

Finding Subcube Heavy Hitters in Analytics Data Streams

Branislav Kveton*
Adobe Research
kveton@adobe.com

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

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

Yikun Xian
Rutgers University
siriusxyk@gmail.com

ABSTRACT

Modern data streams typically have high dimensionality. For example, digital analytics streams consist of user online activities (e.g., web browsing activity, commercial site activity, apps and social behavior, and response to ads). An important problem is to find frequent joint values (heavy hitters) of subsets of dimensions.

Formally, the data stream consists of d -dimensional items and a k -dimensional subcube T is a subset of k distinct coordinates. Given a threshold γ , a subcube heavy hitter query $\text{Query}(T, v)$ outputs YES if $f_T(v) \geq \gamma$ and NO if $f_T(v) < \gamma/4$ where f_T is the ratio of the number of stream items whose coordinates T have joint values v . The all subcube heavy hitters query $\text{AllQuery}(T)$ outputs all joint values v that return YES to $\text{Query}(T, v)$. The problem is to answer these queries correctly for all T and v .

We present a simple one-pass sampling algorithm to solve the subcube heavy hitters problem in $\tilde{O}(kd/\gamma)$ space. $\tilde{O}(\cdot)$ suppresses polylogarithmic factors. This is optimal up to polylogarithmic factors based on the lower bound of Liberty et al. [13]. In the worst case, this bound becomes $\Theta(d^2/\gamma)$ which is prohibitive for large d .

Our main contribution is to circumvent this quadratic bottleneck via a model-based approach. In particular, we assume that the dimensions are related to each other via the Naive Bayes model. 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 $\tilde{O}((k/\gamma)^2)$ time.

We demonstrate the effectiveness of our approach on a synthetic dataset as well as real datasets from Adobe and Yandex. Our work shows the potential of model-based approach to data streams.

KEYWORDS

algorithms, data streams, heavy hitters, graphical models

ACM Reference Format:

Branislav Kveton, S. Muthukrishnan, Hoa T. Vu, and Yikun Xian. 2018. Finding Subcube Heavy Hitters in Analytics Data Streams. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3178876.3186082>

*The authors are listed alphabetically.

†Part of this work was done at Adobe Research.

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW 2018, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5639-8/18/04.

<https://doi.org/10.1145/3178876.3186082>

1 INTRODUCTION

We study the problem of finding heavy hitters in high dimensional data streams. Most companies see transactions with items sold, time, store location, price, etc. that arrive over time. Modern online companies see streams of user web activities that typically have components of user information including ID (e.g. cookies), hardware (e.g., device), software (e.g., browser, OS), and contents such as web properties, apps. Activity streams also include events (e.g., impressions, views, clicks, purchases) and event attributes (e.g., product id, price, geolocation, time). 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. Furthermore, in applications such as Natural Language Processing, streams of documents can be thought of as streams of a large number of bigrams or multi-grams over word combinations [8]. As these examples show, analytics data streams with 100's and 1000's of dimensions arise 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 of items x_1, \dots, x_m where each $x_i \in [n] := \{1, 2, \dots, n\}$. We can look at the count $c(v) = |\{i : x_i = v\}|$ or the frequency ratio $f(v) = c(v)/m$. A heavy hitter value v is one with $c(v) \geq \gamma m$ or equivalently $f(v) \geq \gamma$, for some constant γ . The standard data stream model is that we maintain data structures of size $\text{polylog}(m, n)$ and determine if v is a heavy hitter with probability of success at least $3/4$, that is, if $f(v) \geq \gamma$ output YES and output NO if $f(v) < \gamma/4$ for all v .¹ We note that if $\gamma/4 \leq f(v) < \gamma$, then either answer is acceptable.

Detecting heavy hitters on data streams is a fundamental problem 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, the heavy hitters problem has been studied extensively in theory, databases, networking and signal processing literature. See [4] for an early survey and [19] for a recent survey.

Subcube heavy hitter problems. Our focus is on modern data streams such as in analytics cases, with d dimensions, for large d . The data stream consists of d -dimensional items x_1, \dots, x_m . In

¹The gap constant 4 can be narrowed arbitrarily and the success probability can be amplified to $1 - \delta$ as needed, and we omit these factors in the discussions.

particular,

$$x_i = (x_{i,1}, \dots, x_{i,d}) \text{ and each } x_{i,j} \in [n].$$

A k -dimensional subcube T is a subset of k distinct coordinates $\{T_1, \dots, T_k\} \subseteq [d]$. We refer to the joint values of the coordinates T of x_i as $x_{i,T}$.

The number of items whose coordinates T have joint values v is denoted by $c_T(v)$, i.e., $c_T(v) = |\{i : x_{i,T} = v\}|$. Finally, we use X_T to denote the random variable of the joint values of the coordinates T of a random item. We have the following relationship

$$f_T(v) := \Pr[X_T = v] = \frac{c_T(v)}{m}.$$

For a single coordinate i , we slightly abuse the notation by using f_i and $f_{\{i\}}$ interchangeably. For example, $f_{T_i}(v)$ is the same as $f_{\{T_i\}}(v)$. Similarly, X_i is the same as $X_{\{i\}}$.

We are now ready to define our problems. They take k, γ as parameters and the stream as the input and build data structures to answer:

- *Subcube Heavy Hitter*: $\text{Query}(T, v)$, where $|T| = k$, and $v \in [n]^k$, returns an estimate if $f_T(v) \geq \gamma$. Specifically, 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 k -dimensional subcubes T and $v \in [n]^k$ is at least $3/4$.
- *All Subcube Heavy Hitters*: $\text{AllQuery}(T)$ outputs all joint values v that return YES to $\text{Query}(T, v)$. This is conditioned on the algorithm used for $\text{Query}(T, v)$.

It is important to emphasize that the stream is presented (in a single pass or constant passes) to the algorithm before the algorithm receives any query.

Subcube heavy hitters are relevant wherever one dimensional heavy hitters have found applications: combination of source and destination IP addresses forms the subcube heavy hitters that detect network attacks; combination of stores, sales quarters and nature of products forms the subcube heavy hitters that might be the pattern of interest in the data, etc. Given the omnipresence of multiple dimensions in digital analytics, arguably, subcube heavy hitters learn the significant data properties far more than the single dimensional view.

Related works. 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 $\text{Query}(T, v)$ where $v = (1, \dots, 1) := U_k$. It is known that counting all maximal subcubes T that have a frequent itemset, i.e., $f_T(U_k) \geq \gamma$, is $\#P$ -complete [21]. Furthermore, finding even a single T of maximal size such that $f_T(U_k) \geq \gamma$ is NP-hard [9, 13]. Recently, Liberty et al. showed that any constant-pass streaming algorithm answering $\text{Query}(T, U_k)$ requires $\Omega(kd/\gamma \cdot \log(d/k))$ bits of memory [13]. In the worst case, this is $\Omega(d^2/\gamma)$ for large k , ignoring the polylogarithmic 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 reservoir sampling approach [18] solves subcube heavy hitters problems more efficiently compared to the approaches mentioned above. Our analysis shows that the space we use is within polylogarithmic factors of the lower bound shown in [13] 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 $\tilde{O}(kd/\gamma)$ space. However, this is $\Omega(d^2)$ in worst case.

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 formula. 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 $\text{Query}(T, v)$. This improves upon the kd factor in the space complexity from sampling, without assumptions, to just d with the Naive Bayes assumption, which would make this algorithm 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 their frequencies.
- We present a fast algorithm for answering $\text{AllQuery}(T)$ in $\tilde{O}((k/\gamma)^2)$ 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 Naive Bayes assumption to iteratively construct the subcube heavy hitters 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 [7]. In statistics, using tree constrained models of multidimensional data sometimes improves point and density estimation. In high dimensional distribution testing, model based approach has also been studied to overcome the curse of dimensionality [6].

In the data stream model, [2, 3, 10] studied the problem of testing independence. McGregor and Vu [15] studied the problem of evaluating Bayesian Networks. In another work, Kveton et al. [11] 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) which prior work does not solve. In following such a direction, we have extended the fundamental heavy 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 [17] is a Bayesian network over d features X_1, \dots, X_d and a class variable Y . This model represents a joint probability distribution of the form

$$\begin{aligned} & \Pr[X_1 = x_1, \dots, X_d = x_d, Y = y] \\ &= \Pr[Y = y] \prod_{j=1}^d \Pr[X_j = x_j \mid Y = y], \end{aligned}$$

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 [12, 14], with state-of-the-art performance in spam filtering [1], text classification [12], and others.

Empirical study. We perform detailed experimental study of subcube heavy hitters. We use a synthetic dataset where we generate data that confirms to the Naive Bayes model. We then experiment with real data sets from Yandex (Search) and Adobe (Marketing Cloud) which give multidimensional analytics streams. We experiment with the reservoir sampling based algorithm as a benchmark that works without modeling assumptions, and our two-pass subcube heavy hitters algorithm that improves upon it for data that satisfies the model. We also adopt our approach to give a simpler one-pass algorithm for which theoretical guarantees is weaker. Our experiments show substantial improvement of the model-based algorithms over the benchmark for synthetic as well as real data sets, and further show the benefits of the second pass.

2 THE SAMPLING ALGORITHM

In this section, we show that sampling solves the problem efficiently compared to running one-dimensional heavy hitters algorithms for each of $\binom{d}{k}$ k -dimensional subcubes independently. It also matches the lower bound in [13] up to polylogarithmic factors.

Algorithm details. The algorithm samples $m' = \tilde{O}(\gamma^{-1}kd)$ random items $z_1, \dots, z_{m'}$ from the stream using Reservoir sampling [18]. Let $S = \{z_1, \dots, z_{m'}\}$ be the sample set. Given $\text{Query}(T, v)$, we output YES if and only if the sample frequency of v , denoted by $\hat{f}_T(v)$, is at least $\gamma/2$. Specifically,

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

For all subcubes T and joint values v of T , the expected sample frequency $\hat{f}_T(v)$ is $f_T(v)$. Intuitively, if v is a frequent joint values, then its sample frequency $\hat{f}_T(v) \approx f_T(v)$; otherwise, $\hat{f}_T(v)$ stays small.

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

$$\hat{f}_T(v) = f_T(v) \pm \frac{\max\{\gamma, f_T(v)\}}{4}. \quad (1)$$

It is then straightforward to see that if $f_T(v) < \gamma/4$, then $\hat{f}_T(v) < \gamma/4 + \gamma/4 = \gamma/2$. Otherwise, if $f_T(v) \geq \gamma$, then $\hat{f}_T(v) \geq 3f_T(v)/4 \geq 3\gamma/4 > \gamma/2$. Hence, we output YES for all v where $\hat{f}_T(v) \geq \gamma/2$, and output NO otherwise.

LEMMA 2.1. (*Chernoff bound*) Let X_1, \dots, X_n be independent or negatively correlated binary random variables. Let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. Then,

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

Recall that $S = \{z_1, z_2, \dots, z_{m'}\}$ is the sample set returned by the algorithm. For a fixed $v \in [n]^k$, we use Z_i as the indicator variable for the event $z_{i,T} = v$. Since we sample without replacement, the random variables Z_i are negatively correlated. The following lemma shows that Eq. 1 holds for all v and k -dimensional subcubes T via Chernoff bound.

LEMMA 2.2. For all k -dimensional subcubes T and joint values $v \in [n]^k$, with probability at least 0.9,

$$\hat{f}_T(v) = f_T(v) \pm \frac{\max\{\gamma, f_T(v)\}}{4}.$$

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, i.e., $Z_i = 1$ if $z_{i,T} = v$. Suppose $f_T(v) \geq \gamma$. Appealing to Lemma 2.1, we have

$$\begin{aligned} & \Pr\left[\left|\sum_{i=1}^{m'} \frac{Z_i}{m'} - f_T(v)\right| \geq \frac{f_T(v)}{4}\right] \\ &= \Pr\left[\left|\hat{f}_T(v) - f_T(v)\right| \geq \frac{f_T(v)}{4}\right] \\ &\leq \exp\left(-\frac{f_T(v)m'}{3 \times 16}\right) \leq \frac{1}{10d^k n^k}. \end{aligned}$$

On the other hand, if $f_T(v) < \gamma/4$, then

$$\begin{aligned} \Pr\left[\left|\hat{f}_T(v) - f_T(v)\right| \geq \frac{\gamma}{4}\right] &\leq \exp\left(-\left(\frac{\gamma}{4f_T(v)}\right)f_T(v)\frac{m'}{3}\right) \\ &\leq \frac{1}{10d^k n^k}. \end{aligned}$$

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

We therefore could answer all $\text{Query}(T, v)$ correctly with probability at least 0.9 for all joint values $v \in [n]^k$ and k -dimensional subcubes T . Because storing each sample z_i requires $\tilde{O}(d)$ bits of space, the algorithm uses $\tilde{O}(dk\gamma^{-1})$ space. We note that answering $\text{Query}(T, v)$ requires computing $\hat{f}_T(v)$ which takes $O(|S|)$ time. We can answer $\text{AllQuery}(T)$ by computing $\hat{f}_T(v)$ for all joint values v of coordinates T that appear in the sample set which will take $O(|S|^2)$ time. We summarize the result as follows.

THEOREM 2.3. *There exists a 1-pass algorithm that uses $\tilde{O}(dk\gamma^{-1})$ space and solves k -dimensional subcube heavy hitters. Furthermore, $\text{Query}(T, v)$ and $\text{AllQuery}(T)$ take $\tilde{O}(dk\gamma^{-1})$ and $\tilde{O}\left((dk\gamma^{-1})^2\right)$ time respectively.*

3 THE NEAR-INDEPENDENCE ASSUMPTION

The near-independence assumption. Suppose the random variables representing the dimensions X_1, X_2, \dots, X_d 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

$$f_{\{1, \dots, d\}}(v) \approx f_1(v_1)f_2(v_2) \cdots f_d(v_d).$$

More formally, we assume that the total variation distance is bounded by a small quantity α . Furthermore, we assume that α is reasonable with respect to γ that controls the heavy hitters. For example, $\alpha \leq \gamma/10$ will suffice.

The formal *near-independence* assumption is as follows: There exists $\alpha \leq \gamma/10$ such that for all subcubes T ,

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

We observe that:

- If $f_T(v) \geq \gamma$, then

$$\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq f_T(v) - \gamma/10 > \gamma/2.$$

- If $f_T(v) < \gamma/4$, then

$$\prod_{i=1}^{|T|} f_{T_i}(v_i) \leq f_T(v) + \gamma/10 < \gamma/2.$$

Thus, it suffices to output YES to $\text{Query}(T, v)$ if and only if the marginals product $\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq \gamma/2$. For convenience, let

$$\lambda := \gamma/2.$$

Algorithm details. We note that simply computing $f_i(x)$ for all coordinates $i \in [d]$ and $x \in [n]$ will need $\Omega(dn)$ space. To overcome this, we make following simple but useful observation. We observe that if v is a heavy hitter in the subcube T and if T' is a subcube of T , then $v_{T'}$ is a heavy hitter in the subcube T' .

LEMMA 3.1. *For all subcubes T ,*

$$\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq \lambda \text{ implies } \prod_{i \in \mathcal{V}} f_{T_i}(v_i) \geq \lambda$$

for all $\mathcal{V} \subseteq [|T|]$ (i.e., $\{T_i : i \in \mathcal{V}\}$ is a subcube of T).

The proof is trivial since all $f_{T_i}(v_i) \leq 1$. Therefore, we have the following corollary.

COROLLARY 3.2. *For all subcubes T ,*

$$\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq \lambda \text{ implies } f_{T_i}(v_i) \geq \lambda \text{ for all } i \in [|T|].$$

We therefore only need to compute $f_i(x)$ if x is a heavy hitter in coordinate i . To this end, for each coordinate $i \in [d]$, by using (for example) Misra-Gries algorithm [16] or Count-Min sketch [5], we can 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 $x \notin H_i$. In the second pass, for each $x \in H_i$, we compute $f_i(x)$ exactly to obtain

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

We output YES to $\text{Query}(T, v)$ if and only if all $v_i \in S_{T_i}$ and $\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq \lambda$. Note that if $v \in S_i$, then $f_i(v)$ is available to the algorithm since it is computed exactly in the second pass. The detailed algorithm is as follows.

- (1) First pass: For each coordinate $i \in [d]$, use Misra-Gries algorithm to find H_i .
- (2) Second pass: For each coordinate $i \in [d]$, compute $f_i(x)$ exactly for each $x \in H_i$ to obtain S_i .
- (3) Output YES to $\text{Query}(T, v)$ if and only if $v_i \in S_{T_i}$ for all $i \in [|T|]$ and

$$\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq \lambda.$$

THEOREM 3.3. *There exists a 2-pass algorithm that uses $\tilde{O}(d\gamma^{-1})$ space and solves subcube heavy hitters under the near-independence assumption. The time to answer $\text{Query}(T, v)$ and $\text{AllQuery}(T)$ are $\tilde{O}(k)$ and $\tilde{O}(k\gamma^{-1})$ respectively where k is the dimensionality of T .*

PROOF. The first pass uses $\tilde{O}(d\lambda^{-1})$ space since Misra-Gries algorithm uses $\tilde{O}(\lambda^{-1})$ space for each coordinate $i \in [d]$. Since the size of each H_i is $O(\lambda^{-1})$, the second pass also uses $\tilde{O}(d\lambda^{-1})$ space. Recall that $\lambda = \gamma/2$. We then conclude that the algorithm uses $\tilde{O}(d\gamma^{-1})$ space.

For an arbitrary $\text{Query}(T, v)$, the algorithm's correctness follows immediately from Corollary 3.2 and the observation that if $v_i \in S_{T_i}$, then $f_{T_i}(v_i)$ is available since it was computed exactly in the second pass. Specifically, if

$$\prod_{i=1}^{|T|} f_{T_i}(v_i) \geq \lambda, \tag{2}$$

then $v_i \in S_{T_i}$ for all $i \in [|T|]$ and we could verify the inequality and output YES. On the other hand, suppose Eq. 2 does not hold. Then, if $v_i \notin S_{T_i}$ for some i , we correctly output NO. But if all $v_i \in S_{T_i}$, then we are able to verify that the inequality does not hold (and correctly output NO).

The parameter k only affects the query time. We now analyze the time to answer $\text{Query}(T, v)$ and $\text{AllQuery}(T)$ for a k -dimensional subcube T .

We can easily see that $\text{Query}(T, v)$ takes $\tilde{O}(k)$ time as we need to check if all $v_i \in S_{T_i}$ (e.g., using binary searches) and compute $\prod_{i=1}^k f_{T_i}(v_i)$.

Next, we exhibit a fast algorithm to answer $\text{AllQuery}(T)$. We note that naively checking all combinations (v_1, \dots, v_k) in $S_{T_1} \times S_{T_2} \times \dots \times S_{T_k}$ takes exponential $\Omega(\gamma^{-k})$ time in the worst case.

Our approach figures out the heavy hitters gradually and takes advantage of the near-independence assumption. In particular, define

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

Recall that the goal is to find W_k . Note that $W_1 = S_1$ is obtained directly by the algorithm. We now show that it is possible to construct W_{j+1} from W_j in $\tilde{O}(\lambda^{-1})$ time which in turn means that we can find W_k in $\tilde{O}(k\lambda^{-1})$ time. We use the notation $T_{[j]} := \{T_1, \dots, T_j\}$ and $v_{[j]} := (v_1, v_2, \dots, v_j)$.

We note that $|W_j| \leq 5/(4\lambda)$. This holds since if $y \in W_j$, then $\prod_{i=1}^j f_{T_i}(y_i) \geq \lambda$. Appealing to the near-independence assumption, we have

$$f_{T_{[j]}}(y) \geq \prod_{i=1}^j f_{T_i}(y_i) - \alpha \geq \lambda - \alpha \geq 4/5 \cdot \lambda.$$

For each $y \in W_j$, we collect all $x \in S_{j+1}$ such that

$$\left(\prod_{i=1}^j f_{T_i}(y_i) \right) f_{T_{j+1}}(x) \geq \lambda$$

and put (y_1, \dots, y_j, x) 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, by observing that there could be at most $\lambda^{-1} \prod_{i=1}^j f_{T_i}(y_i)$ such x for each $y \in W_j$, the upper bound for the number of combinations of x and y is

$$\begin{aligned} \sum_{y \in W_j} \frac{1}{\lambda} \prod_{i=1}^j f_{T_i}(y_i) &\leq \sum_{y \in W_j} \frac{1}{\lambda} (f_{T_{[j]}}(y) + \alpha) \\ &= \sum_{y \in W_j} \frac{\alpha}{\lambda} + \sum_{y \in W_j} \frac{f_{T_{[j]}}(y)}{\lambda} \\ &\leq |W_j| + \frac{1}{\lambda} \leq \frac{3}{\lambda}. \end{aligned}$$

The last inequality follows from the assumption that $\alpha \leq \lambda/5$ and $\sum_{y \in W_j} f_{T_{[j]}}(y) \leq 1$. Thus, the algorithm can find W_{j+1} given W_j in $\tilde{O}(\lambda^{-1})$ time. Hence, we obtain W_k in $\tilde{O}(k\lambda^{-1}) = \tilde{O}(k\gamma^{-1})$ time. The correctness of this procedure follows directly from Lemma 3.1 and induction since $v = (v_1, \dots, v_{j+1}) \in W_{j+1}$ implies that $v_{[j]} \in W_j$ and $v_{j+1} \in S_{j+1}$. Thus, by checking all combinations of $y \in W_j$ and $x \in S_{j+1}$, we can construct W_{j+1} correctly. \square

4 THE NAIVE BAYES ASSUMPTION

The Naive Bayes assumption. In this section, we focus on the data streams 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 *class coordinate* that has a value in $\{1, \dots, \ell\}$. The $(d+1)$ th coordinate is also often referred to as the *latent coordinate*.

As in typical in Naive Bayes analysis, we assume ℓ is a constant but perform the calculations in terms of ℓ so its role in the complexity of the problem is apparent.

Informally, this model asserts that the random variables representing coordinates X_1, \dots, X_d are near independent conditioning on a the random variable X_{d+1} that represents the class coordinate.

We introduce the following notation

$$\begin{aligned} f_{T \mid d+1}(v \mid z) &:= \frac{|\{x_i : x_{i,T} = v \wedge x_{i,\{d+1\}} = z\}|}{|\{x_i : x_{i,\{d+1\}} = z\}|} \\ &= \Pr[X_T = v \mid X_{d+1} = z]. \end{aligned}$$

In other words, $f_{T \mid d+1}(v \mid z)$ is the frequency of the joint values v in the T coordinates among the stream items where the class coordinate $d+1$ has value z .

The formal *Naive Bayes* assumption is as follows: There exists $\alpha \leq \gamma/10$ such that for all subcubes T ,

$$\max_{v \in [n]^{|T|}} \left| f_T(v) - \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i \mid d+1}(v_i \mid z) \right| < \alpha.$$

Algorithm details. As argued in the previous section, it suffices to output YES to Query(T, v) if and only if

$$\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i \mid d+1}(v_i \mid z) \geq \gamma/2 \cdot \lambda.$$

However, naively computing all $f_{T_i \mid d+1}(v_i \mid z)$ uses $\Omega(\ell dn)$ space. We circumvent this problem by generalizing Lemma 3.1 as follows. If a joint values v is a heavy hitter in a subcube T in the Naive Bayes formula and T' is a subcube of T , then $v_{T'}$ is a heavy hitter in the subcube T' .

LEMMA 4.1. *For all subcubes T ,*

$$\begin{aligned} q(v) &:= \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i \mid d+1}(v_i \mid z) \geq \lambda \\ \text{implies} &\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in \mathcal{V}} f_{T_i \mid d+1}(v_i \mid z) \geq \lambda \end{aligned}$$

for all $\mathcal{V} \subseteq [|T|]$ (i.e., $\{T_i : i \in \mathcal{V}\}$ is a subcube of T).

PROOF. For a fixed z , observe that

$$\sum_{y_j \in [n]} f_{T_j \mid d+1}(y_j \mid z) = 1.$$

Suppose $q(v) \geq \lambda$ and consider an arbitrary $\mathcal{V} \subseteq [|T|]$. We have

$$\begin{aligned} &\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in \mathcal{V}} f_{T_i \mid d+1}(v_i \mid z) \\ &= \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in \mathcal{V}} f_{T_i \mid d+1}(v_i \mid z) \prod_{j \notin \mathcal{V}} \left(\sum_{y_j \in [n]} f_{T_j \mid d+1}(y_j \mid z) \right) \\ &\geq \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i \in \mathcal{V}} f_{T_i \mid d+1}(v_i \mid z) \prod_{j \notin \mathcal{V}} f_{T_j \mid d+1}(v_j \mid z) \\ &= \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i \mid d+1}(v_i \mid z) = q(v) \geq \lambda. \end{aligned}$$

An alternative proof is by noticing that $q(v)$ is a valid probability density function of $|T|$ variables. The claim follows by marginalizing over the variables that are not in \mathcal{V} . \square

Setting $\mathcal{V} = \{i\}$ for each $i \in [|T|]$ and appealing to the fact that

$$\sum_{z \in [\ell]} f_{d+1}(z) f_{T_i | d+1}(v_i | z) = \sum_{z \in [\ell]} f_{\{T_i, d+1\}}((v_i, z)) = f_{T_i}(v_i),$$

we deduce the following corollary.

COROLLARY 4.2. *For all subcubes T ,*

$$\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i | d+1}(v_i | z) \geq \lambda \text{ implies } f_{T_i}(v_i) \geq \lambda$$

for all $i \in [|T|]$.

Therefore, we only need to compute $f_{i | d+1}(x | z)$ for all coordinates $i \in [d]$, values $z \in [\ell]$ if x is a heavy hitter of coordinate i . Similar to the previous section, for each dimension $i \in [d]$, we find H_i in the first pass and use H_i to find S_i in the second pass. Appealing to Corollary 4.2, we deduce that if

$$q(v) := \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i | d+1}(v_i | z) \geq \lambda$$

then for all $i = 1, 2, \dots, |T|$, we have $f_{T_i}(v_i) \geq \lambda$ which in turn implies that $v_i \in S_{T_i}$. Therefore, we output YES to $\text{Query}(T, v)$ if and only if all $v_i \in S_{T_i}$ and $q(v) \geq \lambda$.

To this end, we only need to compute $f_{i | d+1}(x | z)$ and $f_{d+1}(z)$ for all $x \in H_i$, $z \in [\ell]$, and $i \in [d]$. The detailed algorithm is as follows.

- (1) First pass:
 - (a) For each value $z \in [\ell]$, compute $f_{d+1}(z)$ exactly.
 - (b) For each coordinate $i \in [d]$, use Misra-Gries algorithm to find H_i .
- (2) Second pass:
 - (a) For each coordinate $i \in [d]$ and each value $x \in H_i$, compute $f_i(x)$ exactly to obtain S_i .
 - (b) For each value $z \in [\ell]$, coordinate $i \in [d]$, and $x \in H_i$, compute $f_{i | d+1}(x | z)$ exactly.
- (3) Output YES to $\text{Query}(T, v)$ if and only if $v_i \in S_{T_i}$ for all $i \in [|T|]$ and

$$\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i | d+1}(v_i | z) \geq \lambda.$$

THEOREM 4.3. *There exists a 2-pass algorithm that uses $\tilde{O}(\ell d \gamma^{-1})$ space and solves subcube heavy hitters under the Naive Bayes assumption. The time to answer $\text{Query}(T, v)$ and $\text{AllQuery}(T)$ are $\tilde{O}(\ell k)$ and $O(\ell(k/\gamma)^2)$ respectively where k is the dimensionality of T .*

PROOF. The space to obtain H_i and S_i over the two passes is $\tilde{O}(d\lambda^{-1})$. Additionally, computing $f_{i | d+1}(x | z)$ for all $i \in [d]$, $z \in [\ell]$, and $x \in H_i$ requires $\tilde{O}(\ell d \lambda^{-1})$ bits of space. The overall space we need is therefore $\tilde{O}(\ell d \lambda^{-1}) = \tilde{O}(\ell d \gamma^{-1})$.

The correctness of answering an arbitrary $\text{Query}(T, v)$ follows directly from Corollary 4.2. Specifically, if

$$\sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^{|T|} f_{T_i | d+1}(v_i | z) \geq \lambda, \quad (3)$$

then, $v_i \in S_{T_i} \subseteq H_{T_i}$ for all $i \in [|T|]$ as argued. Hence, $f_{T_i | d+1}(v_i | z)$ is computed exactly in the second pass for all $z \in [\ell]$. As a result, we could verify the inequality and output YES. On the other hand, if Eq. 3 does not hold. Then, if some $v_i \notin S_{T_i}$, we will correctly output NO. Otherwise if all $v_i \in S_{T_i}$, then we can compute the left hand side and verify that Eq. 3 does not hold (and correctly output NO).

Obviously, $\text{Query}(T, v)$ takes $\tilde{O}(\ell k)$ time for a k -dimensional subcube T . We now exhibit a fast algorithm to answer $\text{AllQuery}(T)$ for a k -dimensional subcube T . Define

$$W_j := \{v \in [n]^j : \sum_{z \in [\ell]} f_{d+1}(z) \prod_{i=1}^j f_{T_i | d+1}(v_i | z) \geq \lambda\}.$$

Recall that the goal is to find W_k . We note that $W_1 = S_1$ is obtained directly by the algorithm. Next, we show how to obtain W_{j+1} in $\tilde{O}(\lambda^{-2})$ time from W_j . Note that $|W_j| \leq 5/(4\lambda)$ because if $y \in W_j$, then

$$\sum_{z \in [\ell]} f_{T_1 | d+1}(y_1 | z) \cdots f_{T_j | d+1}(y_j | z) f_{d+1}(z) \geq \lambda$$

and hence $f_{T_{[j]}}(y) \geq \lambda - \alpha = 4/5 \cdot \lambda^{-1}$ according to the Naive Bayes assumption. This implies that $|W_j| \leq 5/(4\lambda)$.

For each $(v_1, \dots, v_j) \in W_j$, we collect all $v_{j+1} \in S_{j+1}$ such that

$$\sum_{z \in [\ell]} f_{T_1 | d+1}(v_1 | z) \cdots f_{T_{j+1} | d+1}(v_{j+1} | z) f_{d+1}(z) \geq \lambda$$

and put (v_1, \dots, v_{j+1}) to W_{j+1} . Since $|W_j| \leq 5/(4\lambda)$ and $|S_{j+1}| \leq 1/\lambda$, this step obviously takes $\tilde{O}(\ell k \lambda^{-2})$ time. Since we need to do this for $j = 2, 3, \dots, k$, we attain W_k in $\tilde{O}(\ell(k/\gamma)^2)$ time. The correctness of this procedure follows directly from Lemma 4.1 and induction since $(v_1, \dots, v_{j+1}) \in W_{j+1}$ implies that (v_1, \dots, v_j) is in W_j and v_{j+1} is in S_{j+1} . Since we check all possible combinations of $(v_1, \dots, v_j) \in W_j$ and $v_{j+1} \in S_{j+1}$, we guarantee to construct W_{j+1} correctly. \square

5 EXPERIMENTAL STUDY

Overview. We experiment with our algorithms on a synthetic dataset generated from a Naive Bayes model, and two real-world datasets from Adobe Marketing Cloud² and Yandex. We thoroughly compare the following approaches:

- The sampling method (Sampling) in Section 2.
- The 2-pass algorithms (TwoPassAlg) described in Section 3 and 4 depending on the context of the experiment.
- The Count-Min sketch heuristic (Heuristic): this heuristic uses Count-Min sketch's point query estimation (see [5]) to estimate the frequencies given by the near-independence formula (instead of making a second pass through the stream to compute their exact values). We note that this approach has no theoretical guarantee.

We highlight the main differences between the theoretical algorithms in Sections 3 and 4 and the actual implementation:

²<http://www.adobe.com/marketing-cloud.html>

- Instead of running our algorithms with the theoretical memory bounds, we run and compare them for different memory limits. This approach is more practical and natural from the implementation perspective.
- In theory, Sampling and TwoPassAlg use a fixed threshold $\gamma^* = \gamma/2$ to decide between outputting YES or NO. We however experiment with different values of γ^* which is helpful when the memory is more limited or when the assumptions are not perfect in real data.

The heavy hitters threshold γ is carefully chosen so that the proportion of the number of heavy hitters to the total number joint values to be reasonably small, i.e., approximately at most 1% in this paper. Therefore, we use different values of γ for each dataset (see Table 1 for the actual parameter values).

5.1 Synthetic dataset

The synthetic dataset is sampled from a pre-trained Naive Bayes model that is used to estimate the probability of a page view. The model was provided by [11] and built on the same Clickstream dataset that we used in Section 5.2. The coordinates consist of one class variable Z and five feature variables (X_1, \dots, X_5) with high cardinalities. The dataset strongly follows the property that X_1, \dots, X_5 are conditionally independent given Z . Specifically, the variables and their corresponding approximate cardinalities are: country (7), city (10,500), page name (8,500), starting page name (6,400), campaign (3,500), browser (300) where country is the class variable.

Warm up experiment. We first evaluate Sampling and Heuristic on this synthetic dataset. As mentioned earlier, we compare the performance of the two approaches for each fixed memory size.³ We take a subset of approximately 135,000 records conditioned on a fixed and most frequent value of Z so that X_1, \dots, X_5 are independent in this subset. We then run experiments on three different subcubes: $\{X_1, X_2, X_3\}$, $\{X_2, X_3, X_4\}$, and $\{X_3, X_4, X_5\}$. In this warm up experiment, the main goal is not to find the heavy hitters but to compare the accuracy of the heavy hitters frequency estimations given by Heuristic and Sampling. We measure the performance via the mean square error (MSE), the mean absolute error (MAE), and the mean absolute percentage error (MAPE). To do this, the true frequencies were pre-computed. We use the frequencies of the top 10 heavy hitters in each of the above subcubes. The results (see Figure 1) indicate that Heuristic outperforms Sampling when restricted to small memory. This warm up experiment gives evidence that knowing the underlying distribution structure helps improving small space heuristic’s performance in estimating the heavy hitters frequencies.

Experiment with the near-independence assumption. We compare performance of the three aforementioned methods on finding heavy hitters under the near-independence assumption. In this experiment, we use the same subset of data and subcubes as in the previous experiment. We fix the memory to be 2% of data size.

Dataset	Mem.	#Subcubes	γ	#HH	HH ratio
Synthetic (fixed Z)	2%	3	0.002	29.7	0.079%
Synthetic (whole)	2%	3	0.002	28.7	0.054%
Clickstream	10%	4	0.002	42.0	0.165%
Yandex	0.2%	8	0.1	2.2	1.51 %

Table 1: Parameter values for each experiment.
(The columns correspond to memory size relative to the dataset, number of the experimented subcubes, average number of heavy hitters, average percentage of heavy hitters.)

We measure the performance, for different values of γ^* , based on the number of true positives and false positives. As shown in Figure 2, for small memory, both Heuristic and TwoPassAlg manage to find more heavy hitters than Sampling. In terms of false positives, TwoPassAlg beats both Heuristic and Sampling for smaller space. One possible explanation is that when γ^* is small (close to γ), TwoPassAlg, with the advantage of the second pass, accurately estimates frequencies of potential heavy hitters whereas the other two methods, especially Heuristic, overestimate the frequencies and therefore report more false positives. For larger γ^* , false positives become less likely and all three approaches achieve similar performances. In general, TwoPassAlg obtains the best performance as seen in the ROC curve.

Experiment with the Naive Bayes assumption. We use the whole dataset of approximately 168,000 records without fixing Z and keep other settings unchanged. We only compared the performance of TwoPassAlg and Sampling because the conditional probabilities cannot be directly derived from Heuristic. In Figure 3, we observe that when restricted to small memory, TwoPassAlg attains a better performance by reporting more true heavy hitters and fewer false heavy hitters. As we allow more space, the performance of Sampling improves as predicted by our theoretical analysis.

5.2 Clickstream dataset

To evaluate TwoPassAlg on real data, we use an advertising dataset called *Clickstream Data Feeds* from Adobe Marketing Cloud. The approximate dataset size is 168,000 and all values have been anonymized in advance.

There are 19 high cardinality variables grouped by categories as follows: geography info (city, region, country, domain, carrier), page info (page name, start page name, first-hit page name), search info (visit number, referrer, campaign, keywords, search engine), external info (browser, browser width/height, plugins, language, OS).

We avoid obvious correlated features in the query subcubes, e.g., “search engine” and “keywords” are highly correlated. For example, some highly correlated variables and their correlations are: start page name & first-hit page name (0.67), browser & OS (0.40), region & country (0.32), search engine & country (0.27).

We carefully select a subset of coordinates that may follow the near independence assumption to query on. For instance, we show our experiment results for the following subcubes, along with the number of heavy hitters recorded: {region, page name, language}, {region, campaign, plugins}, {carrier, first-hit page name, plugins}, {carrier, keywords, OS}.

³We compute the memory use by the Sampling as the product of dimension and the sample size. The memory used by Heuristic is computed as the product of dimension and the Count-Min sketch’s size.

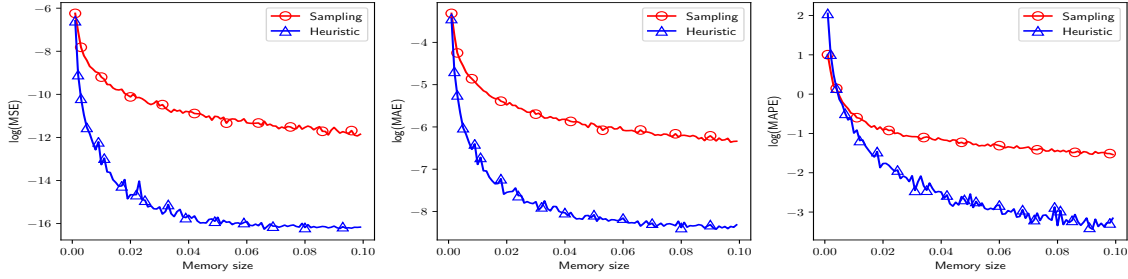


Figure 1: Warm up experiment on synthetic data. Memory size ranges from 0.1% to 10% of data size. We report the error as a function of memory size.

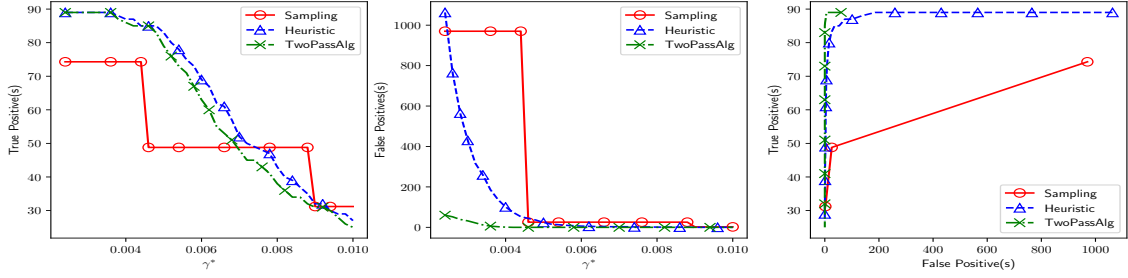


Figure 2: Near-independence experiment on synthetic dataset. We measure the performance based on the number of true and false positives (as a function of γ^*), and the ROC curve.

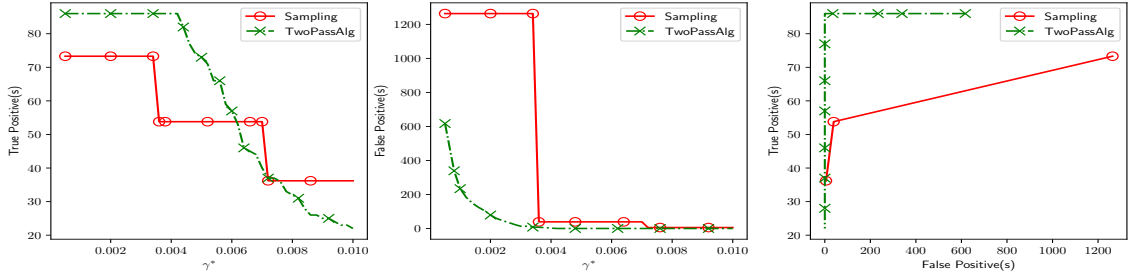


Figure 3: Naive Bayes experiment on synthetic dataset.

Since strong independence property is not guaranteed in this real dataset, we increase memory size to 10% of the data size in order to obtain better estimation for all methods. Recall that the memory used by Sampling and TwoPassAlg is partially determined by the number of dimensions and therefore it is reasonable to use a relatively larger memory size.

In this experiment, all three algorithms are able to find most true heavy hitters (see Figure 4), but TwoPassAlg returns far fewer false positives than the other two methods when γ^* is small. In addition, TwoPassAlg reaches zero false positive for reasonably large γ^* . We can see in the ROC curve that TwoPassAlg performs slightly better than Heuristic and much better than Sampling for small space.

5.3 Yandex dataset

Finally, we consider the *Yandex dataset* [20] which is a web search dataset (with more than 167 millions data points). Each record in the dataset contains a query ID, the corresponding date, the list of 10 displayed items, and the corresponding click indicators of each displayed item. In the pre-processing step, we extracted 10 subsets from the whole dataset according to top 10 frequent user queries. The sizes of subsets range from 61,000 to 454,000. In each subset, we treat the first 10 search results as variables X_1, \dots, X_{10} and “day of query” as the latent variable Z . We conjecture that the search results X_1, \dots, X_{10} are approximately independent conditioned on a given day Z . We observe that web patterns typically experience heavy weekly seasonality and these search results largely depend on user query time for some fixed query. We proceed to evaluate TwoPassAlg under the Naive Bayes assumption on this dataset.

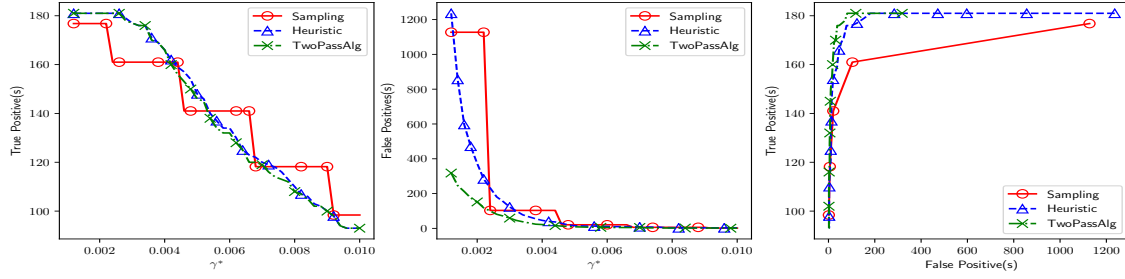


Figure 4: Near-independence experiment with Clickstream dataset.

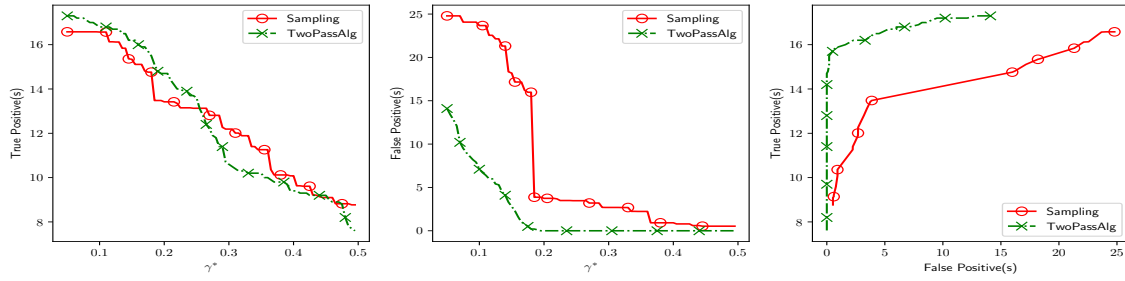


Figure 5: Naive Bayes experiment with Yandex dataset.

We consider 8 subcubes in the form $\{X_i, X_{i+1}, X_{i+2}\}$ and deliberately set a smaller memory size for this experiment because the cardinality of this dataset is relatively low. We note that different subsets of data may have different number of heavy hitters, so we take the average over 10 subsets as the final result.

We report the results in Figure 5. We observe that both Sampling and TwoPassAlg are able to find most true heavy hitters. However, TwoPassAlg performs significantly better in terms of false positives.

6 CONCLUDING REMARKS

Our work demonstrates the power of model-based approach for analyzing high dimensional data that abounds in digital analytics applications. We exhibit algorithms, with fast query time, that overcome worst case space lower bound under the classical Naive Bayes assumption. Our approach to subspace heavy hitters opens several directions for further study. For example,

- Can heavy hitters be detected efficiently under more general models?
- Can these models be learned or fitted over data streams with polylogarithmic space? We believe this is an algorithmic problem of great interest and will have applications in machine learning beyond the context here.
- We assumed that we observe 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.

REFERENCES

- [1] Ion Androustopoulos, Georgios Paliouras, Vangelis Karkaletsis, Georgios Sakakis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. 2000. Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach. *CoRR* cs.CL/0009009 (2000).
- [2] Vladimir Braverman, Kai-Min Chung, Zhenming Liu, Michael Mitzenmacher, and Rafail Ostrovsky. 2010. AMS Without 4-Wise Independence on Product Domains. In *STACS (LIPIcs)*, Vol. 5. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 119–130.
- [3] Vladimir Braverman and Rafail Ostrovsky. 2010. Measuring independence of datasets. In *STOC*. ACM, 271–280.
- [4] Graham Cormode. 2008. Finding Frequent Items in Data Streams. <http://dmac.rutgers.edu/Workshops/WGUNifyingTheory/Slides/cormode.pdf>. (2008). DIMACS Workshop.
- [5] Graham Cormode and S. Muthukrishnan. 2004. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. In *LATIN (Lecture Notes in Computer Science)*, Vol. 2976. Springer, 29–38.
- [6] Constantinos Daskalakis, Nishanth Dikkala, and Gautam Kamath. 2016. Testing Ising Models. *CoRR* abs/1612.03147 (2016). [arXiv:1612.03147](http://arxiv.org/abs/1612.03147)
- [7] Marco F. Duarte, Volkan Cevher, and Richard G. Baraniuk. 2009. Model-based compressive sensing for signal ensembles. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE, 244–250.
- [8] Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian. 2009. 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*. The Association for Computational Linguistics, 512–520. <http://www.aclweb.org/anthology/N09-1058>
- [9] Matthew Hamilton, Rhonda Chaytor, and Todd Wareham. 2006. The Parameterized Complexity of Enumerating Frequent Itemsets. In *IWPEC (Lecture Notes in Computer Science)*, Vol. 4169. Springer, 227–238.
- [10] Piotr Indyk and Andrew McGregor. 2008. Declaring independence via the sketching of sketches. In *SODA*. SIAM, 737–745.
- [11] Branislav Kveton, Hung Hai Bui, Mohammad Ghavamzadeh, Georgios Theodorou, S. Muthukrishnan, and Siqi Sun. 2016. Graphical Model Sketch. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 9851. Springer, 81–97.
- [12] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research* 5 (2004), 361–397.
- [13] Edo Liberty, Michael Mitzenmacher, Justin Thaler, and Jonathan Ullman. 2016. Space Lower Bounds for Itemset Frequency Sketches. In *PODS*. ACM, 441–454.

- [14] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
- [15] Andrew McGregor and Hoa T. Vu. 2015. Evaluating Bayesian Networks via Data Streams. In *COCOON (Lecture Notes in Computer Science)*, Vol. 9198. Springer, 731–743.
- [16] Jayadev Misra and David Gries. 1982. Finding Repeated Elements. *Sci. Comput. Program.* 2, 2 (1982), 143–152.
- [17] Stuart J. Russell and Peter Norvig. 2010. *Artificial Intelligence - A Modern Approach*. Pearson Education.
- [18] Jeffrey Scott Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (1985), 37–57.
- [19] David P. Woodruff. 2016. New Algorithms for Heavy Hitters in Data Streams (Invited Talk). In *ICDT (LIPIcs)*, Vol. 48. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 4:1–4:12.
- [20] Yandex. 2013. Yandex Personalized Web Search Challenge. <https://www.kaggle.com/c/yandex-personalized-web-search-challenge>. (2013).
- [21] Guizhen Yang. 2004. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD*. ACM, 344–353.