The Communication Complexity of Subsequence Detection

Mason DiCicco*  Daniel Reichman*

May 6, 2022

Abstract

We study the communication complexity of deciding whether a binary sequence \( x \) of length \( n \) contains a binary sequence \( y \) of length \( k \) as a subsequence. We give nearly tight bounds for the communication complexity of this problem, and extend most of our results to larger alphabets. Finally, we prove a lower bound for the VC dimension of a family of classifiers that are based on subsequence containment.

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 many of our results extend to arbitrary alphabets.

We provide nearly tight bounds for the communication complexity of this problem under a variety of settings (randomized vs. deterministic communication, different partitions of \( x, y \) between the two parties). 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 \( y \)).

When Alice holds \( x \) and Bob holds \( y \) (i.e. the natural bi-partition) we show that no deterministic protocol is better than the trivial protocol where Bob sends all the bits in \( y \) to Alice. For randomized protocols, we give a lower bound of \( \Omega(\log k) \) bits of communication. Under the worst-case bi-partition, we give a tight bound of \( \Theta(k \log n) \) for the deterministic communication complexity under any alphabet, as well as a randomized lower bound of \( \Omega(k) \).

We also consider the VC dimension of a family of classifiers defined by containing a fixed binary sequence as a sequence. That is given \( y \in \{0,1\}^k \), we let \( y \) positively classify \( x \in \{0,1\}^n \) if and only if \( x \) contains \( y \) as a subsequence. We prove an \( \Omega(\log k) \) lower bound on the VC dimension of (length \( k \)) of the aforementioned family of \( 2^k \) classifiers. We are not aware of previous bounds on the VC dimension of this family.

Our methods are very straightforward. Lower bounds are proved using reductions to the “usual suspects:” Disjointness and Indexing, and upper bounds are proved using simple protocols. However, our proofs are indeed different from those appearing in previous studies of the communication complexity of string related problems [SW07, LNZ06, GGRS19, BYJKK04].

*Computer Science Department, WPI. [mdicipico@wpi.edu, dreichman@wpi.edu]
While our upper bounds in the communication setting hold for all possible values of \( k \leq n \), our lower bounds only hold for specific values of \( k \). Finding tight lower bounds for all \( k \leq n \) is an interesting problem that is left for future work.

Non-contiguity seems to arise in certain applications; important features of an input stream may not always be contiguous, but could be rather fragmented. In particular, time series \([KC17]\), linguistic \([SCC+05]\) and genetic \([TMAM20]\) features are often non-contiguous. Subsequence anomaly detection for time series data as defined in \([KLLVH07]\) is a widely studied problem in computer science with a variety of applications, such as detecting irregular heartbeats \([HNAK16]\), machine degradation in manufacturing \([MMP+13]\), hardware and software faults in data-centers \([PFT+15]\), noise within sensors \([BNR+18]\), and spoofed biometric data \([FAAK19]\).

1.1 Definitions

**Definition 1** (Subsequence Detection). For integers \( n \geq k \geq 1 \) and alphabet \( \Sigma = \{0, 1, \cdots, m\} \) with \( m \geq 1 \), define the Boolean function

\[
SSD_{n,k,m} : \Sigma^n \times \Sigma^k \to \{0, 1\}, \quad SSD_{n,k,m}(x, y) := \begin{cases} 1 & y \text{ is a subsequence of } x, \\ 0 & \text{otherwise.} \end{cases}
\]

When no alphabet is provided, we default to binary; \( SSD_{n,k} := SSD_{n,k,1} \). We also assume that \( m < n \).

**Remark 1.** We justify the assumption that \( m < n \) by the fact that a string of length \( n \) can contain at most \( n \) unique symbols. The symbols themselves have no distinguishable value, so we consider strings to be identical under alphabet re-labeling (i.e. 01010 \( \equiv \) 23232). Thus, we can always re-label to effectively limit the alphabet of any string to the same set of \( n \) characters.

**Example 1.**

\[
SSD_{3,2}(010,00) = 1, \quad SSD_{6,3}(101010,111) = 1, \quad SSD_{6,3,2}(120021,211) = 0.
\]

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 \( \land, \lor \) 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.** Setting \( y \) to equal \( 1^{k+1} \) simplifies \( SSD_{2k,k+1} \) to the MAJORITY Boolean function which is known not to belong to \( AC_0 \) \([Juk12]\). \( \square \)

We now review the relevant definitions of Communication Complexity \([KN96]\).

**Definition 2** (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 it uses to compute \( f \) over all inputs \( (x, y) \).

**Definition 3** (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.
Remark 2 (Bi-partitioning). For any communication protocol, we say the “natural” bi-partition of inputs has Alice hold $x$ and Bob hold $y$. A more general version of this communication problem gives both parties complimentary partitions of $x$ and $y$. For example, if $x$ and $y$ are binary strings, Alice may receive the odd-indexed bits of $(x, y)$ 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 4 (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 $\Sigma = \{0, 1, \ldots, m\}$, we denote $\Sigma^{n \times k} := M_{\text{SSD}, n, k, m}$. This is a $(m + 1)^n \times (m + 1)^k$ binary matrix with

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

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

$$\Sigma^{3 \times 2} = \begin{bmatrix} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{bmatrix}^T.$$

1.2 Related work

Subsequence detection has been described as “one of the most interesting and least studied problems in pattern matching” in [JS21], which studies this problem for random strings over binary and non-binary alphabets. A folklore result (e.g., [FHS04]) is that subsequence detection for a string $x$ of length $n$ admits an $O(n)$ algorithm.

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 $\text{SM}_{n,k}$ denote the contiguous string-matching problem. For $k \leq \sqrt{n}$ and arbitrary partitions, the authors of [GGRS19] prove an upper bound of $D(\text{SM}_{n,k}) = O(n/k \cdot \log k)$ and a lower bound of $R(\text{SM}_{n,k}) = \Omega(n/k \cdot \log \log k)$ bits of communication. We prove significantly smaller bounds for the communication complexity of non-contiguous pattern matching.

1.2.2 Non-contiguous pattern matching

Tight lower bounds are known [SW07, LNVZ06] 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, the authors of [SW07] prove that $R(\text{LCS}-k\text{-decision}) = \Omega(n)$ for $3 \leq k \leq n/2$. These works are different than ours as they consider sequences with arbitrary alphabets and we focus on fixed alphabets allowing us to circumvent the strong lower bounds in [SW07]. Additionally, we focus on detecting a subsequence of length $k$ in a string of length $n$ whereas [SW07, LNVZ06] focus on computing the largest length of a common subsequence in two strings of length $n$. Consequently, our proof ideas differ from those in [SW07, LNVZ06].
Lower bounds on the query complexity of one-sided testers for subsequence-freeness were devised recently in \[RR21\]. While lower bounds for query complexity of testing algorithms can be used to derive lower bounds on VC-dimension \[GGR98\], the lower bounds in \[RR21\] 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 in \[RR21\] applies to testers with one-sided error and arbitrary sequences as opposed to our setting where binary sequences are concerned.

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

The authors of \[BCT18\] show an \(\Omega((k/m)^m)\) lower bound on the one-way communication complexity of subsequence detection. Additionally, they construct a sketch of size \(O(km \log k)\), showing the lower bound is nearly tight. Our lower bound on the randomized two-way communication complexity uses a reduction to indexing similar to that in \[BCT18\]. We stress that the main utility of our lower bound is that it informs the construction of a shattered set for lower bounding the VC dimension. Additionally, both of these reductions build on a construction in \[KNR99\] used to lower bound the one-way communication complexity of the index function.

1.2.3 Reconstructing from subsequences

The problem of reconstructing strings from their subsequences has been previously studied, initiated by the authors of \[MMS91\] and subsequently expanded on in \[Sco97, DS03, ADM15\] 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.

1.3 Binary vs. arbitrary alphabets

Remark 3. By default, we will assume \(x\) and \(y\) are binary sequences. One very simple observation is that the communication complexity of SSD\(_{n,k,m}\) is lower bounded by that of SSD\(_{n,k}\) for \(m > 1\). We formalize this in Proposition 2.

Proposition 2. For all \(n \geq k \geq 1\) and \(m > 1\),

- \(D(\text{SSD}_{n,k,m}) = \Omega(D(\text{SSD}_{n,k}))\).
- \(R(\text{SSD}_{n,k,m}) = \Omega(R(\text{SSD}_{n,k}))\).

Proof. We show that SSD\(_{n,k,m}\) reduces to SSD\(_{n,k}\); let \(x \in \{0,1\}^n\), \(y \in \{0,1\}^k\) be inputs to SSD\(_{n,k}\). Then Alice and Bob simply execute the protocol for SSD\(_{n,k,m}(x, y)\). It follows that lower bounds on the communication complexity of SSD\(_{n,k}\) are lower bounds for SSD\(_{n,k,m}\) as well.

2 Communication complexity

2.1 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 1 ([KN96]). For any function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$,
\[
D(f) \geq \log_2(\text{rank } M_f),
\]
where $\text{rank } M_f$ is equal to the number of linearly independent rows (or columns) of $M_f$.

Proposition 3. Recall that $\Sigma^{n \times k} := M_{\text{SSD}_{n,k,m}}$, with $\Sigma = \{0, 1, \ldots, m\}$. For all $n \geq k \geq 1$, $m \geq 1$,
\[
\text{rank}(\Sigma^{n \times k}) = (m + 1)^k
\]

Proof. For simplicity, we index the rows and columns of $\Sigma^{n \times k}$ lexicographically (i.e. by the sequence $[0^n, 0^{n-1}, \ldots, m^n]$). We claim the first $(m + 1)^k$ rows of $\Sigma^{n \times k}$ are a full-rank lower-triangular matrix.

Firstly, it is clear that a string $s \in \Sigma^k$ does not appear as a subsequence in $\Sigma^n$ until the $s$'th string, $0^{n-k}s$, where it appears as a contiguous subsequence. Thus
\[i < j \implies (\Sigma^{n \times k})_{i,j} = 0.\]
\[i = j \implies (\Sigma^{n \times k})_{i,j} = 1.\]

This completes the proof. \qed

Proposition 4. For all $n \geq k \geq 1$ and $m \geq 1$, under the natural bi-partition of inputs,
\[
D(\text{SSD}_{n,k,m}) = k \log(m + 1) + 1.
\]

Proof. Theorem [1] applied to Proposition [3] yields
\[
D(\text{SSD}_{n,k,m}) \geq \log \text{rank}(\Sigma^{n \times k}) = k \log(m + 1).
\]
It is easy to achieve this bound when Alice holds $x$ and Bob holds $y$; Bob sends Alice all $k$ characters of $y$, each requiring $\log(m + 1)$ bits. Alice then uses 1 bit to share the answer with Bob. \qed

In fact, the same bounds apply to $\text{SM}_{n,k,m}$, the contiguous string-matching problem, because $s$ appears contiguously in $0^{n-k}s$. Thus, we have the following corollary.

Corollary 1. $D(\text{SM}_{n,k,m}) = k \log(m + 1) + 1$.

We now lower bound the randomized communication complexity of this problem via a reduction from the indexing problem. Although the resulting lower bound follows directly from foundations [KN96] Lemma 3.8, we find the reduction useful in later sections.

Definition 5 (Indexing). Let $n > 0$. Given a binary string $x \in \{0, 1\}^n$ and index $i \in [n]$, we define the indexing function $\text{IND}_n : \{0, 1\}^n \times [n] \to \{0, 1\}$ as $\text{IND}_n(x, i) = x_i$.

Theorem 2 ([KNR99]). For all $k \geq 0$,
\[
R(\text{IND}_k) = \Omega(\log k).
\]

Proposition 5. $\text{IND}_k$ reduces to $\text{SSD}_{2k,k+1}$ in the following sense: if there is a communication protocol $Q$ for $\text{SSD}_{2k,k+1}$ for the natural partition, then there is a communication protocol for $\text{IND}_k$ for the natural partition with cost identical to that of $Q$. 

5
Definition 6. For $k \geq 1$, let Alice and Bob hold $x \in \{0, 1\}^k$ and $i \in [k]$ respectively. For a bit $b$, let $\overline{b} = 1 - b$. Without communication, the two players construct inputs to $SSD_{2k,k+1}$ as follows:

- Alice constructs $x' = \overline{x}_1x_1\overline{x}_2x_2 \cdots \overline{x}_kx_k$.
- Bob constructs $y' = 0^i1^{k-i+1}$.

Denote $\alpha_i = \overline{x}_1 \cdots \overline{x}_{i-1}x_{i-1}$, and $\beta_i = \overline{x}_{i+1}x_{i+1} \cdots \overline{x}_k \cdots x_k$. Note that $\alpha_i$ contains exactly $i - 1$ 0's, and $\beta_i$ contains exactly $k - i$ 1's. Now suppose $x_i = 1$. Then

$$x' = \alpha_i \cdot 01 \cdot \beta_i$$

Thus $y'$ is a subsequence of $x'$. Conversely, if $x_i = 0$, then

$$x' = \alpha_i \cdot 10 \cdot \beta_i.$$ 

2.2 Worst-case bi-partition

**Definition 6.** Define $DISJ^n$ as the problem of determining set disjointness. 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 $DISJ^n(a, b) \rightarrow \{0, 1\}$ 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 sets corresponding to $a, b$ are disjoint. We 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 $k$. When considering the communication complexity of $DISJ^k_n(a, b)$ we always assume Alice gets $a$ and Bob gets $b$.

**Theorem 3 ([Raz90], [KS92], [HW07]).** For all $n \geq 0$,

- $D(DISJ^k_n) = \Omega\left(\log\left(\frac{n}{k}\right)\right)$ for every $k \leq n/2$.
- $R(DISJ^k_n) = \Omega(k)$ for every $k \leq n/2$.

**Proposition 6.** For all $n \geq k \geq 1$, there is a bi-partition $B$ such that any communication protocol for $DISJ^k_n$ has cost no larger than the cost of the optimal protocol for $SSD_{3n,4k}$ under $B$.

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

- $y = 1010 \cdots 10 = (10)^{2k}$,
- $x = a_1b_1a_2b_20 \cdots a_nb_n0 = (a_ib_i0)_{1 \leq i \leq n}$. 

Thus $y'$ is a subsequence of $x'$. Conversely, if $x_i = 0$, then

$$x' = \alpha_i \cdot 10 \cdot \beta_i.$$ 

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

$$R(SSD_{n,k}) = \Omega(\log k)$$
This induces the bi-partition of inputs to SSD\(_{n,4k}\) 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.

Proposition 6 and Theorem 3 imply the following lower bounds for SSD\(_{n,k}\).

**Corollary 3.** There is a bi-partition of inputs such that

- \(D(SSD_{n,k}) = \Omega\left(\log \binom{n}{k}\right)\) for every \(k \leq n/2\).
- \(R(SSD_{n,k}) = \Omega(k)\) for every \(k \leq n/2\).

**Proposition 7.** Under any bi-partition of inputs,

\[ D(SSD_{n,k,m}) = O(k \log n). \]

**Proof.** The parties first exchange \(y\) requiring \(O(k \log (m+1))\) 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\(_{n-i,k-1,m}\) 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 \(D(SSD_{n,k,m}) = O(k \log (m+1) + k \log n) = O(k \log n)\) if we assume \(m < n\) as in Definition 1. This achieves the lower bound in Corollary 3, up to a difference of \(k\log k\), as

\[ \Omega\left(\log \binom{n}{k}\right) = \Omega\left(\log \binom{n}{k}^k\right) = \Omega(k \log n - k \log k). \]

---

### 3 VC dimension

For a finite set \(A\), a hypothesis class \(H\) is a collection of functions \(f : A \to \{0, 1\}\). Then, a subset \(B \subseteq A\) is **shattered** by \(H\) if for every subset \(B' \subseteq B\) there exists a function \(f \in H\) which “masks” \(B'\) within \(B\) (i.e. \(B \cap f^{-1}(1) = B'\)). The VC dimension of \(H\), denoted by \(\text{VCdim}(H)\), is the largest size of a subset of \(A\) that is shattered by \(H\). VC dimension essentially characterizes the number of samples needed to PAC-learn \(H\). For a detailed discussion of VC dimension and its relation to supervised learning we refer the reader to [SSBD14].

**Definition 7.** We denote \(H^k\) as the hypothesis class, parameterized by \(\{0, 1\}^k\), defined as follows: for \(\sigma \in \{0, 1\}^k\), the function \(h_\sigma \in H^k\) classifies a string \(s \in \{0, 1\}^n\) as 1 if \(\sigma\) is a subsequence of \(s\).

**Example 3.** For \(k = 2, 3, 4, 5\), and for \(n \leq 12\) we calculate (by brute force) the largest \(S \in \{0, 1\}^n\) that is shattered by \(H^k\).

| \(k\) | \(n\) | \(S \subseteq \{0, 1\}^n\) shattered by \(H^k\) |
|------|------|----------------------------------|
| 2    | 3    | 011, 001                         |
| 3    | 6    | 100001, 111000, 000111           |
| 4    | 5    | 10100, 10010, 01010              |
| 5    | 8    | 11000101, 01110010, 10011010, 10110011 |
To this end we build on the connection between the indexing function and SSD_{n,k}. We first need one more definition:

**Definition 8.** Let $B^n$ be a $2^n \times n$ matrix, with $B^n_{ij}$ corresponding to the $j$'th bit of the (left-padded) binary number $i - 1$.

**Example 4.**

\[
B^3 = \begin{bmatrix}
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
\end{bmatrix}^T.
\]

**Proposition 8.** For all $k > 1$,

\[
\lceil \log_2(k-1) \rceil \leq \text{VCdim}(H^k) \leq k.
\]

**Proof.** **Upper bound:** Let $S \subset \{0, 1\}^n$ be shattered by $H^k$. Then there must be a surjective mapping $f : H^k \to 2^S$. Surjectivity requires the domain to be at least as large as the range;

\[
|2^S| \leq |H^k| \implies 2^{|S|} \leq 2^k \implies |S| \leq k.
\]

Therefore the VC dimension of $H^k$ is bounded above by $k$.

**Lower bound:** Let the set $S$ contain the strings corresponding to the columns of $B^{\lfloor \log k \rfloor}$. Now observe that the $i$'th row of $B^{\lfloor \log k \rfloor}$ is equal to the characteristic vector of the subset containing all strings with $i$'th bit equal to 1, which we denote as $S_i := \{ s \in S : s_i = 1 \}$. Furthermore, the collection of all rows $\{S_i : 1 \leq i \leq 2^{\lfloor \log k \rfloor} \} = 2^S$. Indeed, all rows are distinct and there are $2^{|S|}$ rows. Finally, we map each $s \in S$ to a new string $t(s)$ with length twice that of $s$, yielding the set

\[
S' = \{ t(s) := \overline{s}_1 s_1 \overline{s}_2 s_2 \cdots \overline{s}_k s_k : s \in S \} \subset \{0, 1\}^{2k}.
\]

We now consider the set containing $k$ classifiers, $G = \{ g_i := 0^{i-1} 1^{k-i+1}, 1 \leq i \leq k \} \subset H^{k+1}$. As we saw in the proof of Proposition 5, the string $g_i \in G$ is a subsequence of $t(s) \in S'$ if and only if $t(s)_{2i} = s_i = 1$. Thus, $S'$ is shattered by $G$. Since $|S'| = \lceil \log_2 k \rceil$, this completes the proof. 

4 **Future directions**

There are several questions arising from this work. What is the randomized communication complexity of SSD_{n,k}? We suspect an upper bound of $O(k)$ is achievable no matter how $x, y$ are partitioned between the players. We have an exponential gap between our lower and upper bounds for VC dimension for the set of classifiers defined by containing a fixed length $k$ subsequence. Clos- ing this gap remains an interesting question. Although we were unable to obtain stronger bounds, we suspect $\text{VCdim}(H^k) = o(k)$.

5 **Acknowledgements**

We wish to thank the anonymous reviewers for providing helpful feedback, as well as for bringing [BC18] to our attention.
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.

[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.

[BYJK04] 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.

[DS03] Miroslav Dudik 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.

[FY19] 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.

[FGRS04] Abraham Flaxman, Aram W Harrow, and Gregory B Sorkin. Strings with maximally many distinct subsequences and substrings. *the electronic journal of combinatorics*, 11(1):R8, 2004.

[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.

[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.

[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.

[KLHLV07] 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.

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

[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.

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