String Matching: Communication, Circuits, and Learning

Alexander Golovnev
Harvard University, Cambridge, MA, USA
alex.golovnev@gmail.com

Mika Göös
Institute for Advanced Study, Princeton, NJ, USA
mika@ias.edu

Daniel Reichman
Department of Computer Science, Princeton University, NJ, USA
daniel.reichman@gmail.com

Igor Shinkar
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
ishinkar@sfu.ca

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.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Circuit complexity; Theory of computation → Boolean function learning

Keywords and phrases string matching, communication complexity, circuit complexity, PAC learning

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2019.56

Category RANDOM

Related Version All proofs are deferred to the full version of the paper available at `https://arxiv.org/abs/1709.02034`.

Funding *Alexander Golovnev*: supported by a Rabin Postdoctoral Fellowship.
Mika Göös: supported by the NSF grant No. CCF-1412958.
Igor Shinkar: supported by NSERC discovery grant.

Acknowledgements We thank Paweł Gawrychowski for his useful feedback and Gy. Turán for sharing [16] with us. We are also very grateful to anonymous reviewers for their insightful comments.

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 [7, 25, 15] (with some lower bounds given by [40]). It has also been studied in more modern algorithmic frameworks such as streaming [37], sketching [3], and property testing [5]. 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 [26, 22].) The only related prior work we are aware of is Bar-Yossef et al. [3] 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** (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*

  Deterministic: $O(\log k \cdot n/k)$ *if* $k \leq \sqrt{n}$ *;*
  Randomized: $O(\log n \cdot \sqrt{n})$ *if* $k \geq \sqrt{n}$*.*

- ***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 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 $x[1,k]$, $x[2,k+1],\ldots, x[n-k+1,n]$. 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 U·BPP) communication protocols.

A classic result [51] 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. [9]. 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 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 [50]. We observe that the lower bound follows by a minor modification of existing *information complexity* techniques [8]. 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 $\sum_{i\in[m]} a_i x_i \geq \theta$ 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 [17, 36, 35, 32, 33]. The case of *low-depth* threshold circuits is also interesting. In particular, one line of work [47, 38, 46] has focused on efficient low-depth threshold implementations of arithmetic primitives (addition, comparison, multiplication). As for lower bounds, [17] show an exponential-in-n lower bound for the *mod-2 inner-product* function against depth-2 threshold circuits of low weight (see [12] 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 [24].

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 [16], and proving them is considered challenging [43]. Indeed, Jukna [22] 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 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*

$\Omega(\frac{n \log\log k}{k \log n})$ *if* $k > 1$;
$\Omega(\sqrt{n/k})$ *if* $k \geq 2.1 \cdot \log n$.

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 $k \leq \text{polylog}(n)$, we have nearly linear lower bounds for unbounded-depth threshold circuits computing $\text{SM}_{n,k}$. 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 $\text{SM}_{n,k}$ using a connection between threshold complexity and circuit complexity outlined by [34]. We also prove lower bounds for threshold circuits by reducing the problem of computing a "sparse hard" function to computing $\text{SM}_{n,k}$. 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 $\text{SM}_{n,k}$. 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 4** (Depth-2 DeMorgan circuits). *For the* $\text{SM}_{n,k}(x, y)$ *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 [19] 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 $\text{SM}_{n,k}$. 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 $\text{SM}_{n,k}$ 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; [14] claims an upper bound $O(n \log^2 n)$ without a proof).

▶ **Theorem 5** (General DeMorgan circuits). *For the* $\text{SM}_{n,k}(x, y)$ *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 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 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 $\mathcal{F}$.*

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 [6, 11, 18], and is agnostic PAC-learnable in $\Theta\left(\frac{\mathrm{VC}(\mathcal{F})+\log(1/\delta)}{\varepsilon^2}\right)$ samples [2, 44]. 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:

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

[2] For a precise definition of PAC learning, see Definition 36.

▶ **Theorem 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 37 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 [42]. 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 [29, 30]. 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 [30] as a simple example of 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 [30]. Recently, the two-dimensional version was studied in [48] where a $O(\sqrt{n})$-gate threshold implementation

[3] We refer to [42] for the precise definition of distance used there.

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 [29, 30, 48] 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., [1, 13, 41]). 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 [27, 45]. 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 24). 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 [10]. Hence the result in [27, 45] 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, and also discuss the possibility of a better upper bound.

▶ **Theorem 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*

  Deterministic: $O(\log k \cdot n/k)$ *if* $k \le \sqrt{n}$;
  Randomized: $O(\log n \cdot \sqrt{n})$ *if* $k \ge \sqrt{n}$.

- ***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 9** ([31]). *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 $\le n/2$ are powers of some *primitive period* (shortest period of order $\le n/2$). 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 $k \le \sqrt{n}$ this yields the first upper bound claimed in Theorem 1.

▶ **Lemma 10.** *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.*

Next we supply the protocol for the second upper bound in Theorem 1.

▶ **Lemma 11.** *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).*

▶ Remark 12. 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 [23, 39, 4].

▶ **Observation 13.** $\mathrm{Disj}_{\Omega(n/k)}$ *reduces to* $\mathrm{SM}_{n,k}$ *(under some bipartition of input bits).*

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 15 was conjectured by [50].

▶ **Lemma 14.** $\mathrm{OR}_{\Omega(n/k)} \circ \mathrm{GT}_{\Omega(k)}$ *reduces to* $\mathrm{SM}_{n,k}$ *(under some bipartition of input bits).*

▶ **Lemma 15.** $\mathrm{OR}_m \circ \mathrm{GT}_\ell$ *has randomized communication complexity $\Omega(m \cdot \log \log \ell)$ for any m, ℓ.*

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 [51].

▶ **Definition 16** ($\mathsf{U} \cdot \mathsf{BPP}$ 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. [9] (who disproved the log-approximate-rank conjecture of [28]) 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 U·BPP protocol by a BPP protocol. By contrast, Yannakakis [51, Lemma 1] showed that U·P protocols can be made deterministic efficiently.

Our first bonus result is an efficient U·BPP protocol for determining if a given string has a primitive period. We do not know whether there is an efficient randomized protocol.

▶ **Lemma 17.** *Suppose the bits of $x \in \{0,1\}^n$ are split between two players. There is an* U·BPP *protocol of cost $O(\log^2 n)$ for deciding whether x has a primitive period (and to compute its order).*

If we let R_{pf} denote the randomized communication complexity of the above period finding problem, then we can interpret Lemma 17 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 18.** $\mathrm{SM}_{n,0.9n}$ *admits an* U·BPP *protocol of cost $O(\log n) + R_{\mathrm{pf}}$.*

4 Threshold Circuits

In this section we prove Theorem 3.

▶ **Theorem 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*

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

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 3.

▶ **Lemma 19.** *There is a depth-2 threshold circuit of size $O(n-k)$ computing $\mathrm{SM}_{n,k}$.*

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 [34], and the lower bound on communication complexity of $\mathrm{SM}_{n,k}$ from Theorem 1.

Nisan and Safra [34] 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 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 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 20.** *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 21.** *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 $|C'| \leq |C|$.*

In order to complete the proof of Theorem 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, [22, 24]) with the number of functions in $F_{\ell,t}$.

▶ **Proposition 22.** *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 $\Omega(\sqrt{t - t\log t/\ell})$.*

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 20, and the same lower bound holds (up to a constant factor in the asymptotics).

4.3 Depth-2 Circuits

In Theorem 23 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 3 and Theorem 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 13 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 $x := a_1 b_1 1^{k-2} 0 a_2 b_2 1^{k-2} 0 \dots a_n b_n 1^{k-2} 0$ 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 23.** *For every $1 < k \le n$,*

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

5 DeMorgan Circuits

In this section we prove Theorem 4 and Theorem 5.

▶ **Theorem 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*

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

▶ **Theorem 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 26 we will prove that this DNF is essentially optimal.

▶ **Lemma 24.** *For any $k \le n$ there exists a DeMorgan circuit of depth 2 and size $(n-k+1)\cdot 2^k + 1$ computing $\mathrm{SM}_{n,k}$.*

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

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 24 is optimal (up to a constant factor).

▶ **Lemma 26.**
For every $k > 2$, the DNF*-size of* $\mathrm{SM}_{n,k}$ *is at least*

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

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

▶ **Definition 27.** *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 28.** *For any $k \leq n$, every maxterm of* $\mathrm{SM}_{n,k}$ *has width at least*

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

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

▶ **Lemma 29.** *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

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

▶ **Lemma 30.** *For every k, the* CNF*-size of* $\mathrm{SM}_{n,k}$ *is at least*

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

Discussion

Lemma 26 and Lemma 30 together give the lower bounds of Theorem 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 13 and the known lower bound on the depth-3 complexity of Iterated Disjointness [20] and Disjointness [21].

5.3 Lower bound for unbounded depth

Now we prove the lower bound of Theorem 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 31.** *For $k > 1$, any DeMorgan circuit computing $\mathrm{SM}_{n,k}$ has size at least $n/2$.*

6 Learning

6.1 VC dimension

In this section we prove Theorem 8.

▶ **Theorem 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 32.** *A system $\mathcal{F}$ of subsets of $\{1,\dots,n\}$ 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 33.** *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

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

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

▶ **Lemma 34.** *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.*

▶ **Lemma 35.** *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

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

This concludes the proof of Theorem 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

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

Following the notion of PAC-learning [49, 44], we can now define what we mean by learning $\mathcal{H}_{k,\Sigma}$.

▶ **Definition 36.** *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 36 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

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

namely the measure under $\mathcal{D}$ of all pairs $(x, y) \in \Sigma^n \times \{0, 1\}$ with $h_\tau(x) \neq y$ [44]. 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 33, 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 33, 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 37.** *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 $O(n)$.*

References

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

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

3 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.

4 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.

5 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.

6 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.

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

8 Mark Braverman and Omri Weinstein. A Discrepancy Lower Bound for Information Complexity. *Algorithmica*, 76(3):846–864, 2016. doi:10.1007/s00453-015-0093-8.

9 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.

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

11 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.

12 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.

13 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.

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

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

16 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.

17 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.

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

19 Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, 1987.

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

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

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

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

24 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.

25 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.

26 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

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

28 Troy Lee and Adi Shraibman. *Lower Bounds in Communication Complexity*, volume 3. Now Publishers, 2009. doi:10.1561/0400000040.

29 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.

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

31 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.

32 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.

33 Saburo Muroga. Threshold logic and its application. *Wily-Interscience*, 1971.

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

35 Ian Parberry. *Circuit complexity and neural networks*. MIT press, 1994.

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

37 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.

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

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

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

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

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

43 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.

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

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

46 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.

47 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.

48 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.

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

50 Thomas Watson. Communication Complexity of Statistical Distance. *ACM Transactions on Computation Theory*, 10(1):2:1–2:11, 2018. doi:10.1145/3170708.

51 Mihalis Yannakakis. Expressing combinatorial optimization problems by Linear Programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991. doi:10.1016/0022-0000(91)90024-Y.

A Learning – 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 33 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 38.** *Let Σ be an infinite alphabet, and $k \geq 1$. Then*

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

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 $\mathcal{H}^{\text{and}}_{k,\Sigma}$ and $\mathcal{H}^{\text{or}}_{k,\Sigma}$ follows at once from the following Lemma proved in [6] (Lemma 3.2.3).

▶ **Lemma 39.** *Let $\mathcal{H}_1, \ldots, \mathcal{H}_c$ be classes of functions of* VC *dimension at most $\forall i$: $\text{VC}(\mathcal{H}_i) \leq d$. 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 $\text{VC}(\mathcal{H}^{and}) = O(dc \log c)$ *and* $\text{VC}(\mathcal{H}^{or}) = O(dc \log c)$.

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 $\mathcal{H}^{\text{and}}_{k,\Sigma}$ and $\mathcal{H}^{\text{or}}_{k,\Sigma}$ provided that the number c of (distinct) patterns is not too large. Let us see that the lower bounds of Lemma 35 hold for $\mathcal{H}^{\text{and}}_{k,\Sigma}$ and $\mathcal{H}^{\text{or}}_{k,\Sigma}$. Indeed, for the class $\mathcal{H}^{\text{and}}_{k,\Sigma}$, we use the construction from Lemma 35, 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 $c \leq k$, we take c longest patterns $\{\sigma^{k-c+1}, \ldots, \sigma^k\}$, and our construction gives a shattered set of size

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

For the class $\mathcal{H}^{\text{or}}_{k,\Sigma}$, 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 $c \leq 2^{d-1} - 1$ because $c \leq k$). 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 35 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 40.** *Let $1 \leq c \leq k$ 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 41.** $\text{VC}(\mathcal{E}_{k,\Sigma}) \leq \min(k \log|\Sigma|, \log(n-k+1) + 1)$.

Now we prove the following lower bound:

▶ **Lemma 42.** *Let Σ be a finite alphabet of size $|\Sigma| \geq 2$, then*

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

We remark that for the case of patterns of length *at most* k, Lemma 33 and Lemma 35 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 43.** *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 34 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 34, 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 $q_0 \ldots q_{2^d-1}$ 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 44.**

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