This dissertation/thesis/report entitled:

The Learning and Communication Complexity of Subsequence Containment

written by:

Mason T. DiCicco

has been approved for the degree of:

Master of Science

in

Computer Science

by

Prof. Daniel Reichman, Advisor

1 - Advisor Name

Prof. Gabor Sarkozy, Reader

2 - Department Chair or Committee Member Name

3 - Committee Member Name

4- Committee Member Name

5 - Committee Member Name

6 - Committee Member Name

December 2022

Month YEAR

The Learning and Communication Complexity of Subsequence Containment

Mason DiCicco* Daniel Reichman*

November 21, 2022

Abstract

We consider the learning and communication complexity of subsequence containment. In the learning problem, we seek to learn a classifier that positively labels a binary string x if it contains a fixed binary string y as a subsequence. In the communication problem, x and y are partitioned between two players, Alice and Bob, who wish to determine if x contains y as a subsequence using a minimal amount of communication. We devise asymptotically tight bounds for the sample complexity (VC dimension) of the learning problem and the communication complexity of the communication problem. Our results illustrate that the sample complexity of our learning problem can be considerably larger when the subsequence occurs in non-contiguous locations.

1 Introduction

Given a string x of length n and a string y of length $k \leq n$, we say y is a *subsequence* of x if all of the characters of y appear consecutively (but not necessarily contiguously) within x. The subsequence detection problem is to determine, given x and y, whether y is a subsequence of x. We study the *communication complexity* of this problem: the minimal communication required to compute whether y is a subsequence of x when the characters of x and y are partitioned between two parties, Alice and Bob. We primarily focus on *binary* sequences, but some of our results extend to arbitrary alphabets.

Subsequence detection has been described as "one of the most interesting and least studied problems in pattern matching" by [JS21]. From the learning perspective, non-contiguity seems to arise in certain applications; important features input data could be rather fragmented. In particular, time series (e.g., [KC17]), linguistic (e.g., [SCC+05]) and genetic (e.g., [TMAM20]) features are often non-contiguous. For example, subsequence *anomaly detection* for time series data (as defined by [KLLVH07]) is a widely studied problem in computer science with a variety of applications. It has been used to detect irregular heartbeats by [HNAK16], machine degradation in manufacturing by [MMP+13], hardware and software faults in data-centers by [PFT+15], noise within sensors by [BNR+18], and spoofed biometric data by [FAAK19]. Detecting subsequences is also useful in computational biology and has led to deep theoretical questions such as the study of the expected size of the longest common subsequence between two uniformly random strings [CS75].

In many applications, the strings that are considered with respect to subsequence detection have lengths in the millions. This can lead to a significant slowdown when attempting to find subsequences in a long data stream. End users with limited computational capacity may share their stream with a party with more computational resources, motivating the question of *communication*

*Computer Science Department, WPI. [mtdicicco@wpi.edu, dreichman@wpi.edu]

complexity. That is, how many bits of communication need to be exchanged in order to allow the party with more computational resources to solve the problem. Additionally, the existence or nonexistence of a subsequence can be useful in *classification*, and it is of practical and theoretical interest to understand how the sample complexity of subsequence-dependent classifiers depends on n and k, the lengths of the string and subsequence respectively.

We provide nearly tight bounds for the communication complexity of subsequence detection under a variety of settings (randomized vs. deterministic communication, different partitions of x, y between the two parties, and whether or not the length of y is fixed). We show that, up to log factors, the communication complexity of this problem scales like $O(k)$. This is somewhat surprising as our bounds hold for arbitrary partitions of x and y (not just the natural partition where Alice holds x and Bob holds y). We complement these upper bounds by providing a nearly matching lower bound of $\Omega(k)$. We note that the communication complexity of detecting a substring in *contiguous* locations under arbitrary bi-partitions is $\Omega(n)$ for $k = 2$, as shown by [GGRS19].

Next, we consider the VC dimension of a family of *subsequence* classifiers defined by containing a fixed subsequence. That is, every classifier is parameterized by $y \in \{0,1\}^k$ such that $x \in \{0,1\}^n$ is classified as 1 if and only if x contains y as a subsequence (for a precise definition, see Definition 9). We prove that the VC dimension of this family of (length k) subsequence classifiers is $\Theta(k)$. [SVL14] show that in some cases it is beneficial not to assume any upper bound on the length n of the string being classified. Our bounds on the VC dimension easily extend to this case as well. In either case, we are not aware of previous bounds on the VC dimension of this family. Our methods can also be used to bound the VC dimension of classification based on *super*sequences, where a sequence of length k evaluates to 1 if and only if it occurs in a fixed sequence of length $n \geq k$ as a subsequence. This classification problem resembles trace reconstruction as in [BKKM04] and may be of independent interest.

Our methods are straightforward. Lower bounds are proved using reductions from set-disjointness, and upper bounds are proved using simple protocols. Our reduction between subsequence containment and the disjointness problem is useful also in lower bounding the VC-dimension of the subsequence classifiers we study. This expands on the work of [KNR99] in that it serves as another illustration of the connection between lower bounds from communication complexity and learning theory.

While elementary, our proofs and reductions differ from those appearing in previous studies of the communication complexity of string related problems (e.g., [SW07, LNVZ06, GGRS19, BYJKK04]). Furthermore, our results uncover a qualitative difference between learning complexity of the contiguous versus the arbitrary. When the classifying subsequence is required to occur in *consecutive* locations within the string, the VC dimension of the classification problem was shown by [GGRS19] to be uniformly upper bounded by $O(\log n)$ (regardless of the length of the subsequence k). In contrast, when the contiguity requirement is dropped, the VC dimension of subsequence classifiers turns out to be $\Omega(k)$ – which can be exponentially larger than $\log n$ (for $k = \Omega(n)$). Furthermore, this lower bound holds even if we restrict the occurrence of the classifying substring to have gaps not larger than one; every two characters of the subsequence either appear next to one another or are separated by at most one character. We should remark that we make no attempt to optimize constant factors: we are primarily concerned with the *asymptotic* complexity in the learning setting.

1.1 Definitions

Definition 1 (Alphabets)**.** Let an *alphabet* Σ be any set of symbols (for instance, the binary alphabet $\{0,1\}$). Then, we denote Σ^n the set of length-n strings over Σ. Sometimes we consider

strings of *arbitrary* length by Σ^*.

Definition 2 (Subsequence Detection)**.** For $n \geq 1$ and alphabet Σ, define the Boolean function $\mathrm{SSD}^n(x, y)$ whose inputs are strings $x \in \Sigma^n$ and $y \in \Sigma^*$ and whose output is 1 if and only if y is a subsequence of x.

We also define SSD^n_k (for a positive integer $k \leq n$) where y is guaranteed to belong Σ^k.

Assumption 1. *When considering strings of arbitrary length, we assume that the length of y does not exceed the length of x. Clearly, if the length of y exceeds the length of x, then y cannot be a subsequence of x.*

Assumption 2. *We will always assume that the alphabet size does not exceed the lengths of the strings (i.e. $|\Sigma| \leq n$) as a string of length n can contain at most n unique symbols. However, we will generally assume a binary alphabet unless otherwise specified.*

Example 1.

$$\mathrm{SSD}^3(010, 00) = 1,$$
$$\mathrm{SSD}^6(101010, 111) = 1,$$
$$\mathrm{SSD}^6_3(120021, 211) = 0.$$

Remark 1. A natural question to ask is whether SSD^n_k belongs to AC_0. Namely, whether it can be computed by a Boolean circuit with $\wedge, \vee$ and $\neg$ gates of polynomial size in n and constant depth. The answer is negative:

Proposition 1. *For all $k \geq 1$, SSD^{2k}_{k+1} is not in AC_0.*

Proof. Set y to equal 1^{k+1} simplifies SSD^{2k}_{k+1} to the MAJORITY Boolean function which is known not to belong to AC_0 (as stated by [Juk12]).

□

1.1.1 Communication Complexity

We are mainly interested in the *communication* required to compute SSD^n. We now review the relevant definitions from Communication Complexity by [KN96].

Definition 3 (Communication Protocol)**.** Let $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. Suppose Alice and Bob are two players holding inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively, with the goal of computing $f(x, y)$. A *communication protocol* is a 1-bit message-passing protocol between Alice and Bob. The *cost* of a protocol is the maximum number of messages used to compute f over all inputs (x, y).

Definition 4 (Communication Complexity)**.** Let $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. The *deterministic communication complexity* of f, $D(f)$, is the minimal cost of a deterministic protocol that computes f. The *randomized communication complexity* of f, $R(f)$, is the minimal cost of a randomized protocol that computes f with error probability at most $1/3$.

Of primary interest in this paper is the function f whose inputs are sets $A, B \subseteq [n]$ and whose value is equal to 1 if and only if A and B are *disjoint*.

Definition 5 (Disjointness)**.** Define DISJ^n as the set-disjointness problem. Given subsets $A, B \subseteq [n]$ respectively, Alice and Bob must determine whether A and B are disjoint. We encode subsets of $[n]$ as their characteristic vectors in $\{0, 1\}^n$. Then, $\mathrm{DISJ}^n(a, b)$ is defined as the Boolean function whose inputs are characteristic vectors $a, b \in \{0, 1\}^n$ and whose output is 1 if and only if the corresponding subsets are disjoint.

As in the definition of SSD, we also consider a restricted variation. Define DISJ_k^n as the problem of set disjointness when $|A| = |B| = k$. Namely, it is the same as DISJ^n, with the restriction that both a and b have Hamming weight k. We always assume Alice gets a and Bob gets b.

Theorem 1 ([Raz90] [HW07]). *For all $n \geq k \geq 0$,*

1. $R(\mathrm{DISJ}^n) = \Theta(n)$.
2. $R(\mathrm{DISJ}_k^n) = \Theta(k)$ *for every* $k \leq n/2$.
3. $D(\mathrm{DISJ}_k^n) = \Theta\left(\log \binom{n}{k}\right)$ *for every* $k \leq n/2$.

Several results in this work rely on *reductions*. A reduction is a high-level way to relate two different problems by transforming one into the other. For our purposes, we define a reduction as follows.

Definition 6 (Reduction). Let $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ and $g : \mathcal{A} \times \mathcal{B} \to \{0,1\}$ be two communication problems. We say f *reduces to* g if there exists mappings $\rho : \mathcal{X} \to \mathcal{A}$ and $\phi : \mathcal{Y} \to \mathcal{B}$ such that

$$g(\rho(x), \phi(y)) = f(x,y) \text{ for all } (x,y) \in \mathcal{X} \times \mathcal{Y}.$$

We call the reduction *injective* if ρ and ϕ are both injective (i.e. $\rho(x) = \rho(y)$ if and only if $x = y$).

For instance, we show later that DISJ reduces to SSD. It follows that if we would have a communication protocol P for SSD then we would have a protocol Q for DISJ with the same cost as of P. However, Q cannot contradict existing lower bounds on DISJ. This means that SSD is *at least as "hard"* as DISJ, ignoring technical details discussed in Proposition 5.

Definition 7 (Bi-partition). For any communication problem, the *bi-partition* describes "who gets what" with respect to the input bits. We mainly focus on the *natural* bi-partition in which Alice holds x and Bob holds y. A more general version of this communication problem gives both parties complementary partitions of both x and y. For example, Alice may receive the odd-indexed bits of x while Bob receives the even-indexed bits.

We consider both natural and worst-case bi-partitions, and the partition under consideration will always be clear from the context. We sometimes consider a protocol for every possible bi-partition. In this case, the cost of the protocol is the maximal cost over all possible bi-partitions and inputs.

Definition 8 (Communication Matrix). Let $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. The *communication matrix* M_f is the $|\mathcal{X}| \times |\mathcal{Y}|$ matrix with $(M_f)_{x,y} = f(x,y)$.

For $n \geq k \geq 1$ and alphabet Σ, we denote the communication matrix $\Sigma^{n \times k} := M_{\mathrm{SSD}_k^n}$. This is a $|\Sigma|^n \times |\Sigma|^k$ binary matrix with

$$(\Sigma^{n \times k})_{x,y} = \mathrm{SSD}_k^n(x,y).$$

Example 2. Let $\Sigma = \{0,1\}$. Then

$$\Sigma^{3\times 2} = \left[\begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 000 & 1 & 0 & 0 & 0 \\ 001 & 1 & 1 & 0 & 0 \\ 010 & 1 & 1 & 1 & 0 \\ 011 & 0 & 1 & 0 & 1 \\ 100 & 1 & 0 & 1 & 0 \\ 101 & 0 & 1 & 1 & 1 \\ 110 & 0 & 0 & 1 & 1 \\ 111 & 0 & 0 & 0 & 1 \end{array}\right]$$

1.1.2 Learning Complexity

We also consider subsequence containment as a *learning problem*. See [SSBD14] for a more detailed discussion on statistical learning.

Definition 9 (Subsequence classifiers)**.** Let $\mathcal{H}_k^n$ denote the *hypothesis class* of length-k subsequence classifiers acting on strings of length n. That is, the collection of functions $\{h_y : y \in \{0,1\}^k\}$ where $h_y : \{0,1\}^n \to \{0,1\}$ such that $h_y(x) = 1$ if and only if y is a subsequence of x.

Also let $\mathcal{H}_*^n$ denote the collection of subsequence classifiers of *any length* acting on strings of length n.

Definition 10 (Supersequence classifiers)**.** Let $\mathcal{G}_k^n$ denote the hypothesis class of length-n supersequence classifiers acting on strings of length k. That is, the collection of functions $\{g_x : x \in \{0,1\}^n\}$ where $g_x : \{0,1\}^k \to \{0,1\}$ such that $g_x(y) = 1$ if and only if y is a subsequence of x.

Also let $\mathcal{G}_*^n$ denote the collection of length-n supersequence classifiers acting on strings of any length.

Definition 11 (PAC learning)**.** Let $\mathcal{X} = \{0,1\}^n$ be *labelled* by a fixed $h_y \in \mathcal{H}_k^n$ (which is unknown to the learner). Then, given a distribution $\mathcal{D}$ over $\mathcal{X}$, the *loss* of a hypothesis h_z is equal to

$$L_{\mathcal{D}}(h_z) = \mathbb{P}_{x\sim\mathcal{D}}(h_y(x) \neq h_z(x)).$$

A learning algorithm $\mathcal{A}$ is said to (ϵ, δ)-*PAC*-learn $\mathcal{H}_k^n$ if for every distribution $\mathcal{D}$ over $\mathbf{X}$ and every $h_y \in \mathcal{H}_k^n$, there exists a *sample size* N such that the following holds*:

Given N i.i.d samples $\{(x_1, h_y(x_1)), \cdots, (x_N, h_y(x_N))\}$ from $\mathcal{D}$ as input, $\mathcal{A}$ outputs (with probability $1-\delta$) a hypothesis $h_z \in \mathcal{H}_k^n$ with $L_{\mathcal{D}}(h_z) < \epsilon$.

Finally, we introduce the *Vapnik–Chervonenkis dimension*, a parameter often relevant to statistical learning which is known to be strongly connected to communication complexity (as in [KNR99]).

Definition 12 (VC dimension)**.** For a finite set A, let $\mathcal{H}$ be any collection of functions $f : A \to \{0,1\}$. Then, a subset $B \subseteq A$ is *shattered* by $\mathcal{H}$ if for every subset $B' \subseteq B$ there exists a function $f_{B'} \in \mathcal{H}$ which *realizes* B': For each $b \in B$, $f_{B'}(b) = 1$ iff $b \in B'$. The VC dimension of $\mathcal{H}$, denoted by $\mathrm{VCdim}(\mathcal{H})$, is the largest size of a subset of A that is shattered by $\mathcal{H}$.

For a communication problem $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, we define $\mathrm{VCdim}(f_{\mathcal{X}})$ as the VC dimension of the hypothesis class parameterized by $\mathcal{X}$. That is, the collection of functions $\mathcal{H}_{\mathcal{X}} := \{h_x : x \in \mathcal{X}\}$ such that $h_x(y) = f(x,y)$.

VC dimension essentially characterizes the number of samples needed to PAC-learn a family of classifiers.

Theorem 2 ([Han16, EHKV89])**.** *For any hypothesis class $\mathcal{H}$, the sample size required to PAC-learn $\mathcal{H}$ is equal to $\Theta(\mathrm{VCdim}(\mathcal{H}))$. (Here, Θ hides polynomial dependency of $1/\epsilon$, $1/\delta$, and indicates that the upper bound on the asymptotic learning complexity is tight.)*

*We focus on the realizable case: For a definition of agnostic PAC-learning please see [SSBD14]

1.2 Related work

1.2.1 Contiguous pattern matching

In the classical pattern matching problem, we seek to determine whether a string y of length k appears in *contiguous* locations in a string of length $n \geq k$. Let SM_k^n denote the contiguous string-matching problem. For $k \leq \sqrt{n}$ and arbitrary partitions, [GGRS19] prove an upper bound of $D(\mathrm{SM}_k^n) = O(n/k \cdot \log k)$ and a lower bound of $R(\mathrm{SM}_k^n) = \Omega(n/k \cdot \log \log k)$ bits of communication. We prove significantly smaller bounds for the communication complexity of non-contiguous pattern matching.

1.2.2 Non-contiguous pattern matching

[SW07] and [LNVZ06] proved tight lower bounds for the communication complexity of the LCS-k-decision problem of determining whether two strings of length n have a common subsequence of length k or greater. For example, [SW07] prove that $R(\text{LCS-}k\text{-decision}) = \Omega(n)$. These works are different than ours as they consider sequences with arbitrary alphabets and we focus on fixed alphabets allowing us to circumvent their strong lower bounds. Additionally, we focus on detecting a subsequence of length k in a string of length n whereas these works focus on computing the largest length of a common subsequence in two strings of length n. Consequently, our proof ideas differ from those by [SW07, LNVZ06].

Lower bounds on the query complexity of one-sided testers for subsequence-freeness were devised recently by [RR21]. While lower bounds for query complexity of testing algorithms were used by [GGR98] to derive lower bounds on VC-dimension, their lower bounds as well as those of [RR21] do not seem to imply our lower bounds for the VC dimension of classifiers based on the inclusion of a fixed pattern as a subsequence. This is because the lower bound proven by [RR21] applies to testers with *one-sided* error and arbitrary alphabets as opposed to our setting where binary sequences are concerned.

The *deletion channel* takes a binary string as input and independently deletes each bit with fixed probability d (See [JS21] for a detailed analysis). It was proven by [DSV12] that the problem of determining the capacity of the deletion channel can be exactly formulated as the subsequence detection problem.

[BC18] show an $\Omega((k/|\Sigma|)^{|\Sigma|})$ lower bound on the *one-way* communication complexity of subsequence detection. Additionally, they construct a sketch of size $O(k^{|\Sigma|} \log k)$, showing the lower bound is nearly tight.

1.2.3 Reconstructing from subsequences

The problem of reconstructing strings from their subsequences has been previously studied, initiated by [MMS+91] and subsequently expanded on by [Sco97, DS03, ADM+15] which give various conditions on when a string can be reconstructed from its k-subsequence decomposition. Our problem differs from the reconstruction problem studied in these works. For example, these works all consider the *multiset*-decomposition of subsequences which includes the multiplicities of each subsequence whereas we only consider the *set*-decomposition for the purposes of subsequence detection.

1.2.4 VC dimension

That the *one-way* randomized communication complexity of f is greater than or equal to its VC dimension was observed by [KNR99]. The crux of our discussion on sample complexity is the exponential gap between the VC dimension of *contiguous* and *non-contiguous* subsequence

containment. In particular, [GGRS19] showed recently that the contiguous string-matching problem has VC dimension $\Theta(\log(n))$, whereas we prove a linear lower bound on the VC dimension of non-contiguous subsequences.

2 Communication complexity

2.1 Deterministic protocols for the natural bi-partition

We now consider the natural bi-partition in which Alice holds x and Bob holds y. First, we lower bound the deterministic communication complexity of subsequence detection using the well-known *log-rank method*:

Theorem 3 ([KN96])**.** *For any function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$,*

$$D(f) \geq \log_2(\operatorname{rank} M_f),$$

where $\operatorname{rank} M_f$ *is equal to the number of linearly independent rows (or columns) of M_f.*

We now compute the rank of the communication matrix for subsequence detection, $\Sigma^{n\times k}$.

Lemma 1. *For all $n \geq k \geq 1$,*

$$\operatorname{rank}(\Sigma^{n\times k}) = |\Sigma|^k$$

Proof. For simplicity, we assume $\Sigma = \{0, 1, \cdots, m\}$ and index the rows and columns of $\Sigma^{n\times k}$ lexicographically (by the sequence $[0^n, 0^{n-1}1, \cdots, m^n]$). Then, it is clear that a string $s \in \Sigma^k$ does not appear as a subsequence in Σ^n until the s'th string, $0^{n-k}s$, where it appears as a contiguous subsequence. Thus

- $i < j \implies (\Sigma^{n\times k})_{i,j} = 0.$
- $i = j \implies (\Sigma^{n\times k})_{i,j} = 1.$

In particular, the first $|\Sigma|^k$ rows of $\Sigma^{n\times k}$ are a full-rank lower-triangular matrix.

□

Proposition 2. *For all $n \geq k \geq 1$ and alphabet Σ, under the natural bi-partition of inputs,*

1. $D(\mathrm{SSD}^n_k) = k \log |\Sigma| + 1$
2. $D(\mathrm{SSD}^n) = n \log |\Sigma| + 1$

Proof. Theorem 3 applied to Lemma 1 yields the first lower bound:

$$D(\mathrm{SSD}^n_k) \geq \log \operatorname{rank}(\Sigma^{n\times k}) = k \log |\Sigma|.$$

The second lower bound follows from the fact that the communication matrix for SSD^n contains each $\Sigma^{n\times k}$ as a sub-matrix (for every $k \leq n$) and thus has rank equal to n.

It is easy to achieve these bounds under the natural bi-partition; Bob sends Alice every character of y, each requiring $\log |\Sigma|$ bits. Alice then uses 1 bit to share the answer with Bob.

□

Remark 2. In fact, the same bounds apply to SM^n_k, the contiguous string-matching problem, because s appears contiguously in $0^{n-k}s$.

Corollary 1. *For all $n \geq k \geq 1$ and alphabet Σ, under the natural bi-partition of inputs,*

$$D(\mathrm{SM}^n_k) = k \log |\Sigma| + 1.$$

2.2 Deterministic protocols for arbitrary bi-partitions

We are also able to tightly bound the deterministic communication complexity of subsequence detection under the *worst-case* bi-partition via a reduction from disjointness.

Proposition 3. *For all $n \geq k \geq 1$, there exists a bi-partition B such that* DISJ_k^n *(under the natural bi-partition) reduces to* SSD_{4k}^{3n} *under B.*

Proof. Given inputs $a, b \in \{0,1\}^n$ of DISJ_k^n to Alice and Bob, consider the following inputs to SSD_{4k}^{3n}:

- $y = 1010\cdots10 = (10)^{2k}$,
- $x = a_1b_10a_2b_20\cdots a_nb_n0 = (a_ib_i0)_{1\leq i\leq n}$.

This induces the bi-partition of inputs to SSD_{4k}^{3n} which has Alice hold the a_i's and Bob hold the b_i's. The remaining bits can be partitioned arbitrarily, or even known to both parties simultaneously.

We note that both a and b contain exactly k 1's each. Thus there are $2k$ "isolated" 1's in x (i.e. y is a subsequence) if and only if a and b are disjoint. This completes the proof.

□

Corollary 2. *There is a bi-partition of inputs such that*

$$D(\mathrm{SSD}_k^n) = \Omega\left(\log\binom{n}{k}\right) \text{ for every } k \leq n/2.$$

Proposition 4. *Under any bi-partition of inputs (and any alphabet),*

$$D(\mathrm{SSD}_k^n) = O(k\log n).$$

Proof. Alice and Bob first exchange y requiring $O(k\log|\Sigma|)$ bits. Then they compute i, the first index in which $x_i = y_1$, requiring $O(\log n)$ bits (by exchanging an integer less than or equal to n). If there is no such index, then y is not a subsequence of x. Otherwise, this reduces to an instance of SSD_{k-1}^{n-i} with input $x' := x_{i+1}x_{i+2}\cdots x_n$, and $y' = y_2y_3\cdots y_k$. The bi-partition of inputs remains unchanged, although exchanging y is no longer required.

Continuing iteratively, we have $D(\mathrm{SSD}_k^n) = O(k\log|\Sigma| + k\log n) = O(k\log n)$ if we assume $|\Sigma| \leq n$ as in Definition 2. This achieves the lower bound in Corollary 2, up to a difference of $O(k\log k)$, as

$$\log\binom{n}{k} = O\left(\log\left(\frac{n}{k}\right)^k\right) = O\left(k\log n - k\log k\right)$$

□

2.3 Randomized protocols for the natural bi-partition

We now give a similar reduction from disjointness which proves a *randomized* lower bound for subsequence detection under the natural bi-partition.

Proposition 5. DISJ_k^n *(injectively) reduces to* SSD_{2n+k}^{3n} *under the natural bi-partition.*

Proof. Let Alice and Bob hold $a, b \in \{0,1\}^n$ respectively. Alice and Bob each construct (without communication) strings x and y, which both consist of n "blocks". Each block will consist of either two or three bits.

- Alice constructs block i equal to $a_i\overline{a}_i0$ (i.e. $0 \mapsto 010$ and $1 \mapsto 100$). That is,

$$x = a_1\overline{a}_10 \cdot a_2\overline{a}_20 \cdots a_n\overline{a}_n0$$

- Bob constructs block i equal to 00 if $b_i = 0$, and 010 if $b_i = 1$ (i.e. $0 \mapsto 00$ and $1 \mapsto 010$). Supposing b contains k ones appearing at indices $i_1 < \cdots < i_k$, then

$$y = 0^{2(i_1-1)} \cdot 010 \cdot 0^{2(i_2-i_1-1)} \cdot 010 \cdots 0^{2(i_k-i_{k-1}-1)} \cdot 010 \cdot 0^{2(n-i_k-1)}$$

If a and b are disjoint, then the i'th block of y is a subsequence of the i'th block of x for all i. Thus, y is a subsequence of x. Otherwise, there exists some index i with $a_i = b_i = 1$. Let $\alpha, \beta, \gamma, \delta$ partition x and y around the i'th block as follows.

$$x = \alpha \cdot 100 \cdot \gamma$$
$$y = \beta \cdot 010 \cdot \delta$$

Note that both α and β contain exactly $2i - 2$ zeros, and both terminate in a 0. Thus, $010 \cdot \delta$ must be a subsequence of $100 \cdot \gamma$. Furthermore, γ and δ contain exactly $n - i$ zeros. If y was a subsequence of x, then 010 must be a subsequence of 100, which is not the case. Thus, y is not a subsequence of x.

As x has length $3n$ and y has length $2n + k$, we have proven that $\mathrm{DISJ}_k^n(a, b) = \mathrm{SSD}_{2n+k}^{3n}(x, y)$. As every a maps to a unique x (and similarly b to y), the reduction is injective, concluding the proof.

□

Table 1: An example instance of the reduction when a and b are not disjoint. Note that there are precisely two zeros in every cell of x and y. Thus, for y to be a subsequence of x, every zero in y must match a zero in x *in the same column*. However, we cannot match both zeros in the red column because we must also match the bold "1".

a	0	1	1	0	0	1	1	0	1	1	0
$x(a)$	010	100	100	010	010	100	**1**00	010	100	100	010
$y(b)$	010	00	00	010	00	0**1**0	00	010	00	00	010
b	1	0	0	1	0	1	0	1	0	0	1

Note that the reduction above is fairly restrictive; we require the lengths of x and y to be $3n$ and $2n + k$ respectively. However, a simple padding argument gives us the following.

Corollary 3. *For $n \geq k \geq 0$, under the natural bi-partition of inputs,*

1. $R(\mathrm{SSD}_k^n) = \Theta(k)$
2. $R(\mathrm{SSD}^n) = \Theta(n)$

Proof. Lower bounds follow from two observations regarding the strings constructed in Proposition 5:

1. Alice may *pad* as many 1's as desired to the end of x without compromising the reduction. In particular, if Alice constructs $x \cdot 1^N$ and Bob constructs y as above, we have a reduction from DISJ_k^n to $\mathrm{SSD}_{2n+k}^{3n+N}$ for any value of N. Then, for any fixed value of n, we may make a simple parameterization, $n' = 3n + N$ and $k' = 2n + k$, to obtain

$$R(\mathrm{SSD}_{k'}^{n'}) = \Omega(k) = \Omega(k').$$

2. As x always has length $3n$, the exact same construction reduces DISJ^n to SSD^{3n}.

Indeed, both of these lower bounds are met (up to constant factors) by the trivial deterministic protocol.

□

Remark 3. It is interesting to note that, although y appears non-contiguously in x, it is highly *constrained* (using the language of [FSV06]). That is, y appears in x such that no two consecutive characters are separated by more than 1 character of x.

3 VC dimension

This section amounts to the following lemma, which connects our communication complexity results to subsequent sample complexity results.

Lemma 2. *Let $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ and $g : \mathcal{A} \times \mathcal{B} \to \{0,1\}$ be two communication problems and suppose f has an **injective** reduction to g. Then,*

1. $\mathrm{VCdim}(f_{\mathcal{X}}) \leq \mathrm{VCdim}(g_{\mathcal{A}})$.
2. $\mathrm{VCdim}(f_{\mathcal{Y}}) \leq \mathrm{VCdim}(g_{\mathcal{B}})$.

Proof. (We prove only statement 1 as both arguments are identical.) Recall that a reduction from f to g induces two mappings $\rho : \mathcal{X} \to \mathcal{A}$ and $\phi : \mathcal{Y} \to \mathcal{B}$ such that

$$g(\rho(x), \phi(y)) = f(x, y) \text{ for all } (x, y) \in \mathcal{X} \times \mathcal{Y}.$$

Let $S \subseteq \mathcal{Y}$ be shattered by $f_{\mathcal{X}}$. By definition, for every $T \subseteq S$, there exists an $x_T \in \mathcal{X}$ such that (for all $s \in S$),

$$f(x_T, s) = 1 \text{ if and only if } s \in T. \tag{1}$$

We will show that every such T (which is realized by x_T) uniquely maps to a subset $\phi(T)$ which is realized by $\rho(x_T)$. Indeed, as ρ is injective, every x_T maps to a unique $\rho(x_T) \in \mathcal{A}$ such that

$$g(\rho(x_T), \phi(s)) = f(x_T, s).$$

By (1), and for every $s \in S$, this is equal to 1 if and only if $s \in T$. Now consider the set $\phi(S) = \{\phi(s) : s \in S\} \subseteq \mathcal{B}$. As ϕ is injective, we have $s \in T$ if and only if $\phi(s) \in \phi(T)$. Thus, $\phi(S)$ is shattered by $g_{\mathcal{A}}$.

□

In the same vein as in Proposition 5, we now calculate the VC dimension of *disjointness classifiers*[†] for the sake of lower-bounding the VC dimension of *subsequence classifiers*.

[†]As DISJ is symmetric ($\mathrm{DISJ}(a, b) = \mathrm{DISJ}(b, a)$), we can refer to the corresponding hypothesis class as DISJ^n.

Table 2: For $k = 2, 3, 4, 5$ (and various values of n) we calculate (by brute force) the largest $S \in \{0,1\}^n$ that is shattered by $\mathcal{H}_k^n$.

k	n	$S \subset \{0,1\}^n$ shattered by $\{0,1\}^k$
2	3	$011, 001$
3	6	$100001, 111000, 000111$
4	5	$10100, 10010, 01010$
5	8	$11000101, 01110010, 10011010, 10110011$

Proposition 6 ([KNR99])**.** *For all $n \geq 0$,*

$$\mathrm{VCdim}(\mathrm{DISJ}^n) = n.$$

Proof. Of course, $\mathrm{VCdim}(\mathrm{DISJ}^n)$ cannot be greater than n by a simple surjectivity argument; if S is shattered, then for each $T \in 2^S$, there must exist a unique *classifying* subset $X_T \in 2^{[n]}$. Thus, $|S| \leq n$.

We can construct a shatterable set of size n. In particular, we can take the collection of singletons $S = \{\{1\}, \cdots, \{n\}\}$. Indeed, the subset of singletons $\{\{i_1\}, \cdots, \{i_k\}\} \subseteq S$ is realized by the *complement* of their union, $C = [n] \setminus \{i_1, \cdots, i_k\}$; for each $i \in [n]$, each singleton $\{i\}$ is disjoint from C if and only if $i \in C$.

□

Corollary 4. *For all $n \geq 0$,*

$$\frac{n}{3} \leq \mathrm{VCdim}(\mathcal{H}^n) \leq n + 1.$$

Proof. Recall that we have an injective reduction from DISJ^n to SSD^{3n} (Proposition 5). Thus, the lower bound follows from Lemma 2 applied to Proposition 6. The upper bound in this case again follows from a simple surjectivity argument.

□

Example 3. What follows is the explicit construction for $n = 9$. The shattered strings (which correspond to the singletons $\{1\}, \{2\}, \{3\}$), are

$$s_1 = 100010010$$
$$s_2 = 010100010$$
$$s_3 = 010010100$$

Table 3: For completeness, we enumerate every subsequence-classifier y and the corresponding subset $S_y \subseteq \{s_1, s_2, s_3\}$ (i.e. y is a subsequence of *every string* in S_y, but *no strings* in the complement).

y	$S_y \subseteq \{s_1, s_2, s_3\}$
010010010	$\emptyset$
00010010	s_1
01000010	s_2
01001000	s_3
0000010	s_1, s_2
0001000	s_1, s_3
0100000	s_2, s_3
000000	s_1, s_2, s_3

The restricted version follows similarly, with a slight loss of generality due to the fact that every classifying subset must have size exactly k.

Corollary 5. *For all $k \leq n/2$,*

$$\mathrm{VCdim}(\mathrm{DISJ}_k^n) \geq k.$$

Proof. We can indeed shatter a subset of the singletons, $T = \{\{1\}, \cdots, \{k\}\}$ provided that $k \leq n/2$. Every subset $\{\{i_1\}, \cdots, \{i_\ell\}\} \subset T$ would indeed be realized by the complement $[k] \setminus \{i_1, \cdots, i_\ell\}$ as before, but every classifying subset must have size k (which is not necessarily the case here). Thus, we *pad* these classifiers with elements not in $[k]$. In particular, let $P(m) = \{k+1, \cdots, k+m\}$ consist of m *padding* elements (where $P(0) = \emptyset$). Then, the set $\{\{i_1\}, \cdots, \{i_\ell\}\}$ is realized by the union

$$([k] \setminus \{i_1, \cdots, i_\ell\}) \cup P(\ell)$$

The left-hand side contributes $k-\ell$ elements and the right-hand side ℓ, for a grand total of k elements as desired. At most, we require k padding elements to realize T itself. Thus, T is shattered when $k \leq n/2$.

□

Corollary 6. *For all $n \geq 6k/5 \geq 0$*

$$\frac{k}{5} \leq \mathrm{VCdim}(\mathcal{H}_k^n) \leq k.$$

Proof. By Proposition 5 and the subsequent padding argument from Corollary 3, we have an injective reduction from DISJ_k^n to $\mathrm{SSD}_{2n+k}^{3n+N}$ for all $N \geq 0$. Thus, by Lemma 2 applied to Corollary 5, we have for $k \leq n/2$,

$$k \leq \mathrm{VCdim}(\mathcal{H}_{2n+k}^{3n+N}) \leq 2n + k.$$

Then, by optimizing over n (i.e. substituting $n = 2k$) we obtain $k \leq \mathrm{VCdim}(\mathcal{H}_{5k}^{6k+N}) \leq 5k$, and dividing k by 5 completes the proof.

□

Remark 4. Note that the same bounds apply even for *arbitrary* n. Clearly, $\{0,1\}^n \subset \{0,1\}^*$, so our shattered set is also a subset of $\{0,1\}^*$. Thus, our results apply even to the hypothesis class $\mathcal{H}_k^*$ whose domain includes sequences of any length.

Indeed, Corollary 6 applied to Theorem 2 tells us that non-contiguous subsequences have sample complexity independent of n.

Corollary 7. *For all $k \geq 0$, $n \geq 6k/5$, the sample complexity of PAC-learning length-k subsequence classifiers is $\Theta(k)$*[‡]*.*

Remark 5. Efficiently recovering the subsequence based on the training data via the ERM rule seems intractable even in the realizable case: Computing the longest common subsequence of multiple strings was shown by [JL95] to be NP-hard. When the LCS is promised to have length k, [IF92] give an algorithm which computes a common subsequence of length k in $O(Nn(n-k)^{N-1}) = O(kn(n-k)^{O(k)})$ time.

3.1 Supersequence classification

The previous section analyzed subsequence classifiers, which essentially parameterize the two-argument subsequence detection problem $f(x, y)$ for a fixed subseqeunce y and variable string x, that is $f_y(x)$. We may also consider the analogous *supersequence* classifiers, $f_x(y)$, where the string x is fixed and the subsequence y is varied.

As the disjointness function $\mathrm{DISJ}^n(x, y)$ is symmetric with respect to x and y, the following is a direct consequence of the second statement of Lemma 2 applied to Proposition 6 and Corollary 5.

Proposition 7. *For all $n \geq 6k/5 \geq 0$,*

1. $\dfrac{n}{3} \leq \mathrm{VCdim}(\mathcal{G}^n_*)$.
2. $\dfrac{k}{5} \leq \mathrm{VCdim}(\mathcal{G}^n_k)$.

As usual, we can prove a general upper bound of n by a simple surjectivity argument. However, in some cases it is possible to obtain a tighter bound.

Proposition 8. *For all $n \geq 0$ and $k \geq n/2$,*

$$\mathrm{VCdim}(\mathcal{G}^n_k) \leq n \cdot H(k/n) + 1$$

where $H(x) = -x \log_2 x - (1-x)\log_2(1-x))$ is the binary entropy function.

Before proving Proposition 8, we first prove the following Lemma.

Lemma 3. *Let $E(n, k)$ denote the maximum number of length-n supersequences of any **fixed** binary string of length k. Then,*

$$\mathrm{VCdim}(\mathcal{G}^n_k) \leq \log_2(E(n, k)) + 1.$$

Proof. Let S be some shattered set $(y_1, \cdots, y_d)$. By definition, for each $T \subseteq S$, there exists a unique classifying supersequence x_T which realizes T. In particular, as the string x_1 belongs to precisely 2^{d-1} elements of 2^S (i.e. unique subsets of S), there must exist at least 2^{d-1} unique *supersequences* of y_1. Thus, if $E(n, k) < 2^{d-1}$, we have a contradiction. □

[‡]As in Theorem 2, Θ hides polynomial dependency on $1/\epsilon$, $1/\delta$, and indicates that the upper bound on the asymptotic learning complexity is tight.

Thus, proving Proposition 8 amounts to calculating the value of $E(n,k)$. To begin with, we may consider the subsequence $y = 1^k$. Then, any $x \in \{0,1\}^n$ is a supersequence of y if and only if x contains $\ell \geq k$ ones. We have by counting that there are exactly $\binom{n}{\ell}$ binary strings which contain precisely ℓ zeros. Thus, we obtain the lower bound $E(n,k) \geq \sum_{\ell=k}^{n} \binom{n}{\ell}$. Interestingly, as noted by [Dix13], $E(n,k)$ is *invariant* over the choice of subsequence. Thus, this expression is indeed the exact value of $E(n,k)$. For completeness, we give a self-contained proof below.

Lemma 4. *For all $n \geq k \geq 0$,*

$$E(n,k) = \sum_{\ell=k}^{n} \binom{n}{\ell}$$

Proof. First we define some simplifying notation: For a binary string z, we let z_{-1} denote the *tail* of z (i.e. $z_{-1} = z_2 z_3 \cdots$). Now, let $n \geq k \geq 0$ and fix a binary string $y \in \{0,1\}^k$. Then, let $E(n,y)$ denote the number of supersequences of y. We show inductively that $E(n,y)$ is invariant over the choice of y.

Clearly, when y has length 1, we have $E(n,0) = E(n,1) = 2^n - 1$, and when y has length n we have $E(n,y) = 1$. Then, for $1 < |y| < n$, we have the following.

$$\begin{aligned}
E(n,y) &= \sum_{x \in \{0,1\}^n} \mathbb{1}[x \text{ contains } y \text{ as a subsequence}] \\
&= \sum_{x \in \{0,1\}^n} (\mathbb{1}[x_1 = y_1]\mathbb{1}[x_{-1} \text{ contains } y_{-1} \text{ as a subsequence}] \\
&\qquad + \mathbb{1}[x_1 \neq y_1]\mathbb{1}[x_{-1} \text{ contains } y \text{ as a subsequence}]) \\
&= \sum_{z \in \{0,1\}^{n-1}} \mathbb{1}[z \text{ contains } y_{-1} \text{ as a subsequence}] + \sum_{z \in \{0,1\}^{n-1}} \mathbb{1}[z \text{ contains } y \text{ as a subsequence}] \\
&= E(n-1, y_{-1}) + E(n-1, y).
\end{aligned}$$

The induction step essentially "strips" the first bit from x, regardless of the value of y. Thus, $E(n,y)$ depends only on the length of y. So we may abuse notation and let $E(n,k) := E(n,1^k)$ for the sake of computation. Then, $x \in \{0,1\}^n$ is a supersequence of 1^k if and only if x contains $\ell \geq k$ ones. The result follows from the fact that there are exactly $\binom{n}{\ell}$ binary strings which contain precisely ℓ ones.

□

Now, upper bounding the VC dimension of supersequence classifiers amounts to the following well-known application of Stirling's approximation.

Proof of Proposition 8. The result follows from the estimate (for $\alpha > 1/2$)

$$\sum_{k=\alpha n}^{n} \binom{n}{k} \leq 2^{H(\alpha)n}$$

as derived in [MS77].

□

4 Future directions

There are several questions arising from this work. For the learning problem, we focused on binary alphabets. Studying the VC dimension for larger alphabets is of interest in several applications. Additionally, studying subsequence classifiers based on the occurrence of *multiple* subsequences is of potential interest. Recently the effect contiguity of features in classification tasks on the efficacy of various architectures of neural networks (convolutional vs fully connected nets) was studied by [SS+20]. Future empirical and theoretical study on how the number and magnitude of gaps influences the success of different learning methods for our subsequence-based classifier could be of interest.

5 Acknowledgements

We wish to thank the anonymous reviewers for providing helpful feedback, as well as for bringing [BC18] to our attention. We also thank Cristopher Moore for insightful discussions on this topic.

References

[ADM+15] Jayadev Acharya, Hirakendu Das, Olgica Milenkovic, Alon Orlitsky, and Shengjun Pan. String reconstruction from substring compositions. *SIAM Journal on Discrete Mathematics*, 29(3):1340–1371, 2015.

[BC18] Karl Bringmann and Bhaskar Ray Chaudhury. Sketching, streaming, and fine-grained complexity of (weighted) lcs. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2018.

[BKKM04] Tugkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. *Departmental Papers (CIS)*, page 173, 2004.

[BNR+18] Sara Bahaadini, Vahid Noroozi, Neda Rohani, Scott Coughlin, Michael Zevin, Joshua R Smith, Vicky Kalogera, and A Katsaggelos. Machine learning for gravity spy: Glitch classification and dataset. *Information Sciences*, 444:172–186, 2018.

[BYJKK04] Ziv Bar-Yossef, Thathachar 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.

[CS75] Václáv Chvatal and David Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12(2):306–315, 1975.

[Dix13] John D Dixon. Longest common subsequences in binary sequences. *arXiv preprint arXiv:1307.2796*, 2013.

[DS03] Miroslav Dudık and Leonard J Schulman. Reconstruction from subsequences. *Journal of Combinatorial Theory, Series A*, 103(2):337–348, 2003.

[DSV12] Michael Drmota, Wojciech Szpankowski, and Krishnamurthy Viswanathan. Mutual information for a deletion channel. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 2561–2565. IEEE, 2012.

[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.

[FAAK19] Soroush Fatemifar, Shervin Rahimzadeh Arashloo, Muhammad Awais, and Josef Kittler. Spoofing attack detection by anomaly detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8464–8468. IEEE, 2019.

[FSV06] Philippe Flajolet, Wojciech Szpankowski, and Brigitte Vallée. Hidden word statistics. *Journal of the ACM (JACM)*, 53(1):147–183, 2006.

[GGR98] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

[GGRS19] Alexander Golovnev, Mika Göös, Daniel Reichman, and Igor Shinkar. String matching: Communication, circuits, and learning. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

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

[HNAK16] Medina Hadjem, Farid Naït-Abdesselam, and Ashfaq Khokhar. St-segment and t-wave anomalies prediction in an ecg data using rusboost. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6. IEEE, 2016.

[HW07] Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[IF92] Robert W Irving and Campbell B Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In *Annual Symposium on Combinatorial Pattern Matching*, pages 214–229. Springer, 1992.

[JL95] Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24:1122–1139, 1995.

[JS21] Svante Janson and Wojciech Szpankowski. Hidden words statistics for large patterns. *The Electronic Journal of Combinatorics*, pages P2–36, 2021.

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

[KC17] Takuya Kamiyama and Goutam Chakraborty. Real-time anomaly detection of continuously monitored periodic bio-signals like ecg. In *New Frontiers in Artificial Intelligence*, pages 418–427. Springer International Publishing, 2017.

[KLLVH07] Eamonn Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. Finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*, 11(1):1–27, 2007.

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1996.

[KNR99] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

[LNVZ06] David Liben-Nowell, Erik Vee, and An Zhu. Finding longest increasing and common subsequences in streaming data. *Journal of Combinatorial Optimization*, 11(2):155–175, 2006.

[MMP+13] Katsiaryna Mirylenka, Alice Marascu, Themis Palpanas, Matthias Fehr, Stephan Jank, G. Welde, and D. Groeber. Envelope-based anomaly detection for high-speed manufacturing processes. In *European Advanced Process Control and Manufacturing Conference*, 2013.

[MMS+91] Bennet Manvel, Aaron Meyerowitz, Allen Schwenk, Ken Smith, and Paul Stockmeyer. Reconstruction of sequences. *Discrete Mathematics*, 94(3):209–219, 1991.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

[PFT+15] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment*, 8(12):1816–1827, 2015.

[Raz90] Alexander A Razborov. On the distributional complexity of disjointness. In *International Colloquium on Automata, Languages, and Programming*, pages 249–253. Springer, 1990.

[RR21] Dana Ron and Asaf Rosin. Optimal distribution-free sample-based testing of subsequence-freeness. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 337–256. SIAM, 2021.

[SCC+05] Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. Translating with non-contiguous phrases. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 755–762, 2005.

[Sco97] Alex D Scott. Reconstructing sequences. *Discrete Mathematics*, 175(1-3):231–238, 1997.

[SS+20] Shai Shalev-Shwartz et al. Computational separation between convolutional and fully-connected networks. In *International Conference on Learning Representations*, 2020.

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

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[SW07] Xiaoming Sun and David P Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 336–345. Citeseer, 2007.

[TMAM20] Behrooz Tahmasebi, Mohammad Ali Maddah-Ali, and Seyed Abolfazl Motahari. The capacity of associated subsequence retrieval. *IEEE Transactions on Information Theory*, 67(2):790–804, 2020.