String Matching: Communication, Circuits, and Learning

Alexander Golovnev* Mika Göös† Daniel Reichman‡ Igor Shinkar§

May 3, 2019

Abstract

String matching is the problem of deciding whether a given n-bit string contains a given k-bit pattern. We study the complexity of this problem in three settings.

- **Communication complexity.** For small k, we provide near-optimal upper and lower bounds on the communication complexity of string matching. For large k, our bounds leave open an exponential gap; we exhibit some evidence for the existence of a better protocol.
- **Circuit complexity.** We present several upper and lower bounds on the size of circuits with threshold and DeMorgan gates solving the string matching problem. Similarly to the above, our bounds are near-optimal for small k.
- **Learning.** We consider the problem of learning a hidden pattern of length at most k relative to the classifier that assigns 1 to every string that contains the pattern. We prove optimal bounds on the VC dimension and sample complexity of this problem.

Contents

*Harvard University. Email: alexgolovnev@gmail.com. Research supported by a Rabin Postdoctoral Fellowship.
†Institute for Advanced Study, Princeton, NJ, USA. Email: mika@ias.edu
‡Department of Computer Science, Princeton University. Email: daniel.reichman@gmail.com
§School of Computing Science, Simon Fraser University. Email: ishinkar@sfu.ca

1 Introduction

One of the most fundamental and frequently encountered tasks by minds and machines is that of detecting patterns in perceptual inputs. A basic example is the *string matching* problem, where given a string $x \in \{0,1\}^n$ and a pattern $y \in \{0,1\}^k$, $k \leq n$, the goal is to decide whether x contains y as a substring. Formally, denoting by $x[i,j]$ the bits of x in the interval $[i,j] \coloneqq \{i, i+1, \ldots, j\}$, we define a Boolean function by

$$\mathrm{SM}_{n,k}(x,y) \coloneqq 1 \qquad \text{iff} \qquad x[i,i+k-1] = y \text{ for some } i \in [n-k+1].$$

String matching is well-studied in the context of traditional algorithms: it can be computed in linear time [BM77, KMP77, GS83] (with some lower bounds given by [Riv77]). It has also been studied in more modern algorithmic frameworks such as streaming [PP09], sketching [BJKK04], and property testing [BEKR17]. See Section 2 for more related work.

In this work we study the $\mathrm{SM}_{n,k}$ problem in three models of computation, where it appears to have received relatively little attention.

1. *Communication complexity:* How many bits of communication are required to compute $\mathrm{SM}_{n,k}$ when the input (x,y) is adversarially split between two players?

2. *Circuit complexity:* How many gates are needed to compute $\mathrm{SM}_{n,k}$ by DeMorgan circuits (possibly in low depth)? How about threshold circuits?

3. *Learning:* How many labeled samples of strings must be observed in order to (PAC) learn a classifier assigning 1 to a string if and only if it contains a (fixed) hidden pattern y? What is the VC dimension of this problem?

1.1 Results: Communication Complexity

We first show bounds on the randomized two-party communication complexity of $\mathrm{SM}_{n,k}$. (For standard textbooks on communication complexity, see [KN97, Juk12].) The only related prior work we are aware of is Bar-Yossef et al. [BJKK04] who studied the *one-way* communication complexity of string matching; our focus is on *two-way* communication. Our bounds are near-optimal for small k, but for large $k \geq \Omega(n)$, we leave open a mysterious exponential gap. Our protocols work regardless of how the input bits (x,y) are bipartitioned between the players, whereas our lower bound is proved relative to some fixed hard partition.

Theorem 1.1 (Communication Complexity)**.** *For the* $\mathrm{SM}_{n,k}(x,y)$ *problem:*

- **Upper bound:** *Under any bipartition of the input bits, there is a protocol of cost*

$$\begin{array}{rll} \text{Deterministic:} & O(\log k \cdot n/k) & \text{if } k \leq \sqrt{n}\ ; \\ \text{Randomized:} & O(\log n \cdot \sqrt{n}) & \text{if } k \geq \sqrt{n}. \end{array}$$

- **Lower bound:** *For* $k \geq 2$ *there is a bipartition of the input bits such that every randomized protocol requires* $\Omega(\log\log k \cdot n/k)$ *bits of communication, even for the fixed pattern* $y = 1^k$.

Remark 1.2. Note that the most natural bipartition, where Alice gets x and Bob gets y, is easy. Indeed, for such partition there is a randomized $O(\log n)$-bit protocol, where Bob sends to Alice a hash of y, and Alice compares it with the hashes of the substrings $x[1,k]$, $x[2,k+1]$,..., $x[n-k+1,n]$. Under this bipartition, by setting $k=n$, one can also recover the usual *equality* problem, which is well-known to have deterministic communication complexity $\Omega(n)$. This explains why nontrivial protocols for large k need randomness.

A better protocol? For simplicity of discussion, consider the case $k=n/2$.

What is the randomized communication complexity of $\mathrm{SM}_{n,n/2}$*?*

Our bounds, $\Omega(\log\log n)$ and $O(\log n\cdot\sqrt{n})$, leave open a huge gap. We conjecture that the answer is closer to the lower bound. As formal evidence we show that problems closely related to $\mathrm{SM}_{n,n/2}$ admit efficient "unambiguous randomized" (aka $\mathsf{U{\cdot}BPP}$) communication protocols. A classic result [Yan91] says that any "unambiguous deterministic" (aka $\mathsf{U{\cdot}P}$) protocol can be efficiently simulated by a deterministic one, that is, $\mathsf{U{\cdot}P}=\mathsf{P}$ in communication complexity. A randomized analogue of this, $\mathsf{U{\cdot}BPP}=\mathsf{BPP}$, turns out to be false as a consequence of the recent breakthrough of Chattopadhyay et al. [CMS19]. One can nevertheless interpret our $\mathsf{U}\cdot\mathsf{BPP}$ protocols as evidence for the existence of improved randomized protocols.

Techniques. Our lower bound in Theorem 1.1 requires proving a tight randomized lower bound for composed functions of the form $\mathrm{OR}\circ\mathrm{GT}$ (where GT is the *greater-than* function), which answers a question of Watson [Wat18]. We observe that the lower bound follows by a minor modification of existing *information complexity* techniques [BW16]. For upper bounds, the role of periods in strings plays a central role (Section 3.1). We go on to discuss a natural *period finding* problem, and conjecture that it is easy for randomized protocols. See Section 3.4 for details.

The communication complexity and circuit complexity of $\mathrm{SM}_{n,k}$ are related. As we soon demonstrate, our study of the communication complexity of $\mathrm{SM}_{n,k}$ results with circuit lower bounds for threshold circuits computing $\mathrm{SM}_{n,k}$.

1.2 Results: Circuit Complexity

Threshold circuits. A threshold circuit is a circuit whose gates compute *linear threshold functions* (LTFs). Recall that an LTF outputs 1 on an m-bit input x if and only if $\sum_{i\in[m]}a_ix_i\geq\theta$ for some fixed coefficient vector $a\in\mathbb{R}^m$, and $\theta\in\mathbb{R}$. The study of threshold circuits is often motivated by its connection to neural networks [HMP+93, PS88, Par94, MCPZ13, Mur71]. The case of *low-depth* threshold circuits is also interesting. In particular, one line of work [SBKH93, Raz92a, SB91] has focused on efficient low-depth threshold implementations of arithmetic primitives (addition, comparison, multiplication). As for lower bounds, [HMP+93] show an exponential-in-n lower bound for the *mod-2 inner-product* function against depth-2 threshold circuits of low weight (see [FKL+01] for an extension). Superlinear lower bounds on the number of gates of arbitrary depth-2 as well as low-weight depth-3 threshold circuits were proven recently by Kane and Williams [KW16].

It is important that we measure the size of a threshold circuit as the *number of gates* (excluding inputs), in which case even superconstant lower bounds are meaningful. For example, it is easy to implement the equality function (namely $\mathrm{SM}_{n,n}$) using three threshold gates (albeit, with exponential weights). Thus, in contrast to the case of bounded fanin circuits, proving linear or

even nonconstant lower bounds on the number of gates is not straightforward. Indeed, there are few explicit examples of functions with superconstant lower bounds [GT93], and proving them is considered challenging [ROS94]. Indeed, Jukna [Juk12] writes "even proving non-constant lower bounds ... is a nontrivial task".

We show that $\mathrm{SM}_{n,k}$ admits a linear-size implementation at low depth. Thereafter we focus on its fine-grained complexity, seeking to establish lower bounds as close to $\Omega(n)$ as possible.

Theorem 1.3 (Threshold circuits)**.** *For the* $\mathrm{SM}_{n,k}(x,y)$ *problem:*

- **Upper bound:** *There is a depth-2 threshold circuit of size* $O(n-k)$*.*
- **Lower bound for unbounded depth:** *Any threshold circuit must be of size*

$$\begin{array}{ll} \Omega(\frac{n \log\log k}{k \log n}) & \text{if } k > 1; \\ \Omega(\sqrt{n/k}) & \text{if } k \geq 2.1 \cdot \log n. \end{array}$$

The second lower bound is stronger than the first one in the regime $k = \Omega(n \cdot (\frac{\log\log n}{\log n})^2)$. We note that for $k \leq \mathrm{polylog}(n)$, we have nearly linear lower bounds for unbounded-depth threshold circuits computing $\mathrm{SM}_{n,k}$. We stress that there are no restrictions on the weights of the threshold gates in these lower bounds. We are not able to prove $\Omega(n)$ lower bounds even for depth-2 threshold circuits. Proving such lower bounds (or constructing a threshold circuit of size $o(n)$) remains open. We can prove strong lower bounds for depth-2 circuits in some special cases (see Section 4.3).

Techniques. In Section 4.2 we obtain lower bounds for threshold circuits from the lower bounds on communication complexity of $\mathrm{SM}_{n,k}$ using a connection between threshold complexity and circuit complexity outlined by [Nis93]. We also prove lower bounds for threshold circuits by reducing the problem of computing a "sparse hard" function to computing $\mathrm{SM}_{n,k}$. Perhaps surprisingly, we show that the string matching problem can encode a truth table of an arbitrary sparse (few preimages of 1) Boolean function.

DeMorgan circuits. We consider usual DeMorgan circuits (AND, OR, NOT gates) of *unbounded fan-in* and show upper and lower bounds on the circuit complexity of $\mathrm{SM}_{n,k}$. We emphasize again that we measure the size of a circuit as the *number of gates* (excluding inputs). For example, the n-bit AND can be computed with a circuit of size 1.

We start by analyzing the case of low-depth circuits.

Theorem 1.4 (Depth-2 DeMorgan circuits)**.** *For the* $\mathrm{SM}_{n,k}(x,y)$ *problem:*

- **Depth-2 upper bound:** *There is a depth-2 DeMorgan circuit of size* $O(n \cdot 2^k)$*.*
- **Depth-2 lower bound:** *Any depth-2 DeMorgan circuit must be of size*

$$\begin{array}{ll} \Omega(n \cdot 2^k) & \text{if } 1 < k \leq \sqrt{n}\ ; \\ \Omega(2^{2\sqrt{n-k+1}}) & \text{if } k \geq \sqrt{n}. \end{array}$$

For $k \leq \sqrt{n}$, our depth-2 results are optimal (up to a constant factor). For large k, say $k = n/2$, there is (similarly as for communication) a huge gap in our bounds: $2^{\Omega(\sqrt{n})}$ versus $2^{O(n)}$. We do not know what bound to conjecture here as the correct answer.

For DeMorgan circuits, the celebrated Håstad's switching lemma [Hås87] established exponential lower bounds for bounded depth circuits computing explicit functions (e.g., majority, parity). We note that in contrast to the parity function, the string matching function admits a polynomial size circuit of depth 3. It is unclear (to us) how to leverage known tools for proving lower bounds for small depth circuits (such as the switching lemma) towards proving super linear lower bounds for small depth DeMorgan circuits computing $\mathrm{SM}_{n,k}$. Whether the string matching problem can be computed by a depth 3 (or even unrestricted) DeMorgan circuit of size $O(n)$ remains open.

Next, we prove that the circuit complexity of $\mathrm{SM}_{n,k}$ for general DeMorgan circuits (unrestricted depth and fan-in) must be $\Omega(n)$. We also include a relatively straightforward upper bound (which may have been discovered before; [Gal85] claims an upper bound $O(n \log^2 n)$ without a proof).

Theorem 1.5 (General DeMorgan circuits)**.** *For the* $\mathrm{SM}_{n,k}(x, y)$ *problem:*

- **Upper bound:** *There is a DeMorgan circuit of size* $O(nk)$ *and depth* 3*.*
- **Lower bound:** *Any DeMorgan circuit must be of size at least* $n/2$*.*

Techniques. We prove the lower bound on DNF by exhibiting an explicit set of inputs to $\mathrm{SM}_{n,k}$ each of which requires a separate clause in any DNF. Our lower bound for CNF involves estimating the size of maxterms of $\mathrm{SM}_{n,k}$. For the lower bound against circuits of unrestricted depth, we adjust the gate elimination technique to the case of unbounded fan-in circuits. See Section 5 for details.

1.3 Results: Learning

Finally, we seek to understand the sample complexity of PAC-learning the string matching function $\mathrm{SM}_{n,\ell}(x, \sigma)$, where x is an arbitrary string of length n and σ is a *fixed* pattern of length $\ell \leq k$. Towards this goal we prove (almost) tight bounds on the VC dimension of the class of these functions. The VC dimension essentially determines the sample complexity needed to learn the pattern σ from a set of i.i.d. samples in the PAC learning framework. We formalize these notions below.

Let Σ be a fixed finite alphabet of size $|\Sigma| \geq 2$.[1] By Σ^n we denote the set of strings over Σ of length n, and by $\Sigma^{\leq k}$ we denote the set of strings of length at most k. We study the VC dimension of the class of functions, where each function is identified with a pattern of length at most k, and outputs 1 only on the strings containing this pattern. Recall that the length of the pattern $k = k(n) \leq n$ can be a function of n. We now define the set of functions we wish to learn:

Definition 1.6. For a fixed *finite* alphabet Σ and an integer $k > 0$, let us define the class of Boolean functions $\mathcal{H}_{k,\Sigma}$ over Σ^n as follows. Every function $h_\sigma \in \mathcal{H}_{k,\Sigma}$ is parameterized by a pattern $\sigma \in \Sigma^{\leq k}$ of length at most k. Hence, $|\mathcal{H}_{k,\Sigma}| = \frac{|\Sigma|^{k+1}-1}{|\Sigma|-1}$. For a string $s \in \Sigma^n$, $h_\sigma(s) = 1$ if and only if s contains σ as a substring.

To analyze the sample complexity required to learn a function from $\mathcal{H}_{k,\Sigma}$ we first define *VC dimension*.

Definition 1.7. Let $\mathcal{F}$ be a class of functions from a set D to $\{0, 1\}$, and let $S \subseteq D$. A *dichotomy* of S is one of the possible labellings of the points of S using a function from $\mathcal{F}$. S is *shattered* by $\mathcal{F}$ if $\mathcal{F}$ realizes all $2^{|S|}$ dichotomies of S. The VC *dimension* of $\mathcal{F}$, $\mathrm{VC}(\mathcal{F})$, is the size of the largest set S shattered by $\mathcal{F}$.

[1] In contrast to the circuit and communication setting, for the learning problem we consider nonbinary alphabets.

In particular, $\mathrm{VC}(\mathcal{H}_{k,\Sigma}) = d$ if and only if there is a set S of d strings of length n such that for every $S' \subseteq S$, there exists a pattern P_S of length at most k occurring in all the strings in S' and not occurring in all the strings in $S \setminus S'$.

A class of functions $\mathcal{F}$ is PAC-learnable[2] with accuracy ε and confidence $1-\delta$ in $\Theta\left(\frac{\mathrm{VC}(\mathcal{F})+\log(1/\delta)}{\varepsilon}\right)$ samples [BEHW89, EHKV89, Han16], and is agnostic PAC-learnable in $\Theta\left(\frac{\mathrm{VC}(\mathcal{F})+\log(1/\delta)}{\varepsilon^2}\right)$ samples [AB09, SSBD14]. Thus, tight bounds on the VC dimension of a class of functions give tight bounds on its sample complexity.

Our main result is a tight bound on the VC dimension of $\mathcal{H}_{k,\Sigma}$ (up to low order terms). That is:

Theorem 1.8. *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

$$\mathrm{VC}(\mathcal{H}_{k,\Sigma}) = \min(\log|\Sigma|(k - O(\log k)), \log n + O(\log\log n)) \ .$$

It follows that the sample complexity of learning patterns is $O(\log n)$. We also show that there are efficient polynomial time algorithms solving this learning problem. See Corollary 6.6 for details.

Techniques. We prove our upper bound on the VC dimension by a double counting argument. This argument uses Sperner families to show that shattering implies a "large" family of non-overlapping patterns, which, on the other hand, is constrained by the length n of the strings that we shatter. The lower bound is materialized by the idea to have 2^d patterns $P = \{p_0 \dots p_{2^d-1}\}$ and d strings such that the ith string is a concatenation of all patterns with the binary expansion of their index having the ith bit equal 1. We construct a family of patterns T with the property that for any pair of distinct strings $\alpha, \beta \in T$, their concatenation $\alpha\beta$ does not contain a string $\gamma \in T, \gamma \neq \alpha, \beta$. Using this family (with some additional technical requirements) we are able to show that P shatters a set of d strings implying our lower bound on the VC dimension.

2 More related work

Circuit complexity. *Upper bounds* on the circuit complexity of 2D image matching problem under projective transformations was studied in [Ros16]. In this problem, which is considerably more complicated than the pattern matching problems we study, the goal is to find a projective transformation f such that $f(A)$ "resembles"[3] B for two images A, B. Here, images are 2D square arrays of dimension n containing discrete values (colors). In particular, it is proven that this image matching problem is in TC^1 (it admits a threshold circuit of polynomial size and logarithmic depth in n). These results concern a different problem than the string matching considered here, and do not seem to imply the upper bounds we obtain for circuits solving the string matching problem.

The idea to lower bound the circuit complexity of Boolean functions that arise in feature detection was studied in [LM01, LM02]. These works assumed a setting with two types of features, a and b, with detectors corresponding to the two types situated on a 1D or 2D grid. The binary outputs of these features are represented by an array of n positions: $a_1, ..., a_n$ (where $a_i = 1$ if the feature a is detected in position i, and $a_i = 0$ otherwise) and an array $b_1, ..., b_n$ which is analogously defined with respect to b. The Boolean function P^n_{LR} outputs 1 if there exist i, j with $i < j$ such that $a_i = b_j = 1$, and 0 otherwise. This function is advocated in [LM02] as a simple example of

[2] For a precise definition of PAC learning, see Definition 6.5.

[3] We refer to [Ros16] for the precise definition of distance used there.

a detection problem in vision that requires to identify spatial relationship among features. It is shown that this problem can be solved by $O(\log n)$ threshold gates. A 2-dimensional analogue where the indices $i = (i_1, i_2)$ and $j = (j_1, j_2)$ represent two-dimensional coordinates and one is interested whether there exist indices i and j such that $a_i = b_j = 1$ and j is above and to the right of the location i is studied in [LM02]. Recently, the two-dimensional version was studied in [UYZ15] where a $O(\sqrt{n})$-gate threshold implementation was given along with a lower bound of $\Omega(\sqrt{n/\log n})$ for the size of any threshold circuit for this problem. We remark that the problem studied in [LM01, LM02, UYZ15] is different from ours, and different proof ideas are needed for establishing lower bounds in our setting.

Learning patterns. The language of all strings (of arbitrary length) containing a fixed pattern is regular and can be recognized by a finite automaton. There is a large literature on learning finite automata (e.g., [Ang87, FKR+97, RR97]). This literature is mostly concerned with various active learning models and it does not imply our bounds on the sample complexity of learning $\mathcal{H}_{k,\Sigma}$.

Motivated by computer vision applications, several works have considered the notion of *visual concepts*: namely a set of shapes that can be used to classify images in the PAC-learning framework [KR96, Shv90]. Their main idea is that occurrences of shapes (such as lines, squares etc.) in images can be used to classify images and that furthermore the representational class of DNF's can represent occurrences of shapes in images. For example, it is easy to represent the occurrence of a fixed pattern of length k in a string of size n as a DNF with $n-k$ clauses (see e.g., Lemma 5.1). We note that these works do not study the VC dimension of our pattern matching problems (or VC bounds in general). We also observe that no polynomial algorithm is known for learning DNF's and that there is some evidence that the problem of learning DNF is intractable [DSS16]. Hence the result in [KR96, Shv90] do not imply that our pattern learning problem (represented as a DNF) can be done in polynomial time.

3 Communication Complexity

In this section we prove Theorem 1.1, and also discuss the possibility of a better upper bound.

Theorem 1.1 (Communication Complexity)**.** *For the* $\mathrm{SM}_{n,k}(x, y)$ *problem:*

- **Upper bound:** *Under any bipartition of the input bits, there is a protocol of cost*

$$\begin{array}{rll} \text{Deterministic:} & O(\log k \cdot n/k) & \text{if } k \le \sqrt{n}\ ; \\ \text{Randomized:} & O(\log n \cdot \sqrt{n}) & \text{if } k \ge \sqrt{n}. \end{array}$$

- **Lower bound:** *For* $k \ge 2$ *there is a bipartition of the input bits such that every randomized protocol requires* $\Omega(\log\log k \cdot n/k)$ *bits of communication, even for the fixed pattern* $y = 1^k$.

3.1 Periods in strings

We say a string $x \in \{0,1\}^n$ has *period* $p \in \{0,1\}^i$ *of order* i if x is a prefix of a high enough power p^m (for some $m \ge 1$). Equivalently, x has a period of order i iff $x[i+1, n] = x[1, n-i-1]$. A classic lemma characterizes the orders of short periods in a string.

Lemma 3.1 ([LS62])**.** *If* x *has periods of orders* i, j, $i + j \le |x|$, *then there is one of order* $\gcd(i, j)$.

In particular, all periods of order $\leq n/2$ are powers of some *primitive period* (shortest period of order $\leq n/2$). It is natural to ask: how many bits of communication are required to decide whether a string has a primitive period? We will discuss this in Section 3.4.

3.2 Upper bound

We start by describing an $O(\log k \cdot n/k)$-bit deterministic protocol for $\mathrm{SM}_{n,k}$ assuming the pattern y is fixed (known to both players). This immediately gives a protocol of cost $O(k + \log k \cdot n/k)$ when y is *not* fixed: Alice and Bob simply exchange all bits of the k-bit pattern and then run the protocol that assumes y is fixed. When $k \leq \sqrt{n}$ this yields the first upper bound claimed in Theorem 1.1.

Lemma 3.2. *For every fixed pattern $y \in \{0,1\}^k$ the function $x \mapsto \mathrm{SM}_{n,k}(x,y)$ admits a deterministic protocol of cost $O(\log k \cdot n/k)$ under any bipartition of the input x.*

Proof. Since every occurrence of pattern y in x must start in one of the n/k many intervals $[1,k],[k,2k],\ldots$, it suffices to to design a $O(\log k)$-bit protocol to test whether y occurs starting in a particular interval, and then repeat this protocol for every interval. Let us describe a protocol for the first interval $[1,k] = [k]$.

Suppose Alice is given the bits x_I for $I \subseteq [n]$ and Bob the bits $x_{\bar{I}}$ for $\bar{I} \coloneqq [n] \setminus I$. The protocol proceeds as follows. First, Alice sends two indices $i,j \in [k]$ where i (resp. j) is the smallest (largest) index such that it is consistent with Alice's bits x_I that y could appear in x starting at position i (j). (If there are no such indices, then the players may output "no match".) From i Bob can infer all Alice's bits in the interval $[i, i+|y|-1]$ (the bits agree with y, which is known to Bob), and similarly from j Bob can infer Alice's bits in $[j, j+|y|-1]$. Altogether Bob learns Alice's bits in $[i, i+|y|-1] \cup [j, j+|y|-1] = [i, j+k-1]$. Together with his own bits $x_{\bar{I}}$ Bob can then determine whether y occurs in x with a starting position in $[k]$. The cost of the protocol (sending the two indices and the final output value) is $2\log k + 1$. □

Next we supply the protocol for the second upper bound in Theorem 1.1.

Lemma 3.3. *For $k \geq \sqrt{n}$ the function $\mathrm{SM}_{n,k}$ admits a randomized protocol of cost $O(\log n \cdot \sqrt{n})$ under any bipartition of the input (x,y).*

Proof. At the start of the protocol, the two players exchange the first $2\sqrt{n}$ many bits of y so that they both learn the prefix $p \coloneqq y[1, 2\sqrt{n}]$. We think of p as fixed from now on. Since any occurrence of y in x must start in one of the $\sqrt{n}$ many intervals $[1,\sqrt{n}],[\sqrt{n},2\sqrt{n}],\ldots$ it suffices to design a $O(\log n)$-bit protocol (with error probability $\leq 1/n$) to test whether y starts in a particular interval, and then repeat this protocol for every interval (resulting in error probability $\leq \sqrt{n}/n$ by a union bound). Let us describe a protocol for the first interval $[1,\sqrt{n}] = [\sqrt{n}]$.

For simplicity of presentation, we first assume that p has no period of order $\leq \sqrt{n}$. We will handle a p with short periods later.

No short period. The protocol to test if y occurs in x starting at a position in $[\sqrt{n}]$ is similar to the one in Lemma 3.2. Assuming Alice is given x_I and Bob is given $x_{\bar{I}}$, Alice first sends two indices $i,j \in [\sqrt{n}]$ where i (resp. j) is the smallest (largest) index such that it is consistent with Alice's bits that the prefix p could appear in x starting at position i (j). Bob can again reconstruct all Alice's bits in the interval $[i, j+|p|-1]$ and determine whether p occurs in x with a starting position in $[\sqrt{n}]$. Since we are assuming that p has no period of order $\leq \sqrt{n}$, Bob can find at most one such

starting position, say at coordinate $\ell \in [\sqrt{n}]$. (If there is no starting position for the prefix, there is none for the full pattern y and we may output "no match".) The remaining goal becomes to test whether $x[\ell, \ell + k - 1] = y$. Consider any $i \in [k]$; either

(1) Alice (or Bob) owns both $x_{\ell-1+i}$ and y_i;
(2) Alice owns $x_{\ell-1+i}$ and Bob owns y_i (or vice versa).

For coordinates of type (1), the players may test for equality without communication. For coordinates of type (2), the players can execute a randomized test for equality—a single test for all type-(2) coordinates at once—for which there is a well-known $O(\log n)$-bit protocol (with error probability $\leq 1/n$) [KN97, Example 3.5]. This concludes the description of the $O(\log n)$-bit protocol (for a p without short periods).

Short period. Suppose $p \in \{0,1\}^{2\sqrt{n}}$ has a period of order $\leq \sqrt{n}$. Since the players know p, they can both agree on the shortest one (the primitive period), call it $\bar{p}$, $|\bar{p}| \leq \sqrt{n}$.

The players then proceed to find the largest number m such that $\bar{p}^m$ is a prefix of y. To do this, Alice (resp. Bob) reports the largest m_A (m_B) such that it is consistent with her (his) knowledge of the bits of y that $\bar{p}^{m_A}$ ($\bar{p}^{m_B}$) is a prefix of y. Then $m \coloneqq \min(m_A, m_B)$ is the sought number. This takes $O(\log n)$ bits of communication.

Next, the players can check, with constant communication, whether y is simply a prefix of $\bar{p}^{m+1}$. If yes, both players would fully know y and hence they can run the protocol from Lemma 3.2. Assume otherwise henceforth. In this case the players can find a string q, $|q| \leq |\bar{p}|$, that is not a prefix of $\bar{p}$, and such that $p' \coloneqq \bar{p}^m q$ is a prefix of y. This takes $|q| \leq |\bar{p}| \leq \sqrt{n}$ bits of communication.

We claim that p' has no period of order $\leq \sqrt{n}$. This claim would finish the proof, as the players can finally run the *no-short-period* protocol with p' in place of p (note that the cost of that protocol does not depend on $|p|$). To prove the claim, suppose for contradiction that p' (and hence p) has a period $\hat{p}$ of order $|\hat{p}| \leq \sqrt{n}$. Since $\bar{p}$ is the primitive period for p, $\hat{p}$ must be a power of $\bar{p}$. Therefore p' is a power of $\bar{p}$. But this contradicts our definition of $p' = \bar{p}^m q$. □

Remark 3.4. For $k \geq \sqrt{n \log n}$ the above protocol can be optimized to have cost $O(\sqrt{n \log n})$. Namely, consider a prefix p (and intervals) of length $\Theta(\sqrt{n \log n})$ rather than $\Theta(\sqrt{n})$.

3.3 Lower bound

Next we prove a lower bound of $\Omega(\log \log k \cdot n/k)$, for every $k \leq n$, on the randomized communication complexity of $\mathrm{SM}_{n,k}$. As a warm-up, we first observe that a reduction from the ubiquitous set-disjointness function yields a randomized lower bound of $\Omega(n/k)$ for $\mathrm{SM}_{n,k}$. We then show how to improve this by a factor of $\log \log k$.

Recall that in the m-bit set-disjointness problem, Alice is given $a \in \{0,1\}^m$, Bob is given $b \in \{0,1\}^m$, and their goal is to compute $\mathrm{Disj}_m(a,b) \coloneqq (\mathrm{OR}_m \circ \mathrm{AND}_2)(a,b) = \bigvee_{i \in [m]} (a_i \wedge b_i)$. It is well known that this function has communication complexity $\Omega(m)$ even against randomized protocols [KS92, Raz92b, BJKS04].

Observation 3.5. $\mathrm{Disj}_{\Omega(n/k)}$ *reduces to* $\mathrm{SM}_{n,k}$ *(under some bipartition of input bits).*

Proof. Given inputs (a,b) of Disj_m to Alice and Bob they construct, without communication, inputs to $\mathrm{SM}_{m(k+1),k}$ as follows. We set $y \coloneqq 1^k$ and

$$x \;\coloneqq\; a_1 b_1 1^{k-2} 0 a_2 b_2 1^{k-2} 0 \ldots a_n b_n 1^{k-2} 0.$$

This also implicitly determines the bipartition of input bits of $\mathrm{SM}_{m(k+1),k}$; namely, Alice gets all the coordinates of x with a_is, Bob gets those with b_is, and the rest can be split arbitrarily. It is straightforward to check that $\mathrm{Disj}_m(a,b) = \mathrm{SM}_{m(k+1),k}(x,y)$. □

To improve the above, we give a reduction from a slightly harder function, $\mathrm{OR}_m \circ \mathrm{GT}_\ell \colon [\ell]^m \times [\ell]^m \to \{0,1\}$, which maps $(a,b) \mapsto \bigvee_{i\in[m]} \mathrm{GT}(a_i,b_i)$ where $\mathrm{GT}_\ell \colon [\ell] \times [\ell] \to \{0,1\}$ is the *greater-than* function given by $\mathrm{GT}_\ell(a,b) \coloneqq 1$ iff $a \geq b$. The claimed lower bound $\Omega(\log\log k \cdot n/k)$ for $\mathrm{SM}_{n,k}$ follows from the following two lemmas. As mentioned in the introduction, Lemma 3.7 was conjectured by [Wat18].

Lemma 3.6. $\mathrm{OR}_{\Omega(n/k)} \circ \mathrm{GT}_{\Omega(k)}$ *reduces to* $\mathrm{SM}_{n,k}$ *(under some bipartition of input bits).*

Lemma 3.7. $\mathrm{OR}_m \circ \mathrm{GT}_\ell$ *has randomized communication complexity* $\Omega(m \cdot \log\log \ell)$ *for any* m, ℓ.

Proof of Lemma 3.6. It suffices to describe a reduction from GT_k to $\mathrm{SM}_{4k,2k+2}$ as this reduction can be repeated $\Omega(n/k)$ times in parallel on disjoint inputs (similarly as in the proof of Observation 3.5). Given inputs $(a,b) \in [k] \times [k]$ to GT_k the two players construct inputs (x,y) to $\mathrm{SM}_{4k,2k+2}$ as follows. As before, we set $y \coloneqq 1^{2k+2}$. As for x, Alice will own the even coordinates $I \coloneqq \{2,4,\ldots,4k\}$ of x and Bob the odd coordinates $\bar{I} \coloneqq [4k] \setminus I$. Alice sets $x_I \coloneqq 1^{k+a}0^{k-a}$ and Bob sets $x_{\bar{I}} \coloneqq 0^b 1^{2k-b}$. The longest all-1 pattern in x is then of length $2(k+a-b+1)$, as illustrated below.

$$k = 5,\quad a = b = 2 \qquad \rightsquigarrow \qquad x \coloneqq \begin{array}{l} \texttt{1 1 1 1 1 1 1 0 0 0} \quad \text{(Alice's bits } x_I\text{)} \\ \texttt{0 0}\underbrace{\texttt{1 1 1 1 1 1 1}}_{2(k+a-b+1)}\texttt{ 1 1} \quad \text{(Bob's bits } x_{\bar{I}}\text{)} \end{array}$$

Note that $2(k+a-b+1) \geq 2k+2$ iff $a \geq b$. Hence $\mathrm{GT}_k(a,b) = \mathrm{SM}_{4k,2k+2}(x,y)$, as desired. □

Proof of Lemma 3.7. A standard technique for proving randomized communication lower bounds for functions of the form $\mathrm{OR}_m \circ F$, where $F \colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, is *information complexity* (IC) [CSWY01, BJKS04, Bra12]. We explain how to combine existing methods to obtain the desired lower bound when $F = \mathrm{GT}_\ell$. Our discussion assumes familiarity with the IC technique.

The usual plan is to exhibit a *one-sided* distribution μ_0 over $F^{-1}(0)$ and prove, for any bounded-error protocol Π computing F, a lower bound on the *information cost* $I_{\mu_0} \coloneqq \mathbb{I}(\Pi(X,Y) : X \mid Y) + \mathbb{I}(\Pi(X,Y) : Y \mid X)$ where $XY \sim \mu_0$ (see [Bra12] for details on information cost). Bar-Yossef et al. [BJKS04] proved that the randomized communication complexity of $\mathrm{OR}_m \circ F$ is at least $m \cdot I_{\mu_0}$. Hence our goal is to show, for some one-sided μ_0 over $\mathrm{GT}_\ell^{-1}(0)$,

$$I_{\mu_0} \;\geq\; \Omega(\log\log \ell). \tag{1}$$

Braverman and Weinstein [BW16] already obtained a lower bound like (1) except for a *two-sided* distribution μ over $F^{-1}(0) \cup F^{-1}(1)$. Here we observe that their proof, virtually unchanged, gives the same lower bound also for a one-sided distribution μ_0.

BW simulation. Let us summarize the main technical result of [BW16]. They show a general simulation of any bounded-error, say $\leq 1\%$, protocol Π computing F with information cost I_μ relative to a μ by an "unbounded-error" protocol Π' (of communication cost $O(I_\mu)$) satisfying the following: With high probability, say $\geq 99\%$, over $(x,y) \sim \mu$, the simulation is *"successful"* (event $\mathcal{Z}$ in the proof of [BW16, Thm 2]) meaning that, for some $\delta \coloneqq 2^{-O(I_\mu+1)}$,

$$\forall (x,y) \in \mathcal{Z} : \qquad \mathbb{P}_{\text{coins of } \Pi'}[\,\Pi'(x,y) \text{ outputs } F(x,y)\,] \;\geq\; \tfrac{1}{2} + 0.9 \cdot \delta. \tag{2}$$

A crucial property is that even if the simulation fails for an input $(x, y) \notin \mathcal{Z}$, we are still guaranteed that Π' does not output the wrong answer with too high a probability [BW16, Prop 2]:

$$\forall (x,y): \qquad \mathbb{P}_{\text{coins of } \Pi'}[\,\Pi'(x,y) \text{ outputs } F(x,y)\,] \;\geq\; \tfrac{1}{2} - 0.1 \cdot \delta. \tag{3}$$

By averaging over $(x, y) \sim \mu$ it follows that

$$\begin{aligned} \mathbb{P}_{(x,y)\sim\mu,\,\text{coins of } \Pi'}[\,\Pi'(x,y) \text{ outputs } F(x,y)\,] \;&\geq\; \tfrac{1}{2} + (99\% \cdot 0.9 - 1\% \cdot 0.1)\delta \\ &\geq\; \tfrac{1}{2} + 0.8 \cdot \delta. \end{aligned} \tag{4}$$

In words, Π' achieves a non-trivial bias in guessing F relative to μ. The authors conclude [BW16, Thm 1] that Π' witnesses an $O(\log \delta^{-1}) = O(I_\mu + 1)$ *discrepancy bound* for F relative to μ. Finally, they provide an $\Omega(\log \log \ell)$ discrepancy bound for $F = \mathrm{GT}_\ell$ relative to a two-sided μ, which proves (1) (except for a two-sided μ).

Our modification. Our observation is that the BW simulation can be applied while assuming only an upper bound on I_{μ_0} for every one-sided μ_0, and still conclude (4) for any two-sided μ. Indeed, let μ be any two-sided distribution. We may assume wlog that μ is balanced, $\mu = \frac{1}{2}\mu_0 + \frac{1}{2}\mu_1$, where μ_b is over $F^{-1}(b)$. (If μ is highly unbalanced, say towards $F^{-1}(1)$, the trivial protocol that always outputs 1 satisfies (4); if μ is close to (but not quite) balanced, then one may adapt our following analysis for a balanced μ.) Suppose Π is a protocol for F with information cost I_{μ_0} relative to μ_0. Then from the BW simulation we can obtain Π' such that for some $\delta := 2^{-O(I_{\mu_0}+1)}$,

$$\mathbb{P}_{(x,y)\sim\mu_0,\,\text{coins of } \Pi'}[\,\Pi'(x,y) \text{ outputs } 0\,] \;\geq\; \tfrac{1}{2} + 0.8 \cdot \delta, \tag{5}$$

$$\mathbb{P}_{(x,y)\sim\mu_1,\,\text{coins of } \Pi'}[\,\Pi'(x,y) \text{ outputs } 1\,] \;\geq\; \tfrac{1}{2} - 0.1 \cdot \delta, \tag{6}$$

where the first bound is from (4) (specialized to μ_0) and the second bound is from the failure guarantee (3). The protocol Π' is still not correct with probability $> 1/2$ relative to μ_1 (only $1/2 - 0.1 \cdot \delta$ is guaranteed by (6)), but we may fix this at the cost of lowering the success probability relative to μ_0. Define a third protocol Π'' with a scaled-down probability of outputting 0: $\Pi''(x, y)$ outputs 1 with probability $\delta/2$ and with the remaining probability $1 - \delta/2$ it runs $\Pi'(x, y)$. Now

$$\begin{aligned} &\mathbb{P}_{(x,y)\sim\mu_0,\,\text{coins of } \Pi''}[\,\Pi''(x,y) \text{ outputs } 0\,] \;\geq\; (1-\delta/2)(\tfrac{1}{2} + 0.8 \cdot \delta) \;\geq\; \tfrac{1}{2} + 0.1 \cdot \delta, \\ &\mathbb{P}_{(x,y)\sim\mu_1,\,\text{coins of } \Pi''}[\,\Pi''(x,y) \text{ outputs } 1\,] \;\geq\; \delta/2 + (1-\delta/2)(\tfrac{1}{2} - 0.1 \cdot \delta) \;\geq\; \tfrac{1}{2} + 0.1 \cdot \delta. \end{aligned}$$

This satisfies (4), albeit with coefficient 0.1 rather than 0.8. □

3.4 A better protocol?

As bonus results, we give some evidence for the existence of an improved randomized protocol for $\mathrm{SM}_{n,k}$ when k is large. We first define what *unambiguous randomized* (aka $\mathsf{U}\cdot\mathsf{BPP}$, or unambiguous Merlin–Arthur) protocols are; they generalize the notion of unambiguous deterministic protocols (aka $\mathsf{U}\cdot\mathsf{P}$) introduced by Yannakakis [Yan91].

Definition 3.8 ($\mathsf{U}\cdot\mathsf{BPP}$ protocols)**.** An *unambiguous randomized* protocol Π computes a function $F(x, y)$ as follows. In the first phase the players nondeterministically guess a witness string $z \in \{0,1\}^{c_1}$, and then in the second phase they run a randomized (error $\leq 1/3$) protocol of cost c_2 to decide whether to accept the witness z. The correctness requirement is that for every $(x, y) \in F^{-1}(1)$ there needs to be a unique witness that is accepted; for every $(x, y) \in F^{-1}(0)$ no witness should be accepted. The cost of Π is defined as $c_1 + c_2$.

Unambiguous randomized protocols have not been studied before in communication complexity. However, the recent breakthrough of Chattopadhyay et al. [CMS19] (who disproved the log-approximate-rank conjecture of [LS09]) is closely related. It is not hard to see that the function $F(x, y)$ they study (of the form Sink $\circ$ XOR) admits an $O(\log n)$-cost $\mathsf{U}\cdot\mathsf{BPP}$ protocol. The authors proved that the usual randomized (aka BPP) communication complexity of F is high, $n^{\Omega(1)}$. Consequently, there is no generic simulation of a $\mathsf{U}\cdot\mathsf{BPP}$ protocol by a BPP protocol. By contrast, Yannakakis [Yan91, Lemma 1] showed that $\mathsf{U}\cdot\mathsf{P}$ protocols can be made deterministic efficiently.

Our first bonus result is an efficient $\mathsf{U}\cdot\mathsf{BPP}$ protocol for determining if a given string has a primitive period. We do not know whether there is an efficient randomized protocol.

Lemma 3.9. *Suppose the bits of $x \in \{0,1\}^n$ are split between two players. There is an $\mathsf{U}\cdot\mathsf{BPP}$ protocol of cost $O(\log^2 n)$ for deciding whether x has a primitive period (and to compute its order).*

Proof. Suppose Alice is given the bits x_I, $I \subseteq [n]$, and Bob the bits $x_{\bar{I}}$, $\bar{I} \coloneqq [n] \setminus I$. The idea is that Alice and Bob guess the order of the primitive period, and then verify their guess using randomness. The guess is just a $\log n$-bit number $k \in [n/2]$ having some prime factorization $k = p_1^{e_1} p_2^{e_2} \cdots p_\ell^{e_\ell}$ where $e_i \geq 1$ and $\ell \leq \log n$. In the randomized checking phase Alice and Bob run an $O(\log n)$-bit equality protocol (as in Lemma 3.3) to check whether $x[1, i]$ is a period (namely, they test the equality $x[i+1, n] = x[1, n-i-1]$). If yes, we continue to check that there is no shorter period. Since the shortest (primitive) period divides k, it suffices to check that none of the candidates $\{k/p_1, k/p_2, \ldots, k/p_\ell\}$ is a period. For each such candidate we run an equality protocol. Altogether this checking phase costs $O(\ell \log n) = O(\log^2 n)$ bits of communication. The protocol is indeed unambiguous since the primitive period (should it exist) is unique. □

If we let R_{pf} denote the randomized communication complexity of the above period finding problem, then we can interpret Lemma 3.9 as evidence that $R_{\mathrm{pf}} \leq \mathrm{polylog}(n)$. Assuming period finding is indeed easy, we can then provide similar evidence for the easiness of $\mathrm{SM}_{n,k}$ for large k.

Lemma 3.10. $\mathrm{SM}_{n,0.9n}$ *admits an $\mathsf{U}\cdot\mathsf{BPP}$ protocol of cost $O(\log n) + R_{\mathrm{pf}}$.*

Proof. The idea is that the players guess a position $i \in [n]$ ($\log n$ bits) and then verify, using randomness, that i is the starting point for the *earliest* occurrence of y in x. More precisely, in the verification phase, the players first run the R_{pf}-bit protocol to decide whether y has a primitive period (and compute its order). Observe that if y occurs more than once in x, then since $k = 0.9n$, the occurrences must overlap by $\geq 0.8n$ positions. In this case y has a period of order $\leq 0.1n \leq k/2$, and hence y has a primitive period. Two cases:

- *y does not have a primitive period.* Then y can appear at most once in x. The players run an $O(\log n)$-bit equality protocol (as in Lemma 3.3) to test whether y starts at position i in x.
- *y has a primitive period of order $\ell \in [k/2]$.* Then position i is the earliest occurrence of y in x iff (1) y starts at position i in x, and (2) y does not start at position $i - \ell$ in x. The conditions (1) and (2) can be checked by running an equality protocol twice.

□

4 Threshold Circuits

In this section we prove Theorem 1.3.

Theorem 1.3 (Threshold circuits)**.** *For the* $\mathrm{SM}_{n,k}(x,y)$ *problem:*

- **Upper bound:** *There is a depth-2 threshold circuit of size* $O(n-k)$*.*
- **Lower bound for unbounded depth:** *Any threshold circuit must be of size*

$$\begin{array}{ll} \Omega(\frac{n\log\log k}{k\log n}) & \text{if } k>1; \\ \Omega(\sqrt{n/k}) & \text{if } k \geq 2.1\cdot\log n. \end{array}$$

In Section 4.1 we prove the upper bound, in Section 4.2 we give the lower bounds. Finally, in Section 4.3 we study the complexity of $\mathrm{SM}_{n,k}$ in the models of restricted threshold circuits.

4.1 Upper bound

We start with a construction giving the upper bound of Theorem 1.3.

Lemma 4.1. *There is a depth-2 threshold circuit of size* $O(n-k)$ *computing* $\mathrm{SM}_{n,k}$*.*

Proof. Let GEQ be the gate that gets $2k$ bits $z_1,\ldots,z_k;w_1,\ldots,w_k$ and evaluates to 1 if and only if the number represented in binary by the bits $z_1,...,z_k$ is greater than or equal to the number represented by $w_i,...w_k$. Note that GEQ can be implemented by one threshold gate as follows: $\mathrm{GEQ}(z_1,\ldots,z_k;w_1,\ldots,w_k)=1$ if and only if $(z_1+2\cdot z_2+4\cdot x_3+...+2^{k-1}\cdot z_k)-(w_1+2\cdot w_2+4\cdot w_3+...+2^{k-1}\cdot w_k)\geq 0$.

Now we describe a circuit computing $\mathrm{SM}_{n,k}(x_1,\ldots,x_n;y_1,\ldots,y_k)$. Let the first layer contain $n-k+1$ GEQ gates g_i and $n-k+1$ gates ℓ_i for $i=1,\ldots n-k+1$, where each g_i gets as inputs $x_i,\ldots,x_{i+k-1};y_1,\ldots,y_k$ (and evaluates to 1 if and only if the number represented by the corresponding bits of x is *at least* that represented by y), and ℓ_i gets as inputs $y_1,\ldots,y_k;x_i,\ldots,x_{i+k-1}$ (and evaluates to 1 if and only if the number represented by the corresponding bits of x is *at most* that represented by y). The second (output) layer evaluates to 1 if and only if $\sum_{i=1}^{n-k+1}(g_i+\ell_i)\geq n-k+2$. Clearly the circuit contains $2n-2k+3$ gates.

In order to prove correctness, we note that for every i, at least one of g_i and ℓ_i evaluates to 1, and $x[i,i+k-1]=y$ if and only if both g_i and ℓ_i are equal to 1. Therefore, $\sum_{i=1}^{n-k+1}(g_i+\ell_i)>n-k+1$ if and only if y is a substring of x, i.e., $\mathrm{SM}_{n,k}(x,y)=1$. □

4.2 Lower bounds

In order to prove the first lower bound of $\Omega(\frac{n\log\log k}{k\log n})$ we use the classical result on communication complexity of threshold gates [Nis93], and the lower bound on communication complexity of $\mathrm{SM}_{n,k}$ from Theorem 1.1.

Nisan and Safra [Nis93] proved that for *any* bipartition of the n input bits, the ϵ-error randomized communication complexity of a threshold gate (with arbitrary weights) has communication complexity $O(\log n/\epsilon)$. From this they concluded that for any function f, a lower bound of m on the randomized communication complexity for *some* bipartition of the input implies a lower bound of $\Omega(m/\log n)$ on the threhold complexity of f. Now the lower bound of $\Omega(n\log\log k/k)$ from Theorem 1.1 implies the lower bound of $\Omega(\frac{n\log\log k}{k\log n})$ on the size of an unbounded depth threshold circuit computing $\mathrm{SM}_{n,k}$.

Below we prove the second lower bound stated in Theorem 1.3. The lower bound is shown via a reduction from a hard function $f\colon\{0,1\}^{k/2-1}\to\{0,1\}$ which has n/k preimages of 1: $|f^{-1}(1)|=n/k$.

First, we prove the desired lower bound for the case where k is even and n is a multiple of k. In the end of this section we explain how to adjust the proof to the remaining cases. Let ℓ and t be integers such that $k = 2\ell + 2$ and $n = t \cdot k$. Let $F_{\ell,t} = \{f \colon \{0,1\}^\ell \to \{0,1\} : |f^{-1}(1)| = t\}$ be the class of Boolean functions of ℓ inputs which have exactly t preimages of 1.

We prove this lower bound via a reduction from a hard function $f \in F_{\ell,t}$. Specifically, we show that if $\mathrm{SM}_{n,k}$ can be solved by a circuit of size s, then every function $f \in F_{\ell,t}$ also has a circuit of size s computing it. Then, we show that there are functions in $F_{\ell,t}$ that require large threshold circuits, which implies the corresponding lower bound for the $\mathrm{SM}_{n,k}$ function.

The reduction. Given a string $a \in \{0,1\}^\ell$ define $dup(a) \in \{0,1\}^k$ to be the string obtained from a by repeating each bit of a twice, and concatenating it with 01 in the end. (Note that $2\ell + 2 = k$ by the choice of ℓ). For example $dup(010) = 00110001$.

Observation 4.2. *Given a function $f \in F_{\ell,t}$ define $x_f \in \{0,1\}^{tk}$ to be the concatenation of $dup(a)$ for all $a \in f^{-1}(1)$ in the lexicographic order on $\{0,1\}^\ell$. Note that $|x_f| = tk = n$. Then, for any $y \in \{0,1\}^\ell$ it holds that $f(y) = 1$ if and only if $\mathrm{SM}_{n,k}(x_f, dup(y)) = 1$.*

Indeed, it is immediate to see that if $f(y) = 1$ then $\mathrm{SM}_{n,k}(x_f, dup(y)) = 1$. Duplicating every bit in a and adding 01 to the end of the resulting pattern are done to ensure that if $f(y) = 0$ there will not be a copy of $dup(y)$ in x_f.

Given the observation above, it is not difficult to see that any lower bound on the size of a circuit computing $f \in F_{\ell,t}$ implies a lower bound on $\mathrm{SM}_{n,k}$.

Proposition 4.3. *Let C be a threshold circuit computing $\mathrm{SM}_{n,k}$. Then for every $f \in F_{\ell,t}$, there exists a threshold circuit C' computing f such that $|C'| \leq |C|$.*

Proof. Suppose there exists a circuit C of size at most s computing $\mathrm{SM}_{n,k}$. We denote the input variables of the pattern y by $y_1, y_2 \ldots y_k$, where $k = 2\ell + 2$. We show how to convert C into a circuit C' computing f by fixing some of the input variables of C. This is done by (1) fixing the "text part" (the variables corresponding to x) of the input of $\mathrm{SM}_{n,k}$ to be x_f as defined in Observation 4.2, and (2) replacing every pair of variables y_{2i-1} and y_{2i} for all $i = 1, \ldots, \ell$ by a single variable $\widehat{y}_i$ that is fed to all gates that have inputs y_{2i-1} or y_{2i} (with a proper adjustment of the weight if both y_{2i-1} and y_{2i} are inputs of the gate). Finally, fix $y_{2\ell+1} = 0$ and $y_{2\ell+2} = 1$. It is now easy to see that C' computes f. □

In order to complete the proof of Theorem 1.3, we need to show that there exists a function $f \in F_{\ell,t}$ that requires large threshold circuits. For this, we compare the number of small threshold circuits (see, for example, [Juk12, KW16]) with the number of functions in $F_{\ell,t}$.

Proposition 4.4. *Let $\ell \in \mathbb{N}$ be sufficiently large, and let $t \in \mathbb{N}$. There exists a function $f \in F_{\ell,t}$ such that any threshold circuit (with no restrictions on its depth) computing f must be of size at least $\Omega(\sqrt{t - t\log t/\ell})$.*

Proof. We first upper bound the number of functions that can be represented by threshold circuits of size at most s. This can be obtained using the following result from [RSO94].

Theorem 4.5. *Let $f_1 \ldots f_s \colon \{0,1\}^\ell \to \{0,1\}$ be a set of s Boolean functions. Then, the number of Boolean functions which are realized by a threshold gate $g \colon \{0,1\}^s \to \{0,1\}$ whose s inputs are $f_1 \ldots f_s$ is at most $2^{O(\ell s)}$.*

It follows from Theorem 4.5 that the number of distinct Boolean functions with ℓ variables computed by a threshold circuit of size s is $2^{O(\ell s^2)}$ (as there are at most $2^{O(\ell s)}$ choices for every gate and there are s gates). On the other hand, the number of Boolean functions $f \in F_{\ell,t}$ is $\binom{2^\ell}{t} \geq (\frac{2^\ell}{t})^t = 2^{(\ell - \log t)t}$. Therefore, there exists a function $f \in F_{\ell,t}$ that cannot be computed by a threshold circuit of size $s \geq \Omega(\sqrt{t - t\log t/\ell})$. □

We now derive the desired lower bound on the size of threshold circuits computing the string matching function. Plugging in $k = 2\ell + 2$ and $n = tk$, we get the lower bound of $s \geq \Omega\big(\sqrt{\frac{n}{k} - \frac{2n}{k^2} \cdot \log(\frac{n}{k})}\,\big) = \Omega(\sqrt{\frac{n}{k}})$ assuming $k \geq \Omega(\log n)$.

Now we describe how this proof can be adopted for the case when n is not a multiple of k and the case of odd k. First, in order to handle the case of pattern of *odd* length, one can add the string 010 (instead of 01) to the end of $dup(a)$. If n is not a multiple of k, then in the reduction above we can pad the string x_f with zeros in the end, and the reduction still satisfies the property that $f(y) = 1$ if and only if $\mathrm{SM}_{n,k}(x_f, dup(y)) = 1$ as in Observation 4.2, and the same lower bound holds (up to a constant factor in the asymptotics).

4.3 Depth-2 Circuits

In Theorem 4.6 we prove lower bounds for some restricted classes of depth-2 circuits computing $\mathrm{SM}_{n,k}$. These results should be contrasted with the upper bounds of Theorem 1.3 and Theorem 1.5. Namely, there exists an LTF $\circ$ LTF circuit of size $O(n-k)$ and an OR $\circ$ AND $\circ$ OR circuit of size $O(nk)$ computing $\mathrm{SM}_{n,k}$.

We recall a few definitions. Let ELTF denote the class of *exact* threshold functions (that is, the functions which output 1 on an m-bit input x if and only if $\sum_{i\in[m]} a_i x_i = \theta$ for some fixed coefficient vector $a \in \mathbb{R}^m$, and $\theta \in \mathbb{R}$). Similarly, EMAJ denotes the class of exact majorities which output 1 if and only if the sum of their m Boolean inputs is exactly $m/2$. By SYM we denote the class of all symmetric Boolean functions. For two classes of functions $\mathcal{C}_1$ and $\mathcal{C}_2$, by $\mathcal{C}_1 \circ \mathcal{C}_2$ we denote the class of depth-2 circuits where the output gate is from $\mathcal{C}_1$ and the gates of the first layer are from $\mathcal{C}_2$. For a class of circuits $\mathcal{C}$ and a function f, be $\mathcal{C}(f)$ we denote the minimal size of a circuit from $\mathcal{C}$ computing f.

In proving lower bounds for $\mathrm{SM}_{n,k}$ a simple yet useful property is that Observation 3.5 can be applied to circuits as well. This allows to reduce the disjointness problem to string matching, and get lower bounds for $\mathrm{SM}_{n,k}$ via known circuit lower bounds for disjointness. The point is that a circuit C with strings of length roughly mk for $\mathrm{SM}_{n,k}$ (and patterns of length k) can be used to solve disjointness on strings of length m by feeding C with the string $x := a_1 b_1 1^{k-2} 0 a_2 b_2 1^{k-2} 0 \ldots a_n b_n 1^{k-2} 0$ and the pattern $y = 1^k$. Hence a lower bound of $s(n)$ for circuits computing disjointness implies a lower bound of $\Omega(s(n/k))$ for circuits computing $\mathrm{SM}_{n,k}$.

Theorem 4.6. *For every $1 < k \leq n$,*

1. $\mathrm{OR} \circ \mathrm{LTF}(\mathrm{SM}_{n,k}) \geq \Omega(n-k)$;
2. $\mathrm{AND} \circ \mathrm{LTF}(\mathrm{SM}_{n,k}) \geq 2^{\Omega(n/k)}$;
3. $\mathrm{AND} \circ \mathrm{OR} \circ \mathrm{XOR}(\mathrm{SM}_{n,k}) \geq 2^{\Omega(n/k)}$;
4. $\mathrm{ELTF} \circ \mathrm{SYM}(\mathrm{SM}_{n,k}) \geq 2^{\Omega(n/k)}$;
5. $\mathrm{EMAJ} \circ \mathrm{ELTF}(\mathrm{SM}_{n,k}) \geq 2^{\Omega(n/k)}$.

Proof. 1. We will prove that even for the fixed pattern $y = 1^k$, the number of LTF gates in any $\mathrm{OR} \circ \mathrm{LTF}$ circuit computing $\mathrm{SM}_{n,k}$ must be at least $(n-k+1)/2$. Assume, for the sake of contradiction that there exist $t < (n-k+1)/2$ threshold gates $g_1, \ldots, g_t$ whose OR computes $\mathrm{SM}_{n,k}(x, 1^k)$. For $0 \le i \le n-k$, let $x_i = 0^i 1^k 0^{n-k-i}$ be a string of length n. Note that for every i, $\mathrm{SM}_{n,k}(x_i, 1^k) = 1$, therefore, there exists at least on gate g_j for $1 \le j \le t$ accepting it. Since there are $n-k+1$ strings x_i, and $t < (n-k+1)/2$ gates, at least one gate g_j must accept two non-consecutive x_i's. Without loss of generality assume that g_1 accepts (x_i, y) and (x_j, y) for $j > i+1$. Now let

$$x = 0^i 1 0^{j-i-1} 1^{k-1} 0^{n-k-j+1} ,$$
$$x' = 0^{i+1} 1^{k-1} 0^{j-i-1} 1 0^{n-k-j} .$$

For the fixed pattern $y = 1^k$, suppose g_1 computes the function $g_1(x) = \sum_{m=1}^{n} \alpha_m x[m] \ge \theta$ of the text x. From $g_1(x_i) = g_1(x_j) = 1$, we have that $\sum_{m=i+1}^{i+k} \alpha_m + \sum_{m=j+1}^{j+k} \alpha_m \ge 2\theta$. Now we apply the function g_1 to x and x':

$$g_1(x) + g_1(x') = \left(\alpha_{i+1} + \sum_{m=j+1}^{j+k-1} \alpha_m \right) + \left(\sum_{m=i+2}^{i+k} \alpha_m + \alpha_{j+k} \right) = \sum_{m=i+1}^{i+k} \alpha_m + \sum_{m=j+1}^{j+k} \alpha_m \ge 2\theta .$$

Therefore, at least one of the inputs x and x' is accepted by g_1 (and, therefore, by the $\mathrm{OR} \circ \mathrm{LTF}$ circuit). Note that x and x' each has k ones with a zero in between (since $j > i+1$). Therefore, neither x not x' can be accepted by a circuit computing $\mathrm{SM}_{n,k}(x, 1^k)$, which leads to a contradiction.

2. We will simulate an $\mathrm{OR} \circ \mathrm{LTF} = \neg\mathrm{AND} \circ \mathrm{LTF}$ circuit by a special type of private-coin *"small bounded-error"* protocol against which lower bounds are known for computing $\neg\mathrm{Disj}_m$ for $m := n/k$ [GW16] (and hence $\neg\mathrm{SM}_{n,k}$ by Observation 3.5). Suppose the top fan-in of an $\mathrm{OR} \circ \mathrm{LTF}$ circuit is t and its input $x \in \{0,1\}^m$ is bipartitioned between two players. In the simulation, Alice first uses her private coins to choose a uniform random $i \in [t]$ and sends it to Bob ($\log t$ bits). Then Alice and Bob together evaluate the i-th LTF gate to within error $\epsilon := 0.1/t$, which takes $O(\log(m/\epsilon)) = O(\log m + \log t)$ many bits of communication. This protocol is such that it accepts every 1-input of the circuit with probability at least $\alpha := 1/t \cdot (1-\varepsilon)$ (at least one LTF evaluates to 1); and every 0-input it accepts with probability at most $\epsilon \le \alpha/2$. It is known that every such protocol (with a constant-factor acceptance gap, α vs. $\alpha/2$) for $\neg\mathrm{Disj}_m$ (and hence for $\neg\mathrm{SM}_{n,k}$) needs $\Omega(m)$ bits of communication [GW16, Thm 1.3]. This shows a lower bound of $t \ge 2^{\Omega(m)} = 2^{\Omega(n/k)}$ for the size of any $\mathrm{OR} \circ \mathrm{LTF}$ circuit for $\neg\mathrm{SM}_{n,k}$, or equivalently, any $\mathrm{AND} \circ \mathrm{LTF}$ circuit for $\mathrm{SM}_{n,k}$.

3. The above proof works with the layer of LTF gates replaced by $\mathrm{OR} \circ \mathrm{XOR}$ gates: both types of gates admit an $O(\log(n/\epsilon))$-bit ϵ-error protocol (evaluating $\mathrm{OR} \circ \mathrm{XOR}$ involves computing the *equality* function).

4. This bound follows from the reduction from Disj and Theorem 15 in Hansen and Podolskii [HP10].

5. Follows from the work of Razborov and Sherstov [RS10], and the closedness of $\mathrm{EMAJ} \circ \mathrm{ELTF}$ under AND (see Theorem 20 in [HP10]).

□

5 DeMorgan Circuits

In this section we prove Theorem 1.4 and Theorem 1.5.

Theorem 1.4 (Depth-2 DeMorgan circuits)**.** *For the* $\mathrm{SM}_{n,k}(x,y)$ *problem:*

- **Depth-2 upper bound:** *There is a depth-2 DeMorgan circuit of size* $O(n \cdot 2^k)$*.*
- **Depth-2 lower bound:** *Any depth-2 DeMorgan circuit must be of size*

$$\begin{array}{ll} \Omega(n \cdot 2^k) & \text{if } 1 < k \leq \sqrt{n}\ ; \\ \Omega(2^{2\sqrt{n-k+1}}) & \text{if } k \geq \sqrt{n}. \end{array}$$

Theorem 1.5 (General DeMorgan circuits)**.** *For the* $\mathrm{SM}_{n,k}(x,y)$ *problem:*

- **Upper bound:** *There is a DeMorgan circuit of size* $O(nk)$ *and depth* 3*.*
- **Lower bound:** *Any DeMorgan circuit must be of size at least* $n/2$*.*

In Section 5.1 we give upper bounds for both theorems, in Section 5.2 we prove lower bounds for depth-2 circuits, and in Section 5.3 we provide a lower bound for the unbounded depth case.

5.1 Upper Bounds

We first give a DNF with $2^k(n-k+1)$ clauses computing $\mathrm{SM}_{n,k}$, and in Lemma 5.3 we will prove that this DNF is essentially optimal.

Lemma 5.1. *For any* $k \leq n$ *there exists a DeMorgan circuit of depth 2 and size* $(n-k+1) \cdot 2^k + 1$ *computing* $\mathrm{SM}_{n,k}$*.*

Proof. First we note that equality of two k-bit strings can be implemented using a DNF of width $2k$ and size (number of clauses) 2^k. Indeed, denoting the two inputs by $z = (z_1, \ldots, z_k)$ and $w = w_1, \ldots, w_k$, let

$$\mathrm{EQ}(z_1, \ldots, z_k; w_1, \ldots, w_k) = \bigvee_{a=(a_1,\ldots,a_k)\in\{0,1\}^k} \left(\wedge_{i=1}^k (z_i = a_i) \wedge_{i=1}^k (w_i = a_i) \right) ,$$

where $(w_i = a_i)$ is equal to w_i if $a_i = 1$, and $\neg w_i$ otherwise.

For each $i = 1, \ldots, n-k+1$ let EQ_i be the DNF that outputs 1 if and only if $y = (x_i, \ldots, x_{i+k-1})$. Taking $\bigvee_{i=1}^{n-k+1} \mathrm{EQ}_i$ we obtain a circuit of depth-3 that computes the $\mathrm{SM}_{n,k}$ function. In order to turn it into a depth-2 circuit, note that the second and the third layers consist of $\vee$ gates, and hence can be collapsed to one layer. This way we get a depth-2 circuit of size $(n-k+1) \cdot 2^k + 1$. □

Now we show that already in depth 3, one can compute $\mathrm{SM}_{n,k}$ by a much smaller circuit. This Lemma is likely to have been discovered multiple times, we attribute it to folklore.

Lemma 5.2. *There exists a DeMorgan circuit of depth 3 and size* $O(nk)$ *computing* $\mathrm{SM}_{n,k}$*.*

Proof. First we note that the equality function of two k-bit strings can be implemented using a CNF of width 2 and size (number of clauses) $2k$. Indeed, we can check equality of two bits z and w using the circuit $(z \vee \neg w) \wedge (\neg z \vee w)$. Therefore, we can implement equality of two k-bits strings using the CNF formula

$$\mathrm{EQ}(z_1, \ldots, z_k; w_1, \ldots, w_k) = \bigwedge_{i=1}^{k} ((z_i \vee \neg w_i) \wedge (\neg z_i \vee w_i)) \ .$$

From here on we can proceed as in the previous proof, namely, for each $i = 1, \ldots, n-k+1$ let EQ_i be the CNF that outputs 1 is and only if $y = (x_i, \ldots, x_{i+k-1})$. Taking $\bigvee_{i=1}^{n-k+1} \mathrm{EQ}_i$ we obtain a circuit of depth 3 that computes the $\mathrm{SM}_{n,k}$ function. The output gate has fanin $n-k+1$, the gates in the second layer have fan-in $2k$, and the gates in the first layer have fan-in 2. Therefore, the total size of the circuit is $O(nk)$. □

5.2 Lower bounds for depth 2

We may assume wlog that every optimal circuit of depth 2 is either a CNF or a DNF. First, we show that in the class of DNFs, the construction from Lemma 5.1 is optimal (up to a constant factor).

Lemma 5.3.

For every $k > 2$, the DNF*-size of* $\mathrm{SM}_{n,k}$ *is at least*

$$\mathrm{DNF}(\mathrm{SM}_{n,k}) \geq 2^{k-2}(n-k+1) \ .$$

Proof. Let us consider the set P of 2^{k-2} patterns of length k which all start and end with a 1:

$$P = \{1p1 \colon p \in \{0,1\}^{k-2}\} \ .$$

For any pattern $p \in P$ and integer $0 \leq i \leq n-k$, let $s_{p,i} = 0^i p 0^{n-k-i}$ be the string containing p at the $i+1$th position and having zeros everywhere else. Now let the set S be the set of inputs to the $\mathrm{SM}_{n,k}$ problem (that is a set of pairs of a text and pattern) consisting of all $p \in P$ and the corresponding $s_{p,i}$'s:

$$S = \{(s_{p,i}, p) \colon p \in P, 0 \leq i \leq n-k\} \ .$$

Consider a DNF ϕ computing $\mathrm{SM}_{n,k}$. In order to show that it has at least $2^{k-2}(n-k+1)$ clauses, we will show that no pair of distinct inputs from S can be accepted by the same clause. Indeed, since every input from S must be accepted by ϕ and $|S| = 2^{k-2}(n-k+1)$, we get the corresponding lower bound on the number of clauses in S.

Assume, for the sake of contradiction that $(s_{p_1,i_1}, p_1) \neq (s_{p_2,i_2}, p_2)$ are accepted by the same clause C. Consider the following two cases.

Case 1: $i_1 = i_2, p_1 \neq p_2$. Since $p_1 \neq p_2$, there exists an index $2 \leq j \leq k-1$ such that $p_1[j] \neq p_2[j]$. The clause C cannot depend on the (i_1+j)th character of the text, because if it depended on it it wouldn't accept one of these strings. Let us consider the string s which differs from s_{p_1,i_1} only in the character number $i_1 + j$. Then the input (s, p_1) must still be accepted by ϕ. This contradicts the definition of $\mathrm{SM}_{n,k}$ because the string s contains exactly one string of length k staring and ending with a 1, and that string differs from p_1 in one character.

Case 2: $i_1 \neq i_2$. Wlog assume that $i_1 < i_2$. Then the strings s_{p_1,i_1} and s_{p_2,i_2} differ in the character number $j = i_1 + 1$. (Indeed, by the definition of p_1, s_{p_1,i_1} has a 1 in the jth position, while s_{p_2,i_2} has a 0 since $i_2 > i_1$.) Again, this implies that C does not depend on the jth character of the text. Let us now consider the string s which differs from s_{p_1,i_1} only at the character number j. Then the input (s, p_1) is accepted by ϕ which leads to a contradiction.

□

Now we will prove lower bounds for CNFs computing $\mathrm{SM}_{n,k}$. We will need the following definition.

Definition 5.4. A *maxterm* of a Boolean function f is a set of variables of f, such that some assignment to those variables makes f output 0 irrespective of the assignment to the other variables. The *width* of a maxterm is the number of variables in it.

First we find the minimal width of maxterms of $\mathrm{SM}_{n,k}$.

Lemma 5.5. *For any $k \leq n$, every maxterm of $\mathrm{SM}_{n,k}$ has width at least*

$$\begin{array}{ll} 2\sqrt{n-k+1} & \text{for all } k\ ; \\ k + \frac{n-k+1}{k} & \text{if } k \leq \sqrt{n-k+1}. \end{array}$$

Proof. Consider a substitution ρ which fixes n_1 variables in the text and k_1 variables in the pattern. In order to force $\mathrm{SM}_{n,k}$ to output 0, for every shift $1 \leq i \leq n-k+1$ there must be an index $1 \leq j \leq k$ such that ρ assigns a value to y_j and x_{i+j}. Thus, each of the n_1 assigned variables in the text "covers" at most k_1 shifts. Since the total number of shifts is $n-k+1$, we have that $n_1 \cdot k_1 \geq n-k+1$. Therefore, by the the inequality of arithmetic and geometric means, $n_1 + k_1 \geq 2\sqrt{n_1 \cdot k_1} \geq 2\sqrt{n-k+1}$.

Since $n_1 \cdot k_1 \geq n-k+1$, we have that $n_1 + k_1 \geq \frac{n-k+1}{k_1} + k_1$. The second bound follows by noting that the function $f(k_1) = \frac{n-k+1}{k_1} + k_1$ is monotone decreasing for $k_1 < \sqrt{n-k+1}$.

□

Next we prove tight bounds on the number of non-satisfying inputs of $\mathrm{SM}_{n,k}$.

Lemma 5.6. *For $k \leq n$, let Z denote the set of preimages of 0 of $\mathrm{SM}_{n,k}$. That is,*

$$Z = \{(x, y) \in \{0,1\}^{n+k} : \mathrm{SM}_{n,k}(x; y) = 0\}.$$

Then

$$\begin{array}{ll} |Z| = \Theta\left(2^{n+k}\right) & \text{if } k \geq \log n + 1; \\ |Z| \geq \Omega\left(2^n(1-2^{-k})^n\right) & \text{for all } k. \end{array}$$

Proof. Let O denote the set of preimages of 1 of $\mathrm{SM}_{n,k}$. Since every string of length n contains at most $n-k+1$ different substrings of length k, we have that $|O| \leq n \cdot 2^n$. Now for $k \geq \log n + 1$, from the equation $|Z| + |O| = 2^{n+k}$, we have that $|Z| \geq \Omega(2^{n+k})$.

In order to prove the lower bound $|Z| = \Omega\left(2^n(1-2^{-k})^n\right)$, we consider the pattern string $y_0 = 0^k$. The number F_n of strings x of length n which do not contain y_0 satisfies the generalized Fibonacci recurrence:

$$F_n = \sum_{i=1}^{k} F_{n-i}\ .$$

From the known bounds on the generalized Fibonacci numbers (see, e.g., Lemma 3.6 in [Wol98]), we have $F_n \geq \Omega(2^n(1-2^{-k})^n)$, which implies the corresponding lower bound on $|Z|$.

□

Lemma 5.7. *For every k, the* CNF*-size of* $\mathrm{SM}_{n,k}$ *is at least*

$$\begin{aligned}
&\mathrm{CNF}(\mathrm{SM}_{n,k}) \geq \Omega\left(2^{\frac{n}{10k}}\right) && \text{if } 1 < k \leq \log n + 1;\\
&\mathrm{CNF}(\mathrm{SM}_{n,k}) \geq \Omega\left(2^{k+n/k}\right) && \text{if } \log n + 1 \leq k \leq \sqrt{n};\\
&\mathrm{CNF}(\mathrm{SM}_{n,k}) \geq \Omega\left(2^{2\sqrt{n-k+1}}\right) && \text{if } k \geq \sqrt{n}.
\end{aligned}$$

Proof. We say that a clause of a CNF *covers an input* $w \in \{0,1\}^{n+k}$ if this clause evaluates to 0 on w. Note that a clause of width c covers at most 2^{n+k-c} elements in $\{0,1\}^{n+k}$. For the parameters $k \leq n$ we claim first that every clause of a CNF computing $\mathrm{SM}_{n,k}$ must be of width at least $c = c(k,n)$ depending on the range of k (as follows from Lemma 5.5). This implies that the number of clauses in any CNF computing $\mathrm{SM}_{n,k}$ is at least $|Z|/2^{n+k-c}$. Below, we use Lemma 5.6 and Lemma 5.5 to estimate c and $|Z|$ for different ranges of k.

If $k \leq \log n + 1$, then $|Z| \geq \Omega\left(2^n(1-2^{-k})^n\right)$ by Lemma 5.6. By Lemma 5.5, the width of each maxterm is at least $c \geq k + \frac{n-k+1}{k}$. Thus, the number of clauses in any CNF computing $\mathrm{SM}_{n,k}$ must be at least

$$\Omega\left(|Z|/2^{n+k-c}\right) \geq \Omega\left(|Z|/2^{n-n/k}\right) \geq \Omega\left(2^{n/k}(1-2^{-k})^n\right) \geq \Omega\left(2^{n/10k}\right),$$

where the last bound follows from the inequality $2^{1/k} \cdot (1-2^{-k}) \geq 2^{1/10k}$ which holds for all $k \geq 2$.

For $k \geq \log n + 1$, Lemma 5.6 gives us an $\Omega(2^{n+k})$ lower bound on $|Z|$. Lemma 5.5 provides a lower bound on the width c of maxterms: for $\log n + 1 \leq k \leq \sqrt{n}$, $c \geq k + n/k - 1$, and for $k \geq \sqrt{n}$, $c \geq 2\sqrt{n-k+1}$. The desired bounds on the number of clauses in any CNF computing $\mathrm{SM}_{n,k}$ now follow immediately. □

Discussion. Lemma 5.3 and Lemma 5.7 together give the lower bounds of Theorem 1.4. We observe a curious behavior of CNFs and DNFs for $\mathrm{SM}_{n,k}$. For $k \leq \sqrt{n}$, an optimal depth-2 circuit for $\mathrm{SM}_{n,k}$ is a DNF. It can also be shown that for $k \geq n - O(\frac{n}{\log n})$, an optimal circuit is a CNF. (Indeed, in order to certify that $\mathrm{SM}_{n,k}(x,y) = 0$, it suffices to give mismatches for each of the $(n-k+1)$ shifts of the pattern y in x. This amounts to $k^{O(n-k+1)} < n \cdot 2^k$ clauses.) We leave the exact CNF complexity of $\mathrm{SM}_{n,k}$ for the regime $k > \sqrt{n}$ as an open problem. One way to prove a stronger lower bound in this regime would be to give a lower bound on the width of every maxterm. This approach does not lead to stronger lower bounds because there exist maxterms of width $2\sqrt{n}$. To see this, consider an assignment where the first $\sqrt{n}$ characters of the pattern y are fixed to zeros, and all indices divisible by $\sqrt{n}$ in the text x are fixed to ones. While we cannot prove a stronger lower bound on the width of "most" maxterms, we know that some maxterms must have width at least $n-k+1$. Indeed, consider the text $x = 0^n$ and pattern $y = 10^{k-1}$. Every clause which outputs 0 on this pair, must assign the first $(n-k+1)$ positions of x to 0.

We remark that weaker lower bounds of $2^{\Omega(\sqrt{n/k})}$ and $2^{0.08n/k}$ on the size of CNF computing $\mathrm{SM}_{n,k}$ follow from the reduction from Disjointness in Observation 3.5 and the known lower bound on the depth-3 complexity of Iterated Disjointness [HJP95] and Disjointness [Juk06].

5.3 Lower bound for unbounded depth

Now we prove the lower bound of Theorem 1.5. For circuits with fan-in 2, a linear lower bound follows from the observation that $\mathrm{SM}_{n,k}$ essentially depends on all of its inputs. In the next lemma, we use an extension of the gate elimination technique to show that even in the class of DeMorgan circuits with *unbounded fan-in*, $\mathrm{SM}_{n,k}$ still requires linear size.

Lemma 5.8. *For $k > 1$, any DeMorgan circuit computing $\mathrm{SM}_{n,k}$ has size at least $n/2$.*

Proof. Suppose that a circuit C computes $\mathrm{SM}_{n,k}$, and consider an input (x, y) to the circuit. We prove that C has at least $n/2$ gates as follows. We show that for *any* fixing of the bits $x_1, x_3, x_5, \ldots, x_{2t-1}$ for $1 \leq t \leq n/2 - 1$, the restricted function depends on the bit x_{2t+1}. Since the function depends on x_{2t+1}, any circuit computing it must have x_{2t+1} or $\neg x_{2t+1}$ among its inputs. Without loss of generality we assume that x_{2t+1} appears as an input. Now we show that we can fix the input x_{2t+1} so that at least one gate of the circuit is removed.

Indeed, if x_{2t+1} appears as an input to an AND gate, we can set $x_{2t+1} = 0$, hence setting the output of the gate to 0. This way we can remove the gate from the circuit by setting the output of the AND gate to 0, and propagating it. (It is possible that we also affect other gates). Similarly, if x_{2t+1} appears as an input to an OR gate, we can set $x_{2t+1} = 1$, hence setting the outputs of the OR gate to 1, and remove the gate from the circuit. Therefore, we can remove at least $n/2 - 1$ gates from the circuit, and hence the size of the original circuit computing $\mathrm{SM}_{n,k}$ was at least $n/2$.

Therefore, it is left to prove the following claim

Claim 5.9. *Let $k \geq 2$. For* any *fixing of the bits $x_1, x_3, x_5, \ldots, x_{2t-1}$ for $1 \leq t \leq n/2 - 1$, the restricted function depends on the bit x_{2t+1}.*

Proof. Let $x^* = (x_1^*, x_3^*, \ldots, x_{2t-1}^*)$ be the values of the t fixed bits of x. In order to show that the restricted function depends on x_{2t+1}, we show that there exist two inputs: (x, y) and (x', y), such that $0 = \mathrm{SM}_{n,k}(x, y) \neq \mathrm{SM}_{n,k}(x', y) = 1$ and (x, y) and (x', y) are extensions of x^* which differ only in the position $2t + 1 : x_{2t+1} \neq x'_{2t+1}$.

We set all non-fixed bits of x to 0, except for $x_{2t+2} = 1$. Now we set x' to be equal to x everywhere except for the position $2t + 1$, where $x'_{2t+1} = 1$. Now we see that the string x does not contain two ones in a row, while x' does. Since $k \geq 2$, we can set y to be an arbitrary substring of x' of length k which contains x'_{2t+1} and x'_{2t+2}. By the definition of y we have $\mathrm{SM}_{n,k}(x', y) = 1$ and $\mathrm{SM}_{n,k}(x, y) = 0$ because x does not contain the substring 11. □

□

6 Learning

6.1 VC dimension

In this section we prove Theorem 1.8.

Theorem 1.8. *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

$$\mathrm{VC}(\mathcal{H}_{k,\Sigma}) = \min(\log|\Sigma|(k - O(\log k)), \log n + O(\log\log n)) .$$

We begin by upper bounding the VC dimension. In the proof we will use the following folklore construction of a Sperner system.

Definition 6.1. A system $\mathcal{F}$ of subsets of $\{1, \ldots, n\}$ is called a *Sperner system* if no set in $\mathcal{F}$ contains another one:

$$\forall A, B \in \mathcal{F} \colon A \neq B \implies A \not\subseteq B \; .$$

For any n, there exists a Sperner system of size $\binom{n}{\lfloor n/2 \rfloor}$. Indeed, one can take $\mathcal{F}$ to be the family of all sets of size exactly $\lfloor n/2 \rfloor$.

Lemma 6.2. *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

$$\mathrm{VC}(\mathcal{H}_{k,\Sigma}) \leq \min(\lceil k \log |\Sigma| \rceil, \log n + 0.5 \log \log n + 2) \; .$$

Proof. Since $|\mathcal{H}_{k,\Sigma}| = \frac{|\Sigma|^{k+1}-1}{|\Sigma|-1} < 2|\Sigma|^k$, $\mathcal{H}_{k,\Sigma}$ cannot shatter a set of strings S of size $|S| \geq k \log |\Sigma| + 1$. Hence, $\mathrm{VC}(\mathcal{H}_{k,\Sigma}) \leq \lceil k \log |\Sigma| \rceil$. We now give a different upper bound on $\mathrm{VC}(\mathcal{H}_{k,\Sigma})$.

Suppose one can shatter some d strings $X = \{x_1, \ldots, x_d\}$, where $x_i \in \Sigma^n$. That is, for any dichotomy of the strings from X, there is a pattern from $\Sigma^{\leq k}$ which realizes it. We will show an upper bound on $d = \mathrm{VC}(\mathcal{H}_{k,\Sigma})$.

Consider a Sperner system of size $D = \binom{d-1}{\lfloor (d-1)/2 \rfloor}$ of the set $\{1, \ldots, d-1\}$. Now add the element d to each of these sets. This way we have D sets containing the element d, such that none of them is a subset of another. Let us denote this family of D sets by $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_D\}$. Consider the following set $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_D\}$ of D dichotomies of X: $\mathcal{D}_i$ labels x_j with one if and only if $j \in \mathcal{S}_i$.

From the assumption that X can be shattered, we have that there exist D patterns $p_1, \ldots, p_D$ which realize all D dichotomies from $\mathcal{D}$. Since each of these dichotomies labels x_d with one, the string x_d must contain all patterns p_i. If one of the patterns p_i was a substring of another pattern p_j, then we would have that $\mathcal{S}_i \subseteq \mathcal{S}_j$, which contradicts the definition of Sperner systems.

Thus, there must be D patterns which are contained in the string x_d, and none of these patterns is a substring of another one. Let us sort the occurrences of these D patterns in x_d by their starting position. Since one pattern cannot be a substring of another one, their ending positions must form an increasing sequence. Therefore, the length n of x_d is at least D. This gives us that

$$\frac{2^{d-1}}{\sqrt{2(d-1)}} \leq \binom{d-1}{\lfloor (d-1)/2 \rfloor} \leq D \leq n \; ,$$

or, $\mathrm{VC}(\mathcal{H}_{k,\Sigma}) \leq \log n + 0.5 \log \log n + 2$. □

To lower bound the VC dimension of $\mathcal{H}_{k,\Sigma}$ we need the following lemma.

Lemma 6.3. *Let m be an integer $m \geq 1$, and Σ be an alphabet of size $|\Sigma| \geq 2$. There exists a set T_m of at least $|\Sigma|^{m-1}$ strings from $\Sigma^{m+\lceil \log m \rceil + 2}$ with the following property. For any two distinct strings $\tau_1, \tau_2 \in T_m$, their concatenation $\tau = \tau_1 \circ \tau_2$ doesn't contain any string from $T_m \setminus \{\tau_1, \tau_2\}$ as a substring.*

Proof. Since $|\Sigma| \geq 2$, we can fix two distinct characters $0, 1 \in \Sigma$. Let S_m be the set of all strings from Σ^m which don't contain $0^{\lceil \log m \rceil + 1}$ as a substring. Note that each string containing $0^{\lceil \log m \rceil + 1}$ as a substring is uniquely defined by the starting position of $0^{\lceil \log m \rceil + 1}$ and $m - \lceil \log m \rceil - 1$ remaining characters. Therefore, the number of strings containing $0^{\lceil \log m \rceil + 1}$ doesn't exceed $m|\Sigma|^{m - \lceil \log m \rceil - 1}$, and $|S_m| \geq |\Sigma|^m - m|\Sigma|^{m-\lceil \log m \rceil - 1} \geq |\Sigma|^m - |\Sigma|^{m-1} \geq |\Sigma|^{m-1}$.

For each string $s \in S_m$, we include in T_m the string s appended with the string $0^{\lceil \log m \rceil + 1}1$ ($\lceil \log m \rceil + 1$ zeros followed by a one) in the end. Note that the number of strings in T_m is at least

$|\Sigma|^{m-1}$, and each string in this set is of length $m + \lceil \log m \rceil + 2$. Now we'll prove that for any $\tau_1, \tau_2 \in T_m$, $\tau = \tau_1 \circ \tau_2$ doesn't contain any string from $T_m \setminus \{\tau_1, \tau_2\}$.

Assume, towards contradiction, that τ contains a string $\tau_3 \in T_m \setminus \{\tau_1, \tau_2\}$. Recall that $\tau_1 = s_1 \circ 0^{\lceil \log m \rceil + 1}1, \tau_2 = s_2 \circ 0^{\lceil \log m \rceil + 1}1, \tau_3 = s_3 \circ 0^{\lceil \log m \rceil + 1}1$, where s_1, s_2, and s_3 are distinct strings from S_m. Thus,

$$\tau = s_1 \circ 0^{\lceil \log m \rceil + 1}1 \circ s_2 \circ 0^{\lceil \log m \rceil + 1}1 \ .$$

Since τ_3 ends with $0^{\lceil \log m \rceil + 1}1$ and neither $s_1 \circ 0^{\lceil \log m \rceil + 1}$ nor $s_2 \circ 0^{\lceil \log m \rceil + 1}$ contains this substring, τ_3 must be equal to τ_1 or τ_2. □

Lemma 6.4. *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

$$\mathrm{VC}(\mathcal{H}_{k,\Sigma}) \geq \min((k - \log k - 5) \log |\Sigma|, \log n - \log \log n) \ .$$

Proof. We will show that there exist d strings of length n: $x_0, \ldots, x_{d-1} \in \Sigma^n$, and 2^d patterns $p_0, \ldots, p_{2^d-1} \in \Sigma^{\leq k}$ of length at most k, such that each dichotomy of $\{x_0, \ldots, x_{d-1}\}$ is realized by some pattern p_i. This will prove that the VC dimension of $\mathcal{H}_{k,\Sigma}$ is at least d.

Let

$$m = \left\lfloor \min\left(k - \log k - 3, \frac{\log n}{\log |\Sigma|} - \frac{\log \log n}{\log |\Sigma|} + 1\right)\right\rfloor \ ,$$

and let $d = \lfloor (m-1) \log |\Sigma| \rfloor$. If $m < 1$, then the Lemma statement follows trivially, hence, assume that $m \geq 1$. We will show that $\mathcal{H}_{k,\Sigma}$ shatters d strings, and this will finish the proof.

Let T_m be the set of strings from Lemma 6.3 for the chosen value of m. Since the size $|T_m| \geq |\Sigma|^{m-1} \geq 2^d$, we can choose 2^d patterns from T_m. Let us call these patterns $p_0 \ldots p_{2^d-1}$. For $0 \leq i \leq d-1$, we define x_i to be the concatenation (in arbitrary order) of all strings p_j such that the ith bit of the binary expansion of j is 1. If the length of x_i is less than n, we pad it with ones in the end.

1. The length of each pattern p_i is

$$m + \lceil \log m \rceil + 2 \leq m + \log m + 3 \leq k - \log k - 3 + \log k + 3 = k \ .$$

2. Each string x_i can be padded to a string of length n, because it is a concatenation of 2^{d-1} patterns of total length

$$2^{d-1}(m + \lceil \log m \rceil + 2) \leq 2^{(m-1)\log|\Sigma| - 1}(m + \log m + 3) \leq 2^{\log n - \log\log n - 1}(\log n + 4) \leq n \ .$$

3. Consider now a subset $I \subseteq [d-1]$ of the strings $x_0, \ldots, x_{d-1}$ to be shattered. Let $0 \leq j \leq 2^d - 1$ be the number whose binary expansion is the indicator vector of I. We claim that the pattern p_j realizes the set I. First, by the definition of the strings x_i, the pattern p_j was among the patterns concatenated in x_i if and only if $i \in I$. Second, by Lemma 6.3, no x_i with $i \notin I$ contains p_j as a substring.

□

This concludes the proof of Theorem 1.8.

6.2 Learning $\mathcal{H}_{k,\Sigma}$

In this section we discuss an efficient algorithm for learning the hypothesis class $\mathcal{H}_{k,\Sigma}$. For completeness we state the definition of PAC learning:

Let $\mathcal{D}$ be a distribution over Σ^n. Suppose we are trying to learn h_σ for $\sigma \in \Sigma^{\leq k}$. Given $\tau \in \Sigma^{\leq k}$, the loss of h_τ with respect to h_σ is defined as

$$L_{\mathcal{D},\sigma}(\tau) = \mathrm{P}_{x\sim D}[h_\tau(x) \neq h_\sigma(x)] \ .$$

Following the notion of PAC-learning [Val84, SSBD14], we can now define what we mean by learning $\mathcal{H}_{k,\Sigma}$.

Definition 6.5. An algorithm $\mathcal{A}$ is said to PAC-learn $\mathcal{H}_{k,\Sigma}$ if for every distribution $\mathcal{D}$ over Σ^n and every $h_\sigma \in \mathcal{H}_{k,\Sigma}$ for all $\epsilon, \delta \in (0, 1/2)$ the following holds. Given $m := m(\epsilon, \delta, n, k)$ i.i.d. samples $(x_1, h_\sigma(x_1)), \ldots, (x_m, h_\sigma(x_m))$ where each x_i is sampled according to the distribution $\mathcal{D}$, $\mathcal{A}$ returns with probability at least $1 - \delta$ a function $h_\tau \in \mathcal{H}_{k,\Sigma}$ such that $L_{\mathcal{D},\sigma}(\tau) \leq \epsilon$. Here the probability is taken with respect to the m i.i.d. samples as well as the possible random choices made by the algorithm $\mathcal{A}$.

Throughout, we refer to δ as the confidence parameter and ϵ as the accuracy parameter.

In Definition 6.5 we consider the *realizable* case. Namely there exists $h_\sigma \in \mathcal{H}_{k,\Sigma}$ that we want to learn. One can also consider the *agnostic* case. Consider a distribution $\mathcal{D}$ over $\Sigma^n \times \{0, 1\}$. We now define the loss of h_τ as

$$L_{\mathcal{D}}(\tau) = \mathrm{P}_{x\sim D}[h_\tau(x) \neq y] \ ,$$

namely the measure under $\mathcal{D}$ of all pairs $(x, y) \in \Sigma^n \times \{0, 1\}$ with $h_\tau(x) \neq y$ [SSBD14]. In the agnostic case we wish to find, given m i.i.d. samples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$, a pattern $\sigma' \in \Sigma^{\leq k}$ such that $L_{\mathcal{D}}(\sigma') \leq \min_\tau L_{\mathcal{D}}(\tau) + \epsilon$ (where the minimum is taken over all $\tau \in \Sigma^{\leq k}$). Thus agnostically PAC-learning generalizes the realizable case where $\min_\tau L_{\mathcal{D}}(\tau) = 0$.

Recall that a function $h_\sigma \in \mathcal{H}_{k,\Sigma}$ (parameterized by the pattern σ of length at most k) can be learned with error ϵ and confidence δ by considering $m = O(\mathrm{VC}(\mathcal{H}_{k,\Sigma}))$ samples $(x_1, h_\sigma(x_1)), \ldots, (x_m, h_\sigma(x_m))$ (where the constant in the O term depends on ϵ, δ) and following the ERM (expected risk minimization) rule: Finding σ' that minimizes the loss

$$L(h_{\sigma'}) := \frac{|\{i \in [m] : h_{\sigma'}(x_i) \neq h_\sigma(x_i)\}|}{m} \ .$$

In words, to PAC learn h_σ we simply look for a string σ' of length at most k such that the fraction of sample points that are misclassified by $h_{\sigma'}$ is minimized (the ERM rule applies both for the agnostic and realizable settings).

By Lemma 6.2, the number of samples needed to PAC-learn h_σ is at most $O(\log n)$ (ignoring the dependency on ϵ, δ). Clearly we can implement the ERM by considering all possible substrings of length at most k that occur in the $m = O(\log n)$ strings $x_1 \ldots x_m$ and finding the substring σ' minimizing $L(h_{\sigma'})$. The number of such substrings is at most $O(\log n \sum_{i=1}^{k}(n-k+1)) \leq O(kn \log n)$. Since for every substring we can check whether it occurs in a string of length n in time $O(n)$, we can implement the ERM rule by going over every substring η of length at most k and checking for every string x_i (with $i \in [m]$) whether η occurs in x_i. By keeping track of the pattern which has minimal classification error with respect to the sample $(x_1, h_\sigma(x_1)), \ldots, (x_m, h_\sigma(x_m))$ we can thus implement the ERM rule in time $O(kn^2 \log^2 n)$.

We can do better if the number of substrings of length at most k which is upper bounded by $2|\Sigma|^k$ is smaller than $(n-k+1)\log n$. Suppose for example, that $k \leq \frac{\log n}{\log|\Sigma|}$. By Lemma 6.2, the VC-dimension of $\mathcal{H}_{k,\Sigma}$ is then upper bounded by $k\log|\Sigma|$. Hence in this case we can assume the number of strings m in our sample is at most $k\log|\Sigma|$, and we can implement the ERM rule in time $O(|\Sigma|^k kn\log|\Sigma|)$. When $k, |\Sigma|$ are constants independent of n we can thus learn h_σ in time $O(n)$.

We summarize this discussion with the following corollary:

Corollary 6.6. *The hypothesis class $\mathcal{H}_{k,\Sigma}$ is PAC-learnable in time $O(kn^2\log^2 n)$, where the O symbol contains constants depending on ϵ, δ but not on n, k. If $k, |\Sigma|$ are constants independent of n, then $\mathcal{H}_{k,\Sigma}$ can be learned in time $O(n)$.*

6.3 Extensions

Infinite alphabet. So far we have been considering the case of finite alphabet Σ. For an infinite Σ the VC dimension is essentially $\log n$ for every value of $k \geq 1$. Note that the upper bound of $\mathrm{VC}(\mathcal{H}_{k,\Sigma}) \leq \log n + 0.5\log\log n + 2$ from Lemma 6.2 holds even for infinite alphabets Σ. Indeed, this upper bound counts the number of different patterns which have to occur in one string and compares it to the length of the string n. In the following lemma we give a lower bound of $\log n$ for all values of $k \geq 1$.

Lemma 6.7. *Let Σ be an infinite alphabet, and $k \geq 1$. Then*

$$\mathrm{VC}(\mathcal{H}_{k,\Sigma}) = (1+o(1))\log n\ .$$

Proof. For the lower bound, we pick 2^d+1 distinct elements $\perp, a_0, \ldots, a_{2^d-1} \in \Sigma$. Let $d = \lfloor \log n \rfloor + 1$. We construct d strings $X = \{x_1, \ldots, x_d\}, x_i \in \Sigma^n$ such that any dichotomy of X is realizable in $\Sigma^{\leq k}$. Now, for $0 \leq i \leq d-1$, we define x_i to be a concatenation (in arbitrary order) of all a_j such that the ith bit of the binary expansion of the number j is 1. Note that now the length of each x_i is at most $2^{d-1} \leq n$, so we pad each x_i with the element $\perp$ so that $x_i \in \Sigma^n$.

Now we need to show that each dichotomy of X is realizable. For a dichotomy $\mathcal{D}$ of X, consider the set $I \subseteq [d]$ such that $\mathcal{D}$ labels x_i with one if and only if $i \in I$. In order to realize $\mathcal{D}$, we take the pattern a_j such that the binary expansion of $0 \leq j \leq 2^d - 1$ equals the indicator vector of I. By the definition of x_i, x_i contains a_j if and only if $i \in I$. Therefore, a_j realizes the dichotomy $\mathcal{D}$. Note that the pattern $a_j \in \Sigma \subseteq \Sigma^{\leq k}$. □

Learning multiple patterns. In this section we make a few simple observations regarding the VC dimension of classifiers defined by the occurrences of multiple patterns. The main observation is that learning a *constant* number of patterns does not change the asymptotics of the VC dimension so long as the number of patterns is upper bounded by the length of the pattern k. Let us consider two natural classes $\mathcal{H}^{\mathrm{and}}_{k,\Sigma}$ and $\mathcal{H}^{\mathrm{or}}_{k,\Sigma}$ of multi-pattern Boolean functions over Σ^n. Each function $h^{\mathrm{and}}_\sigma \in \mathcal{H}^{\mathrm{and}}_{k,\Sigma}$ is parameterized by $c > 0$ patterns $\sigma = (\sigma_1, \ldots, \sigma_c) \in \left(\Sigma^{\leq k}\right)^c$. Now, for an $s \in \Sigma^n$, $h^{\mathrm{and}}_\sigma(s) = 1$ if and only if s contains *each* $\sigma_i, 1 \leq i \leq c$ as a substring (for brevity we omit from notation the dependence of $\mathcal{H}^{\mathrm{and}}_{k,\Sigma}$ and $\mathcal{H}^{\mathrm{or}}_{k,\Sigma}$ on c). Similarly, a function $h^{\mathrm{or}}_\sigma \in \mathcal{H}^{\mathrm{or}}_{k,\Sigma}$ takes the value one: $h^{\mathrm{or}}_\sigma(s) = 1$ if and only if s contains *at least one* σ_i as a substring. We stress that we assume that the set of patterns $\sigma_i, i \in [c]$ are distinct.

An upper bound on the VC dimension of $\mathcal{H}^{\mathrm{and}}_{k,\Sigma}$ and $\mathcal{H}^{\mathrm{or}}_{k,\Sigma}$ follows at once from the following Lemma proved in [BEHW89] (Lemma 3.2.3).

Lemma 6.8. *Let $\mathcal{H}_1, \ldots, \mathcal{H}_c$ be classes of functions of* VC *dimension at most $\forall i\colon \mathrm{VC}(\mathcal{H}_i) \le d$. Let*

$$\mathcal{H}^{and} = \{f_{h_1,\ldots,h_c}(x) = h_1(x) \wedge \ldots \wedge h_c(x)\colon h_1 \in \mathcal{H}_1, \ldots, h_c \in \mathcal{H}_c\} ,$$
$$\mathcal{H}^{or} = \{f_{h_1,\ldots,h_c}(x) = h_1(x) \vee \ldots \vee h_c(x)\colon h_1 \in \mathcal{H}_1, \ldots, h_c \in \mathcal{H}_c\} .$$

Then $\mathrm{VC}(\mathcal{H}^{and}) = O(dc \log c)$ *and* $\mathrm{VC}(\mathcal{H}^{or}) = O(dc \log c)$.

We now turn to the lower bound. Our result here is rather modest: We show that the lower bound on the VC dimension of a single pattern also holds for $\mathcal{H}^{\mathrm{and}}_{k,\Sigma}$ and $\mathcal{H}^{\mathrm{or}}_{k,\Sigma}$ provided that the number c of (distinct) patterns is not too large. Let us see that the lower bounds of Lemma 6.4 hold for $\mathcal{H}^{\mathrm{and}}_{k,\Sigma}$ and $\mathcal{H}^{\mathrm{or}}_{k,\Sigma}$. Indeed, for the class $\mathcal{H}^{\mathrm{and}}_{k,\Sigma}$, we use the construction from Lemma 6.4, where for every pattern σ in that construction we consider a set of k patterns $\{\sigma^1, \ldots, \sigma^k\}$. We define $\sigma^i = \sigma_1 \ldots \sigma_i$ to be the prefix of length i of σ. For example, for the pattern 11010 we take the patterns $\{1, 11, 110, 1101, 11010\}$. We remark that we obtain k distinct subpatterns of σ. Since every string from the shattered set contains σ if and only if it contains every pattern from $\{\sigma^1, \ldots, \sigma^k\}$, all dichotomies are realized by the "last" pattern $\sigma^k = \sigma$. Since $c \le k$, we take c longest patterns $\{\sigma^{k-c+1}, \ldots, \sigma^k\}$, and our construction gives a shattered set of size

$$\mathrm{VC}(\mathcal{H}^{\mathrm{and}}_{k,\Sigma}) \ge \min\left(\log|\Sigma|(k - O(\log k)), \log n + O(\log\log n)\right) .$$

For the class $\mathcal{H}^{\mathrm{or}}_{k,\Sigma}$, we can take $T'_m \subseteq T_m$ with $|T'_m| = |T_m|/2$ and shatter a set of size $d-1$. Now for every $\sigma \in T'_m$ define a c-tuple of patterns by adding to σ $c-1$ patterns in $T_m \setminus T'_m$ (where $c \le 2^{d-1} - 1$ because $c \le k$). Since none of the strings in the shattered set contains a pattern from $T_m \setminus T'_m$, all dichotomies are realized by the "first" pattern σ_1. Again, our construction from Lemma 6.4 gives a shattered set of size $\min\left(\log|\Sigma|(k - O(\log k)), \log n + O(\log\log n)\right) - 1$.

To conclude, we have proved:

Theorem 6.9. *Let $1 \le c \le k$ be a fixed constant. Then*

$$VC(\mathcal{H}^{and}_{k,\Sigma}), VC(\mathcal{H}^{or}_{k,\Sigma}) = \Theta\left(\min\left(\log|\Sigma|(k - O(\log k)), \log n + O(\log\log n)\right)\right) .$$

Patterns of length k. One can also consider learning patterns of length *exactly* k. We consider this case separately since it seems that getting tight bounds on VC-dimension in this case is a harder task. In particular, we are not able to get tight bounds for the regime $k = n^{1-o(1)}$ and leave this as an open question.

For a fixed *finite* alphabet Σ and an integer $k > 0$, the class of functions $\mathcal{E}_{k,\Sigma}$ over Σ^n is defined as follows. Every Boolean function $h_\sigma \in \mathcal{E}_{k,\Sigma}$ is parameterized by a pattern $\sigma \in \Sigma^k$ of length exactly k. Therefore, $|\mathcal{E}_{k,\Sigma}| = |\Sigma|^k$. For a string $s \in \Sigma^n$, $h_\sigma(s) = 1$ if and only if s contains σ as a substring. We use a simple double counting argument to prove:

Lemma 6.10. $\mathrm{VC}(\mathcal{E}_{k,\Sigma}) \le \min(k \log|\Sigma|, \log(n-k+1)+1)$.

Proof. Since $|\mathcal{E}_{k,\Sigma}| = |\Sigma|^k$, the upper bound of $\mathrm{VC}(\mathcal{E}_{k,\Sigma}) \le k\log|\Sigma|$ follows immediately. For the other upper bound, suppose we can shatter a set X of d strings $x_1 \ldots x_d$ of length n. Then we have 2^d distinct patterns which realize all dichotomies of X. For a fixed $i \in [d]$, the number of dichotomies of X which label x_i with one is 2^{d-1}. Therefore, every string x_i contains at least 2^{d-1} distinct patterns of length k. On the other hand, any string of length n can contain at most $n-k+1$ distinct patterns of length k. Thus, we have $2^{d-1} \le n-k+1$, or, $d \le \log(n-k+1)+1$. □

Now we prove the following lower bound:

Lemma 6.11. *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

$$\mathrm{VC}(\mathcal{E}_{k,\Sigma}) \geq \min((k - \log k - 5)\log|\Sigma|, \log n - \log k)\ .$$

Proof. Let

$$\begin{aligned} d &= \lfloor \min((k - \log k - 5)\log|\Sigma| + 1, \log n - \log k + 1) \rfloor\ , \\ m &= \lfloor k - \log k - 2 \rfloor\ . \end{aligned}$$

By Lemma 6.3, we have the set T_m of $|\Sigma|^{m-1} \geq 2^{(k-\log k-5)\log|\Sigma|} \geq 2^d$ strings of length k. We choose 2^d arbitrary strings $p_0, \ldots, p_{2^d-1}$ from T_m. Now we essentially use the construction from Lemma 6.4: we construct d strings $x_0, \ldots, x_{d-1} \in \Sigma^n$ such that x_i contains p_j if and only if the ith bit of the binary expansion of j is 1, and pad x_i with ones to have $x_i \in \Sigma^n$.

We have d strings which can be shattered by $\mathcal{E}_{k,\Sigma}$. We also know that the length of each pattern is k. We only need to show that before the padding step, each string x_i had length at most n. Since each x_i is a concatenation of 2^{d-1} patterns of length k, we have that its length is at most

$$2^{d-1} \cdot k \leq 2^{\log n - \log k} \cdot k = n\ .$$

□

We remark that for the case of patterns of length *at most* k, Lemma 6.2 and Lemma 6.4 give essentially tight bounds for all regimes of the parameters. Here, in the case of patterns of length *exactly* k, we have a gap between lower and upper bounds for the regime $k = n^{1-o(1)}$.

2D patterns. Our bounds for learning one dimensional strings generalize to the 2D case. Here we have an $n \times n$ image over an alphabet Σ and am $m \times m$ pattern σ where $m \leq k \leq n$. An image is classified as 1 if and only if it contains σ.

Definition 6.12. For a fixed *finite* alphabet Σ and an integer $k > 0$, let us define the class of Boolean functions $\mathcal{G}_{k,\Sigma}$ over $\Sigma^{n\times n}$ as follows. Every function $g_\sigma \in \mathcal{G}_{k,\Sigma}$ is parameterized by a square 2D pattern $\sigma \in \Sigma^{m\times m}$ of dimension $m \leq k$. For a 2D image $s \in \Sigma^{n\times n}$ of dimension n, $g_\sigma(s) = 1$ if and only if s contains σ as a consecutive sub-matrix (sub-image).

We give tight bounds (up to low order terms) on $\mathrm{VC}(\mathcal{G}_{k,\Sigma})$. Since the proofs are very similar to the 1D case, we only sketch the arguments here.

Since $|\mathcal{G}_{k,\Sigma}| = \sum_{1\leq i\leq k} |\Sigma|^{i^2}+1 \leq \sum_{1\leq i\leq k} |\Sigma|^{ik}+1 < 2|\Sigma|^{k^2}$, we have that $\mathrm{VC}(\mathcal{G}_{k,\Sigma}) \leq \lceil k^2 \log|\Sigma| \rceil$. Suppose that $\mathcal{G}_{k,\Sigma}$ shatters a set of d 2D images from $\Sigma^{n\times n}$. By considering a Sperner system over $\{1, \ldots, d-1\}$ of size $D = \binom{d-1}{\lfloor (d-1)/2 \rfloor}$ and adding the element d to each subset, we get a family of $D = \binom{d-1}{\lfloor (d-1)/2 \rfloor}$ patterns all lying in a single $n \times n$ image such that no pattern contains another one. We have that the bottom right corners of all these patterns are distinct, and thus $\frac{2^{d-1}}{\sqrt{2(d-1)}} \leq D \leq n^2$ implying that $d \leq 2\log n + 0.5 \log\log n + 3$. Hence,

$$\mathrm{VC}(\mathcal{G}_{k,\Sigma}) \leq \min(\lceil k^2 \log|\Sigma| \rceil, 2\log n + 0.5\log\log n + 3).$$

For the lower bound, the main observation is that we can generalize Lemma 6.3 to the two dimensional case having a set R_m of $(m+2\lceil\log m\rceil+2)\times(m+2\lceil\log m\rceil+2)$ 2D patterns of cardinality $|\Sigma|^{m^2-1}$ such that for any four distinct patterns $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ from R_m, their concatenation (fitting the four patterns into a $2(m+2\lceil\log m\rceil+2)\times 2(m+2\lceil\log m\rceil+2)$ square image in each of the 4! possible ways) does not contain any $\alpha_5\neq\alpha_i$ for $1\leq i\leq 4$ from R_m. We achieve this by taking all $m\times m$ templates not containing the all 0 2D square template of size $(2\lceil\log m\rceil+1)\times(2\lceil\log m\rceil+1)$, padding them by an all zero strip of width $2\lceil\log m\rceil+1$ on the right and bottom, and then adding a boundary of ones on those two sides. Similarly to Lemma 6.3, it can be verified that R_m satisfies the desired condition.

We now set

$$m=\left\lfloor \min\left(k-2\log k-4,\sqrt{\frac{2\log n}{\log|\Sigma|}-\frac{3\log\log n}{\log|\Sigma|}}\right)\right\rfloor .$$

Let R_m be a set of $|\Sigma|^{m^2-1}$ templates whose construction was described in the paragraph above and set $d=\lfloor(m^2-1)\log|\Sigma|\rfloor$. Since $|R_m|=|\Sigma|^{m^2-1}\geq 2^d$, we can choose 2^d distinct 2D patterns $q_0\dots q_{2^d-1}$ from R_m. The dimension of each pattern q_i is $m+2\lceil\log m\rceil+2$ which by the choice of m is at most k.

Define a set of $n\times n$ images $Y:=\{y_0\dots y_{d-1}\}$ where y_i is an image containing all the patterns q_j from R_m such that the binary expansion of j equals 1 in the ith location. This way, each image from Y must contain at most 2^{d-1} patterns, while we can fit $\left\lfloor\frac{n}{m+2\lceil\log m\rceil+2}\right\rfloor^2$ patterns into an image of size $n\times n$. It can be verified that for the chosen values of m and d, $2^{d-1}\leq\left\lfloor\frac{n}{m+2\lceil\log m\rceil+2}\right\rfloor^2$. Thus, we have that each y_i can be padded to an $n\times n$ image if necessary by assigning 1 to all unassigned positions. Finally, it follows in a similar fashion to the 1D case that the set of patterns $q_0\dots q_{2^d-1}$ shatters Y. Hence R_m shatters Y. Since $|Y|=d$ the VC dimension of the set of all 2D patterns of dimensions at most k is at least d.

We conclude this discussion with the following Theorem:

Theorem 6.13.

$$\mathrm{VC}(\mathcal{G}_{k,\Sigma})=\min\left((k-O(\log k))^2\log|\Sigma|,2\log n-O(\log\log n)\right) .$$

Acknowledgements

We thank Paweł Gawrychowski for his useful feedback and Gy. Turán for sharing [GT93] with us. We are also very grateful to anonymous reviewers for their insightful comments.

References

[AB09] Martin Anthony and Peter L. Bartlett. *Neural network learning: Theoretical foundations.* Cambridge University Press, 2009.

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[BEKR17] Omri Ben-Eliezer, Simon Korman, and Daniel Reichman. Deleting and testing forbidden patterns in multi-dimensional arrays. In *International Proceedings in Informatics*, volume 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[BJKK04] Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. The sketching complexity of pattern matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 261–272. Springer, 2004.

[BJKS04] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[BM77] Robert S. Boyer and J. Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

[Bra12] Mark Braverman. Interactive information complexity. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 505–524. ACM, 2012.

[BW16] Mark Braverman and Omri Weinstein. A discrepancy lower bound for information complexity. *Algorithmica*, 76(3):846–864, 2016.

[CMS19] Arkadev Chattopadhyay, Nikhil Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. In *Proceedings of the 51st Symposium on Theory of Computing*, 2019. To appear.

[CSWY01] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS)*, pages 270–278. IEEE, 2001.

[DSS16] Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning DNF's. In *Conference on Learning Theory*, pages 815–830, 2016.

[EHKV89] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261, 1989.

[FKL+01] Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 171–182. Springer, 2001.

[FKR+97] Yoav Freund, Michael Kearns, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. *Information and Computation*, 138(1):23–48, 1997.

[Gal85] Zvi Galil. Optimal parallel algorithms for string matching. *Information and Control*, 67(1-3):144–157, 1985.

[GS83] Zvi Galil and Joel Seiferas. Time-space-optimal string matching. *Journal of Computer and System Sciences*, 26(3):280–294, 1983.

[GT93] Hans Dietmar Groeger and György Turán. A linear lower bound for the size of threshold circuits. *Bulletin-European Association For Theoretical Computer Science*, 50:220–220, 1993.

[GW16] Mika Göös and Thomas Watson. Communication complexity of set-disjointness for all probabilities. *Theory of Computing*, 12(9):1–23, 2016.

[Han16] Steve Hanneke. The optimal sample complexity of PAC learning. *The Journal of Machine Learning Research*, 17(1):1319–1333, 2016.

[Hås87] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, 1987.

[HJP95] Johan Håstad, Stasys Jukna, and Pavel Pudlák. Top-down lower bounds for depth-three circuits. *Computational Complexity*, 5(2):99–112, 1995.

[HMP+93] 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.

[HP10] Kristoffer Arnsfelt Hansen and Vladimir V Podolskii. Exact threshold circuits. In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 270–279. IEEE, 2010.

[Juk06] Stasys Jukna. On graph complexity. *Combinatorics, Probability and Computing*, 15(6):855–876, 2006.

[Juk12] Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[KMP77] Donald E. Knuth, James H. Morris, Jr, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KR96] Eyal Kushilevitz and Dan Roth. On learning visual concepts and DNF formulae. *Machine Learning*, 24(1):65–85, 1996.

[KS92] Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[KW16] Daniel M. Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 633–643. ACM, 2016.

[LM01] Robert A. Legenstein and Wolfgang Maass. Foundations for a circuit complexity theory of sensory processing. *Advances in neural information processing systems*, pages 259–265, 2001.

[LM02] Robert A. Legenstein and Wolfgang Maass. Neural circuits for pattern recognition with small total wire length. *Theoretical Computer Science*, 287(1):239–249, 2002.

[LS62] R. C. Lyndon and M. P. Schützenberger. The equation $a^m = b^n c^p$ in a free group. *Michigan Mathematical Journal*, 9:289–298, 1962.

[LS09] Troy Lee and Adi Shraibman. *Lower Bounds in Communication Complexity*, volume 3. Now Publishers, 2009.

[MCPZ13] James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted Boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2877–2885, 2013.

[Mur71] Saburo Muroga. Threshold logic and its application. *Wily-Interscience*, 1971.

[Nis93] Noam Nisan. The communication complexity of threshold gates. *Combinatorics, Paul Erdos is Eighty*, 1:301–315, 1993.

[Par94] Ian Parberry. *Circuit complexity and neural networks*. MIT press, 1994.

[PP09] Benny Porat and Ely Porat. Exact and approximate pattern matching in the streaming model. In *Foundations of Computer Science, 2009. 50th Annual IEEE Symposium on*, pages 315–323. IEEE, 2009.

[PS88] Ian Parberry and Georg Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, 36(3):278–302, 1988.

[Raz92a] Alexander A. Razborov. On small depth threshold circuits. In *Scandinavian Workshop on Algorithm Theory*, pages 42–52. Springer, 1992.

[Raz92b] Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[Riv77] Ronald L. Rivest. On the worst-case behavior of string-searching algorithms. *SIAM Journal on Computing*, 6(4):669–674, 1977.

[ROS94] Vwani P. Roychowdhury, Alon Orlitsky, and Kai-Yeung Siu. Lower bounds on threshold and related circuits via communication complexity. *IEEE Transactions on Information Theory*, 40(2):467–474, 1994.

[Ros16] Christian Rosenke. The exact complexity of projective image matching. *Journal of Computer and System Sciences*, 82(8):1360–1387, 2016.

[RR97] Dana Ron and Ronitt Rubinfeld. Exactly learning automata of small cover time. *Machine Learning*, 27(1):69–96, 1997.

[RS10] Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of ac^0. *SIAM Journal on Computing*, 39(5):1833–1855, 2010.

[RSO94] Vwani Roychowdhury, Kai-Yeung Siu, and Alon Orlitsky. Neural models and spectral methods. In *Theoretical Advances in Neural Computation and Learning*, pages 3–36. Springer, 1994.

[SB91] Kai-Yeung Siu and Jehoshua Bruck. On the power of threshold circuits with small weights. *SIAM Journal on Discrete Mathematics*, 4(3):423–435, 1991.

[SBKH93] Kai-Yeung Siu, Jehoshua Bruck, Thomas Kailath, and Thomas Hofmeister. Depth efficient neural networks for division and related problems. *IEEE Transactions on information theory*, 39(3):946–956, 1993.

[Shv90] Haim Shvaytser. Learnable and nonlearnable visual concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):459–466, 1990.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[UYZ15] Kei Uchizawa, Daiki Yashima, and Xiao Zhou. Threshold circuits for global patterns in 2-dimensional maps. In *International Workshop on Algorithms and Computation*, pages 306–316. Springer, 2015.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Wat18] Thomas Watson. Communication complexity of statistical distance. *ACM Transactions on Computation Theory*, 10(1):2:1–2:11, 2018.

[Wol98] D. A. Wolfram. Solving generalized Fibonacci recurrences. *The Fibonacci Quarterly*, 36.2:129–145, 1998.

[Yan91] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.