CS369G: Algorithmic Techniques for Big Data Spring 2015-2016

Graphical Model Sketch

Authors: Kveton et al. *Presenters: Akshay Agrawal, Yifan Lu*

1 Overview

We present and discuss graphical model sketches [KBG+16], a set of three algorithms for approximating the maximum likelihood estimates of the marginal and conditional probabilities within a Bayesian network. In particular, we motivate the problem at hand, highlight the commonalities among the three algorithms and contextualize them within the framework of frequency estimation, providing color along the way.

1.1 A motivating example

Consider the following problem [KBG+16]: we have a stream of n vectors of dimension K with each vector $x^{(t)} \in [M]^K$ drawn from some distribution. Each component of the vector is drawn from a universe of size M. We wish to estimate the probability of seeing a particular x given that x is drawn from the same unknown distribution from which the stream was drawn. Without any other information, we can do this with the estimate

$$\tilde{P}(x) = \frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{x = x^{(t)}\}$$

The naive implementation of this estimator is to store the entire stream. But that requires $O(M^K)$ space, and we assume that both M and K can be very large. We can use the Count-Min sketch [CM05] to get a good estimate that is space efficient.

1.2 Count-Min Sketch

We begin by reviewing the Count-Min sketch, and contextualize it in the setting of estimating $\tilde{P}$. This formulation will motivate later algorithms and will familiarize us with the notations.

Recall that the Count-Min sketch uses d hash functions $h^1, h^2, \ldots, h^d$; these has functions are drawn uniformly at random from a pairwise independent family. In order to sketch $\tilde{P}$, we require hash functions of the form $h^i : [M]^K \to [m]$. We use these hash functions to maintain a table $c \in \mathbb{N}^{d \times m}$. The update and query functions are as defined in Algorithm 1 and Algorithm 2. We refer to these algorithms collectively as Count-Min-Vec, since they operate upon observation vectors.

Theorem 1 ([CM05]). *Let $d = \log(1/\delta)$ and $m = e/\epsilon$. Then for any $x \in [M]^K$, we have $\tilde{P}(x) \leq P_{CM}(x) \leq \tilde{P}(x) + \epsilon$ with probability at least $1 - \delta$. The algorithm requires $O((e/\epsilon)\log(1/\delta))$ space.*

Algorithm 1 COUNT-MIN-VEC.UPDATE ▷ Given stream element $x^{(t)} = (x_1^{(t)}, x_2^{(t)}, \ldots, x_K^{(t)})$ and table c

```
for $i = 1, \ldots, d$ do
    $c(i, h^i(x^{(t)})) \leftarrow c(i, h^i(x^{(t)})) + 1$
end for
```

Algorithm 2 COUNT-MIN-VEC.QUERY: $P_{\text{CM}}(x)$ ▷ Given table c

```
return $P_{\text{CM}}(x) \leftarrow \frac{1}{n} \min_{i \in [d]} c(i, h^i(x))$
```

This algorithm is good for estimating $\tilde{P}$ and also has low space usage. However, $\tilde{P}$ itself does not give us granular information about the relationships among our random varaibles. Without any additional information, we won't be able to do much more. But what if we knew something about the *structure* of the distribution? Could we then do better?

1.3 Bayesian Networks and Inference

A probabilistic graphical model is a directed graph that encodes the conditional dependencies among a set of random variables X_1, X_2, $\ldots$, X_K [KF09]. A probabilistic graphical model is called a *Bayesian network* if it is directed and acyclic. Formally, a Bayesian network is a directed and acyclic graph $\mathcal{G} = (V, E)$, where $V = \{X_1, X_2, \ldots, X_K\}$ and $(i, j) \in E$ for all i, j such that $P(X_i | X_j) \neq P(X_i)$. A key property of Bayesian networks is that the joint density of its random variables factors into the product

$$P(X_1, \ldots, X_K) = \prod_{i=k}^{K} P(X_k | X_{\text{pa}(k)})$$

where $X_{\text{pa}(k)}$ is the set of parents of X_k. Going forward, we will restrict our attention to networks that are trees rooted at X_1; this assumption is made for the sake of simplicity, and all the results that we will present would hold even if we were to lift it [KBG$^+$16].

Assuming that our graphs $\mathcal{G}$ are in fact trees, we have that

$$P(X_1, \ldots, X_K) = P(X_1) \prod_{i=k}^{K} P(X_k | X_{\text{pa}(k)}),$$

where $X_{\text{pa}(k)}$ is now the single element that is the parent of X_k, instead of the set containing that element. From here on, the notation $\text{pa}(k)$ will refer to the index of the parent of node k.

For our particular setting, let X_1, X_2, $\ldots$, X_K be discrete random variables with support $[M]$. For convenience, define

$$\begin{aligned} P_k(i) &= P(X_k = i) \\ P_k(i, j) &= P(X_k = i, X_{\text{pa}(k)} = j) \\ P_k(i | j) &= P(X_k = i | X_{\text{pa}(k)} = j). \end{aligned}$$

We call $\{P_k(i)\}_{\forall k,i}$ the marginal probabilities and $\{P_k(i|j)\}_{\forall k,i,j}$ the conditional probabilities. Together, the KM marginals and conditionals $(K-1)M^2$ conditionals give us a complete description of the joint probabilitiy density of $(X_1, X_2, \ldots, X_K)$.

In a practical setting, the priors and conditionals are often unknown. Given a stream of observations $\{x^{(t)}\}_{t=1}^n$, we can estimate the parameters θ of $\mathcal{G}$ by maximum likelihood estimation. That is, we desire

$$\bar{\theta} = \arg\max_{\theta} P(\{x^{(t)}\}_{t=1}^n; \theta, \mathcal{G}).$$

It turns out that the maximum likelihood estimates take on intuitive forms:

$$\begin{aligned}
\forall i \in [M] : \bar{P}_k(i) &= \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\{x_k^{(t)} = i\} \\
\forall (i,j) \in [M] \times [M] : \bar{P}_k(i,j) &= \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\{x_k^{(t)} = i, x_{\mathrm{pa}(k)}^{(t)} = j\} \\
\forall (i,j) \in [M] \times [M] : \bar{P}_k(i|j) &= \frac{\bar{P}_k(i,j)}{\bar{P}_{\mathrm{pa}(k)}(j)}.
\end{aligned}$$

Given our maximum likelihood estimates, we can approximate the joint distribution of X_1, $\ldots$, X_K as

$$\bar{P}(X_1, \ldots, X_K) = \bar{P}(X_1) \prod_{i=k}^{K} \bar{P}(X_k | X_{\mathrm{pa}(k)})$$

An exact computation of the parameter estimates would take a prohibitive $O(KM^2)$ space, motivating the need for sketches. We will, in particular, employ sketches to approximate the frequencies needed to compute the maximum likelihood estimates. In the remainder of this exposition, $\hat{P}_k$ will denote a sketch of the corresponding MLE $\bar{P}_k$.

1.4 Prior Work

Kveton et al. remark in [KBG$^+$16] that they are the first to sketch graphical models, or at least to publish their methodology. At their cores, however, the algorithms presented in [KBG$^+$16] reduce to sketching frequencies, certainly a precedented techinque. Each algorithm relies heavily upon the sketching techniques of the Count-Min [CM05] and Count-Sketch structures [CCFC02]. Indeed, the flagship algorithm of [KBG$^+$16] (see Section 2.3) reduces to maintaining Count-Min sketches all the marginal and joint frequencies.

In the context of machine learning, [Fox14] uses sketches to compactly represent feature vectors for classification tasks and [Bil15] suggests sketching to estimate tabulated probability distributions.

2 Algorithms

As we have seen, the MLE of the Bayesian network will give us a better estimator of the distribution than naively calculating the empirical frequency. We now present a couple of ways of estimating the MLE.

2.1 GMHash

Our first attempt will be to represent $n\bar{P}(\cdot)$ (on a variable) and $n\bar{P}(\cdot,\cdot)$ (on a variable-parent pair) as two tables: $c, b \in \mathbb{N}^{K\times m}$. For $j = 1, 2, \ldots, K$, row j of c will estimate the frequencies of $1, 2, \ldots, M$ with respect to X_j. Similarly, row j of b will estimate the number of times $X_j = s$ and $X_{\mathrm{pa}(j)} = t$, for all pairs $(s,t) \in [M] \times [M]$.

More concretely, define a hash function for each variable and variable-parent pair: $h_k : [M] \to [m]$ and $g_k : [M] \times [M] \to [m]$. Let $h = (h_1, \ldots, h_K)$ and $g = (g_1, \ldots, g_K)$ be tuples of such hash functions. The update and query procedures are described in Algorithm 3 and Algorithm 4.

Algorithm 3 GMHash.Update ▷ Given stream element $x^{(t)} = (x_1^{(t)}, \ldots, x_K^{(t)})$ and tables c and b

```
for $k = 1, \ldots, K$ do                                                        ▷ $K$ variables
    $c(j, h_k(x_k^{(t)})) \leftarrow c(j, h_k(x_k^{(t)})) + 1$                    ▷ Update for one random variable
end for
for $k = 2, \ldots, K$ do                                                        ▷ $K-1$ variable-parent pairs
    $b(j, g_k(x_k^{(t)}, x_{\mathrm{pa}(k)}^{(t)})) \leftarrow b(j, g_k(x_k^{(t)}, x_{\mathrm{pa}(k)}^{(t)}))) + 1$   ▷ Update for variable-parent pair
end for
```

Algorithm 4 GMHash.Query: $\hat{P}(x)$ ▷ Given observation $x = (x_1, x_2, \ldots, x_K)$ and tables c and b

```
$\hat{P}_1(x_1) \leftarrow \frac{c(1, h_1(x_1))}{n}$
for all $k = 2, \ldots, K$ do
    $\hat{P}_k(x_k | x_{\mathrm{pa}(k)}) \leftarrow \frac{b(k, g_k(x_k, x_{\mathrm{pa}(k)}))}{c(\mathrm{pa}(k), h_{\mathrm{pa}(k)}(x_{\mathrm{pa}(k)}))}$   ▷ Ratio of buckets
end for
return $\hat{P}(x) \leftarrow \hat{P}_1(x_1) \prod_{k=2}^{K} \hat{P}_k(x_k | x_{\mathrm{pa}(k)})$
```

The main idea here is that each conditional $\hat{P}_k(x_k|x_{\mathrm{pa}(k)})$ is estimated as the ratio of two hashing bins. We estimate $\bar{P}(x)$ by taking the product of $K-1$ conditionals and a prior. In the Count-Min approach, we hashed each K dimensional vector to d buckets with d hash functions. Now, we only hash one or two components of each vector to their own bucket with their own hash function. The hope is that this does not "blow up" the error. Indeed that is the case.

Theorem 2. *For any x,*

$$\bar{P}(x) \prod_{k=1}^{K} (1 - \epsilon_k) \leq \hat{P}(x) \leq \bar{P}(x) \prod_{k=1}^{K} (1 + \epsilon_k)$$

holds with probability at least $3/4$, where:

$$\epsilon_1 = \frac{8K}{\bar{P}_1(x_1)m}$$

$$k = 2, \ldots, K : \epsilon_k = \frac{8K}{\bar{P}_k(x_k, x_{\mathrm{pa}(k)})m}$$

The proof is very similar to that for the error of COUNT-MIN. We show that it is unlikely for either estimate in any term to be significantly overestimated. Then we bound the bad events by Markov's inequality. We first declare without proof a couple of lemmas that will be useful in the theorem.

Lemma 3. *Let:*

$$\left|\frac{u_h}{v_h} - \frac{u}{v}\right| > \epsilon$$

for any $u_h \geq u, v_h \geq v, v \geq u$, and $v \geq \alpha n$. Then either $v_h - v > \epsilon\alpha n$ or $u_h - u > \epsilon\alpha n$.

The proof is purely algebraic and is omitted for brevity.

Lemma 4. *Let X be a discrete random variable on $\mathbb{N}$ and $(x^{(t)})_{t=1}^{n}$ be its n observations. Let $h : \mathbb{N} \to [m]$ be any random hash function. Then for any $x \in \mathbb{N}$, $m \geq 1$, and $\epsilon \in (0,1)$:*

$$\Pr\left[\left(\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left\{h(x^{(t)}) = h(x)\right\} - \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left\{x^{(t)} = x\right\}\right) > \epsilon\right] < \frac{1}{m\epsilon},$$

where the randomness is with respect to h.

The proof is a result of Markov's inequality. We will now show the error bound.

Proof. (Theorem 2) First, we find that

$$\Pr[|\hat{P}_1(x_1) - \bar{P}_1(x_1)| > \epsilon_1] < \frac{1}{m\epsilon_1} \tag{1}$$

This follows directly from Lemma 4 and the fact that $\hat{P}_1(x_1) \geq \bar{P}_1(x_1)$.

Next, for all $1 < k \leq K$, we show that

$$\Pr[|\hat{P}_k(x_k|x_{\mathrm{pa}(k)}) - \bar{P}_k(x_k|x_{\mathrm{pa}(k)})| > \epsilon_k] < \frac{2}{m\epsilon_k\alpha_k} \tag{2}$$

where $\alpha_k = \bar{P}_{\mathrm{pa}(k)}(x_{\mathrm{pa}(k)})$.

We can write the probability as

$$\Pr\left[\left|\frac{b(k, g_k(x_k, x_{\mathrm{pa}(k)}))}{c(\mathrm{pa}(k), h_{\mathrm{pa}(k)}(x_{\mathrm{pa}(k)}))} - \frac{\sum_{t=1}^{n} \mathbf{1}\{x_k^{(t)} = x_k, x_{\mathrm{pa}(k)}^{(t)} = x_{\mathrm{pa}(k)}\}}{\sum_{t=1}^{n} \mathbf{1}\{x_{\mathrm{pa}(k)}^{(t)} = x_{\mathrm{pa}(k)}\}}\right| > \epsilon_k\right] \tag{3}$$

For convenience we define the following events. Let A be the event that

$$\frac{1}{n}c(\mathrm{pa}(k), h_{\mathrm{pa}(k)}(x_{\mathrm{pa}(k)})) - \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\{x_{\mathrm{pa}(k)}^{(t)} = x_{\mathrm{pa}(k)}\} > \epsilon_k\alpha_k$$

and B be the event that

$$\frac{1}{n}b(k, g_k(x_k, x_{\mathrm{pa}(k)})) - \frac{1}{n}\mathbf{1}\{x_k^{(t)} = x_k, x_{\mathrm{pa}(k)}^{(t)} = x_{\mathrm{pa}(k)}\} > \epsilon_k\alpha_k$$

By Lemma 3 and the union bound, to prove (2) it suffices to show that $\Pr[A] + \Pr[B] < \frac{2}{m\epsilon_k \alpha_k}$. We can apply Lemma 4 and get exactly that.

Now by the union bound with (1) and (2), we find that the probability of any bad event is at most

$$\frac{1}{m}\sum_{k=1}^{K} \frac{2}{\epsilon_k \alpha_k}$$

and is at most 1/4 for $m \geq 4\sum_{k=1}^{K} \frac{2}{\epsilon_k \alpha_k}$.

Now we can choose our $\epsilon_1, \ldots, \epsilon_k$. Let $\epsilon_k = 8K/(\alpha_k m)$. This is valid for any $m \geq 1$ since:

$$m \geq 4\sum_{k=1}^{K} \frac{2}{\epsilon_k \alpha_k} = 4\sum_{k=1}^{K} \frac{m}{4K} = m$$

□

Note that the approximation improves with the number of bins m because each $\epsilon_k = O(1/m)$. The accuracy of the approximation also depends on the frequency of interaction between the values in x. If $\bar{P}_k(x_k, x_{\mathrm{pa}(k)})$ is sufficiently large for all k, then that means X_k and $X_{\mathrm{pa}(k)}$ co-occur frequently (and x is not too sparse). This trade-off can be demonstrated more precisely. Let

$$\begin{aligned} m &\geq \frac{1}{8K^2}\bar{P}_1(x_1) \\ k = 2, \ldots, K : m &\geq \frac{1}{8K^2}\bar{P}_k(x_k, x_{\mathrm{pa}(k)}), \end{aligned}$$

then we get $\epsilon_k \leq 1/K$ and for $K \geq 2$ we have the error bound

$$\frac{2}{3e}\bar{P}(x) \leq \hat{P}(x) \leq e\bar{P}(x).$$

Since GMHash stores $2K - 1$ hash tables where each table contains m bins, the algorithm uses $O(Km)$ space.

Another way of thinking about GMHash is in context of Count-Min. Recall that when we used Count-Min to estimate $\tilde{P}(x)$ (in Section 2), we hashed the entire $x^{(t)}$ using d different hash functions. GMHash hashed each $x^{(t)}$ using $2k$ different hash functions but instead of hashing the entire vector each time, it only hashes each component with two hash functions. The error guarantee is sufficient but the variance is much worse than Count-Min. This is inevitable because GMHash is akin to running Count-Min on each component of $x^{(t)}$ and each variable-parent pair in $x^{(t)}$ with only one ($d = 1$) hash function. We also take the ratio of the two bin estimates, which gives a two sided error rather than the one-sided guarantee of Count-Min.

2.2 GMSketch

When we proved the variance of Count-Min we used the "medium trick". Essentially, we hashed each item d times and took the minimum (in the case of Count-Min, the minimum is better than the median because we only had a one-sided error). Then we used Markov's inequality to get a concentration tail bound. We can apply the same trick here. We run the algorithm for GMHash d times and then take the median. Then we find the optimal d for a $1-\delta$ success probability.

Let h and g be hash tuples as defined above. We now create d copies of these tuples ($h^1, \dots, h^d$ and $g^1, \dots, g^d$). We require each h^i and h^j to be pairwise independent as well as each g^i and g^j. That means for any $k \in [K]$ we require h^i_k and h^j_k to be pairwise independent (same for g). We also have $2d$ tables $c^i, b^i \in \mathbb{N}^{d\times m}$ where each table uses a different hash tuple. For completeness, the full definition for update and query is defined in Algorithm 5 and Algorithm 6; the authors present these algorithms together as the GMSketch.

Algorithm 5 GMSketch.Update ▷ Given stream element $x^{(t)}$ and tables $c_1, \dots, c_d$ and $b_1, \dots, b_d$

```
  for $i = 1, \dots, d$ do
      GMHash.Update($x^{(t)}; c_i, b_i$)
  end for
```

Algorithm 6 GMSketch.Query: $\hat{P}(x)$ ▷ Given tables $c_1, \dots, c_d$ and $b_1, \dots, b_d$

```
  for $i = 1, \dots, d$ do
      $\hat{P}^i(x) \leftarrow$ GMHash.Query($x; c_i, b_i$)
  end for
  return $\hat{P}(x) \leftarrow \text{median}_{i\in[d]} \hat{P}^i(x)$
```

It is easy to see that this algorithm uses $O(Kmd)$ space. We will now find d that gives us a good variance.

Theorem 5. *For any $d \geq 8\log(1/\delta)$ x:*

$$\bar{P}(x)\prod_{k=1}^{K}(1-\epsilon_k) \leq \hat{P}(x) \leq \prod_{k=1}^{K}(1+\epsilon_k)$$

holds with probability at least $1-\delta$ with each ϵ_k defined the same as in Theorem 2.

The proof is by Hoeffding's inequality (the "median trick") which we'll omit here. With $d = 8\log(1/\delta)$, we get that the median is not in the interval with probability δ. Therefore, using space $O(Km\log(1/\delta))$ we get the same error guarantees as GMHash but with probability at least $1-\delta$.

2.3 GMFactorSketch

While GMHash and GMSketch attain multiplicate error bounds, both bounds depend upon K. The dependence is such that we must choose $m = \Omega(K)$ in order to get a constant-factor error bound, giving space requirements of $\tilde{O}(K^2)$. Such a space requirement may be acceptable for small K; however, given the sheer wealth of data available for use in, for example, online advertising models (the motivating example for [KBG$^+$16]) we would like the ability to design algorithms for large K. And so we find ourselves asking: Can we do better?

It turns out that we can indeed do better. The third and final algorithm that Kveton, et al present [KBG$^+$16], dubbed the GMFactorSketch, attains the same error bounds as the first two while using K times less storage. Whereas GMHash and GMSketch rely upon the methodology of the Count-Min sketch, GMFactorSketch uses the Count-Min sketch directly. In particular, it maintains $2K-1$ Count-Min sketches in order to approximate the marginal and joint distributions.

To be more precise, let $c_1, c_2, \ldots, c_K, b_2, b_3, \ldots, b_K$ be independent Count-Min sketches, each with d rows and m columns each. Sketch c_i will approximate the frequencies of $j \in [M]$ with respect to X_i, and sketch b_i will approximate the frequencies of $(s,t) \in [M] \times [M]$ with respect to $(X_i, X_{\mathrm{pa}(i)})$. Let Count-Min.Update be the standard update function, and let Count-Min.Query be the standard query function. Algorithms 7 and 8 fully specify GMFactorSketch.

Algorithm 7 GMFactorSketch.Update ▷ Given stream element $x^{(t)}$

```
for $k = 1, \ldots, K$ do                                   ▷ Update the marginal frequencies
    Count-Min.Update($x_k^{(t)}; c_k$)
end for
for $k = 2, \ldots, K$ do                                   ▷ Update the joint frequencies
    Count-Min.Update($x_k^{(t)}, x_{\mathrm{pa}(k)}^{(t)}; c_k$)
end for
```

Algorithm 8 GMFactorSketch.Query: $\hat{P}(x)$ ▷ Given observation $x = (x_1, x_2, \ldots, x_K)$

```
for $k = 1, \ldots, K$ do                                   ▷ Marginal probabilities
    $\hat{P}_k(x_k) = \frac{1}{n} \cdot$ Count-Min.Query($x_k; c_k$)
end for
for $k = 2, \ldots, K$ do                                   ▷ Joint probabilities
    $\hat{P}_k(x_k, x_{\mathrm{pa}(k)}) = \frac{1}{n} \cdot$ Count-Min.Query($x_k, x_{\mathrm{pa}(k)}; b_k$)
end for
for $k = 2, \ldots, K$ do                                   ▷ Conditional probabilities
    $\hat{P}_k(x_k | x_{\mathrm{pa}(k)}) = \frac{\hat{P}_k(x_k, x_{\mathrm{pa}(k)})}{\hat{P}_{\mathrm{pa}(k)}(x_{\mathrm{pa}(k)})}$
end for
return $\hat{P}(x) \leftarrow \hat{P}_1(x_1) \prod_{k=2}^{K} \hat{P}_k(x_k | x_{\mathrm{pa}(k)})$
```

Theorem 6 bounds the error probability of GMFactorSketch; observe in particular that the errors ϵ_k no longer depend upon K. The upshot is that GMFactorSketch provides us with substantial space savings: while GMSketch required $\Omega(K)$ bins to attain a constant-factor multiplicative error bound, GMFactorSketch requires a factor of K fewer bins to attain the same bound.

Theorem 6. *For any* $d \geq \log(2K/\delta)$ *x:*

$$\bar{P}(x) \prod_{k=1}^{K} (1 - \epsilon_k) \leq \hat{P}(x) \leq \prod_{k=1}^{K} (1 + \epsilon_k)$$

holds with probability at least $1 - \delta$*, where:*

$$\begin{aligned} \epsilon_1 &= \frac{e}{\bar{P}_1(x_1)m} \\ k = 2, \ldots, K : \epsilon_k &= \frac{e}{\bar{P}_k(x_k, x_{\mathrm{pa}(k)})m}. \end{aligned}$$

The proof is analagous to that of Theorem 2; we leave the details as an exercise for the reader. The key step is to find the requirements on m such that

$$\begin{aligned} \hat{P}_k(x_{\mathrm{pa}(k)}) - \bar{P}_k(x_{\mathrm{pa}(k)}) &\leq e_k \alpha_k \\ \hat{P}_k(x_k, x_{\mathrm{pa}(k)}) - \bar{P}_k(x_k, x_{\mathrm{pa}(k)}) &\leq e_k \alpha_k, \end{aligned}$$

where $\alpha_1 = 1$ and $\alpha_k = \bar{P}_k(x_{\mathrm{pa}(k)})$ for $k \in \{2, 3, \ldots, K\}$. Since our estimates $\hat{P}_k(x_{\mathrm{pa}(k)})$ and $\hat{P}_k(x_k, x_{\mathrm{pa}(k)})$ are Count-Min estimates, we can use the results from [CM05] to show that, for general errors $\epsilon_1, \epsilon_2, \ldots, \epsilon_K$ we require $m \geq e/(\epsilon_k \alpha_k)$. It is then easy to show that the particular assignments of ϵ_k from Theorem 6 work.

2.4 Discussion

Kveton et al. present three sketching algorithms for approximating the MLE of the parameters θ of a Bayesian network $\mathcal{G}$: GMHash, GMSketch, and GMFactorSketch. GMHash simply runs Count-Min on each component of the observation vectors $x^{(t)}$ with only one hash function per Count-Min structure; the algorithm only takes $O(Km)$ space and, as we should expect, its probabilistic guarantees are weak.

GMSketch attempts to improve upon GMHash by bringing the variance reduction ideas of Count-Min: it runs d copies of GMHash in parallel and uses as its estimate the median of GMHash. GMSketch thus maintains $2K-1$ GMHash data structures for a total of $O(Km \log(1/\delta))$ space and with errors proportional to K.

GMFactorSketch improves upon GMSketch by replacing the GMHash data structures with bonafide Count-Min sketches. The errors of GMFactorSketch do not depend upon K, so this particular algorithm achieves the same error bounds as GMSketch with a factor of K less space.

We can say, without qualification, that GMFactorSketch is the best of the three algorithms presented. Its space usage is but a log factor greater than GMHash and its error bounds are significantly tighter than those provided by either of the two other algorithms. This result should come as no surprise. We hardly need to resort to a mathematical analysis to observe that the variance of GMHash is intolerably high. It is also clear that GMSketch should require a larger amount of space than GMFactorSketch does in order to attain the same constant multiplicative error: each GMHash has a two sided error because it takes a ratio of two independent hash buckets–each with one sided error while each Count-Min has only a one-sided error. The argument essentially boils down to the following question: is it better to perform the variance reduction early or late in the algorithm? While it might seem that earlier is better because we can exploit the structure of the problem more, here we find that such an intuition is wrong (or better proof techniques should be developed). Indeed, appealing to intuition, the conditional probabilities estimated by GMFactorSketch should be closer to the true MLE probabilities than those estimated by GMSketch because the former takes the minimum of the joint and marginal probabilities *before* taking their ratio, whereas the latter takes the median of several ratios of approximate joints and marginals. In other words, GMFactorSketch uses better estimates (or, at least, estimates that are no worse) for the joints and marginals than GMSketch does.

3 Comparing GMFactorSketch and Count-Min-Vec

Kveton et al. emphasize that a key result of [KBG+16] is that, for a fixed amount of space and for a certain class of problems, GMFactorSketch attains a tighter error bound than Count-Min. But, as the authors acknowledge, this statement is difficult to parse: GMFactorSketch is nothing more than an ensemble of $2K-1$ Count-Min sketches. What the authors mean to say is that for a fixed amount of space and for a certain class of problems, GMFactorSketch has provably tighter error bounds than Count-Min-Vec; they go on to empirically corroborate that the former outperforms the latter on a particular dataset. The theoretical result is outlined below.

Theorem 7. *Assume that we have a stream such that $\tilde{P} = \bar{P}$. Let P be the distribution of $K+1$ variables such that*

$$P(x) = P_1(x_2) \prod_{k=2}^{K+1} P_k(x_k | x_1),$$

where $P_k(x_k|x_1) \leq 1/2$ for $k \in \{2, 3, \ldots, K+1\}$. Let m be the number of bins in GMFactorSketch *and let $m' = 2Km$ be the number of bins in* Count-Min-Vec*. Then if $4eK \leq m \leq 2^K/K$, then*

$$P(x) \prod_{k=2}^{K+1} (1 + \epsilon_k) \leq P(x) + e/m'.$$

The proof is not particularly insightful; we omit it for the sake of brevity. Instead, we briefly discuss the theorem's assumptions and implications. Note that the assumption that $\tilde{P} = \bar{P}$ is somewhat reasonable. If the observations are sampled i.i.d. from P, then the two estimators are in fact consistent as $n \to \infty$ [KBG+16]. Since our model of interest is the streaming one, it is not unreasonable to think of the number of observations growing unbounded positive. Note, however, that the distribution P is precisely a naive Bayes graphical model. The assumption of conditional independence given x_1 is perhaps too strong and likely will not hold for most real-world phenomena, and we are left wondering how the error bounds behave as the assumption degrades. It is encouraging nonetheless that GMFactorSketch empircally outperformed Count-Min-Vec, though we might expect as much: the latter algorithm coarsely hashes the entire observation vectors, whereas the former maintains fine-grained frequency estimates.

References

[Bil15] Misha Bilenko. Big learning made easy with counts. 2015.

[CCFC02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, languages and programming*, pages 693–703. Springer, 2002.

[CM05] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, April 2005.

[Fox14] Emily Fox. Tackling an unknown number of features with sketching. 2014.

[KBG$^+$16] Branislav Kveton, Hung Hai Bui, Mohammad Ghavamzadeh, Georgios Theocharous, S. Muthukrishnan, and Siqi Sun. Graphical model sketch. *CoRR*, abs/1602.03105, 2016.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.