Nearest Neighbor Complexity and Boolean Circuits

Mason DiCicco, Vladimir Podolskii, Daniel Reichman

June 5, 2024

Abstract

A nearest neighbor representation of a Boolean function \( f \) is a set of vectors (anchors) labeled by 0 or 1 such that \( f(x) = 1 \) if and only if the closest anchor to \( x \) 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 more expressive model of \( k \)-nearest neighbors.

We initiate the 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]. As an outcome of this connection 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 \( k > 1 \).

Next, we further extend the connection between nearest neighbor representations and circuits to the \( k \)-nearest neighbors case. As a result, 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 Hajnal et al. [2022] 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 explicit functions.

1 Introduction

The capacity of a learning model to economically represent a target function is essential in designing machine learning algorithms. For example, the expressivity of machine learning models such as Boolean circuits and neural networks has received considerable attention, with a recent focus on properties that allow for efficient representations [Cybenko, 1989, Hornik et al., 1989, Daniely et al., 2016, Hellerstein and Servedeio, 2007, Eldan and Shamir, 2016, Hajnal et al., 1993, Telgarsky, 2016, Martens et al., 2013]. Here, we study the nearest neighbor and \( k \)-nearest neighbors rules over the Boolean hypercube and seek to understand which Boolean functions can be represented efficiently by these rules.

A nearest-neighbor representation of a function \( f \) is a set of vectors, called “anchors,” say \( S = P \cup N \) such that \( f(x) = 1 \) if and only if the nearest anchor to \( x \) (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 in

*Worcester Polytechnic Institute. [mtdicicco@wpi.edu, dreichman@wpi.edu]
†Tufts University. [vladimir.podolskii@tufts.edu]
Hajnal et al. [2022], Kilic et al. [2023], who advocated the study of Boolean functions admitting efficient representations. In a similar vein, Hajnal et al. [2022] also consider the $k$-nearest neighbors model, where the value of $f$ on input $x$ is computed as a majority vote of the $k$ nearest anchors to $x$.

The nearest neighbor rule is the subject of extensive research both with respect to statistical properties and efficient algorithms [Andoni and Indyk, 2017, Andoni, 2009, Shakhnarovich et al., 2005, Dasgupta and Kpotufe, 2020, Chaudhuri and Dasgupta, 2014, Cover and Hart, 1967, Clarkson, 1997]. To our knowledge, much less is known about the representational efficiency of this rule: Namely, which set of functions can be represented efficiently; with polynomially many anchors.

Our approach to understanding nearest neighbor complexity is to relate it to the well-studied field of Boolean circuit complexity, following a long tradition in computational learning theory and computational complexity e.g., [Linial et al., 1993, Jackson et al., 2002, Klivans and Servedio, 2004, 2006, Williams, 2018, Vardi et al., 2021]. Our results establish both (unconditional) lower bounds and upper bounds on the nearest neighbor complexity of explicit functions. We also describe efficient $k$-nearest neighbors representations – for appropriate values of $k$ – for a large class of Boolean functions for which we do not know how to prove super-polynomial circuit lower bounds. Our proofs uncover new connections between nearest neighbors and computational models such as linear decision lists and depth-two circuits that may be of independent interest.

Our results To study the computational power of nearest neighbor representations, we introduce the closure of the classes NN and HNN under substitution and duplication of variables: These are subfunctions of Boolean functions with polynomial nearest neighbor complexity (NN) as well as polynomial Boolean nearest neighbor complexity (HNN). Note that standard complexity classes based on circuit-like models of computation are closed under these operations. Hence, we can give a precise characterization (equality) of the closures of NN and HNN in terms of min-plus polynomial threshold functions of polynomial complexity (mpPTF). This adds to previous results of Hajnal et al. [2022] showing the containment of NN and HNN in the class of mpPTFs. 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 establish that proving lower bounds for kNN for arbitrary $k$ would require a breakthrough in circuit complexity: See Theorems 6 and 8.

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 matches the upper bound proved in Hajnal et al. [2022].

---

1We only deal with Boolean functions, but some results extend to the non-Boolean case as well, see Remark 6.

2By “explicit,” we mean a function that can be computed in polynomial time by a Turing machine.
**Related work** Nearest neighbor complexity (under Euclidean distance) was formalized by Hajnal et al. [2022]. They observe 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 NN ⊆ mpPTF is proper. Relations to the class mpPTF are of interest because mpPTF has connections to circuit complexity. For instance, Hansen and Podolskii [2015] establish that systems of mpPTF’s compute exactly the class of AND ◦ OR ◦ THR circuits.

Hajnal et al. [2022] prove that the functions THR\(^{[n/3]}\) 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 \(\binom{n}{t} / (2^n)\) anchors for any \(t\), which gives exponential lower bound for \(t/n\) bounded away from 1/2, but is vacuous for THR\(^{n/2}\). It was subsequently shown in Kilic et al. [2023] that any symmetric Boolean function \(f\) has an NN representation with \(I(f)\) anchors, where \(I(f)\) denotes the number of intervals of \(f\), and this bound is optimal when all intervals have the same length. This extends the result of Hajnal et al. [2022] that every symmetric function has nearest neighbor complexity at most \(n + 1\).

The expressive power of \(k\)-nearest neighbors rule was also studied by Hajnal et al. [2022]. 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 IP. They also state the open problem of proving stronger lower bounds for the \(k\)-nearest neighbors complexity of an explicit function.

Nearest-neighbor complexity with respect to non-Euclidean distance measures has been studied before by Globig [1996], Satoh [1998]. The former studies the problem of optimizing the distance metric itself to enable polynomial-size NN representations for a given class of Boolean functions. The latter defines “nearest” with respect to a partial ordering based on the set of differing coordinates between anchors and, in this case, proves that CNFs and DNFs have polynomial-size representations. Salzberg et al. [1995] studies the minimum number of anchors required to represent geometric concepts in \(\mathbb{R}^n\). As real functions are considered, the techniques and results in Salzberg et al. [1995] are markedly different than our work which studies Boolean functions.

It was shown in Kilic et al. [2023] 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. 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, Indyk and Wagner [2018] 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.

The bit-complexity of the weights in polynomial-size threshold circuits has been studied extensively (see, e.g., surveys [Razborov, 1992, Saks, 1993]). For example, it was proved by [Goldmann et al., 1992, Goldmann and Karpinski, 1998] 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).

Constructions of Boolean circuits computing nearest neighbor classification are known. Murphy [1992] constructs the “canonical” OR ◦ AND ◦ 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 Chen et al. [1995]. Note that the weights of the above circuits are bounded by a polynomial in \(n\).

There is extensive research on the learning complexity – in particular, statistical parameters such as sample complexity – of real nearest neighbor representations (e.g., Devroye et al. [2013]). There is evidence that increasing \(k\) in the \(k\)-nearest neighbors rule can decrease its estimation error Devroye [1981]. In addition, the algorithmic complexity of nearest neighbors was considered by Wilfong [1991], who establishes

\[ \text{THR}^t(x) = 1[x_1 + \cdots + x_n \geq t] \]
NP-hardness of finding a set of anchors with minimal cardinality within some ground set (i.e., the consistent subset problem).

**Organization** Section 2 outlines basic definitions required in subsequent sections. Section 3 establishes the equivalence between HNN, 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 kNN. Here, we also derive several connections to circuit classes such as SYM o MAJ. Section 5 contains several 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

The use the following notation throughout the paper:

- Vectors are written in bold (i.e. \( x = (x_1, \ldots, x_n) \)).
- The \( k \)’th statistic of \( x \), denoted \( x^{(k)} \), is the \( k \)’th smallest element of \( x \).
- In particular, \( x^{(k)} = x_{\sigma(k)} \iff x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)} \) for some \( \sigma \in S_n \).
- \( \Delta(x, y) := ||x - y||_2^2 \) denotes the distance between \( x, y \).
- \( \langle x, y \rangle \) denotes the real inner product (dot product), \( x_1 y_1 + \cdots + x_n y_n \).
- \( \text{poly}(n) \) refers to an arbitrary polynomial in the variable \( n \).
- \( 1[P] \) 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, \( \Delta(x, y) = \sum_{i \leq n} 1[x_i \neq y_i] \), so the Hamming weight of a Boolean vector \( p \) is denoted \( \Delta(p) := \Delta(p, 0) = ||p||_2^2 \).

### 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 \( w \in \mathbb{R}^n \) and a threshold \( \theta \in \mathbb{R} \) such that

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

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(s). 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. Nodes with fan-out 0 correspond to output nodes. The depth of the circuit is the length of the longest path from an input node to an output node.

**Remark 1.** 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 Jukna [2012] for a comprehensive introduction to circuit complexity.

**Definition 2.** 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
• \( f(x) = 1 \) if there exists a \( p \in P \) with \( \Delta(x, p) < \Delta(x, q) \) for all \( q \in N \).

• \( f(x) = 0 \) if there exists a \( q \in N \) with \( \Delta(x, q) < \Delta(x, p) \) for all \( p \in P \).

A Hamming Nearest Neighbor (HNN) representation is defined identically for Boolean anchors.

**Definition 3.** A k-Nearest Neighbors (kNN) representation of a function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) is defined by two disjoint sets of positive and negative anchors \( P, N \subseteq \mathbb{R}^n \) and an integer \( k \) such that

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

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

A kHNN representation is defined identically for Boolean anchors.

**Definition 4 (Hansen and Podolskii [2015]).** A min-plus polynomial threshold function (mpPTF) is a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) defined by two sets of linear forms with integer coefficients\(^4\) \( \{L_1, \ldots, L_{\ell_1}\} \cup \{R_1, \ldots, R_{\ell_2}\} \) satisfying

\[
f(x) = 1 \iff \min_{i \leq \ell_1} L_i(x) \leq \min_{j \leq \ell_2} R_j(x) \quad (2)
\]

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 5 (Rivest [1987]).** A linear decision list (LDL) representation of a Boolean function \( f \) is a sequence of instructions “if \( f_i(x) = 1 \), then output \( c_i \)” for \( 1 \leq i \leq m \), followed by “output 0.” Here, \( f_1, \ldots, f_m \) are threshold gates and \( c_1, \ldots, c_m \in \{0, 1\} \). 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 6.** We consider the following well-known Boolean functions.
- The majority function, \( \text{MAJ}(x_1, \ldots, x_n) = 1 \) \( x_1 + \cdots + x_n \geq n/2 \)
- The disjointness function, \( \text{DISJ}(x, y) = 1 \) \( (x, y) = 0 \)
- The inner product mod 2 function, \( \text{IP}(x, y) = \langle x, y \rangle \mod 2 \)
- The odd-max-bit function, \( \text{OMB}(x_1, \ldots, x_n) = \max\{i : x_i = 1\} \mod 2 \)

### 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. \( \text{MAJ} \subseteq \text{THR} \) is the set of threshold gates with polynomial weights.\(^6\) \( \text{AC}^0 \) is the class of constant-depth circuits consisting of a polynomial number of AND, OR, and NOT gates. A CNF is an AND of clauses, where each clause is an OR of (possibly negated) variables.

\(^4\)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., \( \text{THR} \circ \text{THR} \) refers to depth two threshold circuits of polynomial size.)

**Definition 7.** \( \text{NN} \) is the class of Boolean functions that have nearest neighbor representations with polynomially-many anchors. \( \text{HNN} \) is the same class where anchors are Boolean. \( k\text{NN} \) and \( k\text{HNN} \) are defined in the same manner for a positive integer \( k \).

**Definition 8.** \( \text{mpPTF}(\infty) \) 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. \( \text{mpPTF}(\text{poly}(n)) \) is the same class with polynomially-bounded maximum weight.

**Definition 9.** \( \text{LDL} \) is the class of Boolean functions representable by linear decision lists with polynomial length. \( d\text{LDL} \) is the same class with polynomially-bounded maximum weight. \( \text{ELDL} \) and \( \overline{\text{ELDL}} \) 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) \( \text{NN} \), \( \text{HNN} \) and \( \text{mpPTF} \).

**Definition 10.** Define a substitution of variables as a function \( v : \{0, 1\}^n \rightarrow \{0, 1\}^{\bar{n}} \) where \( \sim \) duplicates variables or adds constant variables (e.g., \( x_1 x_2 \sim x_1 x_2 x_2 x_2 x_2 x_2 \)). Then, a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) is a subfunction of \( g : \{0, 1\}^{\bar{n}} \rightarrow \{0, 1\} \) if \( \bar{n} = \text{poly}(n) \) and there exists a substitution of variables \( v \) such that \( f(x) = g(v(x)) \) for all \( x \in \{0, 1\} \).

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

**Definition 11.** 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 “\( \text{NN} \) representation” if it is a subfunction of some \( g \in \text{NN} \).

**Theorem 1.**

\[ \text{NN} = \text{mpPTF}(\infty), \quad \text{HNN} = \text{mpPTF}(\text{poly}(n)) \]

Theorem 1 and some consequences are proved in Appendix A. Namely, we observe that any \( n \)-variate function in \( \overline{\text{NN}} \) is a sub-function of an \((n+1)\)-variate \( \text{NN} \) representation, and that \( \text{mpPTF}(\text{poly}(n)) \) captures precisely the power of \( \overline{\text{NN}} \) representations with bit-complexity \( O(\log n) \). Then, using the results of Hajnal et al. [2022] and Hansen and Podolskii [2015], we immediately establish the following two corollaries.

**Corollary 1.** \( \text{HNN} \subseteq \overline{\text{HNN}} \)

**Corollary 2.** \( \overline{\text{NN}} \) representations of \( \text{IP} \), \( f_n(x, y) := \wedge_{i=1}^{n} \vee_{j=1}^{n^2} (x_{i,j} \land y_{i,j}) \) require \( 2^{\Omega(n)} \) anchors.

Theorem 1 also yields lower bounds for the circuit complexity of functions belonging to \( \text{HNN} \). (A direct construction in Appendix B shows that \( \text{HNN} \subseteq \text{THR} \circ \text{MAJ} \).)

\(^7\)Note that classes based on circuits are already closed under this operation.
Theorem 2.  
\[
\text{HNN} \not\subseteq \text{MAJ} \circ \text{MAJ}
\]

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 \(\text{poly}(n)\) gates.

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

\[
\min \left\{ L_1(x, y), L_3(x, y), \cdots \right\} \leq \min \left\{ -1, L_2(x, y), L_4(x, y), \cdots \right\}
\]

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

Second, it is known that \(\text{OMB} \circ \text{AND}_2 \not\subseteq \text{MAJ} \circ \text{MAJ}\) by \([\text{Buhrman et al.}, 2007, \text{Hajnal et al.}, 1993]\).

Thus, if HNN was in MAJ \circ MAJ, then we could use the HNN representation described above to get a MAJ \circ MAJ circuit computing OMB \circ AND_2, which is a contradiction.

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

Lemma 1.  
\[
\text{mpPTF}(\text{poly}(n)) \subseteq \text{LDL}.
\]

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

Remark 2. This lemma enables another technique to prove lower bounds for 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 Chattopadhyay et al. [2020].

Lemma 2. LDL \(\subseteq\) mpPTF(\(\infty\)).

Proof. It was shown in [Hansen and Podolskii, 2015, Lemma 22] that OMB \circ THR \(\subseteq\) mpPTF(\(\infty\)). Our lemma follows since OMB is complete for the class of decision lists – See [Hansen and Podolskii, 2015, Lemma 22].

It is open whether LDL and LDL are equal by Chattopadhyay et al. [2020]. Lemmas 1 and 2 immediately allow us to relate this problem to the problem of separating HNN and NN.

Corollary 3. If LDL \(\not=\) LDL, then HNN \(\not=\) NN.

Proof. From Theorem 1 and Lemmas 1, 2, we have the following sequence of inclusions.

\[
\text{HNN} = \text{mpPTF}(\text{poly}) \subseteq \text{LDL} \subseteq \text{LDL} \subseteq \text{mpPTF}(\infty) = \text{NN},
\]

If HNN = NN, then the whole sequence of inclusions collapses and, in particular, LDL = 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 establish connection of $k$-nearest neighbors representations for constant $k > 0$ to $\NN$ and prove a lower bound on $k$-nearest neighbors complexity for sublinear $k$.

**Theorem 3.** 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 1, the distances from anchors to a query point $x$ are linear forms $L_1(x), \cdots , L_m(x)$. 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 $A^+(x) = \{L_i(x) + \cdots + L_{i_k}(x) \mid \ell_{i_1} + \cdots + \ell_{i_k} \geq 0\}$ and the compliment $A^-(x) = \{L_i(x) + \cdots + L_{i_k}(x) \mid \ell_{i_1} + \cdots + \ell_{i_k} < 0\}$. The resulting mpPTF with $\binom{m}{k}$ terms, $\mathbb{1}[\min A^+(x) \leq \min A^-(x)]$, 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 $\NN$ with $\binom{m}{k}$ anchors. These results generalize to both weighted and non-Boolean inputs. See Appendix A for a discussion.

As a consequence of Theorem 3, sign-rank lower bounds (e.g., Corollary 2) 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 Hajnal et al. [2022] regarding $k$-nearest neighbors complexity.

**Corollary 4.** Any $kNN$ representation of $IP$ or $f_n(x, y) := \bigwedge_{i=1}^n \bigvee_{j=1}^{m^2} (x_{i,j} \land y_{i,j})$ requires $2^{\Omega(n/k)}$ anchors.

*Proof.* Assume that $IP$ (or $f_n$) has a $kNN$ representation with $m$ anchors. By Theorems 1 and 3, IP has an $\NN$ representation with $\binom{m}{k} \leq m^k$ anchors. By Corollary 2, 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 1 to the closure of $kNN$, yielding further connections between nearest neighbors and circuit complexity. All proofs and further discussion can be found in Appendix A.

**Definition 12.** Define by $kSTAT$ the class of functions $f: \{0,1\}^n \rightarrow \{0,1\}$ 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 $k_r$,

$$f(x) = 1 \iff \langle L_1(x), \cdots , L_{\ell_1}(x) \rangle_{(k_l)} < \langle R_1(x), \cdots , R_{\ell_2}(x) \rangle_{(k_r)}$$

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 $k\widehat{STAT}$ where all coefficients are bounded by a polynomial in $n$.

---

8The constant in $\Omega$-notation does not depend on $k$ or $n$.
9mpPTF can be viewed as a special case of $kSTAT$ in which $k_l = k_r = 1$. 

8
9
Note that we can reduce Definition 12 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 4.**

\[
\text{\( k\text{HNN} = k\text{STAT} \), \( k\text{NN} = k\text{STAT} \).}
\]

Next, we provide another equivalent form of \( k\text{STAT} \) that is sometimes more convenient.

**Theorem 5.** The class \( k\text{STAT} \) consists exactly of functions \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) for which there exist linear forms \( \{L_1, \cdots, L_p\} \) with \( p = \text{poly}(n) \), a positive integer \( k \), and a labelling function \( \text{label} : \{1, \cdots, p\} \rightarrow \{0, 1\} \), such that for all \( x \),

\[
f(x) = 1 \iff (L_1(x), \cdots, L_p(x))(k) = L_i(x) \text{ for some } i \text{ with } \text{label}(i) = 1.
\]

The class \( \widehat{k\text{STAT}} \) consists exactly of functions with the same representation with polynomial-size coefficients in the linear forms.

Now we show that some well-known circuit classes, for which we do not have any known lower bounds, are computable by \( k\text{HNN} \).

**Theorem 6.** \( \text{SYM} \circ \text{MAJ} \subseteq k\text{STAT} \).

Using the same strategy, we can embed a large complexity class into \( k\text{NN} \) directly:

**Theorem 7.** \( \text{SYM} \circ \text{AND} \subseteq k\text{NN} \).

**Remark 3.** Note that \( \text{SYM} \circ \text{AND} \subseteq \text{SYM} \circ \text{MAJ} \) and \( \text{SYM} \circ \text{AND} \) is known to simulate the whole class of \( \text{ACC}^0 \) within quasi-polynomial size Beigel and Tarui [1994]. Related classes are of interest in the context of obtaining lower bounds through circuit satisfiability algorithms [Vyas and Williams, 2023, Conjecture 1].

As a result of Theorem 7, if we prove for some explicit function \( f \) that \( f \notin k\text{NN} \), it will follow that \( f \notin \text{SYM} \circ \text{AND} \), and this would be a major breakthrough in circuit complexity. Also note that IP \( \in \text{SYM} \circ \text{AND} \) and thus, by Theorem 7, IP \( \in k\text{NN} \). Together with Corollary 2, this gives a separation between NN and kNN. This also shows that in Corollary 4 we cannot get rid of \( k \) in the lower bound.

**Theorem 8.** \( \text{ELDL} \subseteq k\text{STAT} \), \( \widehat{\text{ELDL}} \subseteq k\text{STAT} \).

**Remark 4.** The class ELDL is known to be contained in \( \text{THR} \circ \text{THR} \) and proving super-polynomial lower bounds for ELDL is an open problem (See Dahiya et al. [2024]).

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 9.** 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 $N = \{ q := (\frac{1}{2}, \ldots, \frac{1}{2}) \}$ and note that $d(x, q) = n/4$ for every input $x \in \{0, 1\}^n$ (where $d$ is the squared Euclidean distance). For each clause, say $C(x) = (x_1 \land \cdots \land x_k)$, introduce a positive anchor
\[
PC = \left( \frac{1}{2}, \frac{3}{2}, \ldots, \frac{3}{2}, \ldots, \frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2} \right)_{k \rightarrow n-k}
\]
If any variable is negated, replace the corresponding $\frac{3}{2}$ (or 1) with $-\frac{1}{2}$ (or 0).

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

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

**Theorem 10.**

\[
\text{DISJ} \in \text{HNN}
\]

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

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

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

**Remark 5.** It can be shown that the number of anchors in Theorem 10 is nearly tight; based on the $\Omega(n)$ lower bound for DISJ of Razborov [1990], 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 13.** The Hamming cube graph is an undirected graph with vertices $V = \{0, 1\}^n$ and edges $E = \{(u, v) \in V : \Delta(u, v) = 1\}$. 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 3.** 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 $p \in P$ be the nearest anchor to some $x \in C$. Assume for contradiction that $p \notin C$. Note that $\Delta(x, p)$ is equal to the length of the shortest path from $x$ to $p$ in the Hamming cube graph, which by assumption must contain some $y \in \delta(C)$. (In particular, $\Delta(x, p) = \Delta(x, y) + \Delta(y, p)$.) Thus, there must exist some negative anchor $q \in N$ with $\Delta(y, q) < \Delta(y, p)$. By the triangle inequality,

\[
\Delta(x, q) \leq \Delta(x, y) + \Delta(y, q) < \Delta(x, y) + \Delta(y, p) = \Delta(x, p)
\]

which contradicts the minimality of $p$. 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 11.** 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 [Hajnal et al., 2022, 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 2, then \( P = \{1^n\} \) and \( N = \{0^n\} \) would suffice as an HNN representation for MAJ for all \( n \).

**Theorem 12.** 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 \( x \in \{0, 1\}^n \) satisfying \( \Delta(x) = n/2 \), there is a positive anchor \( p \neq 1 \) with \( x \preceq p \) in coordinate-wise order:

It follows from Hajnal et al. [2022] that the nearest anchor \( p \) to \( x \) satisfies \( x \preceq p \). Indeed, for some \( i \) it holds that \( x_i = 1 \), so suppose for contradiction that \( p_i = 0 \). Then, construct \( y = x - e_i \) and let \( q \in N \) be the nearest anchor to \( y \). This yields \( \Delta(x, p) = \Delta(y, p) + 1 > \Delta(y, q) + 1 \), contradicting the fact that

\[
\Delta(x, p) < \Delta(x, q) \leq \Delta(y, q) + 1.
\]

A similar argument shows that \( q \preceq y \). Hence, \( \Delta(y, q) \leq \frac{n}{2} - 1 \), and (5) becomes \( \Delta(x, p) < \frac{n}{2} - 1 \), which implies that \( \Delta(p) \leq n - 1 \), 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 \( x \in \{0, 1\}^n \) with \( \Delta(x) = \frac{n}{2} \) for which there is no positive anchor \( p \neq 1 \) with \( x \preceq p \), leading to a contradiction: For each \( p \in P \setminus 1 \), arbitrarily select some \( i \) where \( p_i = 0 \) and set \( x_i = 1 \), ensuring \( x \preceq p \). After this process, \( \Delta(x) \leq |P| \leq \frac{n}{2} \). Arbitrarily fixing more coordinates of \( x \) to 1 so that \( \Delta(x) = \frac{n}{2} \) 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 several questions we did not pursue. Can we find new connections between nearest neighbor complexity and other complexity measures of Boolean functions? For what classes of Boolean functions does polynomial nearest neighbor complexity translate to efficient learning algorithms? Similar ideas for establishing NP-hardness results of learning depth-two neural networks Blum and Rivest [1988] can be shown to imply that properly learning Boolean functions representable by three real anchors is NP-hard. Devising efficient learnability results for nearest neighbor representations that circumvent worst-case barriers could be of interest. Finally, studying representations of Boolean functions using the well developed theory of approximate nearest neighbor search [Indyk and Motwani, 1998, Kushilevitz et al., 1998] could lead to new insights and more compact representations avoiding the curse of dimensionality.

### 7 Acknowledgements

The second and third authors thank the Simons Institute for the Theory of Computing for their hospitality. Their collaboration on this project began during the Meta-Complexity workshop at the institute.
References


A Omitted proofs

A.1 Proof of Theorem 1

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

Lemma 4.

\[ \mathbb{NN} \subseteq \text{mpPTF}(\infty), \quad \mathbb{HNN} \subseteq \text{mpPTF}(\text{poly}(n)) \]

More precisely, any \( \mathbb{NN} \) representation with \( m \) anchors is equivalent to an mpPTF with \( m \) terms, and any \( \mathbb{HNN} \) representation with \( m \) anchors in \( \bar{n} \) dimensions is equivalent to an mpPTF with \( m \) terms and maximum weight \( \bar{n} \).

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

\[
\sum_i (x_i - p_i)^2 = \sum_i [x_i^2 - 2p_i x_i + p_i^2] = \sum_i [(1 - 2p_i)x_i + p_i^2] = \langle 1 - 2p, x \rangle + ||p||^2_2.
\]

We can observe that NN representations essentially compute \( \mathbb{1}[\min_{p \in P} \Delta(x, p) \leq \min_{q \in N} \Delta(x, q)] \), which is an mpPTF. Subfunctions merely multiply coefficients and add constants to each linear form – For example, \( d(x_1, x_1, 0, p_1p_2p_3) = 2 \cdot (1 - 2p_1)x_1 + (p_1^2 + 2p_2^2) \).

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

Lemma 5.

\[ \text{mpPTF}(\infty) \subseteq \mathbb{NN}, \quad \text{mpPTF}(\text{poly}(n)) \subseteq \mathbb{HNN} \]

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

Proof. We start with the mpPTF(poly(n)) case. Let \( \mathbb{1}[\min_{i \leq t_1} L_{1i}(x) \leq \min_{j \leq t_2} L_{2j}(x)] \) be an arbitrary mpPTF(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 \( \Theta \) (to be decided later) to all linear forms. Then, every linear form is equal to \( L_{ij}(x) = a_{ijk}x_1 + \cdots + a_{ij}x_n + \theta_{ij} + \Theta \), for positive, even constants \( a_{ijk}, \theta_{ij} \leq 8W \).

Define \( n \) block sizes \( t_1, \ldots, t_n \) 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 \( \bar{n} := t_1 + \cdots + t_n + C \). Inputs \( x \in \{0, 1\}^n \) are mapped (\( \sim \)) to query points \( \bar{x} \in \{0, 1\}^\bar{n} \) and linear forms \( L_{ij} \) are mapped to anchors \( \bar{p}_{ij} \in \{0, 1\}^n \) such that \( \Delta(\bar{x}, \bar{p}_{ij}) = L_{ij}(x) \). In particular,

\[
\bar{x} := x_{1\ldots1\ldots1} \cdots x_{n\ldots1\ldots1} \cdot 1\ldots1
\]

\( t_1 \) many \( t_n \) many \( C \) many
and
\[
\tilde{p}_{ij} = \underbrace{0 \cdots 0}_{(t_1 + a_{i1})/2} \underbrace{1 \cdots 1}_{(t_1 - a_{i1})/2} \cdots \underbrace{0 \cdots 0}_{(t_n + a_{ijn})/2} \underbrace{1 \cdots 1}_{(t_n - a_{ijn})/2} z_{ij} \quad C - z_{ij}
\]

where \(z_{ij}\) will be chosen momentarily. (Let \(P = \{\tilde{p}_{1j}\}_{j \leq \epsilon_1}\) and \(N = \{\tilde{p}_{2j}\}_{j \leq \epsilon_2}\).) The distance between \(\tilde{x}\) and \(\tilde{p}_{ij}\) is equal to
\[
\Delta(\tilde{x}, \tilde{p}_{ij}) = z_{ij} + \sum_k \left(\frac{t_k + a_{ijk}}{2}\right)x_k + \left(\frac{t_k - a_{ijk}}{2}\right)(1 - x_k)
\]
\[
= z_{ij} + \langle a_{ij}, x \rangle + \sum_k \left(\frac{t_k - a_{ijk}}{2}\right)
\]

Now let \(z_{ij} = \Theta + \theta_{ij} - \sum_k \left(\frac{t_k - a_{ijk}}{2}\right)\) so that \(\Delta(\tilde{x}, \tilde{p}_{ij}) = \langle a_{ij}, x \rangle + \Theta + \theta_{ij}\). This is valid (i.e., \(z_{ij}\) is a non-negative integer) if we choose a large enough value for \(\Theta\): The minimal value of \(\Theta\) such that \(z_{ij} \geq 0\) for all \(i, j\) is
\[
\Theta = \max_{i,j} \left(\sum_k \left(\frac{t_k - a_{ijk}}{2}\right) - \theta_{ij}\right) \leq \sum_k \frac{t_k}{2} \leq 4nW.
\]

Thus, for \(\Theta = 4nW\), we may always choose \(0 \leq z_{ij} \leq \Theta + \theta_{ij} \leq C\). Observe that \(x \sim \tilde{x}\) 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 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 \(p_{ij} = (1 - a_{ij})/2\) and all constants \(\theta_{ij}\) can be corrected using one additional dimension. \(\square\)

From this we can also deduce the following:

**Theorem 13.** Any function with an \(m\)-anchor NN representation with bit-complexity \(O(\log n)\) is equivalent to an mpPTF\((\text{poly}(n))\) with \(m\) terms. Any function of \(n\) inputs with mpPTF\((\text{poly}(n))\) 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 4 and 5 – for NN and mpPTF\((\infty)\) – the bit-complexity of NN and the logarithms of weights of mpPTF are linearly related. \(\square\)

### A.2 Proof of Corollary 1

**Proof.** It is shown in Hajnal et al. [2022] that XOR has a unique HNN representation with \(2^n\) anchors. Furthermore, it is established in Hansen and Podolskii [2015] that XOR \(\in\) mpPTF\((\text{poly}(n))\): In particular, XOR(x) = 1 if and only if \(\min\{L_0(x), L_2(x), \cdots\} \leq \min\{L_1(x), L_3(x), \cdots\}\) where \(L_i(x) = i^2 - 2i \cdot (x_1 + \cdots + x_n)\). \(\square\)

### A.3 Proof of Corollary 2

**Proof.** It was shown by Hajnal et al. [2022] 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 NN through Theorem 1: The number of terms in an mpPTF computing \(f\) is also bounded below by the sign-rank of \(f\), by Hansen and Podolskii [2015].

Forster [2002] and Razborov and Sherstov [2010] respectively establish that the sign rank of IP is equal to \(2^{n/2}\) and the sign rank of \(f_n\) is \(2^\Omega(n)\). \(\square\)
A.4 Proof of Lemma 1

Proof. Consider a function \( f \in \text{mpPTF}(\text{poly}(n)) \) and let \( \min_{i \leq \ell_1} L_{1i}(x) \leq \min_{j \leq \ell_2} L_{2j}(x) \) 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 its 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 \( a_{ij1}, \ldots, a_{ijt} \) 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 \( L_{ij}(x) \leq a_{ijk} \). 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. \( \square \)

A.5 Consequences of Theorem 3

Corollary 5. Any Boolean function with a \( k\text{NN} \) representation with \( m \) anchors has an \( \text{NN} \) representation with \( \binom{m}{k} \) anchors. (Similarly, Boolean function with a \( k\text{HNN} \) representation with \( m \) anchors has an \( \text{HNN} \) representation with \( \binom{m}{k} \) anchors.)

Remark 6. Theorem 3 and Corollary 5 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 7. Theorem 3 and Corollaries 5 and 4 can be extended to the case of weighted \( k\text{NN} \). Indeed, in Theorem 3, 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 \text{mpPTF} representation. If the weights in the weighted \( k\text{NN} \) representation are small and the bit-complexity of anchors is small, this results in a \( \text{HNN} \) representation and if there are no restrictions of weights and bit-complexity, we get \( \text{NN} \) representation. The proof of Corollary 4 still works despite the increase of the number of anchors to \( \frac{m!}{(m-k)!} \).

A.6 Proof of Theorem 4

We first make the following general observation: Patrick and Fischer III [1970] 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 \( k\text{NN} \).

Lemma 6. 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 \( k\text{NN} \), 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 \( B \). \( \square \)

We now proceed with the proof of Theorem 4.
Proof. For the inclusion $\overline{\text{kNN}} \subseteq \overline{\text{kSTAT}}$, consider any function $f$ in $\overline{\text{kNN}}$. It is a subfunction of some function $g$ with a kNN representation $P \cup N$. As in Lemma 4, the distances between $x$ and each anchor are linear forms $A = \{L_1(x), \cdots, L_p(x)\}$ and $B = \{R_1(x), \cdots, R_{|N|}(x)\}$ which we assume have integer coefficients by the usual finite precision argument. By definition $g(x) = 1$ if and only if the set $S$ of $k$-nearest neighbors satisfies $|P \cap S| \geq |N \cap S|$. By Lemma 6, this happens if and only if $\Lambda (t) < \Lambda (t)$, taking $t = \left\lceil \frac{k+1}{2} \right\rceil$. Hence, $g \in \overline{\text{kSTAT}}$. As $\overline{\text{kSTAT}}$ is closed under taking subfunctions, $f \in \overline{\text{kSTAT}}$ as well.

For the inclusion $\overline{\text{kSTAT}} \subseteq \overline{\text{kNN}}$, assume that $f$ has a kSTAT representation. By adding dummy linear forms we can have $k_l = k_r$. By Lemma 6, 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 $\overline{\text{kNN}}$.

The case of $\overline{\text{kNN}}$ and $\overline{\text{kSTAT}}$ is analogous. \qed

A.7 Proof of Theorem 5

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 $f \in \overline{\text{kSTAT}}$. 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 $\text{label}(i) = 1$, then place the form $L_i(x)$ on the left-hand side and $L_i(x) + 1$ on the right. If $\text{label}(i) = 0$, put the $L_i(x)$ on right-hand side and $L_i(x) + 1$ 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 $L_i(x)$ and $L_i(x) + 1$ (not necessarily in that order), where $L_i(x)$ is the $k$’th statistic of the original representation. Hence, the inequality in (3) holds if and only if $\text{label}(i) = 1$.

For the other direction, assume we have a function $f \in \overline{\text{kSTAT}}$ 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

$$L_{ij}(x) := L_i(x) + \frac{j}{k_l + k_r}$$

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

$$R_{ij}(x) := R_i(x) + \frac{j}{k_l + k_r + 1}$$

for all $j \in \{0, 1, \cdots, k_l + k_r \}$. (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 $L_{i\alpha}$ or $R_{i\alpha}$ and its label is 1 if and only if $a \geq k_l$ as desired. \qed
A.8 Proof of Theorem 6

Proof. Suppose we are given a function \( f \in \text{SYM} \circ \text{MAJ} \) and a circuit computing it. We are going to construct a kSTAT representation of \( f \) in the form given by Theorem 5.

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 \( g: \{0, 1\}^s \rightarrow \{0, 1\} \) the symmetric function at the top of the circuit. Here, \( s \) is the size of the circuit. Now, construct a kSTAT representation with the following linear forms:

\[
(s + 2)L_1(x), \cdots, (s + 2)L_s(x), 1, 2, \cdots, s + 1.
\]

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 \( g(x) = 1 \) for inputs of weight \( i = 1 \). As a results we get the desired representation for \( f \) and show that \( f \in \text{kSTAT} \).

Remark 8. The well-known argument that shows \( \text{MAJ} \circ \text{THR} = \text{MAJ} \circ \text{MAJ} \) (see Goldmann et al. [1992]) can be straightforwardly adapted to show that \( \text{SYM} \circ \text{THR} = \text{SYM} \circ \text{MAJ} \). Thus, \( \text{SYM} \circ \text{THR} \subseteq \text{kSTAT} \) follows from Theorem 6 as well.

A.9 Proof of Theorem 7

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

For each \( i = 1, \cdots, n \) and \( j = 0, 1 \) introduce an anchor \( p_{ij} = a + (\frac{1}{2} - \frac{1}{4})(e_i + e_{i+n}) \). If for some \( (x, y) \) we have \( x_i = y_i = 1 \), then

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

If, on the other hand, \( x_i = 0 \) or \( y_i = 0 \), then

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

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

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

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

It is easy to see that for a given \( (x, y) \) among the \( k \) closest anchors we have all pairs of anchors \( p_{i0}, p_{i1} \) 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 \( q_{i,0,t}, q_{i,1,t} \) for an appropriate \( l \) and for \( i = n + 1, \cdots, t + 2 \). 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 \( q_{t+1,t,t} \). 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 any function in \( \text{SYM} \circ \text{AND} \). Consider a function \( f(x) = g(f_1(x), \cdots, f_s(x)) \), where each \( f_i \) has the form

\[
f_i(x) = \left( \bigwedge_{s \in S_i} x_s \right) \wedge \left( \bigwedge_{s \in T_i} \neg x_s \right)
\]

for some disjoint \( S_i, T_i \subseteq [n] \).

For each \( f_i \) we let \( \epsilon_{i1} > \epsilon_{i0} > 0 \) be a couple of parameters to be fixed later. We introduce a pair of anchors \( p_{i1}, p_{i0} \) in the following way: we let the \( j \)th coordinate of \( p_{i,j} \) to \( \frac{1}{2} \) if \( i \notin S_i \cup T_i \), to \( \frac{3}{2} - \epsilon_{i,j} \) for \( i \in S_i \) and to \( -\frac{1}{2} + \epsilon_{i,j} \) for \( i \in T_i \). It is easy to see that for \( x \) such that \( f_i(x) = 1 \) we have \( \Delta(p_{i,j}, x) = \frac{n}{4} - |S \cup T| \) and for \( x \) such that \( f_i(x) = 0 \) we have \( \Delta(p_{i,j}, x) \geq \frac{n}{4} - |S \cup T| - 1 |(\epsilon_{i,j} - \epsilon_{i,j}^2)| + 1 \) for \( \epsilon_{i,j} < \frac{1}{2} \).

We fix \( \epsilon_{i,j} \) in such a way that \( \frac{n}{4} - |S \cup T| |(\epsilon_{i,j} - \epsilon_{i,j}^2)| < \frac{n}{2} - \frac{1}{2} \) and \( \frac{n}{4} - |S \cup T| - 1 |(\epsilon_{i,j} - \epsilon_{i,j}^2)| + 1 > \frac{n}{4} + \frac{1}{2} \). We set label\( (p_{i,j}) = j \).

We construct anchors \( q_{i,j} \) for \( i = 1, \cdots, s+1 \) and \( j = 0, 1 \) the same way as above and assign label\( (q_{11}) \) to be equal to \( g(y) \) for \( y \) of weight \( i - 1 \) and label\( (q_{i0}) \) 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 8

**Proof:** Consider a function \( f \in \text{ELDL} \) and suppose the linear forms in its representation are \( L_1, \cdots, L_n \). Here \( L_i \) corresponds to the \( i \)'th query. As in the proof of Theorem 6, 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 5. 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 \( L_i(x) = 0 \) and all previous linear forms are non-zero. We then have that \( (s+1)L_i(x) + i - 1 = i - 1 \). It is not hard to see that for \( j < i \) we have that among forms \( (s+1)L_j(x) + j - 1 \) and \( -(s+1)L_j(x) - j + 1 \) exactly one is greater than \( i - 1 \): it is the first one if \( L_j(x) > 0 \) and the second one if \( L_j(x) < 0 \). For \( j > i \) in a similar way we can see that among the forms \( (s+1)L_j(x) + j - 1 \) and \( -(s+1)L_j(x) - j + 1 \) exactly one is greater than \( i - 1 \): it is the first one if \( L_j(x) \geq 0 \) and the second one if \( L_j(x) < 0 \). As a result there are exactly \( s - 1 \) forms that are greater than \( (s+1)L_i(x) + i - 1 \). We assign to this form the same label \( L_i(x) \) 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 \( \text{ELDL} \subseteq \text{kSTAT} \).

**Remark 9:** Note that decision lists are computable in \( \text{AC}^0 \) and thus can be computed by quasi-polynomial-size \( \text{SYM} \circ \text{AND} \) circuits. As a result, ELDL can be computed by quasi-polynomial-size circuit in \( \text{SYM} \circ \text{AND} \circ \text{ETHR} = \text{SYM} \circ \text{ETHR} = \text{SYM} \circ \text{MAJ} \), where the second equality follows since \( \text{ETHR} \) is closed under \( \text{AND} \) operation. Still, Theorem 8 gives a polynomial reduction that translates to the case of small coefficients.
A.11 Proof of Theorem 11

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

**Lemma 7.** For any even integer \( k > 0 \), there exists a CNF with \( n \) variables and \( n2^{k-o(k)}/k \) clauses with \( 2^{(1-o(k))n} \) components.

**Proof.** Assume \( k \) divides \( n \). Divide the set of variables to \( n/k \) disjoint sets \( S_1, \ldots, S_{n/k} \) 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 \( S_i \) are equal to 1. This can be achieved with \( \binom{k}{k/2} = 2^{k-o(k)} \) clauses.

Then, the CNF \( C = C_1 \land \cdots \land C_{n/k} \) has exactly \( 2^{(1-o(k))n} \) 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 11 follows from Lemmas 3 and 7 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 Murphy [1992] 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 Kilic et al. [2023].

**Theorem 14 (Murphy [1992]).**

- \( \text{NN} \subseteq \text{OR} \circ \text{AND} \circ \text{THR} \), \( \text{AND} \circ \text{OR} \circ \text{THR} \)
- \( \text{HNN} \subseteq \text{OR} \circ \text{AND} \circ \text{MAJ} \), \( \text{AND} \circ \text{OR} \circ \text{MAJ} \)

Namely, every NN (HNN) representation is computed by a depth-three \( \text{AC}^0 \circ \text{THR} \) (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 8.**

\( \text{HNN} \subseteq \text{OR} \circ \text{AND} \circ \text{MAJ} \)

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

**Proof.** Note that \( 1[\Delta(x,p) \leq i] \) is computed by a threshold gate \( f_{\leq i}^p(x) \) defined by \( w = p - \bar{p} \) and \( \theta = \Delta(p) - i. \) (And similarly \( 1[\Delta(x,p) \geq i] \).) Suppose \( f \) has an HNN representation \( P \cup N \). Then,

\[
f(x) = \bigvee_{i \leq n} \left( f_{\leq i}^p(x) \land \bigwedge_{q \in N} f_{\geq i}^q(x) \right)
\]

\]
Note that the threshold circuits from Theorem 14 and Lemma 8 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 15.**

\[ \text{HNN} \subseteq \text{THR} \circ \text{MAJ}. \]

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

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

\[ g_i^p(x) := f_{\leq i}^p(x) + f_{\geq i}^p(x) - 1 \]

and note that $g_i^p(x) = 1[\Delta(x, p) = i]$. We can then write the output gate as

\[ h(x) = 1 \left( \sum_{p \in P, i \leq n} m^{3(n-i)+1} g_i^p(x) - \sum_{q \in N, i \leq n} m^{3(n-i)} g_i^q(x) \geq 0 \right). \]

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

\[ \sum_{p \in P, i \leq n} m^{3(n-i)+1} g_i^p(x) \geq m^{3(n-j)+1} \geq \sum_{q \in N, i \leq n} m^{3(n-i)} g_i^q(x). \]

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

\[ \sum_{p \in P, i \leq n} m^{3(n-i)+1} g_i^p(x) \leq m^{3(n-j)-1} < m^{3(n-j)} \leq \sum_{q \in N, i \leq n} m^{3(n-i)} g_i^q(x). \]

Remark 10. Theorem 15 can be obtained through Theorem 1, as a consequence of the following result derived from Hansen and Podolskii [2015]. We include the direct construction to avoid the slight increase in circuit size.

**Lemma 9.**

\[ \text{mpPTF}(\text{poly}(n)) \subseteq \text{THR} \circ \text{MAJ} \]

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

**Proof.** Let $\text{PTF}_{1,2}$ refer to Boolean functions (over $\{1, 2\}$) equal to the sign of an $n$-variate polynomial. Hansen and Podolskii [2015] prove that any $\text{PTF}_{1,2}$ with $\ell$ 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 $\ell$ terms and maximum weight $W$ can be represented by a $\text{PTF}_{1,2}$ with $\ell$ terms and degree at most $2W \cdot \log \ell$.

Remark 11. 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.