Fast Approximation of Empirical Entropy via Subsampling

Chi Wang
wang.chi@microsoft.com
Microsoft Research
Redmond, Washington

Bailu Ding
badin@microsoft.com
Microsoft Research
Redmond, Washington

ABSTRACT

Empirical entropy refers to the information entropy calculated from the empirical distribution of a dataset. It is a widely used aggregation function for knowledge discovery, as well as the foundation of other aggregation functions such as mutual information. However, computing the exact empirical entropy on a large-scale dataset can be expensive. Using a random subsample, we can compute an approximation of the empirical entropy efficiently. We derive probabilistic error bounds for the approximation, where the error bounds reduce in a near square root rate with respect to the subsample size. We further study two applications which can benefit from the error-bounded approximation: feature ranking and filtering based on mutual information. We develop algorithms to progressively subsample the dataset and return correct answers with high probability. The sample complexity of the algorithms is independent of data size. The empirical evaluation of our algorithms on large-scale real-world datasets demonstrates up to three orders of magnitude speedup over exact methods with almost no error.

ACM Reference Format:
Chi Wang and Bailu Ding. 2019. Fast Approximation of Empirical Entropy via Subsampling. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330938

1 INTRODUCTION

Data scientists often query a large dataset for an aggregated answer during data exploration. As the data volume has been growing rapidly in the past decade, computing such an aggregated answer over large-scale datasets can be expensive. Subsampling is an attractive technique to speed up aggregation queries over Big Data [7]. Instead of computing an exact answer using the full data, which is slow and expensive, using subsampling can provide an approximate and fast answer with a random subset of the data. A slew of new data exploration algorithms and systems in the Big Data era [21, 32, 38, 40] show that subsampling can be used to sharply reduce the processing cost while maintaining guarantees on accuracy using theoretical error bound analysis.

In this work, we study the case of approximating a new aggregation function that is prevalent in data mining: empirical entropy. *Information entropy* for a discrete random variable X with probability mass function $P(X)$ is defined as $H(X) = E[-\log P(X)]$. When calculating the entropy from a large dataset, the most typical approach is to use the empirical distribution of X. We refer to the entropy calculated with this approach as *empirical entropy*. Empirical entropy is used in a wide variety of knowledge discovery tasks. For example, Entropy/IP [12] computes empirical entropy of each bit in 3.5 billion IPv6 addresses to discover Internet address structure. Decision tree learning heavily relies on entropy-based splitting criterion evaluation [3, 30]. In addition, entropy-based aggregation functions such as *mutual information* have also been used as criteria for feature selection and feature transformation [20, 24, 28], objectives of clustering [4], and test criteria for graphical model structure learning [6].

Given a large dataset D, computing the exact empirical entropy $\mathcal{H}_D$ is expensive. Fortunately, for many downstream tasks in data exploration, an error-bounded approximation of empirical entropy is often sufficient. For example, if the downstream task needs to compare the empirical mutual information of two variables with a threshold [20], we can compute the lower bound and the upper bound of the empirical mutual information based on the error-bounded approximation of empirical entropy, and it is sufficient to know if the lower bound is above the threshold or the upper bound is below the threshold. To compare two empirical mutual information values, it is sufficient to know if the lower bound of one of them is higher than the upper bound of the other.

We aim to analytically bound the difference between $\mathcal{H}_D$ and the empirical entropy $\mathcal{H}_S$ calculated from a random subset S of D, without actually computing $\mathcal{H}_D$. The error bounds reduce as the size of the subsample increases. Note that our goal is different from the long existing effort of *information entropy estimation*, i.e., using D to estimate the true entropy of an underlying distribution, assuming that D consists of independent samples generated from that distribution [1, 19, 26, 27, 35, 36, 39]. Their primary focus is to address the scarcity of observations and unbounded support size, and their method is not applicable to our problem as we consider subsampling from a finite dataset D, regardless of how D is obtained. See Section 6 for more discussion about related work.

In this work, we make the following contributions:

• We derive probabilistic bounds for empirical entropy based on approximation using a random subsample, where the probabilistic bounds for the approximation error reduce in a near square root rate with respect to the subsample size. (Section 3)

• We study two representative applications that benefit from the error-bounded approximation of empirical entropy: feature ranking and filtering queries based on mutual information. We develop algorithms to progressively sample data and return correct answers with high probability based on the error bounds. We further prove that their sample complexity is independent of data size. (Section 4)

KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330938

• We empirically evaluate our algorithms with large-scale real-world datasets. Our methods achieve up to 1000× speedup over exact methods with almost no error. (Section 5)

2 PRELIMINARIES

In this section, we first give an overview of empirical entropy and entropy-based aggregation functions, including mutual information, conditional entropy, variation of information, and conditional mutual information. We then introduce a specific concentration inequality in our problem context, which will be used to develop theoretical results in Section 3.

2.1 Entropy for a Single Variable

Consider a sequence D of N elements, corresponding to the observations of a random discrete variable X. X takes integer values from 1 to c, or we say the *support* of X is $[c]$. Let n_i be the number of elements in D with value $i \in [c]$. We have $N = \sum_{i=1}^{c} n_i$. The *empirical entropy* $\mathcal{H}_D(X)$ is defined as:

$$\mathcal{H}_D(X) = -\sum_{i=1}^{c} \frac{n_i}{N} \log \frac{n_i}{N} \tag{1}$$

Let S be a random subsample of D of size M. Let m_i be the number of elements in S with value $i \in [c]$. We have $M = \sum_{i=1}^{c} m_i$. The empirical entropy $\mathcal{H}_S(X)$ over the subsample S is:

$$\mathcal{H}_S(X) = -\sum_{i=1}^{c} \frac{m_i}{M} \log \frac{m_i}{M} \tag{2}$$

Our goal is to use $\mathcal{H}_S(X)$ to approximate $\mathcal{H}_D(X)$. In the following sections, we will omit the random variable X when there is no ambiguity.

2.2 Multivariate Entropy-based Aggregations

Th definition of empirical entropy can be generalized to multiple discrete variables. For example, for two discrete variables X_1 and X_2, with support $[c_1]$ and $[c_2]$ respectively, D is a sequence of N pairs of observed values for these two variables. We denote the number of pairs in D with value i for X_1 and value j for X_2 as $n_{i,j}$, and $N = \sum_{i=1}^{c_1} \sum_{j=1}^{c_2} n_{i,j}$. The empirical entropy $\mathcal{H}_D(X_1, X_2)$ is defined as:

$$\mathcal{H}_D(X_1, X_2) = -\sum_{i=1}^{c_1} \sum_{j=1}^{c_2} \frac{n_{i,j}}{N} \log \frac{n_{i,j}}{N} \tag{3}$$

With multivariate empirical entropy, we can define a number of entropy-based aggregation functions. The empirical *mutual information* of two variables $\mathcal{I}_D(X_1, X_2)$ is defined as:

$$\mathcal{I}_D(X_1, X_2) = \mathcal{H}_D(X_1) + \mathcal{H}_D(X_2) - \mathcal{H}_D(X_1, X_2) \tag{4}$$

The empirical *conditional entropy* of X_2 given X_1 is defined as:

$$\mathcal{H}_D(X_2|X_1) = \mathcal{H}_D(X_1, X_2) - \mathcal{H}_D(X_1) \tag{5}$$

The empirical *variation of information* between X_1 and X_2 is defined as:

$$\mathcal{VI}_D(X_1, X_2) = \mathcal{H}_D(X_1) + \mathcal{H}_D(X_2) - 2\mathcal{I}_D(X_1, X_2) \tag{6}$$

The empirical *conditional mutual information* of X_1 and X_2 given X_3 is defined as:

$$\mathcal{I}_D(X_1, X_2|X_3) = \mathcal{H}_D(X_1|X_3) + \mathcal{H}_D(X_2|X_3) - \mathcal{H}_D(X_1, X_2|X_3) \tag{7}$$

2.3 Concentration Inequality

Hoeffding's inequality [16] and McDiarmid's inequality [25] are two common concentration inequalities used in probabilistic bound analysis. Since the entropy function cannot be expressed as a mean of samples, Hoeffding's inequality does not apply. Since we consider sampling from a finite population D without replacement, McDiarmid's inequality is neither applicable. The main theoretical gadget we employ in this work is a concentration inequality developed by El-Yaniv and Pechyony [11] for a transductive learning problem. Part of our contribution is to identify the applicability of this concentration inequality to empirical entropy approximation. We restate the inequality for our problem context, which will be used to derive the probabilistic error bounds of our approximation.

A random subsample S of size M can be obtained in the following process. First, we shuffle the sequence D randomly. Second, we take the first M elements from the shuffled sequence. We index all the elements in D from 1 to N. The shuffling process corresponds to a permutation of integers 1 to N (denoted as I_1^N). We define $\mathbf{Z} \triangleq (Z_j)_{j=1}^N$ as the random permutation vector where the variable $Z_j \in [N]$ indicates the index of the j-th element in the shuffled sequence. The subsample $S(\mathbf{Z})$ consists of the M elements from D with indices $\{Z_j\}_{j=1}^M$. Now we can define a function f on the permutations I_1^N as:

$$f(\mathbf{Z}) = f(Z_1, \dots, Z_N) = \mathcal{H}_{S(\mathbf{Z})}(X) \tag{8}$$

Let $\mathbf{Z}^{ij}$ be a perturbed permutation vector obtained by exchanging Z_i and Z_j in $\mathbf{Z}$. From Eq. (2), we know that f is symmetric on $Z_1, \dots, Z_M$ as well as $Z_{M+1}, \dots, Z_N$. Such a function is called $(M, N-M)$ symmetric permutation function.

Let $\kappa(M, N) = (N-M)^2 \sum_{j=N-M+1}^{N} \frac{1}{j^2}$. It can be verified that $\kappa(M, N) < \frac{M(N-M)}{N} < M$. The following lemma is a restatement of Lemma 2 in [11].

Lemma 2.1. *Let $\mathbf{Z}$ be a random permutation vector. Let $f(\mathbf{Z})$ be an $(M, N-M)$-symmetric permutation function satisfying $|f(\mathbf{Z}) - f(\mathbf{Z}^{ij})| \le \beta$ for all $i \in I_1^M, j \in I_{M+1}^N$. Then*

$$P_{\mathbf{Z}}\{f(\mathbf{Z}) - E_{\mathbf{Z}}[f(\mathbf{Z})] \ge \epsilon\} \le \exp\left(-\frac{\epsilon^2}{2\beta^2 \kappa(M, N)}\right) \tag{9}$$

3 THEORETICAL BOUNDS

Given a random subsample $S \subset D, |S| = M$, we aim to bound the value of $\mathcal{H}_D(X)$ with high probability. For a given significance threshold $\alpha \in (0, 0.5)$, we would like to compute $\underline{\mathcal{H}}_\alpha$ and $\overline{\mathcal{H}}_\alpha$ from S, such that with probability at least $1 - 2\alpha$, $\mathcal{H}_D(X) \in [\underline{\mathcal{H}}_\alpha, \overline{\mathcal{H}}_\alpha]$. The bounds are probabilistic, and $[\underline{\mathcal{H}}_\alpha, \overline{\mathcal{H}}_\alpha]$ is a *confidence interval* for $\mathcal{H}_D(X)$. The confidence intervals of other entropy-based aggregation functions can be defined similarly.

3.1 Entropy

Theorem 3.1. *Given $\alpha \in (0,1)$, with probability at least $1-\alpha$,*

$$\mathcal{H}_D \geq \underline{\mathcal{H}}_\alpha \triangleq \mathcal{H}_S - \sqrt{\frac{8(N-M)\log\frac{1}{\alpha}}{MN}}\log M$$

Given $\alpha \in (0,1)$, with probability at least $1-\alpha$,

$$\mathcal{H}_D \leq \overline{\mathcal{H}}_\alpha \triangleq \mathcal{H}_S + \sqrt{\frac{8(N-M)\log\frac{1}{\alpha}}{MN}}\log M + \log\left(1+\frac{(c-1)(N-M)}{M(N-1)}\right)$$

Given $\alpha \in (0, 0.5)$, with probability at least $1-2\alpha$, $\mathcal{H}_D \in [\underline{\mathcal{H}}_\alpha, \overline{\mathcal{H}}_\alpha]$.

Proof. Our proof makes use of Eq. (9) in Lemma 2.1. Following Section 2.3, $f(\mathbf{Z}) = \mathcal{H}_{S(\mathbf{Z})}$ is a $(M, N-M)$ symmetric permutation function. We first notice that

$$\begin{aligned} E_\mathbf{Z}[f(\mathbf{Z})] &= E_S[\mathcal{H}_S] \\ P_\mathbf{Z}\{f(\mathbf{Z}) - E_\mathbf{Z}[f(\mathbf{Z})] \geq \epsilon\} &= P_S\{\mathcal{H}_S - E_S[\mathcal{H}_S] \geq \epsilon\} \end{aligned} \tag{10}$$

To apply Eq. (9), we show that $|f(\mathbf{Z}) - f(\mathbf{Z}^{ij})|$ can be bounded by $\frac{2\log M}{M}$. Without loss of generality, we assume that the i-th element in D has value 1 and the j-th element in D has value 2. We have

$$\begin{aligned} |f(\mathbf{Z}) - f(\mathbf{Z}^{ij})| = \Big| &\frac{m_1}{M}\log\frac{m_1}{M} + \frac{m_2}{M}\log\frac{m_2}{M} \\ &- \frac{m_1-1}{M}\log\frac{m_1-1}{M} - \frac{m_2+1}{M}\log\frac{m_2+1}{M}\Big| \\ =&\frac{1}{M}|m_1\log m_1 + m_2\log m_2 \\ &-(m_1-1)\log(m_1-1) - (m_2+1)\log(m_2+1)| \end{aligned} \tag{11}$$

Let

$$\begin{aligned} \Delta(m_1,m_2) \triangleq & m_1\log m_1 + m_2\log m_2 \\ &-(m_1-1)\log(m_1-1) - (m_2+1)\log(m_2+1) \end{aligned} \tag{12}$$

Δ is monotonically increasing with m_1 and decreasing with m_2. So

$$\begin{aligned} |f(\mathbf{Z}) - f(\mathbf{Z}^{ij})| &\leq \frac{1}{M}\max\{\Delta(1, M-1), \Delta(M, 0)\} \\ &= \log\frac{M}{M-1} + \frac{\log(M-1)}{M} < \frac{2\log M}{M} \end{aligned} \tag{13}$$

Applying Eq. (9) and (10), we have

$$P_S\{\mathcal{H}_S - E_S[\mathcal{H}_S] \geq \epsilon\} \leq \exp\left(-\frac{\epsilon^2M^2}{8\log^2 M\kappa(M,N)}\right) \tag{14}$$

The above process can be repeated for $-f(\mathbf{Z}) = -\mathcal{H}_{S(\mathbf{Z})}$, which yields:

$$P_S\{E_S[\mathcal{H}_S] - \mathcal{H}_S \geq \epsilon\} \leq \exp\left(-\frac{\epsilon^2M^2}{8\kappa(M,N)\log^2 M}\right) \tag{15}$$

To bound $\mathcal{H}_D$, we now compute the difference between $E_S[\mathcal{H}_S]$ and $\mathcal{H}_D$. For convenience, we denote $p_i = m_i/M, q_i = n_i/N$. In our case, m_i is the number of value i in sample S, where S is a random subsample of D. We can characterize m_i using a *hypergeometric distribution*. Hypergeometric distribution describes the probability of seeing m_i elements with value i in M draws, from a finite population of size N that contains n_i elements with value i. The variance of m_i is $\frac{Mn_i(N-n_i)(N-M)}{N^2(N-1)}$, and the expectation of m_i is $\frac{Mn_i}{N}$. We have $E_S(p_i) = q_i, \forall i \in [c]$.

$$\begin{aligned} \mathcal{H}_D - E_S[\mathcal{H}_S] &= E_S\left[\sum_{i=1}^{c}(p_i\log p_i - q_i\log q_i)\right] \\ =&E_S\left[\sum_{i=1}^{c}(p_i\log p_i - p_i\log q_i + p_i\log q_i - q_i\log q_i)\right] \\ =&E_S\left[\sum_{i=1}^{c}(p_i\log\frac{p_i}{q_i} + (p_i-q_i)\log q_i)\right] = E_S[d_{KL}(\mathbf{p},\mathbf{q})] \end{aligned} \tag{16}$$

where d_{KL} refers to the Kullback-Leibler divergence. From Gibbs and Su [14], we know:

$$0 \leq d_{KL}(\mathbf{p},\mathbf{q}) \leq \log\left(1 + \sum_{i=1}^{M}\frac{(p_i-q_i)^2}{q_i}\right)$$

Taking expectation over it and using Jensen's inequality [18], we have:

$$\begin{aligned} 0 \leq E_S[d_{KL}(\mathbf{p},\mathbf{q})] &\leq E_S\left[\log\left(1+\sum_{i=1}^{c}\frac{(p_i-q_i)^2}{q_i}\right)\right] \\ &\leq \log\left(1 + E_S\left[\sum_{i=1}^{c}\frac{(p_i-q_i)^2}{q_i}\right]\right) \\ &= \log\left(1+\sum_{i=1}^{c}\left(E\left[\frac{m_i^2}{M^2q_i}\right] - 2E[p_i] + q_i\right)\right) \\ &= \log\left(1+\sum_{i=1}^{c}\frac{Var[m_i]+E[m_i]^2}{M^2q_i} - 1\right) \\ &= \log\left(\sum_{i=1}^{c}(\frac{(1-q_i)(N-M)}{(N-1)M} + q_i)\right) \\ &= \log\left(1+\frac{(c-1)(N-M)}{M(N-1)}\right) \triangleq b \end{aligned} \tag{17}$$

Combining Eq. (14)-(17), we have:

$$P_S\{\mathcal{H}_D \leq \mathcal{H}_S - \epsilon\} \leq \exp\left(-\frac{\epsilon^2M^2}{8\kappa(M,N)\log^2 M}\right) \tag{18}$$

$$P_S\{\mathcal{H}_D \geq \mathcal{H}_S + \epsilon + b\} \leq \exp\left(-\frac{\epsilon^2M^2}{8\kappa(M,N)\log^2 M}\right) \tag{19}$$

Setting $\alpha = \exp\left(-\frac{\epsilon^2M^2}{8\kappa(M,N)\log^2 M}\right)$, we have

$$\epsilon = \frac{\log M}{M}\sqrt{-8\kappa(M,N)\log\alpha} < \sqrt{\frac{8(N-M)\log\frac{1}{\alpha}}{MN}}\log M \tag{20}$$

Therefore, with probability at least $1-\alpha$,

$$\mathcal{H}_D \geq \mathcal{H}_S - \sqrt{\frac{8(N-M)\log\frac{1}{\alpha}}{MN}}\log M$$

Similarly, with probability at least $1-\alpha$,

$$\mathcal{H}_D \leq \mathcal{H}_S + \sqrt{\frac{8(N-M)\log\frac{1}{\alpha}}{MN}}\log M + \log\left(1+\frac{(c-1)(N-M)}{M(N-1)}\right)$$

By union bound, with probability at least $1-2\alpha$, $\mathcal{H}_D \in [\underline{\mathcal{H}}_\alpha, \overline{\mathcal{H}}_\alpha]$. ◻

Discussion. From the theorem, we see that the lower bound depends on α, N, M, and the upper bound depends on α, N, M and c. When α, N and c are fixed, the probabilistic error bounds on both sides, $\mathcal{H}_D - \underline{\mathcal{H}}_\alpha$ and $\overline{\mathcal{H}}_\alpha - \mathcal{H}_D$, decrease in a near square root rate, $O(M^{-1/2} \log M)$. The tightness of these bounds is an open question.

3.2 Entropy-based Aggregations

Our probabilistic bounds can be easily generalized to other entropy-based aggregations. Based on the definition of mutual information in Eq. (4), the bounds for $\mathcal{I}_D(X_1, X_2)$ can be simply derived from union bound:

$$\begin{aligned}\overline{\mathcal{I}}_\alpha(X_1,X_2) &= \overline{\mathcal{H}}_{\alpha/3}(X_1) + \overline{\mathcal{H}}_{\alpha/3}(X_2) - \underline{\mathcal{H}}_{\alpha/3}(X_1,X_2)\\ \underline{\mathcal{I}}_\alpha(X_1,X_2) &= \underline{\mathcal{H}}_{\alpha/3}(X_1) + \underline{\mathcal{H}}_{\alpha/3}(X_2) - \overline{\mathcal{H}}_{\alpha/3}(X_1,X_2)\end{aligned} \tag{21}$$

for all $\alpha \in (0,1)$.

Likewise, we have the following bounds for conditional entropy, variation of information, and conditional mutual information.

$$\begin{aligned}\overline{\mathcal{H}}_\alpha(X_2|X_1) &= \overline{\mathcal{H}}_{\alpha/2}(X_1,X_2) - \underline{\mathcal{H}}_{\alpha/2}(X_1)\\ \underline{\mathcal{H}}_\alpha(X_2|X_1) &= \underline{\mathcal{H}}_{\alpha/2}(X_1,X_2) - \overline{\mathcal{H}}_{\alpha/2}(X_1)\\ \overline{\mathcal{VI}}_\alpha(X_1,X_2) &= \overline{\mathcal{H}}_{\alpha/2}(X_1,X_2) - \underline{\mathcal{I}}_{\alpha/2}(X_1,X_2)\\ \underline{\mathcal{VI}}_\alpha(X_1,X_2) &= \underline{\mathcal{H}}_{\alpha/2}(X_1,X_2) - \overline{\mathcal{I}}_{\alpha/2}(X_1,X_2)\\ \overline{\mathcal{I}}_\alpha(X_1,X_2|X_3) &= \overline{\mathcal{H}}_{\alpha/3}(X_1|X_3) + \overline{\mathcal{H}}_{\alpha/3}(X_2|X_3) - \underline{\mathcal{H}}_{\alpha/3}(X_1,X_2|X_3)\\ \underline{\mathcal{I}}_\alpha(X_1,X_2|X_3) &= \underline{\mathcal{H}}_{\alpha/3}(X_1|X_3) + \underline{\mathcal{H}}_{\alpha/3}(X_2|X_3) - \overline{\mathcal{H}}_{\alpha/3}(X_1,X_2|X_3)\end{aligned}$$

4 APPLICATION

Empirical entropy-based aggregation functions are commonly used as information-theoretic criteria for measuring feature's importance in a variety of applications. For example, mutual information can be used to measure the relevance of a feature with respect to a target column for prediction [23]. As another example, mutual information can also be used to evaluate information gain when adding a feature as a data split criterion for decision tree learning [30]. In addition, conditional mutual information is used to measure the redundancy between features [5]. We observe that, in these applications, the empirical entropy is used to answer the following primitive queries:

- Ranking. Find the highest entropy-based aggregation score among a set of scores. For example, in sequential forward feature selection [37] and decision tree learning [3], an algorithm needs to repeatedly query which feature has the highest score. Each query is evaluated on different conditions or subsets. A generalization is to find top-K scores as used in [20].
- Filtering. Determine whether an entropy-based aggregation score is above or below a threshold, such that the corresponding feature can be selected or filtered. For example, Kaul et al. [20] uses this primitive to preprocess the features and decide which features are promising for deriving engineered features. The engineered features are then filtered again based on their mutual information.

To answer both queries exactly, it requires computing all the scores to rank or filter. To reduce the cost, we can use the approximation with bounded error to answer the queries while ensuring the correctness of the answers with high probability.

Next, we will describe two algorithms to answer ranking and filtering queries respectively, and analyze their theoretical guarantees.

4.1 Ranking

Consider a query asking for the top K scores among C entropy-based aggregation scores. Without loss of generality, assume each score is empirical mutual information computed from N pairs of observations. We denote the C mutual information scores as $\{\mathcal{I}_a\}_{a=1}^C$. The ranking problem is formulated as the following: Given a positive integer K, return the top-K indices $\mathcal{A}^* \subset [C]$, such that $\forall a \in \mathcal{A}^*, \forall a' \in [C] \setminus \mathcal{A}^*, \mathcal{I}_a \geq \mathcal{I}_{a'}$.

For each $a \in [C]$, computing the exact score $\mathcal{I}_a$ requires scanning N pairs of observations and collecting a count matrix $\{n_{i,j}\}$ (ref. Eq. (4)). The exact computation of all the scores requires $O(NC)$ cost. To reduce the cost and answer the top-K query with high confidence, we can leverage subsample-based approximation to differentiate the top-K from the rest of the scores. If the minimal lower confidence bound of K scores is higher than the maximal upper confidence bound of the rest of the scores, we can answer the query with high confidence. Based on the theoretical results in Section 3, the larger is the size of the subsample used to approximate a score, the narrower is the confidence interval for that score. In the meantime, large subsample size incurs high computational cost, which is what we aim to avoid in the beginning. So it is most desirable to use as small subsample size as possible, as long as the top-K scores can be differentiated from the rest with high probability. Since such minimal sufficient subsample size is unknown *a priori*, we progressively sample the data until the confidence of identifying top-K is sufficiently high.

We present our fast ranking algorithm EntropyRank (Algorithm 1), leveraging the confidence intervals derived in Section 3. For each $a \in [C]$, we partition all the N pairs of observations required to compute $\mathcal{I}_a$ into $(B+1)$ random batches, such that the first batch is small, and the remaining B batches are of the same size. We use $\mathcal{M}(a, x)$ to represent the count matrix $\{m_{i,j}\}$ collected from batch $x, x \in [B+1]$. Each batch is a random subsample. For a batch of size M, $\{m_{i,j}/M\}$ is an approximation of $\{n_{i,j}/N\}$.

Let $[\underline{\mathcal{I}_a}, \overline{\mathcal{I}_a}]$ be the confidence interval for a fixed α. We define

$$\widetilde{\mathcal{I}_a}(\mathcal{A}) = \begin{cases}\overline{\mathcal{I}_a} & a \notin \mathcal{A}\\ \underline{\mathcal{I}_a} & a \in \mathcal{A}\end{cases}$$

That is, $\widetilde{\mathcal{I}_a}$ takes the lower bound if a is in $\mathcal{A}$, and upper bound otherwise.

EntropyRank starts by loading the first batch and computing the count matrix $M[a] \leftarrow \mathcal{M}(a, 1)$ for each $a \in [C]$ (line 2). Since the size of the first batch is small, it is fast to compute $\mathcal{M}(a, 1)$. In each following iteration, it selects an index a^* to load one more batch of data (line 5-7). At each iteration, $\mathcal{A}$ is the current top-K indices based on the center of the confidence intervals of each score (line 5). It is not certain that $\mathcal{A}$ is the true top-K, but we know that the true score for $a \in [C]$ falls into the confidence interval $[\underline{\mathcal{I}_a}, \overline{\mathcal{I}_a}]$ with high probability. $\mathcal{A}'$ represents the top-K indices that can potentially become true top-K. Consider a very unlucky case: We had overestimated the score for all the indices in $\mathcal{A}$, and underestimated the score for all the other indices. Then $\mathcal{A}'$

Algorithm 1: EntropyRank

```
   Input: $K$
   Output: Top-$K$ indicies $\mathcal{A} \subset [C]$
1  for $a \in [C]$ do
2      $M[a] \leftarrow \mathcal{M}(a, x_a \leftarrow 1)$;
3      Estimate $[\underline{\mathcal{I}_a}, \overline{\mathcal{I}_a}]$;
4  repeat
5      $\mathcal{A} \leftarrow$ top-K indices ranked by $\overline{\mathcal{I}} + \underline{\mathcal{I}}$;
6      $\mathcal{A}' \leftarrow$ top-K indices ranked by $\widetilde{\mathcal{I}}(\mathcal{A})$;
7      $a^* \leftarrow \arg\max_{a \in \mathcal{A} \oplus \mathcal{A}'}(\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a})$;
8      $x_{a^*} \leftarrow x_{a^*} + 1$;
9      $M[a] \leftarrow M[a] + \mathcal{M}(a^*, x_{a^*})$;
10     Update $[\underline{\mathcal{I}_{a^*}}, \overline{\mathcal{I}_{a^*}}]$;
11 until $\mathcal{A} = \mathcal{A}'$;
12 return $\mathcal{A}$
```

Algorithm 2: EntropyFilter

```
   Input: $\eta$
   Output: Indicies $\mathcal{B} \subset [C]$ which pass the filter
1  $\mathcal{B} \leftarrow \emptyset$;
2  for $a \in [C]$ do
3      $M[a] \leftarrow \mathcal{M}(a, x_a \leftarrow 1)$;
4      Estimate $[\underline{\mathcal{I}_a}, \overline{\mathcal{I}_a}]$;
5      while $\eta \in (\underline{\mathcal{I}_a}, \overline{\mathcal{I}_a}]$ do
6          $x_a \leftarrow x_a + 1$;
7          $M[a] \leftarrow M[a] + \mathcal{M}(a, x_a)$;
8          Update $[\underline{\mathcal{I}_a}, \overline{\mathcal{I}_a}]$;
9      if $\eta \le \underline{\mathcal{I}_a}$ then
10         $\mathcal{B} \leftarrow \mathcal{B} \cup \{a\}$
11 return $\mathcal{B}$
```

represents the alternative potential top-K indices (line 6). $\mathcal{A} \oplus \mathcal{A}'$ (i.e., $\mathcal{A} \cup \mathcal{A}' \setminus (\mathcal{A} \cap \mathcal{A}')$) represents the competing indices, some of which can be included in $\mathcal{A}^*$ but not all. Among the competing indices, the algorithm chooses the one with the longest confidence interval to increase its sample size (line 7). With the additional batch of data, it updates the count matrix $M[a^*]$ as well as the confidence interval for a^* (line 8-10). The algorithm terminates when $\mathcal{A} = \mathcal{A}'$, i.e., there is no index in $[C] \setminus \mathcal{A}$ with an upper bound that is higher than any of the lower bounds of the indices in $\mathcal{A}$ (line 11). Finally, we return $\mathcal{A}$ as the top-K indices.

4.2 Filtering

The filtering query follows a similar setup as the ranking query, except that the goal is to find all the indices with scores above a threshold. Again using mutual information as an example, the filtering query is formalized as the following: Given a threshold $\eta \in R$, return the indices $\mathcal{B}^* \subset [C]$, such that $\forall a \in \mathcal{B}^*, \mathcal{I}_a \ge \eta$, and $\forall a' \in [C] \setminus \mathcal{B}^*$, $\mathcal{I}_{a'} < \eta$. We aim to answer this query with high probability of correctness, using subsample-based approximation.

Like the case of ranking, we cannot predetermine the minimal sample size needed for each $a \in [C]$, as the difference between $\mathcal{I}_a$ and η is unknown. We present a progressive subsampling algorithm EntropyFilter, leveraging the confidence intervals derived in Section 3.

EntropyFilter (Algorithm 2) progressively loads the batches to approximate each score and determine whether it is below or above the threshold. If η is below or equal to the current lower bound of $\mathcal{I}_a$, we have high confidence that $\mathcal{I}_a$ can pass the filter (line 9). Likewise, if η is above the current upper bound, we have high confidence that $\mathcal{I}_a$ cannot pass the filter. If neither is true, i.e., η is contained by the current confidence interval (line 5), the algorithm samples more data to narrow the confidence interval (line 6-8).

4.3 Analysis

The cost of EntropyRank and EntropyFilter are determined by the number of batches x_a used for each $a \in [C]$ when the algorithms terminate. Let $X = \sum_{a=1}^{C} x_a$. The total cost is $O(\frac{XN}{B})$. In the worst case, all the batches are used for each $a \in [C]$, i.e., $X = BC$, and the total cost is $O(CN)$.

We prove probabilistic upper bounds of the total cost for the two algorithms. For simplicity of the analysis, we assume the size of each batch is a constant much smaller than N.

EntropyRank. We show that the sample size required by each score is related to the gap between the score and the top-K boundary. We define:

$$\Delta_a \triangleq \begin{cases} \mathcal{I}_a - \mathcal{I}_{[K+1]} & a \in \mathcal{A}^* \\ \mathcal{I}_{[K]} - \mathcal{I}_a & a \notin \mathcal{A}^* \end{cases} \tag{22}$$

where $\mathcal{I}_{[K]}$ is the K-th largest score. So $\mathcal{I}_{[K]}$ and $\mathcal{I}_{[K+1]}$ are the boundaries of the top-K: the indices with score above $\mathcal{I}_{[K+1]}$ are top-K and these with the scores below $\mathcal{I}_{[K]}$ are not. Intuitively, the smaller the gap Δ_a is, the harder it is to decide whether a is in $\mathcal{A}^*$.

Theorem 4.1. *With probability at least $1 - 2\alpha C$, EntropyRank returns the correct top-K set $\mathcal{A}^*$. With probability at least $1 - 2\alpha BC$, the total cost $O(\frac{XN}{B}) = O(\log \alpha^{-1} \sum_{a=1}^{C} \Delta_a^{-2} \log^2 \Delta_a)$.*

Proof. The correctness guarantee can be easily proved by union bound. We focus on proving the upper bound of the cost. During each iteration, $Pr\{\underline{\mathcal{I}_a} \le \mathcal{I}_a \le \overline{\mathcal{I}_a}\} \ge 1 - 2\alpha$. By union bound, all the true scores fall into all the corresponding confidence intervals ever created with probability at least $1 - 2\alpha BC$. In the rest of the proof, we assume that event happens. Under this assumption, the termination condition $\mathcal{A} = \mathcal{A}'$ ensures that the returned $\mathcal{A}$ is the true top-K indices $\mathcal{A}^*$.

We prove by contradiction that $\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}} \ge \frac{\Delta_{a^*}}{2}$ in every iteration. Assume $\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}} < \frac{\Delta_{a^*}}{2}$. We discuss the following 4 cases:

(1) $a^* \in \mathcal{A}^*, a^* \in \mathcal{A} \setminus \mathcal{A}'$. $\Delta_{a^*} = \mathcal{I}_{a^*} - \mathcal{I}_{[K+1]}$. Since $\mathcal{A}' \neq \mathcal{A}^*$, there exists $a \in \mathcal{A}' \setminus \mathcal{A}^*$. If $a \in \mathcal{A}$, $\mathcal{I}_a \ge \underline{\mathcal{I}_a} \ge \underline{\mathcal{I}_{a^*}} \ge \mathcal{I}_{a^*} - (\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}}) > \mathcal{I}_{[K+1]}$. If $a \notin \mathcal{A}$, $\mathcal{I}_a \ge \overline{\mathcal{I}_a} - (\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a}) \ge \underline{\mathcal{I}_{a^*}} - (\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}}) > \mathcal{I}_{[K+1]}$. Both imply $a \in \mathcal{A}^*$ as a contradiction.

(2) $a^* \in \mathcal{A}^*, a^* \in \mathcal{A}' \setminus \mathcal{A}$. $\Delta_{a^*} = \mathcal{I}_{a^*} - \mathcal{I}_{[K+1]}$. Since $\mathcal{A} \neq \mathcal{A}^*$, there exists $a \in \mathcal{A} \setminus \mathcal{A}^*$. If $a \notin \mathcal{A}'$, $\mathcal{I}_a \ge \underline{\mathcal{I}_a} \ge \underline{\mathcal{I}_{a^*}} \ge \mathcal{I}_{a^*} - (\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}}) > \mathcal{I}_{[K+1]}$ which is a contradiction. If $a \in \mathcal{A}'$, we know that

$\mathcal{A}' \neq \mathcal{A}^*$, so there exists $a' \in \mathcal{A}^* \setminus \mathcal{A}'$. If $a' \in \mathcal{A}$, $\mathcal{I}_a \geq \underline{\mathcal{I}_a} \geq \underline{\mathcal{I}_{a'}} \geq \underline{\mathcal{I}_{a^*}} > \mathcal{I}_{[K+1]}$ which is a contradiction. If $a' \notin \mathcal{A}$, $\mathcal{I}_a \geq \underline{\mathcal{I}_a} \geq \overline{\mathcal{I}_{a'}} \geq \overline{\mathcal{I}_{a'}}$. Then it implies $a \in \mathcal{A}^*$, which is a contradiction.

(3) $a^* \notin \mathcal{A}^*, a^* \in \mathcal{A} \setminus \mathcal{A}'$. $\Delta_{a^*} = \mathcal{I}_{[K]} - \mathcal{I}_{a^*}$. Since $\mathcal{A}^* \neq \mathcal{A}$, there exists $a \in \mathcal{A}^* \setminus \mathcal{A}$. If $a \in \mathcal{A}'$, $\mathcal{I}_a \leq \overline{\mathcal{I}_a} \leq \overline{\mathcal{I}_{a^*}} \leq \mathcal{I}_{a^*} + (\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}}) < \mathcal{I}_{[K]}$ which is a contradiction. If $a \notin \mathcal{A}'$, since $\mathcal{A}' \neq TrueTopK$, there exists $a' \in \mathcal{A}' \setminus \mathcal{A}^*$. If $a' \notin \mathcal{A}$, $\mathcal{I}_a \leq \overline{\mathcal{I}_a} \leq \overline{\mathcal{I}_{a'}} \leq \overline{\mathcal{I}_{a^*}} < \mathcal{I}_{[K]}$ which is a contradiction. If $a' \in \mathcal{A}$, $\mathcal{I}_a \leq \overline{\mathcal{I}_a} \leq \underline{\mathcal{I}_{a'}} \leq \mathcal{I}_{a'}$. Then it implies $a \notin \mathcal{A}^*$, which is a contradiction.

(4) $a^* \notin \mathcal{A}^*, a^* \in \mathcal{A}' \setminus \mathcal{A}$. $\Delta_{a^*} = \mathcal{I}_{[K]} - \mathcal{I}_{a^*}$. Since $\mathcal{A}^* \neq \mathcal{A}'$, there exists $a \in \mathcal{A}^* \setminus \mathcal{A}'$. If $a \in \mathcal{A}$, $\mathcal{I}_a \leq \overline{\mathcal{I}_a} - \underline{\mathcal{I}_a} + \underline{\mathcal{I}_a} \leq \overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}} + \overline{\mathcal{I}_{a^*}} < \mathcal{I}_{[K]}$. If $a \notin \mathcal{A}$, $\mathcal{I}_a \leq \overline{\mathcal{I}_a} \leq \overline{\mathcal{I}_{a^*}} \leq \overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}} + \mathcal{I}_{a^*} < \mathcal{I}_{[K]}$. Both imply $a \notin \mathcal{A}^*$, which is a contradiction.

Thus, all the 4 cases lead to contradiction. So $\overline{\mathcal{I}_{a^*}} - \underline{\mathcal{I}_{a^*}} \geq \frac{\Delta_{a^*}}{2}$. That implies $\forall a \in [C]$, x_a will not increase as soon as

$$\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a} < \frac{\Delta_a}{2} \tag{23}$$

because for x_a to increase, a must be chosen as a^* in some iteration.

Our approximation error bounds in Section 3 imply there exists a constant C_0 s.t.

$$\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a} < \log m[a] \sqrt{\frac{C_0 \log \frac{1}{\alpha}}{m[a]}} \tag{24}$$

where $m[a] = \frac{N x_a}{B}$. From Eq. (23) and (24), we know that x_a will not increase as soon as

$$\log m[a] \sqrt{\frac{C_0 \log \frac{1}{\alpha}}{m[a]}} \leq \frac{\Delta_a}{2} \Leftrightarrow m[a] \geq \frac{4C_0 \log^2 m[a] \log \frac{1}{\alpha}}{\Delta_a^2} \tag{25}$$

Since $m[a] = \frac{N x_a}{B}$, we have

$$x_a \leq \frac{16BC_0 \log \frac{1}{\alpha}}{N\Delta_a^2} \log^2 \frac{4C_0 \log \frac{1}{\alpha}}{\Delta_a^2} \leq \left\lceil \frac{16BC_0 \log \frac{1}{\alpha}}{N\Delta_a^2} \log^2 \frac{4C_0 \log \frac{1}{\alpha}}{\Delta_a^2} \right\rceil \tag{26}$$

Summing up for all $a \in [C]$, we have:

$$X = \mathrm{O}\left(C + \frac{B \log \frac{1}{\alpha}}{N} \sum_{a \in [C]} \Delta_a^{-2} \log^2 \Delta_a \right)$$

i.e., the total cost is $\mathrm{O}(\frac{XN}{B}) = \mathrm{O}(\log \alpha^{-1} \sum_{a \in [C]} \Delta_a^{-2} \log^2 \Delta_a)$. □

EntropyFilter. The sample size required by each score is related to the gap between the score and the threshold η. We define:

$$\delta_a = |\mathcal{I}_a - \eta| \tag{27}$$

Intuitively, the smaller is the gap δ_a, the harder it is to decide whether $\mathcal{I}_a \geq \eta$.

Theorem 4.2. *With probability at least $1 - \alpha C$, EntropyFilter returns the correct answer $\mathcal{B}^*$. With probability at least $1 - \alpha C$, the total cost $\mathrm{O}(\frac{XN}{B}) = \mathrm{O}(\log \alpha^{-1} \sum_{a=1}^{C} \delta_a^{-2} \log^2 \delta_a)$.*

Proof. (Sketch) The correctness guarantee is provable via union bound. To prove the upper bound of the cost, we first prove that with probability $1 - 2\alpha$, x_a does not increase as soon as:

$$\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a} < \frac{\delta_a}{2} = \frac{|\mathcal{I}_a - \eta|}{2} \tag{28}$$

To prove that, we notice that Eq. (28) implies either $\mathcal{I}_a - \eta > 2(\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a})$ or $\eta - \mathcal{I}_a > 2(\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a})$. If the former is true, then

$$\eta < \mathcal{I}_a - 2(\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a}) \leq 2\underline{\mathcal{I}_a} - \overline{\mathcal{I}_a} \leq \underline{\mathcal{I}_a} \tag{29}$$

with probability at least $1 - \alpha$. If the latter is true, then

$$\eta > \mathcal{I}_a + 2(\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a}) \geq 2\overline{\mathcal{I}_a} - \underline{\mathcal{I}_a} \geq \overline{\mathcal{I}_a} \tag{30}$$

with probability at least $1 - \alpha$. Both imply that the condition in line 5 of Algorithm 2 is false. Therefore, x_a will not be further increased with probability at least $1 - \alpha$.

The remainder of the proof is similar to the proof of Theorem 4.1. □

Discussion. The cost bounds are not used for precomputing the cost required to answer a query, since Δ and δ are unknown. Rather, they are used to show that the upper bounds of the cost are independent of the size N of the data. That is a distinguishing character from the exact method whose complexity grows linearly with N.

5 EXPERIMENTS

In this section, we evaluate our algorithms for two applications: ranking and filtering by mutual information. By evaluating the effectiveness of EntropyRank and EntropyFilter in producing fast and accurate answers, we aim to understand whether the subsample-based theoretical bounds are useful in practice.

5.1 Setup

Since approximate answer is most relevant for Big Data scenario, we use the largest datasets we can find from Kaggle[1] for evaluation. Among all the datasets with tabular format that are publicly available from Kaggle as of June, 2018, we evaluate our algorithms on all the large-scale datasets, i.e., datasets with more than half a million records and more than a hundred columns. Table 1 shows the information of the datasets. For each dataset, we filtered out all the columns that are not categorical using heuristics, i.e., removing columns with more than 10000 distinct values.

Table 1: Dataset description

ID	Name	Rows	Columns
1	bls_american-time-use	942073	232
2	cdc_behavioral-risk	506467	394
3	census-american_hus	2972991	226
4	census-american_pus	9412410	279
5	gbonesso_enem-2016	8620630	159

To generate the ranking and filtering queries, for each dataset, we create a test case by randomly choosing a categorical column as the target variable. We repeat the process 5 times to select 5 target variables. For ranking queries, we vary K from 1 to 10; and we vary the threshold η from 0.1 to 0.5 for filtering queries. For each query, we compute the exact mutual information score with respect to the

[1]www.kaggle.com

Table 2: Speedup ratio and sampling rate of EntropyRank and EntropyFilter vs. Exact per dataset

(a) EntropyRank

	Speedup ratio (×)				Sampling rate			
ID	K=1	K=3	K=5	K=10	K=1	K=3	K=5	K=10
1	21.8	13.5	10.2	7.5	5.5%	10%	15%	28%
2	25.1	19.3	17.2	21.0	4.4%	9.2%	9.2%	9.4%
3	17.8	17.2	13.8	10.0	7.2%	8.0%	10%	21%
4	**285**	142	227	116	**0.4%**	0.9%	0.9%	1.7%
5	87.2	120	52.1	10.4	2.0%	3.3%	4.8%	8.4%

(b) EntropyFilter

	Speedup ratio (×)				Sampling rate			
ID	η=.2	η=.3	η=.4	η=.5	η=.2	η=.3	η=.4	η=.5
1	15.2	21.9	22.9	23.0	4.5%	4.5%	4.5%	4.5%
2	34.9	43.8	42.5	51.8	1.6%	1.5%	1.5%	1.3%
3	29.7	27.0	36.6	36.0	2.7%	2.9%	2.5%	2.5%
4	441	465	446	**482**	**0.1%**	**0.1%**	**0.1%**	**0.1%**
5	317	317	320	322	0.2%	0.2%	0.2%	0.2%

target variable for each non-target variable, and rank or filter them according to K or η. We refer to this baseline approach as Exact. We implement EntropyRank, EntropyFilter, and Exact in C#. All the experiments are run on a machine with Intel Xeon CPU E5-2650 v4@2.20GHz, 192GB memory, and 64-bit Windows Server 2016.

We compare the running time between the exact approach and the subsampling approach. For ranking, we measure the accuracy of the returned top-K answer by EntropyRank, $\mathcal{A}$, compared to the true top-K answer from Exact, $\mathcal{A}^*$, i.e., $\frac{|\mathcal{A}\cap\mathcal{A}^*|}{K}$. For filtering, we measure the accuracy whether each feature is correctly filtered by EntropyFilter compared to the true decision made by Exact, i.e., $1 - \frac{|\mathcal{B}\oplus\mathcal{B}^*|}{C}$.

5.2 Performance Comparison

Efficiency for ranking. On each dataset, we report the speedup ratio and sampling rate of EntropyRank per K from 1 to 10. Each metric is averaged over 5 test cases. Table 2a shows the results for four representative K values for clarity. EntropyRank computes the top-1 feature by 285× faster than Exact in the largest dataset #4, using 0.4% of the data. Overall, the speedup ratio reduces as K becomes larger, though not monotonically. The gap of the scores between lower-ranked features is generally smaller, and the sampling rate increases as analyzed in Theorem 4.1.

Table 3a shows the distribution of the speedup ratios across all the test cases. For half of the test cases, the ranking is accelerated by at 27-374× when $K = 1$, and 10-149× when $K = 10$.

Efficiency for filtering. Table 2b shows the speedup ratio and sampling rate of EntropyFilter on each dataset per η from 0.2 to 0.5. EntropyFilter filters the columns with mutual information by more than 400× faster than Exact in the largest dataset #4, using 0.1% of the data.

Table 3: Speedup ratio (×) of EntropyRank and EntropyFilter vs. Exact at different percentiles, and overall accuracy

(a) EntropyRank

K	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Accuracy
1	10.6	15.4	17.0	22.9	27.0	31.5	39.0	214	274	374	**100.0%**
3	6.0	8.0	11.0	15.8	16.2	21.6	35.9	97.1	156	380	**100.0%**
5	3.6	5.8	6.8	9.8	13.7	25.3	26.5	37.4	170	**410**	**100.0%**
10	1.8	4.7	6.5	7.5	10.8	13.5	18.9	30.5	128	149	**100.0%**

(b) EntropyFilter

η	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Accuracy
0.1	19.6	22.6	23.2	29.2	41.9	50.1	121	284	474	**1050**	99.88%
0.2	14.9	18.7	24.9	33.2	35.1	45.9	307	322	448	585	**100.0%**
0.3	20.8	22.2	25.1	31.3	34.9	56.5	317	322	454	494	99.99%
0.4	21.7	25.8	25.9	35.6	40.2	55.2	316	356	437	458	99.91%
0.5	22.4	24.9	30.5	39.5	56.3	60.8	316	336	470	513	99.98%

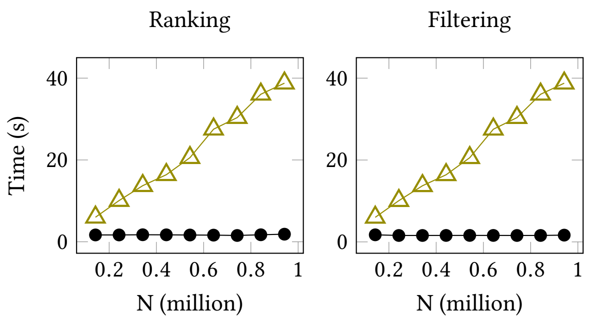
Table 3b shows the distribution of speedup ratios across all the test cases. For 90% of the test cases, the filtering is accelerated by 20-1050× when $\eta = 0.1$, and 22-513× when $\eta = 0.5$.

Accuracy. The last column in Table 3 shows the overall accuracy of EntropyRank and EntropyFilter. EntropyRank makes no single mistake in all the test cases with 100% accuracy, and the accuracy of EntropyFilter is very close to 100%. This validates Theorem 4.1 and 4.2 which guarantee the result is correct with high probability.

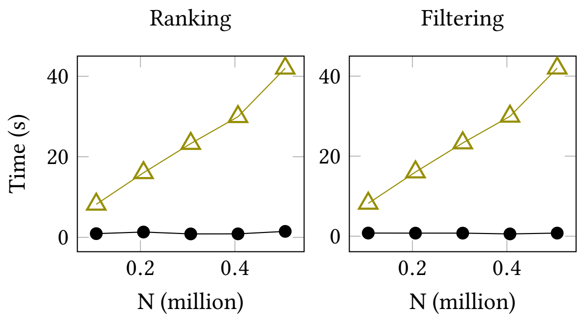
Impact of data size. To study how the cost of the algorithms changes with the size of datasets, we evaluate the performance of EntropyRank and EntropyFilter on random subsets of different sizes from the same dataset. Each subset has a similar data distribution with the original dataset. We run the algorithms on these subsets with the same query as on the original dataset. Figure 1 shows a case study of the running time of EntropyRank and EntropyFilter on two of our datasets, varying the size of the subsets. When the size is large enough, the running time of Exact grows linearly with the number of rows, while the running time of EntropyRank and EntropyFilter is almost constant. This confirms the guarantee from Theorem 4.1 and 4.2, and implies that the speedup of our algorithms over Exact can be higher for even larger datasets.

Impact of batch size. We study the impact of batch size to the performance of our algorithms, which is assumed to be a constant much smaller than N during our theoretical analysis in Section 4.3. On the one hand, smaller batch size encourages more frequent updates to the confidence intervals, avoiding excessive data access if a small sample size is sufficient for top-K ranking or η filtering. On the other hand, each update to a confidence interval adds overhead to the algorithms and increases the running time. Thus, a good batch size should be small enough to allow early termination and large enough to avoid being backfired by the overhead from frequent updates of confidence intervals.

Figure 2 shows a case study on one test case in our dataset for top-1 query and filter query with $\eta = 0.1$ using various batch sizes. While the sampling rate monotonically increases with larger batch



(a) bls_american-time-use-survey



(b) cdc_behavioral-risk

Figure 1: The running time varying data size

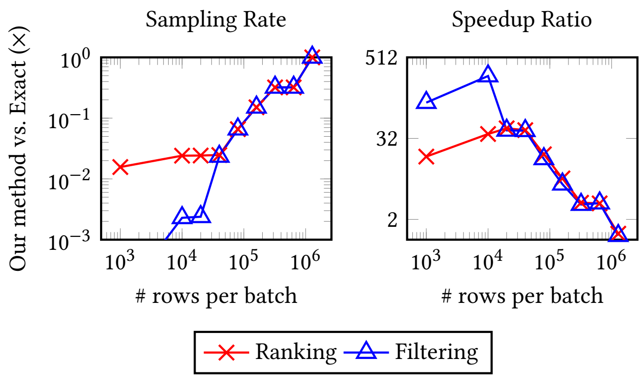


Figure 2: The sampling rate and speedup ratio on bls_american-time-use-survey varying batch size

sizes, the speedup ratio first increases and then decreases. When the batch size is too small, e.g., 10^3, the overhead from frequent updates to confidence intervals outweighs the benefit of sampling fewer data and leads to smaller speedup ratio. After a certain batch size, e.g., 20K, the cost of accessing data dominates, and the speedup ratio decreases with larger batch sizes, i.e., larger sampling rate. When the batch size reaches the full data size, i.e., 10^6, our algorithms access all the data and degenerate to Exact, and both sampling rate and speedup ratio become 1.

Fortunately, for a wide range of batch sizes, e.g., from 1K to 100K, the speedup is significant. We observe similar results on other test cases and datasets.

6 RELATED WORK

The idea of using random subsamples to speed up the extraction of frequent patterns dates back to 1996 [34]. Riondato and Upfal [31] presents the best approximation error bound for a single subsample. They further develop a progressive subsampling method based on Rademacher averages [32]. With high probability, their method returns a superset of the frequent itemsets, and no itemset with frequency much lower than the threshold will be included in it. Their work is an exemplar of how large scale data mining tasks can be practically accelerated via theoretically justified subsampling without losing quality. Scheffer and Wrobel [33] generalizes the association rule discovery problem to best hypothesis mining and discusses a variety of utility functions, which include association rule mining as a special case. They present an algorithm that works for all 'instance averaging' aggregation functions [33], leveraging the insight that the difference between sample mean and population mean can be probabilistically bounded. They also prove that there exists no subsampling algorithm for certain utility functions. There is no generic theoretical result covering all utility functions, though more results for specific tasks and functions are developed over the years. For example, Grosskreutz et al. [15] presents tight bounds for quality functions used in subgroup discovery. A recent study by Wang and Chakrabarti [38] proves the error bounds for $\ell 1$ and $\ell 2$ distance between two unknown distributions under subsampling for developing a near-optimal algorithm to recommend attributes that best distinguish two ad-hoc subsets of a large dataset. Huang et al. [17] approximates the accuracy of a classifier by training the classifier on subsamples, and further proposes an efficient progressively sampling algorithm to select an approximate best pipeline among many given pipelines for AutoML. Approximation error bounds with subsampling are studied for other tasks, such as linear regression [8], graph embedding [29] and triangle counting [10].

The problem of information entropy estimation has a rich history in information theory, statistics and theoretical computer science. The early work dates back to 1950's [26], and the modern interest is revived by Antos and Kontoyiannis [1] and Paninski [27]. They bound the difference between empirical entropy and information entropy. Valiant and Valiant [35] shows a breakthrough result about the minimal sample size required for consistent estimation ($\frac{c}{\epsilon \log c}$). Wu and Yang [39] and Jiao et al.[19] design estimators based on best polynomial approximation, which achieve optimal minimax mean-square error rate $\frac{c^2}{M^2 \log^2 c} + \frac{\log^2 c}{M}$. The fundamental difference with our work is that they all consider the observed data as samples from an underlying distribution. Therefore, one can theoretically keep sampling infinite i.i.d samples from that distribution, but the exact information entropy is never known from finite samples. The most suitable application scenario for their estimators is the setting of streaming data and unbounded support size. For our empirical entropy estimation problem, the given dataset D is the entire universe to subsample from. We can take all the $|D|$ points to calculate the exact empirical entropy.

There are studies on the entropy monitoring problem for streams or distributed streams [9, 13, 22]. They focus on space-efficient or communication-efficient algorithms for continuously approximating the entropy from streams or distributed streams, to detect the changes of empirical entropies calculated from different portions of the streaming data.

Our EntropyRank solution is inspired by the studies of *best arm identification* from a *multi-armed bandit (MAB)* [2]. Finding top-K entropy-based aggregation scores is analogous to finding top-K arms. The difference is that the arms are ranked by expected rewards, which can be approximated by the average of sampled rewards with known confidence intervals, while the entropy-based aggregation functions are more complex, which cannot be approximated in the same way. That challenge is addressed by our work.

7 CONCLUSION

We investigate the idea of using subsampling to approximate empirical entropy for large-scale data. We solve a major challenge of bounding the difference between the entropy computed from subsamples and that computed from the full data with high probability. We use these bounds to construct useful confidence intervals of empirical entropy, and propose algorithms to speed up entropy-based ranking and filtering applications. Our algorithms progressively sample the data as needed until we can return the correct answer with high probability. Empirically, our algorithms are faster than the exact methods by up to three orders of magnitude with almost no error in large-scale real-world datasets. We further show that the speedup ratio is expected to grow with data size, due to the sublinear complexity of our algorithms.

Our work can be extended from both the theory and the application side. For theory, tighter bound analysis with respect to the subsample size is an open problem. For applications such as full-fledged feature selection, automated feature engineering, and entropy-based clustering and classification, it is promising to develop algorithms like EntropyRank and EntropyFilter to perform fast computation with theoretical guarantee.

REFERENCES

[1] András Antos and Ioannis Kontoyiannis. 2001. Convergence properties of functional estimates for discrete distributions. *Random Structures & Algorithms* 19, 3-4 (2001), 163–193.

[2] Jean-Yves Audibert and Sébastien Bubeck. 2010. Best Arm Identification in Multi-Armed Bandits. In *COLT'10.*

[3] B. L. Aurelian. 2018. An information entropy based splitting criterion better for the Data Mining Decision Tree algorithms. In *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC).*

[4] Daniel Barbará, Yi Li, and Julia Couto. 2002. COOLCAT: An Entropy-based Algorithm for Categorical Clustering. In *CIKM'02.*

[5] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *J. Mach. Learn. Res.* 13 (Jan. 2012), 27–66.

[6] Luis M de Campos. 2006. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *J. Mach. Learn. Res.* 7, Oct (2006), 2149–2187.

[7] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. 2017. Approximate Query Processing: No Silver Bullet. In *SIGMOD'17*. 511–519.

[8] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. 2015. Uniform Sampling for Matrix Approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS'15).*

[9] Graham Cormode. 2013. The continuous distributed monitoring model. *ACM SIGMOD Record* 42, 1 (2013), 5–14.

[10] Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. 2017. Approximately counting triangles in sublinear time. *SIAM J. Comput.* 46, 5 (2017), 1603–1646.

[11] Ran El-Yaniv and Dmitry Pechyony. 2006. Stable Transductive Learning. In *COLT'06.*

[12] Pawel Foremski, David Plonka, and Arthur Berger. 2016. Entropy/IP: Uncovering Structure in IPv6 Addresses. In *Proceedings of the 2016 Internet Measurement Conference (IMC'16).*

[13] Moshe Gabel, Daniel Keren, and Assaf Schuster. 2017. Anarchists, Unite: Practical Entropy Approximation for Distributed Streams. In *KDD'17.*

[14] Alison L. Gibbs and Francis Edward Su. 2002. On Choosing and Bounding Probability Metrics. *International Statistical Review* 70, 3 (2002), 419–435.

[15] Henrik Grosskreutz, Stefan Rüping, and Stefan Wrobel. 2008. Tight Optimistic Estimates for Fast Subgroup Discovery. In *ECMLPKDD'08.*

[16] Wassily Hoeffding. 1962. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58, March (1962), 13–30.

[17] Silu Huang, Chi Wang, Bolin Ding, and Surajit Chaudhuri. 2019. Efficient Identification of Approximate Best Configuration of Training in Large Datasets. In *AAAI'19.*

[18] Johan Ludwig William Valdemar Jensen. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica* 30, 1 (1906), 175–193.

[19] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. 2015. Minimax estimation of functionals of discrete distributions. *IEEE Transactions on Information Theory* 61, 5 (2015), 2835–2885.

[20] A. Kaul, S. Maheshwary, and V. Pudi. 2017. AutoLearn âĂŤ Automated Feature Generation and Selection. In *ICDM'17.*

[21] Albert Kim, Eric Blais, Aditya G. Parameswaran, Piotr Indyk, Samuel Madden, and Ronitt Rubinfeld. 2015. Rapid Sampling for Visualizations with Ordering Guarantees. *PVLDB* 8, 5 (2015).

[22] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. 2006. Data Streaming Algorithms for Estimating Entropy of Network Traffic. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'06/Performance'06).*

[23] David D. Lewis. 1992. Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of the Workshop on Speech and Natural Language (HLT'91).*

[24] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *Comput. Surveys* 50, 6 (12 2017).

[25] Colin McDiarmid. 1989. *On the method of bounded differences.* Cambridge University Press, 148–188.

[26] G. MILLER. 1955. Note on the bias of information estimates. *Information theory in psychology : Problems and methods* 2 (1955), 95–100.

[27] L. Paninski. 2003. Estimation of Entropy and Mutual Information. *Neural Computation* 15, 6 (2003), 1191–1253.

[28] Hanchuan Peng, Fuhui Long, and C. Ding. 2005. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27 (2005), 1226–1238.

[29] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *WWW'19.*

[30] J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning* 1, 1 (Mar 1986), 81–106.

[31] Matteo Riondato and Eli Upfal. 2014. Efficient Discovery of Association Rules and Frequent Itemsets Through Sampling with Tight Performance Guarantees. *ACM Trans. Knowl. Discov. Data* 8, 4 (Aug. 2014), 20:1–20:32.

[32] Matteo Riondato and Eli Upfal. 2015. Mining Frequent Itemsets Through Progressive Sampling with Rademacher Averages. In *KDD'15.*

[33] Tobias Scheffer and Stefan Wrobel. 2000. A Sequential Sampling Algorithm for a General Class of Utility Criteria. In *KDD'00.*

[34] Hannu Toivonen. 1996. Sampling Large Databases for Association Rules. In *VLDB'96*. 134–145.

[35] Gregory Valiant and Paul Valiant. 2011. Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing.*

[36] Paul Valiant and Gregory Valiant. 2013. Estimating the unseen: improved estimators for entropy and other properties. In *NIPS'13.*

[37] Jorge R. Vergara and Pablo A. Estévez. 2014. A review of feature selection methods based on mutual information. *Neural Computing and Applications* 24, 1 (Jan 2014), 175–186.

[38] Chi Wang and Kaushik Chakrabarti. 2018. Efficient Attribute Recommendation with Probabilistic Guarantee. In *KDD'18.*

[39] Yihong Wu and Pengkun Yang. 2016. Minimax rates of entropy estimation on large alphabets via best polynomial approximation. *IEEE Transactions on Information Theory* 62, 6 (2016), 3702–3720.

[40] Ying Yan, Liang Jeff Chen, and Zheng Zhang. 2014. Error-bounded Sampling for Analytics on Big Sparse Data. In *VLDB'14.*

Table 4: Dataset source and randomly chosen target columns

Name	URL	Target column index				
bls_american-time-use	https://www.kaggle.com/bls/american-time-use-survey#atuscps.csv	32	92	156	176	233
cdc_behavioral-risk	https://www.kaggle.com/cdc/behavioral-risk-factor-surveillance-system#2011.csv	17	72	314	349	379
census-american_hus	https://www.kaggle.com/census/2015-american-community-survey#ss15husa.csv https://www.kaggle.com/census/2015-american-community-survey#ss15husb.csv https://www.kaggle.com/census/2013-american-community-survey#ss13husa.csv https://www.kaggle.com/census/2013-american-community-survey#ss13husb.csv	13	17	18	20	96
census-american_pus	https://www.kaggle.com/census/2015-american-community-survey#ss15pusa.csv https://www.kaggle.com/census/2015-american-community-survey#ss15pusb.csv https://www.kaggle.com/census/2014-american-community-survey#ss14pusa.csv https://www.kaggle.com/census/2014-american-community-survey#ss14pusb.csv https://www.kaggle.com/census/2013-american-community-survey#ss13pusa.csv https://www.kaggle.com/census/2013-american-community-survey#ss13pusb.csv	78	147	148	152	154
gbonesso_enem-2016	https://www.kaggle.com/gbonesso/enem-2016#microdados_enem_2016_coma.csv	18	78	127	130	132

REPRODUCIBILITY SUPPLEMENT

This section contains reproducibility details that are not described in the main paper.

We evaluate our algorithms on five real-world large-scale datasets from Kaggle as shown in Table 4.

- American Time Use Survey (2013-2015). The American Time Use Survey (ATUS) is the Nation's first federally administered, continuous survey on time use in the United States. The goal of the survey is to measure how people divide their time among life's activities. In ATUS, individuals are randomly selected from a subset of households that have completed their eighth and final month of interviews for the Current Population Survey (CPS). ATUS respondents are interviewed only one time about how they spent their time on the previous day, where they were, and whom they were with. The survey is sponsored by the Bureau of Labor Statistics and is conducted by the U.S. Census Bureau. The Activity file we used contains information about each household member of all individuals selected to participate in ATUS.
- Behavioral Risk Factor Surveillance System (2011). The Behavioral Risk Factor Surveillance System (BRFSS) is the nation's premier system of health-related telephone surveys that collect state data about U.S. residents regarding their health-related risk behaviors, chronic health conditions, and use of preventive services. Factors assessed by the BRFSS include tobacco use, health care coverage, HIV/AIDS knowledge or prevention, physical activity, and fruit and vegetable consumption. Data are collected from a random sample of adults (one per household) through a telephone survey. Kaggle has the survey data from 2011 to 2015 in separate csv files. Since the schema for every year's data has a large variation, we did not merge them. Only the dataset in 2011 has more than half a million rows.
- American Community Survey (2013-2015). The American Community Survey is an ongoing survey from the US Census Bureau. In this survey, approximately 3.5 million households per year are asked detailed questions about who they are and how they live. Many topics are covered, including ancestry, education, work, transportation, internet use, and residency. There are two types of survey data provided, housing and population. For the housing data, each row is a housing unit, and the characteristics are properties like rented vs. owned, age of home, etc. For the population data, each row is a person and the characteristics are properties like age, gender, whether they work, method/length of commute, etc. Each data set is divided in two pieces, "a" and "b" (where "a" contains states 1 to 25 and "b" contains states 26 to 50). We merged the available population survey data from year 2013 to 2015, using the common fields in their schema. We merged the available housing survey data for year 2013 and 2015, using the common fields in their schema. The housing survey data in 2014 is unavailable in Kaggle.
- ENEM 2016. The original dataset is provided by INEP (http://portal.inep.gov.br/microdados), a department from the Brazilian Education Ministry. It contains data from the applicants for the 2016 National High School Exam. Inside this dataset there are not only the exam results, but the social and economic context of the applicants.

To create a test case, we randomly choose one categorical column as the target variable, and repeat the process for five times. Table 4 shows the index of the columns chosen as the target variable.

By default, we choose 100K as the batch size for EntropyRank and EntropyFilter. The size of the first small batch is then N modulo 100K. α is set to $\frac{50}{C}$ and 1 for EntropyRank and EntropyFilter, respectively. For experiments varying data sizes, we create subsets of the datasets with multiples of 100K data points, e.g., 100K, 200K, 300K, etc, that are randomly drawn from the original datasets. For both varying data size and varying batch size experiments, the target column index is 12 for bls_american-time-use and 398 for cdc_behavioral-risk.