Finding Subcube Heavy Hitters in Data Streams

Branislav Kveton  
Adobe Research  
kveton@adobe.com

S. Muthukrishnan  
Rutgers University  
muthu@cs.rutgers.edu

Hoa T. Vu  
University of Massachusetts  
hvu@cs.umass.edu

Abstract

Data streams typically have items of large number of dimensions. We study the fundamental heavy-hitters problem in this setting. Formally, the data stream consists of $x_1, \ldots, x_m$ where each $x_i = (x_{i,1}, \ldots, x_{i,d})$ is a $d$-dimensional item, and each $x_{i,j} \in [n]$. A $k$-dimensional subcube $T$ is a subset of distinct coordinates $\{T_1, \ldots, T_k\} \subseteq [d]$. A subcube heavy hitter query $\text{Query}(T, v)$, $v \in [n]^k$, outputs YES if $f_T(v) \geq \gamma$ and NO if $f_T(v) < \gamma/4$, where $f_T$ is the ratio of number of stream items whose coordinates $T$ have joint values $v$ and $\gamma$ is some suitable constant for a heavy hitter. The all subcube heavy hitters query $\text{AllQuery}(T)$ outputs all joint values $v$ that return YES to $\text{Query}(T, v)$. The one dimensional version of this problem where $d = 1$ was heavily studied in data stream theory, databases, networking and signal processing. The subcube heavy hitters problem is applicable in all these cases.

We present a simple reservoir sampling based one-pass streaming algorithm to solve the subcube heavy hitters problem in $\tilde{O}(kd/\gamma)$ space. This is optimal up to poly-logarithmic factors given the $\Omega(kd/\gamma)$ lower bound for the Frequent Itemset problem in data mining [9] which is a special case of our problem. In the worst case, this is $\Theta(d^2/\gamma)$ which is prohibitive for large $d$, and our goal is to circumvent this quadratic bottleneck.

Our main contribution is a model-based approach to the subcube heavy hitters problem. In particular, we assume that the dimensions are related to each other via the Naive Bayes model, with or without a latent dimension. We present a new two-pass, $\tilde{O}(d/\gamma)$-space algorithm for our problem, and a fast algorithm for answering $\text{AllQuery}(T)$ in $O(k/\gamma^2)$ time. Our work develops the direction of model-based data stream analysis, with much that remains to be explored.

\textsuperscript{1}$\tilde{O}(\cdot)$ suppresses polylog factors.
1 Introduction

Data streams typically have a large number of dimensions. Most companies see transactions with items sold, time, store location, price, and so on that arrive over time. Modern online companies see user activities online on web over time, which typically have components of user information including ID (e.g., cookies), hardware (e.g., device), software (such as browser, OS), content (such as web properties, apps), events (impressions, views, clicks, purchases), event attributes (such as product id, price, geolocation, time) etc. Even classical IP traffic streams have many dimensions including source and destination IP addresses, port numbers and other features of an IP connection such as application type and other features. Furthermore, in applications such as Natural Language Processing, streams of documents can be thought of as streams of large number of bigrams or multi-grams over word combinations [5]. As these examples show, streams of data have 100’s and 1000’s of dimensions in many applications. Motivated by this, we study the problem of finding heavy hitters on data streams focusing on d, the number of dimensions, as a parameter. Given d one sees in practice, $d^2$ in space usage is prohibitive, for solving the heavy hitters problem on such streams.

Formally, let us start with a one-dimensional stream $x_1, \ldots, x_m$ of items where each $x_i \in [n] := \{1, 2, \ldots, n\}$. We can look at the count $c(y) = |\{i : x_i = y\}|$ or frequency ratio $f(y) = \frac{c(y)}{m}$. A heavy hitter item $y$ is one with $c(y) \geq \gamma m$ or equivalently $f(y) \geq \gamma$, for some constant $\gamma$. The standard data stream model is that we maintain data structures of size polylog($m,n$) and determin if $y$ is a heavy hitter with probability of success at least 3/4, that is, if $f(y) \geq \gamma$ output YES and output NO if $f(y) < \gamma/4$ for all $y$\(^2\) We note that if $\gamma/4 \leq f(y) < \gamma$, then either answer is acceptable.

Detecting heavy hitters on data streams is a fundamental problem in data stream theory that arises in guises such as finding elephant flows and network attacks in networking, finding hot trends in databases, finding frequent patterns in data mining, finding largest coefficients in signal analysis, and so on. Therefore, this heavy hitter problem has been studied extensively in theory, databases, networking and signal processing literature under various models (cash-register or turnstile streams, windowed streaming, random order streams) and variations (identify all heavy hitters versus identify the query heavy hitter or estimate frequency of each heavy hitter). See [2] for an early survey and [14] for a recent survey.

Subcube Heavy Hitter Problems. Our focus is on modern data streams with $d$ dimensions, for large $d$. Thus, the data stream consists of $x_1, \ldots, x_m$ where each $x_i = (x_{i,1}, \ldots, x_{i,d})$ is a $d$-dimensional item, for $i = 1, 2, \ldots, m$ and each $x_{i,j} \in [n]$. The number of items whose $j$th coordinate has value $y$ is denoted by $c_j(y)$, i.e., $c_j(y) = |\{i : x_{i,j} = y\}|$. We define the random variable $X_j$ as $j$th coordinate’s value of a random data stream item. Therefore, $$f_j(y) = \Pr[X_j = y] = \frac{c_j(y)}{m}.$$ A $k$-dimensional subcube $T$ is a subset of distinct coordinates $\{T_1, \ldots, T_k\} \subseteq [d]$ where $T_1 < T_2 < \cdots < T_k$. We say $\dim(T) = k$ and use $X_T$ to denote the random variable of the joint values of the coordinates $T$ of a random item.

We refer to the joint values of the coordinates $T$ of $x_i$ as $x_{i,T}$. Suppose $v = (v_1, \ldots, v_k) \in [n]^k$, i.e., $v$ is indexed as $[n] \times [n] \times \cdots \times [n]$, we write $X_T = v$ to denote $X_{T_1} = v_1, \ldots, X_{T_k} = v_k$. In the same fashion, we define $c_T(v) := |\{i : x_{i,T} = v\}|$ and $f_T(v) := c_T(v)/m = \Pr[X_T = v]$.

We define our problems. They take $k, \gamma$ as parameters and the stream as the input and build data structures to answer:

- **Subcube Heavy Hitter.** Query($T, v$), where $\dim(T) = k$, $v \in [n]^k$, returns an estimate if $f_T(v) \geq \gamma$. In particular, output YES if $f_T(v) \geq \gamma$ and NO if $f_T(v) < \gamma/4$. If $\gamma/4 \leq f_T(v) < \gamma$, then either output is acceptable. The required success probability for all $T \in \binom{[d]}{k}$ and $v \in [n]^k$ is at least $3/4$.

- **All Subcube Heavy Hitters.** AllQuery($T$), where $\dim(T) = k$, outputs all joint values $v$ that return YES to Query($T, v$). This is conditioned on the algorithm used for Query($T, v$).

Subcube heavy hitters are relevant wherever one dimensional heavy hitters have found applications: combination of source and destination IP addresses forms the subcube heavy hitter that might detect network attacks;\(^2\)The error can be narrowed to any $\epsilon$ and success probability can be amplified to $1 - \delta$ for any parameters $\alpha, \epsilon$ with cost $1/\epsilon, \log(1/\delta)$ as needed, and we omit these factors in the discussions.
combination of stores, sales quarters and nature of products forms the subcube heavy hitter that might be the pattern of interest in the data, etc. Arguably, given the multiple dimensions in most data streams, subcube heavy hitters may limit the significant data properties far more than the single dimensional view.

Related Work. The problem we address is directly related to frequent itemset mining studied in the data mining community. In frequent itemset mining, each dimension is binary \((n = 2)\), and we consider Query\((T, v)\) where \(v = (1, \ldots, 1) := U_k\). It is known that counting all maximal subcubes \(T\) that have a frequent itemset \(f_r(U_k) \geq \gamma\) is \#P-complete [15]. Furthermore, finding even a single \(T\) of maximal size such that \(f_r(U_k) \geq \gamma\) is NP-hard [8, 9]. Recently, Liberty et al. showed that any constant-pass streaming algorithm answering Query\((T, U_k)\) needs to use \(\Omega(kd/\gamma \cdot \log(d/k))\) bits of memory [9]. In the worst case, this is \(\Omega(d^2/\gamma)\) for large \(k\), ignoring the poly-logarithmic factors. For this specific problem, sampling algorithms will nearly meet their lower bound for space. Our problem is more general, with arbitrary \(n\) and \(v\).

Our Contributions. Clearly, the case \(k = 1\) can be solved by building one of the many known single dimensional data structures for the heavy hitters problem on each of the \(d\) dimension; the \(k = d\) case can be thought of as a giant single dimensional problem by linearizing the space of all values in \([n]^k\); for any other \(k\), there are \(\binom{d}{k}\) distinct choices for subcube \(T\), and these could be treated as separate one-dimensional problems by linearizing each of the subcubes. In general, this entails \(\binom{d}{k}\) and \(\log(n^d)\) cost in space or time bounds over the one-dimensional case, which we seek to avoid. Also, our problem can be reduced to the binary case by unary encoding each dimension by \(n\) bits, and solving frequent itemset mining: the query then has \(kn\) dimensions. The resulting bound will have an additional \(n\) factor which is large.

First, we observe that the well known reservoir sampling method [13] solves subcube heavy hitters problems for arbitrary dimensions and query vectors \(v\). Our analysis shows that the space we use is within poly-logarithmic factors of the lower bound shown in [9] for binary dimensions and query vector \(U_{k}\), which is a special case of our problem. Therefore, the subcube heavy hitters problem can be solved using \(O(d^2)\) space. However, this is \(\Omega(d^2)\) in worst case.

Second, our main contribution is to avoid this quadratic bottleneck for finding subcube heavy hitters. We adopt the notion that there is an underlying probabilistic model behind the data, and in the spirit of the Naive Bayes model, we assume that the dimensions are nearly (not exactly) mutually independent given an observable latent dimension. This could be considered as a low rank factorization of the dimensions. In particular, one could formalize this assumption by bounding the total variational distance between the data’s joint distribution and that derived from the Naive Bayes estimation. This assumption is common in statistical data analysis and highly prevalent in machine learning. Following this modeling, we make two main contributions:

- We present a two-pass, \(\tilde{O}(d/\gamma)\)-space streaming algorithm for answering Query\((T, v)\). This improves upon the \(kd\) factor in the space from sampling, without assumptions, to just \(d\) with the Naive Bayes assumption, which would make this algorithm quite practical for large \(k\). Our algorithm uses sketching in each dimension in one pass to detect heavy hitters, and then needs a second pass to precisely estimate the frequencies of the heavy hitters to compute the overall joint probability accurately.

- We present a fast algorithm for answering AllQuery\((T)\) in \(O(k/\gamma^2)\) time. The naive procedure would take exponential time \(\Omega((1/\gamma)^k)\) by considering the Cartesian product of the heavy hitters in each dimension. Our approach, on the other hand, uses the structure of the subcube heavy hitters to iteratively construct them one dimension at a time.

Our work develops the direction of model-based data stream analysis. Model-based data analysis has been effective in other areas. For example, in compressed sensing, realistic signal models that include dependencies between values and locations of the signal coefficients improve upon unconstrained cases [11]. In statistics, using tree constrained models of multidimensional data sometimes improves point and density estimation. Recently, McGregor and Vu [11] studied the problem of evaluating Bayesian Networks in data streams. In another work, Kveton et al. [7] assumed a tree graphical model and designed a one-pass algorithm that estimates the joint frequency; their work however only solved the \(k = d\) case for the joint frequency estimation problem. Our model is a bit different and more importantly, we solve the subcube heavy hitters problem (addressing all the \(\binom{d}{k}\) subcubes). In following such a direction, we have extended the fundamental heavy

2
hitters problem to higher dimensional data. Given that many implementations already exist for the sketches we use for one-dimensional heavy hitters as a blackbox, our algorithms are therefore easily implementable.

**Background on the Naive Bayes model and its use in our context.** The Naive Bayes Model is a Bayesian network over \( d \) features \( X_1, \ldots, X_d \) and the class variable \( Y \). This model represents a joint probability distribution of the form

\[
Pr [X_1 = x_1, \ldots, X_d = x_d, Y = y] = Pr [Y = y] \prod_{j=1}^{d} Pr [X_j = x_j | Y = y],
\]

which means that the values of the features are conditionally independent given the value of the class variable. The simplicity of the Naive Bayes model makes it a popular choice in text processing and information retrieval, with state-of-the-art performance in spam filtering, text classification and others.

For the purposes of the discussion in this paper, we focus on text classification where the features are individual words in text, which is also known as the bag-of-words model; and the class label summarizes one aspect of the text, such as its topic of the text or the content type. Therefore, the features are discrete variables of a very high cardinality and the class is the topic of the text. The number of topics is typically on the order of tens. Many probabilistic queries can be of interest in the Naive Bayes model. Given such streams of text viewed as streams of features and class labels, an example query of interest is

\[
\{ (x_{i_1}, x_{i_2}) : Pr [X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}] \geq \varepsilon \}
\]

all bi-grams over positions \( i_1 \) and \( i_2 \) that appear frequently in the text. Such a query can, for instance, help in understanding the generative aspect of the model, which sequences of words are likely to be generated by the model. In this work, we show how to answer such a query in a small space for \( k \)-grams of any subset of up to \( d \) features.

### 2 The Sampling Approach

In this section, we show that sampling solves the problem efficiently compared to running one-dimensional heavy hitters for all \( \binom{n}{k} \) subcubes independently. Furthermore, this upper bound matches the lower bound in \cite{Guha et al.} up to poly-logarithmic factors.

The algorithm samples \( m' = O(\gamma^{-1}kd) \) random items \( z_1, \ldots, z_{m'} \) from the stream using Reservoir sampling. Let \( S = \{z_1, \ldots, z_{m'}\} \) be the sample set. Given Query \((T, v)\), output YES if and only if the sample frequency of \( v \), denoted by \( \hat{f}_T(v) \), is at least \( \gamma/2 \). Specifically,

\[
\hat{f}_T(v) := \frac{|\{x_i : x_i \in S \text{ and } x_{i,T} = v\}|}{m'}.
\]

See the algorithm in Figure \ref{fig:reservoir}. For all subcubes \( T \) and joint values \( v \) of \( T \), the expected sample frequency \( \hat{f}_T(v) \) is \( f_T(v) \). If \( v \) is a frequent joint values, then according to Chernoff bound, its sample frequency \( f_T(v) \approx \hat{f}_T(v) \) with high probability; otherwise, \( \hat{f}_T(v) \) stays low.

Let us fix a \( k \)-dimensional subcube \( T \) and suppose that for all \( v \in [n]^k \), we have

\[
\hat{f}_T(v) = f_T(v) \pm \frac{\max\{\gamma, f_T(v)\}}{10}.
\]

It is then straightforward to see that if \( f_T(v) \leq \gamma/4 \), then \( \hat{f}_T(v) < \gamma/2 \) and if \( f_T(v) \geq \gamma \), then \( \hat{f}_T(v) \geq \gamma/2 \). As a result, one could output YES for all \( v \) where \( \hat{f}_T(v) \geq \gamma/2 \), and output NO otherwise.

We continue with our analysis as follows. A set of boolean random variables \( Z_1, \ldots, Z_t \) are negatively correlated if and only if for an arbitrary \( Z_i \) and an arbitrary subset \( S \subseteq [t] \setminus \{i\} \),

\[
Pr [Z_i = 1 | Z_j = 1 \text{ for all } j \in S] \leq Pr [Z_i = 1].
\]

We shall rely on the following Chernoff bound.
(1) Sample $m' = \tilde{O}(\gamma^{-1}kd)$ random items $S = \{z_1, \ldots, z_{m'}\}$.

(2) Output YES to Query($T, v$) if and only if $\hat{f}_T(v) \geq \gamma/2$.

Figure 1: Sampling algorithm

**Lemma 1.** (Chernoff bound for negatively correlated variables) Let $X_1, \ldots, X_n$ be negatively correlated boolean random variables. Let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. Then,

$$\Pr[|X - \mu| \geq \epsilon \mu] \leq 3 \exp(-\epsilon^2 \mu/3).$$

Recall that $S = \{z_1, z_2, \ldots, z_{m'}\}$ is the sample set returned by the algorithm. For a fixed $v \in [n]^k$, let the indicator random variables $Z_i$ denote the event $z_{i,T} = v$. We observe that the random variables $Z_i$ are negatively correlated. This is because

$$\Pr[\{Z_i = 1\} \mid Z_j = 1 \text{ for all } j \in S] = \frac{c_T(v) - |S|}{m - |S|} \leq \frac{c_T(v)}{m} = f_T(v).$$

**Lemma 2.** For all $k$-dimensional subcubes $T$ and joint values $v \in [n]^k$, with probability at least 0.9,

$$\hat{f}_T(v) = f_T(v) \pm \frac{\max(\gamma, f_T(v))}{10}.$$

**Proof.** Let $m' = c\gamma^{-1}\log(d^k \cdot n^k)$ for some sufficiently large constant $c$. We first consider a fixed $v \in [n]^k$ and define the random variables $Z_i$ as above. Specifically, $Z_i$ is the indicator variable for the event that $v$ is the joint values of the coordinates $T$ of the $i$th item in the sample set $S$, i.e., $Z_i = 1$ if $z_{i,T} = v$. Suppose $f_T(v) \geq \gamma$, by appealing to Lemma 1 we have

$$\Pr\left[\left|\frac{1}{m'} \sum_{i=1}^{m'} Z_i - f_T(v)\right| \geq \frac{f_T(v)}{10}\right] \leq 3 \exp\left(-\frac{f_T(v)m'}{300}\right) \leq \frac{1}{10d^k n^k}.$$

Next, suppose $f_T(v) < \gamma/4$, we have

$$\Pr\left[\left|f_T(v) - f_T(v)\right| \geq \frac{2\gamma}{10}\right] \leq 3 \exp\left(-\left(\frac{\gamma}{10f_T(v)}\right)^2 f_T(v)m'\right) \leq \frac{1}{10d^k n^k}.$$

Therefore, by taking the union bound over all $\frac{d^k}{k} \cdot \binom{n^k}{k} \leq d^k \cdot n^k$ possible combinations of $k$-dimensional subcubes and the corresponding joint values $v$, we deduce the claim.

Since we have proven the desired property of $\hat{f}_T(v)$, we could answer all Query($T, v$) correctly with probability at least 0.9 for all $v \in [n]^k$ and $k$-dimensional subcube $T$. Because storing each sample $z_i$ requires $\tilde{O}(d)$ bits of space, the total space use is $\tilde{O}(dk\gamma^{-1})$. We summarize the result as the follows.

**Theorem 3.** There exists a single-pass algorithm that uses $\tilde{O}(dk\gamma^{-1})$ space and solves $k$-dimensional subcube heavy hitters. The time to answer Query($T, v$) and AllQuery($T$) are $\tilde{O}(dk\gamma^{-1})$ and $\tilde{O}((dk\gamma^{-1})^2)$ respectively.
3 Naive Bayes Approach: The Near-Independence Case

Suppose the random variables representing the dimensions are near independent. We show that there is a 2-pass algorithm that uses less space and has faster query time. At a high level, we make the assumption that the joint probability is approximately factorized for all subcubes \( T \subseteq [n] \):

\[
f_T(v) \approx f_{T_1}(v_1)f_{T_2}(v_2)\cdots f_{T_k}(v_k).
\]

More formally, we assume that the total variation distance is bounded by a small quantity \( \alpha \) as follows. Suppose, \( \dim(T) = h \), then

\[
\sum_{v \in [n]^h} \left| f_T(v) - \prod_{i=1}^h f_{T_i}(v_i) \right| < \alpha/2
\]

which is equivalent to

\[
\max_{v \in [n]^h} \left| f_T(v) - \prod_{i=1}^h f_{T_i}(v_i) \right| < \alpha.
\]

We refer to the above as the near-independence assumption. Furthermore, we assume that \( \alpha \) is reasonable with respect to \( \gamma \) that controls the heavy hitters. For example, \( \alpha < \gamma/10 \) will suffice.

We observe that if \( f_T(v) \geq \gamma \), then \( \prod_{i=1}^h f_{T_i}(v_i) \geq \gamma - \gamma/10 \geq \gamma/2 \) and if \( f_T(v) < \gamma/4 \), then \( \prod_{i=1}^h f_{T_i}(v_i) < \gamma/4 + \gamma/10 \leq \gamma/2 \). We therefore want to output YES to Query(\( T, v \)) if and only if \( \prod_{i=1}^h f_{T_i}(v_i) \geq \gamma/2 \). For convenience, we define \( \lambda := \gamma/2 \). We make following simple but useful observation.

**Claim 4.** We have that

\[
\prod_{i=1}^h f_{T_i}(v_i) \geq \lambda \quad \text{implies} \quad \prod_{i \in \mathcal{V}} f_{T_i}(v_i) \geq \lambda
\]

for all \( \mathcal{V} \subseteq [h] \) (in other words, \( \{T_i : i \in \mathcal{V}\} \) is a subcube of \( T \)).

Hence, if \( \prod_{i=1}^h f_{T_i}(v_i) \geq \lambda \), then it must be the case that \( f_{T_1}(v_1) \geq \lambda \), \( f_{T_2}(v_2) \geq \lambda \), and so on. To this end, by using (for example) the Count-Min sketch \[3\], with high probability, we could find a set \( H_i \) such that if \( f_i(x) \geq \lambda/2 \), then \( x \in H_i \) and if \( f_i(x) < \lambda/4 \), then \( x \notin H_i \). And by taking another pass over the stream, for each \( x \in H_i \), we compute \( f_i(x) \) exactly to obtain

\[
S_i := \{ x \in [n] : f_i(x) \geq \lambda \}
\]

for all \( i \in [n] \).

Appealing to Claim \[4\] if \( \prod_{i=1}^k f_{T_i}(v_i) \geq \lambda \), then \( v_i \in S_i \) for all \( i \). Hence, we output YES to Query(\( T, v \)) if and only if all \( v_i \in S_{T_i} \) and \( \prod_{i=1}^k f_{T_i}(v_i) \geq \lambda \). See the algorithm in Figure \[2\].

| 1st pass: | 2nd pass: | 3rd pass: |
|-----------|-----------|-----------|
| (a) For each coordinate \( i \in [d] \), use the Count-Min sketch to find \( H_i \). | (a) For each \( i \in [d] \), compute \( f_i(x) \) exactly for all \( x \in H_i \) to obtain \( S_i \). | (3) Output YES to Query(\( T, v \)) if and only if all \( v_i \in S_{T_i} \) and \( \prod_{i=1}^k f_{T_i}(v_i) \geq \lambda \). |

Figure 2: Subcube heavy hitters algorithm under the near-independence assumption
Theorem 5. Assuming $\alpha \leq \gamma/10$, there exists a two-pass algorithm that uses $\tilde{O}(d\gamma^{-1})$ space and solves subcube heavy hitters under the near-independence assumption. If $\dim(T) = k$, then the time to answer $\text{Query}(T,v)$ and $\text{AllQuery}(T)$ are $\tilde{O}(k)$ and $O(k\gamma^{-1})$ respectively.

Proof. The first pass uses $\tilde{O}(d\gamma^{-1})$ space since the Count-Min data structure uses $\tilde{O}(\lambda^{-1})$ space for each coordinate $i \in [d]$. We then observe that the size of each $H_i$ is $O(\lambda^{-1})$ which means that the second pass also uses $\tilde{O}(d\gamma^{-1})$ space.

We already established correctness of the algorithm for an arbitrary $\text{Query}(T,v)$ based on the near-independence assumption above. Next, we exhibit a fast algorithm to answer $\text{AllQuery}(T)$. We note that naively checking all combinations $v_1, \ldots, v_k \in S_{T_1} \times S_{T_2} \times \cdots \times S_{T_k}$ takes exponential $\Omega(\lambda^{-k})$ time in the worst case.

Our approach works out the frequent joint values gradually and takes advantage of the near-independence assumption. In particular, define

$$W_j := \{v \in [n]^j : f_{T_1}(v_1) \cdots f_{T_j}(v_j) \geq \lambda\}.$$ 

Then, the goal becomes finding $W_k$. Note that $W_1 = S_1$ is obtained directly by the algorithm. Suppose we have $W_j$. We shall show that it is possible to construct $W_{j+1}$ in $\tilde{O}(\lambda^{-1})$ time. For convenience, we define $T_{1,j} := \{T_1, \ldots, T_j\}$ and $v_{1,j} = (v_1, v_2, \ldots, v_j)$.

We note that $|W_j| \leq 5/4 \cdot \lambda^{-1}$ because if $\prod_{i=1}^j f_{T_i}(v_j) \geq \lambda$ then $f_{T_{1,j}}(v_{1,j}) \geq \lambda - \alpha \geq 4/5 \cdot \lambda$ according to the near-independence assumption. For each $(v_1, \ldots, v_j) \in W_j$, we collect all $v_{j+1} \in S_{j+1}$ such that

$$f_{T_{j+1}}(v_{j+1}) \geq \frac{\lambda}{\prod_{i=1}^j f_{T_i}(v_i)}$$

and put $(v_1, \ldots, v_{j+1})$ into $W_{j+1}$. Since $|W_j| \leq 5/4 \cdot \lambda^{-1}$ and $|S_{j+1}| \leq \lambda^{-1}$, this step obviously takes $O(\lambda^{-2})$ time. However, a tighter analysis yields $\tilde{O}(\lambda^{-1})$ time by observing that there could be at most $\lambda^{-1} \prod_{i=1}^j f_{T_i}(v_i)$ such $v_{j+1}$ for each $(v_1, \ldots, v_j) \in W_j$. As a result, the time complexity of this step is

$$\tilde{O}\left(\sum_{v' \in W_j} \lambda^{-1} \prod_{i=1}^j f_{T_i}(v'_i)\right) = \tilde{O}\left(\sum_{v' \in W_j} \lambda^{-1}(f_{T_{1,j}}(v') + \alpha)\right) = \tilde{O}(\lambda^{-1}).$$

Thus, we could find $W_{j+1}$ given $W_j$ in $\tilde{O}(\lambda^{-1})$ time and therefore obtain $W_k$ in $\tilde{O}(k\lambda^{-1}) = \tilde{O}(k\gamma^{-1})$ time. The correctness of this procedure follows directly from Claim since $(v_1, \ldots, v_{j+1}) \in W_{j+1}$ implies that $(v_1, \ldots, v_j) \in W_j$ and $v_{j+1} \in S_{j+1}$. Thus, by checking all combinations of $(v_1, \ldots, v_j) \in W_j$ and $v_{j+1} \in S_{j+1}$, we ensure to completely construct $W_{j+1}$. \hfill \Box

4 The Naive Bayes Model: The General Case

In this final section, we focus on the data stream inspired by the Naive Bayes model which is strictly more general than the near-independence assumption. In particular, we assume that the coordinates are near-independent given an extra $(d+1)^{th}$ observable coordinate that has a value range $1, \ldots, \ell$. As in typical Naive Bayes analysis, we assume $\ell$ is a constant but perform the calculations in terms of $\ell$ so it’s role in the complexity of the problem is apparent. This model asserts that for all subcubes $T$ where $\dim(T) = h$, joint values $v \in [n]^h$, and $z \in [\ell]$,

$$f_{T \cup \{d+1\}}((v,z)) \approx f_{T_1 | d+1}(v_1 | z) \cdots f_{T_h | d+1}(v_d | z) f_{d+1}(z)$$

where $f_{T_i | d+1}(v_i | z)$ is defined as the conditional probability $Pr [X_i = v_i | X_{d+1} = z]$. Marginalizing over $z$ gives

$$f_T(v) \approx \sum_{z \in [\ell]} f_{T_1 | d+1}(v_1 | z) \cdots f_{T_h | d+1}(v_d | z) f_{d+1}(z).$$

\footnote{This requires sorting $W_{j+1}$ based on the frequency.}
The \((d + 1)\)th dimension is often referred to as the *latent dimension*. Similar to the previous section, we formalize our model-based assumption as follows. For all \(h\)-dimensional subcubes \(T \subseteq [n]\),
\[
\max_{v \in [n]^h} \left| f_T(v) - \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^h f_{T_i | d+1}(v_i | z) \right| < \alpha
\]
where we again assume \(\alpha < \gamma/10\). As argued in the previous section, it suffices to output YES to Query\((T, v)\) if and only if
\[
\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^h f_{T_i | d+1}(v_i | z) \geq \lambda.
\]

(1) 1st pass:
(a) For each value \(z \in [\ell]\), compute \(f_{d+1}(z)\) exactly.
(b) For each coordinate \(i \in [d]\), use the Count-Min sketch to find the set \(H_i\).

(2) 2nd pass:
(a) For each \(i \in [d]\), compute \(f_i(x)\) exactly for all \(x \in H_i\) and obtain \(S_i\).
(b) For each value \(z \in [\ell]\), each dimension \(i \in [d]\) and \(x \in H_i\), compute \(f_i | d+1(x | z)\) exactly.

(3) Output YES to Query\((T, v)\) if and only if all \(v_i \in S_{T_i}\) and
\[
f_T(v) := \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^k f_{T_i | d+1}(v_i | z) \geq \lambda.
\]

Figure 3: Subcube heavy hitters algorithm under the Naive Bayes assumption

First, we generalize Claim 4 as follows.

**Claim 6.** *For all \(k\)-dimensional subcubes \(T\), we have*
\[
\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^k f_{T_i | d+1}(v_i | z) \geq \lambda
\]

*implies that for all subcubes \(T'\) of \(T\),*
\[
\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in T'} f_{T_i | d+1}(v_i | z) \geq \lambda.
\]

**Proof.** For a fixed \(z\), observe that \(\sum_{v_j \in [n]} f_{T_j | d+1}(v_j | z) = 1\). We simply rewrite
\[
\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in T'} f_{T_i | d+1}(v_i | z) = \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in T'} f_{T_i | d+1}(v_i | z) \cdot \prod_{j \notin T'} f_{T_j | d+1}(y_j | z) \geq \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^k f_{T_i | d+1}(v_i | z) \geq \lambda.
\]

The last inequality follows by considering the term where \(y_j = v_j\). □
We now demonstrate how to derive \( f_i \) with \( \alpha \) assuming Theorem 7. The correctness of this procedure follows directly from Claim 6 since \( \sum_{z \in [d]} f_{T_i | d+1}(v_i | z) \geq \lambda \) for all \( i = 1, 2, \ldots, k \) which in turn implies that \( v_i \in S_{T_i} \).

Therefore, it becomes apparent that we output YES to Query\((T, v)\) if and only if all \( v_i \in S_{T_i} \) and

\[
\sum_{z \in [d]} f_{T_i | d+1}(v_i | z) \geq \lambda.
\]

To this end, we need to compute \( f_{i | d+1}(x | z) \) and \( f_{d+1}(z) \) for all \( x \in S_i \) and for all \( z \in [d] \). See the algorithm in Figure 7.

**Theorem 7.** Assuming \( \alpha \leq \gamma/10 \), there exists a two-pass algorithm that uses \( \tilde{O}(kd\gamma^{-1}) \) space and solves subcube heavy hitters under the Naive Bayes assumption. The time to answer Query\((T, v)\) and AllQuery\((T)\) are \( O(k) \) and \( O(kd\gamma^{-2}) \) respectively.

**Proof.** The total space to obtain \( H_i \) and \( S_i \) over the two passes is \( \tilde{O}(d\lambda^{-1}) \). Additionally, storing different \( f_{i | d+1}(\cdot) \) requires \( \tilde{O}(kd\lambda^{-1}) \) bits of space. Therefore, the overall space complexity is \( O(d\lambda^{-1}) \).

We already established the correctness of answering Query\((T, v)\) above. Next, we exhibit a fast post-processing time algorithm to answer AllQuery\((T)\). Define \( W_j := \{ v \in [n]^j : \sum_{z \in [d]} f_{T_i | d+1}(v_i | z) \geq \lambda \} \).

We need to find \( W_k \) and note that \( W_1 = S_1 \) is obtained directly by the algorithm. Suppose we have \( W_j \). We now demonstrate how to derive \( W_{j+1} \) in \( O(\lambda^{-2}) \) time.

Observe that \( |W_j| = 5/4 \cdot \lambda^{-1} \) because if

\[
\sum_{z \in [d]} f_{T_i | d+1}(v_i | z) \geq \lambda
\]

then \( f_{T_j}(v_j) \geq \lambda - \alpha = 4/5 \cdot \lambda^{-1} \) according to the Naive Bayes assumption. For each \( (v_1, \ldots, v_j) \in W_j \), we collect all \( v_{j+1} \in S_{j+1} \) such that

\[
\sum_{z \in [d]} f_{T_i | d+1}(v_i | z) \geq \lambda
\]

and put \( (v_1, \ldots, v_{j+1}) \) to \( W_{j+1} \). Since \( |W_j| \leq 5/4 \cdot \lambda^{-1} \) and \( |S_{j+1}| \leq \lambda^{-1} \), this step obviously takes \( O(\lambda^{-2}) \) time. We achieve \( W_{j+1} \) given \( W_j \) in \( O(\lambda^{-2}) \) time and therefore find \( W_k \) in \( O(kd\lambda^{-2}) = O(kd\gamma^{-2}) \) time. The correctness of this procedure follows directly from Claim 6 since \( (v_1, \ldots, v_{j+1}) \in W_{j+1} \) implies that \( (v_1, \ldots, v_j) \in W_j \) and \( v_{j+1} \in S_{j+1} \). Thus, by checking all combinations of \( (v_1, \ldots, v_j) \in W_j \) and \( v_{j+1} \in S_{j+1} \), we ensure to completely construct \( W_{j+1} \).

**5 Concluding Remarks**

Data streams have very large dimensions. So, we study the fundamental problem of heavy hitters of size \( \gamma \) with \( d \), the number of dimensions of the items in the stream, as a parameter. We defined subspace heavy hitter problems with subspace of size \( k, k \leq d \). A special case of this is frequent itemset mining for which a lower bound of \( \Omega(kd/\gamma) \) is known 7 on space needed, ignoring polylog factors. We showed a simple reservoir sampling based algorithm that meets this bound, upto polylog factors. But the focus of our work is to avoid this quadratic dependence on the number of dimensions in the worst case.

We presented a model-based direction, adopting the classical Naive Bayes approach to assume that the dimensions are nearly independent, with or without conditioning on a latent variable. We present two pass streaming algorithms using \( O(d/\gamma) \) space, and further present an efficient algorithm to list all the subcube heavy hitters.

Our work demonstrates the power of model-based data stream analysis for high dimensional case. Our approach to subspace heavy hitters opens several directions for further study. For example,
• Can heavy hitters be detected efficiently under more general models among dimensions, such as graphical models with hidden variables?

• Can these models be learned or fitted over data streams with polylog space? We believe this is an algorithmic problem of great interest and will have applications in machine learning beyond the context here.

• There are some nuances in our results, in particular, with respect to the parameters. We have $\lambda$, which defines the heavy hitting frequency; $\alpha$ which defines the variational distance between the model and the data; $\ell$, the cardinality of the hidden dimension. From the various application scenarios, $\alpha$ like $\gamma/10$ sufficed; $\ell$ is typically a constant $O(1)$ in realistic models. It is interesting to study the relationship between these parameters. Of particular importance is $\ell$. The general assumption is that the space of dimensions has low rank (constant $\ell$) representation in terms of the values of the hidden dimension. Can the latent dimension size $\ell$ be large in applications and if yes, is the dependence of our algorithms on $\ell$ optimal? We assumed we know the latent dimension. Can this be learned from the data stream?

• Can the model-based approach be extended to other problems besides heavy hitters, including clustering, anomaly detection, geometric problems and others which have been studied in the streaming literature.

We believe that model-based approach to algorithmic problems is a fruitful area for collaboration between algorithms and statistics or machine learning.

References

[1] Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Georgios Sakkis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. CoRR, cs.CL/0009009, 2000.

[2] Graham Cormode. Finding frequent items in data streams. http://dmac.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf, 2008. DIMACS Workshop.

[3] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In LATIN, volume 2976 of Lecture Notes in Computer Science, pages 29–38. Springer, 2004.

[4] Marco F. Duarte, Volkan Cevher, and Richard G. Baraniuk. Model-based compressive sensing for signal ensembles. In Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on, pages 244–250. IEEE, 2009.

[5] Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian. Streaming for large scale NLP: language modeling. In Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA, pages 512–520. The Association for Computational Linguistics, 2009.

[6] Matthew Hamilton, Rhonda Chaytor, and Todd Wareham. The parameterized complexity of enumerating frequent itemsets. In IWPEC, volume 4169 of Lecture Notes in Computer Science, pages 227–238. Springer, 2006.

[7] Branislav Kveton, Hung Hai Bui, Mohammad Ghavamzadeh, Georgios Theocharous, S. Muthukrishnan, and Siqi Sun. Graphical model sketch. In ECML/PKDD (1), volume 9851 of Lecture Notes in Computer Science, pages 81–97. Springer, 2016.

[8] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research, 5:361–397, 2004.
[9] Edo Liberty, Michael Mitzenmacher, Justin Thaler, and Jonathan Ullman. Space lower bounds for itemset frequency sketches. In *PODS*, pages 441–454. ACM, 2016.

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[11] Andrew McGregor and Hoa T. Vu. Evaluating bayesian networks via data streams. In *COCOON*, volume 9198 of *Lecture Notes in Computer Science*, pages 731–743. Springer, 2015.

[12] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, 2010.

[13] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

[14] David P. Woodruff. New algorithms for heavy hitters in data streams (invited talk). In *ICDT*, volume 48 of *LIPIcs*, pages 4:1–4:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[15] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD*, pages 344–353. ACM, 2004.