terms in the mpPTF representation. If the weights in the weighted kNN representation are small and the bit-complexity of anchors is small, this results in a $\overline{\mathsf{HNN}}$ representation and if there are no restrictions of weights and bit-complexity, we get $\overline{\mathsf{NN}}$ representation. The proof of Corollary 23 still works despite the increase of the number of anchors to $\frac{m!}{(m-k)!}$.

A.6 Proof of Theorem 25

We first make the following general observation: [32] show that finding the k'th nearest positive anchor and k'th nearest negative anchor and classifying based on which is closest is equivalent to computing a $(2k-1)$-nearest neighbors representation. This fact can be generalized, considering the closure of kNN.

▶ **Lemma 45.** *Let A and B be two sets of numbers and let S be the k smallest elements of $A \cup B$. Then,*

$$|A \cap S| \geq |B \cap S| \iff A_{(t)} < B_{(t)}$$

where $t = \left\lfloor \frac{k+1}{2} \right\rfloor$. (As in kNN, we assume S exists and is unique).

Proof. A contains a majority of the elements in S if and only if $|A \cap S| \geq t$. This happens if and only if the t'th smallest element in A is smaller than the t'th smallest element in B. ◀

We now proceed with the proof of Theorem 25.

Proof. For the inclusion $\overline{\mathsf{kNN}} \subseteq \mathsf{kSTAT}$, consider any function f in $\overline{\mathsf{kNN}}$. It is a subfunction of some function g with a kNN representation $P \cup N$. As in Lemma 39, the distances between $\boldsymbol{x}$ and each anchor are linear forms $A = \{L_1(\boldsymbol{x}), \cdots, L_{|P|}(\boldsymbol{x})\}$ and $B = \{R_1(\boldsymbol{x}), \cdots, R_{|N|}(\boldsymbol{x})\}$ which we assume have integer coefficients by the usual finite precision argument. By definition $g(\boldsymbol{x}) = 1$ if and only if the set S of k-nearest neighbors satisfies $|P \cap S| \geq |N \cap S|$. By Lemma 45, this happens if and only if $A_{(t)} < B_{(t)}$, taking $t = \left\lfloor \frac{k+1}{2} \right\rfloor$. Hence, $g \in \mathsf{kSTAT}$. As kSTAT is closed under taking subfunctions, $f \in \mathsf{kSTAT}$ as well.

For the inclusion $\mathsf{kSTAT} \subseteq \overline{\mathsf{kNN}}$, assume that f has a kSTAT representation. By adding dummy linear forms we can have $k_l = k_r$. By Lemma 45, the inequality (3) holds if and only if the $2k_l - 1$ smallest linear forms consist of more linear forms from the left-hand side than the right. Representing each inequality by an anchor, we obtain a representation of the same function in $\overline{\mathsf{kNN}}$.

The case of $\overline{\mathsf{kHNN}}$ and $\widehat{\overline{\mathsf{kSTAT}}}$ is analogous. ◀

A.7 Proof of Theorem 26

Proof. Suppose a Boolean function f has a representation $\{L_1, \cdots, L_p\}$ satisfying (4) for some function label and integer k. We will show that $f \in \mathsf{kSTAT}$. First, we assume that all coefficients in all linear form are integers and ensure that all values of all linear forms are distinct and even. For this, multiply all forms by $2p$ and shift each form by its own even remainder modulo $2p$.

For each $i \leq p$, we add one linear form to each side of (3). If $\mathsf{label}(i) = 1$, then place the form $L_i(\boldsymbol{x})$ on the left-hand side and $L_i(\boldsymbol{x}) + 1$ on the right. If $\mathsf{label}(i) = 0$, put the $L_i(\boldsymbol{x})$ on right-hand side and $L_i(\boldsymbol{x}) + 1$ on the left. It is easy to see that the k'th statistics in the left and right-hand sides of the resulting kSTAT representation are $L_i(\boldsymbol{x})$ and $L_i(\boldsymbol{x}) + 1$ (not necessarily in that order), where $L_i(\boldsymbol{x})$ is the k'th statistic of the original representation. Hence, the inequality in (3) holds if and only if $\mathsf{label}(i) = 1$.

For the other direction, assume we have a function $f \in \mathsf{kSTAT}$ given by (3). We again assume that all coefficients are integers and all values of all linear forms are distinct. Now we construct the required representation of f. For each form L_i we add to the representation the forms $L_{ij}(\boldsymbol{x}) := L_i(\boldsymbol{x}) + \frac{j}{k_l + k_r}$ for all $j \in \{0, 1, \cdots, k_l + k_r - 1\}$, and for each form R_i we add to the representation the forms $R_{ij}(\boldsymbol{x}) := R_i(\boldsymbol{x}) + \frac{j}{k_l + k_r + 1}$ for all $j = \{0, 1, \cdots, k_l + k_r\}$.

(That is, we have $k_l + k_r$ copies of each form L_i and $k_l + k_r + 1$ copies of each form R_i). To each L_{ij}, R_{ij} we assign the label 0 if $j < k_l$, and 1 if $j \geq k_l$. Finally, we set $k = (k_l + k_r - 1)(k_l + k_r + 1) + 1$.

Now, observe that the inequality (3) holds if and only if, among the $k_l + k_r - 1$ smallest forms, there are at least k_l forms L_i. Assume that there are precisely a forms L_i and b forms R_i. In particular, $a + b = k_l + k_r - 1$. Then, in the new representation, these linear forms give us

$$a(k_l + k_r) + b(k_l + k_r + 1) = (a + b)(k_l + k_r + 1) - a = (k_l + k_r - 1)(k_l + k_r + 1) - a$$

smallest forms. By construction, the next smallest forms are either $L_{i0} \leq \cdots \leq L_{i(k_l+k_r)}$ or $R_{i0} \leq \cdots \leq R_{i(k_l+k_r+1)}$ for some i. Thus, the k'th smallest form is either L_{ia} or R_{ia} and its label is 1 if and only if $a \geq k_l$ as desired. ◀

A.8 Proof of Theorem 27

Proof. Suppose we are given a function $f \in \mathsf{SYM} \circ \mathsf{MAJ}$ and a circuit computing it. We are going to construct a $\widehat{\mathsf{kSTAT}}$ representation of f in the form given by Theorem 26.

We can assume that all MAJ gates in the circuit have the same threshold $t = 0$. For this we can just add dummy variables and fix them to constants. Denote the linear forms for MAJ gates by $L_1, \cdots, L_s$ (all weights are integers) and denote by $g\colon \{0,1\}^s \to \{0,1\}$ the symmetric function at the top of the circuit. Here, s is the size of the circuit. Now, construct a $\widehat{\mathsf{kSTAT}}$ representation with the following linear forms:

$$(s+2)L_1(\boldsymbol{x}), \cdots, (s+2)L_s(\boldsymbol{x}), 1, 2, \cdots, s+1. \tag{6}$$

That is, we multiply each linear form by $(s + 2)$ and add $(s + 1)$ constant linear forms with values $1, \cdots, s + 1$. We let $k = s + 1$.

It is easy to see that the k'th statistic of (6) is always one of the constant linear forms. It is the form i if and only if $i - 1$ of the linear forms among $L_1, \cdots, L_s$ are positive. We assign label 1 to the form i if and only if $g(\boldsymbol{x}) = 1$ for inputs of weight $i - 1$. As a result, we get the desired representation for f and show that $f \in \widehat{\mathsf{kSTAT}}$. ◀

▶ Remark 46. The well-known argument that shows $\mathsf{MAJ}\circ\mathsf{THR} = \mathsf{MAJ}\circ\mathsf{MAJ}$ (see [14]) can be straightforwardly adapted to show that $\mathsf{SYM}\circ\mathsf{THR} = \mathsf{SYM}\circ\mathsf{MAJ}$. Thus, $\mathsf{SYM}\circ\mathsf{THR} \subseteq \widehat{\mathsf{kSTAT}}$ follows from Theorem 27 as well.

A.9 Proof of Theorem 28

Proof. First, as a warm-up, we show that $\mathsf{IP} \in \mathsf{kNN}$. Recall that $\mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) = \bigoplus_{i=1}^{n}(x_i \wedge y_i)$. Denote by $\boldsymbol{a} = (\frac{1}{2}, \cdots, \frac{1}{2})$ an $2n$-dimensional vector with $\frac{1}{2}$ in each coordinate. Note that $\Delta(\boldsymbol{a}, (\boldsymbol{x}, \boldsymbol{y})) = \frac{n}{2}$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \{0,1\}^{2n}$.

For each $i = 1, \ldots, n$ introduce two anchors $\boldsymbol{p_{i0}} = \boldsymbol{a} + \frac{1}{2}(\boldsymbol{e_i} + \boldsymbol{e_{i+n}})$ and $\boldsymbol{p_{i1}} = \boldsymbol{a} + \frac{1}{4}(\boldsymbol{e_i} + \boldsymbol{e_{i+n}})$. If for some $(\boldsymbol{x}, \boldsymbol{y})$ we have $x_i = y_i = 1$, then

$$\Delta((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{p_{ij}}) \leq \frac{n}{2} - 2\left(\frac{1}{4} - \frac{1}{16}\right) = \frac{n}{2} - \frac{3}{8}.$$

If, on the other hand, $x_i = 0$ or $y_i = 0$, then

$$\Delta((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{p_{ij}}) \geq \frac{n}{2} - \left(\frac{1}{4} - \frac{1}{16}\right) - \left(\frac{1}{4} - \frac{9}{16}\right) = \frac{n}{2} + \frac{1}{8} > \frac{n}{2}.$$

For each $i = 1, \cdots, n+1$ and $j = 0, 1$ and $l = 0, 1$ introduce an anchor $\boldsymbol{q_{i,j,l}} = \boldsymbol{a} + (-1)^l \frac{2i+j}{8n} \boldsymbol{e_1}$. For $(\boldsymbol{x}, \boldsymbol{y})$ with $x_1 = 1$ it is not hard to see that

$$\frac{n}{2} - \frac{3}{8} < \Delta((\boldsymbol{x},\boldsymbol{y}), \boldsymbol{q_{n+1,1,0}}) < \Delta((\boldsymbol{x},\boldsymbol{y}), \boldsymbol{q_{n+1,0,0}}) < \cdots < \\ < \Delta((\boldsymbol{x},\boldsymbol{y}), \boldsymbol{q_{1,1,0}}) < \Delta((\boldsymbol{x},\boldsymbol{y}), \boldsymbol{q_{1,0,0}}) < \frac{n}{2}$$

and $\Delta((\boldsymbol{x},\boldsymbol{y}), \boldsymbol{q_{i,j,1}}) > \frac{n}{2}$ for all i, j. The situation is symmetric for $x_1 = 0$. We assign label j to the anchor $\boldsymbol{p_{ij}}$. We assign label 1 to the anchor $\boldsymbol{q_{ijl}}$ iff $i + j$ is odd. We let $k = 2n + 1$.

It is easy to see that for a given $(\boldsymbol{x}, \boldsymbol{y})$ among the k closest anchors we have all pairs of anchors $\boldsymbol{p_{i0}}, \boldsymbol{p_{i1}}$ for all i such that $x_i = y_i = 1$. Denote the number of such i by t. Also among the k closest anchors we will have pairs of anchors $\boldsymbol{q_{i,0,l}}, \boldsymbol{q_{i,1,l}}$ for an appropriate l and for $i = n + 1, \ldots, t + 2$. In each of these pairs the labels of anchors are opposite and they cancel out when we compute the majority. Finally, one last anchor we will have among the k closest anchors is $\boldsymbol{q_{t+1,1,l}}$. The label of this anchor determines the majority among the k closest anchors and it is 1 iff t is odd. As a result, we get the desired representation for IP with $6n + 4$ anchors.

Now we extend this argument to SYM $\circ$ AND. Consider a function $f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), \cdots, f_s(\boldsymbol{x}))$, where each f_i has the form $f_i(\boldsymbol{x}) = \left(\bigwedge_{i \in S_i} x_i\right) \wedge \left(\bigwedge_{i \in T_i} \neg x_i\right)$ for some disjoint $S_i, T_i \subseteq [n]$. For each f_i we let $\frac{1}{2} > \epsilon_{i1} > \epsilon_{i0} > 0$ be a couple of parameters to be fixed later. We introduce a pair of anchors $\boldsymbol{p_{i1}}, \boldsymbol{p_{i0}}$ in the following way: Set the kth coordinate of $\boldsymbol{p_{ij}}$ to

$$\boldsymbol{p_{ij}}^{(k)} = \begin{cases} 1/2 & k \notin S_i \cup T_i \\ 3/2 - \epsilon_{ij} & k \in S_i \\ \epsilon_{ij} - 1/2 & k \in T_i \end{cases}$$

It is easy to see that for $\boldsymbol{x}$ such that $f_i(\boldsymbol{x}) = 1$ we have $\Delta(\boldsymbol{p_{ij}}, \boldsymbol{x}) = \frac{n}{4} - |S_i \cup T_i|(\epsilon_{ij} - \epsilon_{ij}^2)$ and for $\boldsymbol{x}$ such that $f_i(\boldsymbol{x}) = 0$ we have $\Delta(\boldsymbol{p_{ij}}, \boldsymbol{x}) \geq \frac{n}{4} - (|S_i \cup T_i| - 1)(\epsilon_{ij} - \epsilon_{ij}^2) + 1$. We fix ϵ_{ij} in such a way that $\frac{n}{4} - |S_i \cup T_i|(\epsilon_{ij} - \epsilon_{ij}^2) < \frac{n}{4} - \frac{1}{2}$ and $\frac{n}{4} - (|S_i \cup T_i| - 1)(\epsilon_{ij} - \epsilon_{ij}^2) + 1 > \frac{n}{4} + \frac{1}{2}$. We set $\mathsf{label}(\boldsymbol{p_{ij}}) = j$.

We construct anchors $\boldsymbol{q_{ijl}}$ for $i = 1, \cdots, s + 1$ and $j = 0, 1$ the same way as above and assign $\mathsf{label}(\boldsymbol{q_{i1l}})$ to be equal to $g(\boldsymbol{y})$ for $\boldsymbol{y}$ of weight $i - 1$ and $\mathsf{label}(\boldsymbol{q_{i0l}})$ to be the opposite. We let $k = 2s + 1$. The same argument as for IP shows that we get the desired representation of f with $6s + 4$ anchors. ◀

A.10 Proof of Theorem 30

Proof. Consider a function $f \in$ ELDL and suppose the linear forms in its representation are $L_1, \cdots, L_s$. Here L_i corresponds to the i'th query. As in the proof of Theorem 27, we can assume that all thresholds in all linear forms are 0.

We are going to construct a representation for f of the form provided by Theorem 26. We add to this representation the following linear forms:

$$(s+1)L_1, -(s+1)L_1, (s+1)L_2+1, -(s+1)L_2-1, \cdots, (s+1)L_s+s-1, -(s+1)L_s-s+1.$$

That is, for each form L_i in ELDL representation, we add the two forms $(s + 1)L_i + (i - 1)$ and $-(s + 1)L_i - (i - 1)$. We set $k = s$.

Assume that for some x we have $L_i(\boldsymbol{x}) = 0$ and all previous linear forms are non-zero. We than have that $(s+1)L_i(\boldsymbol{x}) + i - 1 = i - 1$. It is not hard to see that for $j < i$ we have that among forms $(s+1)L_j(\boldsymbol{x}) + j - 1$ and $-(s+1)L_j(\boldsymbol{x}) - j + 1$ exactly one is greater than $i - 1$: it is the first one if $L_j(\boldsymbol{x}) > 0$ and the second one if $L_j(\boldsymbol{x}) < 0$. For $j > i$ in a similar way we can see that among the forms $(s+1)L_j(\boldsymbol{x}) + j - 1$ and $-(s+1)L_j(\boldsymbol{x}) - j + 1$ exactly one is greater than $i - 1$: it is the first one if $L_j(\boldsymbol{x}) \geq 0$ and the second one if $L_j(\boldsymbol{x}) < 0$. As a result there are exactly $s - 1$ forms that are greater than $(s+1)L_i(\boldsymbol{x}) + i - 1$. We assign to this form the same label $L_i(\boldsymbol{x})$ has in ELDL. From this it follows that the constructed representation computes the same function.

Clearly, the coefficients in the constructed form are polynomially related to the coefficients in the original forms. Thus, the same proof gives $\widehat{\mathsf{ELDL}} \subseteq \widehat{\mathsf{kSTAT}}$. ◀

▶ Remark 47. Note that decision lists are computable in AC^0 and thus can be computed by quasi-polynomial-size $\mathsf{SYM} \circ \mathsf{AND}$ circuits. As a result, ELDL can be computed by quasi-polynomial-size circuit in $\mathsf{SYM} \circ \mathsf{AND} \circ \mathsf{ETHR} = \mathsf{SYM} \circ \mathsf{ETHR} = \mathsf{SYM} \circ \mathsf{MAJ}$, where the second equality follows since ETHR is closed under AND operation. Still, Theorem 30 gives a polynomial reduction that translates to the case of small coefficients.

A.11 Proof of Theorem 37

Such constructions are likely known; we outline a simple one for completeness.

▶ **Lemma 48.** *For any even integer $k > 0$, there exists a CNF with n variables and $n2^{k-o_k(k)}/k$ clauses with $2^{(1-o_k(1))n}$ components.*

Proof. Assume k divides n. Divide the set of variables to n/k disjoint sets $S_1, \cdots, S_{n/k}$ of size k. For each set S_i, define a CNF C_i which evaluates to 1 if and only if exactly half of the variables in S are equal to 1. This can be achieved with $\binom{k}{k/2} = 2^{k-o_k(k)}$ clauses.

Then, the CNF $C = C_1 \wedge \cdots \wedge C_{n/k}$ has exactly $\binom{k}{k/2}^{n/k} = 2^{(1-o_k(1))n}$ satisfying assignments, and the Hamming distance between any two of such assignments is at least 2. Thus, each of them constitutes a component. ◀

Hence, Theorem 37 follows from Lemmas 36 and 48 by taking k to be a constant independent of n. It is easy to extend the construction above to odd k. We omit the simple details.

B Circuits computing nearest neighbors

In this section we describe a straightforward construction of a *depth-three* circuit computing HNN and then compress it to *depth-two* at the cost of exponential weights. The folklore result of [31] is that any NN representation with m anchors can be computed by a depth three threshold circuit with size $O(m^2)$. A short proof can be found in [24].

▶ **Theorem 49** ([31])**.**
- $\mathsf{NN} \subseteq \mathsf{OR} \circ \mathsf{AND} \circ \mathsf{THR}$, $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{THR}$
- $\mathsf{HNN} \subseteq \mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MAJ}$, $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{MAJ}$

Namely, every NN *(*HNN*) representation is computed by a depth-three* $\mathsf{AC}^0 \circ \mathsf{THR}$ *(*MAJ*) circuit with size* $|P||N| + \min\{|P|, |N|\} + 1$.

Note that the only difference between the circuits for HNN and NN is that the first-level threshold gates are guaranteed to have polynomial weights (in the case of HNN). It turns out that the size of the HNN circuit can be improved (when $n \ll |P| + |N|$).

▶ **Lemma 50.**

$$\mathsf{HNN} \subseteq \mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MAJ}$$

In particular, every HNN *representation with m anchors is computed by an* $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{THR}$ *circuit with size $(n+1)m + (n+1)|P| + 1$.*

Proof. Note that $\mathbb{1}[\Delta(\boldsymbol{x}, \boldsymbol{p}) \leq i]$ is computed by a threshold gate $f^{\boldsymbol{p}}_{\leq i}(\boldsymbol{x})$ defined by $\boldsymbol{w} = \boldsymbol{p} - \overline{\boldsymbol{p}}$ and $\theta = \Delta(\boldsymbol{p}) - i$. (And similarly $\mathbb{1}[\Delta(\boldsymbol{x}, \boldsymbol{p}) \geq i]$.) Suppose f has an HNN representation $P \cup N$. Then, $f(\boldsymbol{x}) = \bigvee_{\substack{i \leq n \\ \boldsymbol{p} \in P}} \left(f^{\boldsymbol{p}}_{\leq i}(\boldsymbol{x}) \wedge \bigwedge_{\boldsymbol{q} \in N} f^{\boldsymbol{q}}_{\geq i}(\boldsymbol{x}) \right)$ ◀

Note that the threshold circuits from Theorem 49 and Lemma 50 have size $O(m^2)$ and $O(mn)$ respectively. In fact, the latter circuit can be compressed to a depth-two threshold circuit with exponential weights.

▶ **Theorem 51.**

$$\mathsf{HNN} \subseteq \mathsf{THR} \circ \mathsf{MAJ}.$$

Namely, every HNN *representation with m anchors is computed by a threshold of $2nm$ majority gates.*

Proof. The first level will consist of $2mn$ gates $f^{\boldsymbol{p}}_{\leq i}$, $f^{\boldsymbol{p}}_{\geq i}$ which output 1 if and only if $\Delta(\boldsymbol{x}, \boldsymbol{p}) \leq i$ and $\Delta(\boldsymbol{x}, \boldsymbol{p}) \geq i$, respectively, for $1 \leq i \leq n$. Define the sum

$$g^p_i(\boldsymbol{x}) := f^{\boldsymbol{p}}_{\leq i}(\boldsymbol{x}) + f^{\boldsymbol{p}}_{\geq i}(\boldsymbol{x}) - 1$$

and note that $g^p_i(\boldsymbol{x}) = \mathbb{1}[\Delta(x, p) = i]$. We can then write the output gate as

$$h(\boldsymbol{x}) = \mathbb{1}\left(\sum_{p \in P, i \leq n} m^{3(n-i)+1} g^p_i(\boldsymbol{x}) - \sum_{q \in N, i \leq n} m^{3(n-i)} g^q_i(\boldsymbol{x}) \geq 0 \right).$$

If some positive anchor is at distance at most j and all negative anchors are at distance at least j to $\boldsymbol{x}$, then

$$\sum_{p \in P, i \leq n} m^{3(n-i)+1} g^p_i(\boldsymbol{x}) \geq m^{3(n-j)+1} \geq \sum_{q \in N, i \leq n} m^{3(n-i)} g^q_i(\boldsymbol{x}).$$

Conversely, if some negative anchor is at distance at most j and all positive anchors are at distance at least $j+1$, then

$$\sum_{p \in P, i \leq n} m^{3(n-i)+1} g^p_i(\boldsymbol{x}) \leq m^{3(n-j)-1} < m^{3(n-j)} \leq \sum_{q \in N, i \leq n} m^{3(n-i)} g^q_i(\boldsymbol{x}).$$ ◀

▶ Remark 52. Theorem 51 can be obtained through Theorem 14, as a consequence of the following result derived from [18]. We include the direct construction to avoid the slight increase in circuit size.

▶ **Lemma 53.**

$$\mathsf{mpPTF}(\mathrm{poly}(n)) \subseteq \mathsf{THR} \circ \mathsf{MAJ}$$

Every mpPTF *with ℓ terms and maximum weight W is computed by a linear threshold of at most $4 \cdot W\ell \log \ell$ majority gates.*

Proof. Let $\mathsf{PTF}_{1,2}$ refer to Boolean functions (over $\{1,2\}$) equal to the sign of an n-variate polynomial. [18] prove that any $\mathsf{PTF}_{1,2}$ with ℓ terms and degree at most d is computed by a linear threshold (with exponential weights) of at most $2\ell d$ majority gates (replacing $\{1,2\}$ with $\{0,1\}$), and any mpPTF with ℓ terms and maximum weight W can be represented by a $\mathsf{PTF}_{1,2}$ with ℓ terms and degree at most $2W \log \ell$. ◀

▶ Remark 54. It is not hard to see that the circuits constructed in this section are *polynomial-time uniform*; they can be generated by a Turing machine given the set of anchors in polynomial time.