Notes on Sparse Matrix Hashing for Model Counting

Aaryan Gupta

March 20, 2022

1 Short Summary and Need

1.1 Before sparse hashing

The problem is stated as follows- given a CNF formula, count approximate number of solutions with probabilistic guarantees. We hash the domain to a randomised new domain with a value x of the domain hashed as Ax$+b$ with 0's and 1's in both A and b randomly chosen with probabililty 1/2 each. Instead of counting and iterating over all possible x in domain, we restrict ourselves to a small random cell (chosen using b) in hashed domain and count number of solutions in that hashed domain using a SAT solver. We then multiply this number by the number of such small cells to get an approximate value of number of solutions of original CNF formula. Probabilistic guarantees are relatively easy to prove as the hashing is 2-universal. I read about this stuff from the IJCAI 2016 paper.

1.2 Need for sparse hashing

An hashing is equivalent to a set of constraints to the variables involved. SAT solvers much work faster if the constraints involve fewer variables. However, 2-universal hashing for a CNF formula of n variables involves constraints with $n/2$ variables on average. So matrices A such that the number of 1's are much less than the number of 0's are called sparse matrices and will lead to smaller constraints. Experimentally, this seems to work very well. However, theoretical probabilistic guarantees are hard to prove. This is pretty much what the LICS 2020 paper aims to tackle.

1.3 Selecting a Special Hash family

We choose the hashing matrix A such that in the ith row, a 1 occurs with the probability p_i, which is a function of i. This function involves the inverse binary entropy function and is approximated as $\mathcal{O}(\frac{\log_2 i}{i})$. A key result in the paper is that this special hashing (called Rennes) has a bounded dispersion index. This bounded dispersion index is used to then make probabilistic guarantees.

1.4 Changes to the algorithm

The main approxMC algorithm remains more or less the same with a few changes in the parameters. The main change occurs in the approxMCCore algorithm and it applies the sparse hashing method developed in the paper for counting the solutions. The algorithms are pretty small and neat for such long papers and hairy math :)

2 Problem Setup

2.1 (ε, δ)-guarantees

A *probably approximately correct* (or PAC) counter is a probabilistic algorithm $\mathsf{ApproxCount}(\cdot,\cdot,\cdot)$ that takes as inputs a formula F, a tolerance $\varepsilon > 0$, and a confidence $\delta \in (0, 1]$, and returns a (ε, δ)-estimate c, i.e., $\mathsf{Pr}\Big[\frac{|sol(F)|}{1+\varepsilon} \le c \le (1+\varepsilon)|sol(F)|\Big] \ge 1-\delta$. PAC guarantees are also sometimes referred to as (ε, δ)-guarantees.

2.2 Prefix Hashing

Let h be a hash function sampled. Then $h^{(m)}$ denotes the m-th prefix slice. We determine the m-value in prefix slices according to the thresh to offer maximum accuracy.

2.3 Concentrated Hash families

Let $\mathsf{qs}, k \in \mathbb{N}$, $\rho \in (0, 1/2]$. A family of hash functions $\mathcal{H}(n, n)$ is *prefix-(ρ, qs, k)-concentrated*, if for each m with $\mathsf{qs} \leq m \leq n$, and $S \subseteq \{0,1\}^n$ where $|S| \leq k \cdot 2^m$, $\alpha \in \{0,1\}^n, h \xleftarrow{R} \mathcal{H}$, we have

$$\mathsf{E}[\mathsf{Cnt}_{\langle S,m\rangle}] = \frac{|S|}{2^m} \tag{1}$$

$$\frac{\sigma^2[\mathsf{Cnt}_{\langle S,m\rangle}]}{\mathsf{E}[\mathsf{Cnt}_{\langle S,m\rangle}]} \leq \rho \tag{2}$$

3 Bounding the Dispersion Index

3.1 Setup

The first step is to obtain a closed form expression for the upper bound on dispersion index for an arbitrary set $S \subseteq \{0,1\}^n$. To this end, we focus on obtaining an expression that depends on n, $|S|$ and the range of hash function, i.e., m for $h^{(m)}$.

For $1 \leq i \leq n-1$, $p_i \in (0, \frac{1}{2}]$, consider the family $\mathcal{H}_{\{p_i\}}(n,n) \triangleq \{h : \{0,1\}^n \to \{0,1\}^n\}$ of functions of the form $h(x) = \boldsymbol{A}x + \boldsymbol{b}$ with $\boldsymbol{A} \in \mathbb{F}_2^{n\times n}$ and $\boldsymbol{b} \in \mathbb{F}_2^{n\times 1}$ where the entries of $\boldsymbol{A}[i]$ (for $1 \leq i \leq n$) and $\boldsymbol{b}$ are independently generated according to Bern(p_i) and Bern($\frac{1}{2}$) respectively. For $1 \leq m \leq n$, let

$$q(w,m) = \prod_{j=1}^{m} \left(\frac{1}{2} + \frac{1}{2}(1-2p_j)^w\right)$$

$$r(w,m) = q(w,m) - \frac{1}{2^m}$$

$c_S(w,x)$ is the number of vectors in S that are at a Hamming distance of w from x. $c_S(w)$ is the number of pairs of vectors in S that are at Hamming distance w from each other.

3.2 Exact Expressions for $\sigma^2[\mathsf{Cnt}_{\langle S,m\rangle}]$ i.e. Variance

For all $\tau \in \{0,1\}^n$, we have

$$Pr(\boldsymbol{A}^{(m)}\tau = \boldsymbol{0}) = q(w,m)$$

First Equation-
$\sigma^2[\mathsf{Cnt}_{\langle S,m\rangle}]$=

$$\sum_{y_1,y_2 \in S} Pr[h^{(m)}(y_1) = \alpha^{(m)}, h^{(m)}(y_2) = \alpha^{(m)}] - E[\mathsf{Cnt}^2_{\langle S,m\rangle}]$$

Upon substituting the probability term-

$$\sigma^2[\mathsf{Cnt}_{\langle S,m\rangle}] = 2^{-m} \sum_{w=0}^{n} c_S(w) r(w,m)$$

3.3 Maximising Variance

S is a down set if it has the following property: if $x2 \in S$, then for all $x1 \subseteq x2$, $x1 \in S$.

S is a left compressed set if it has the following property: if $x2 \in S$, then for all lexicographically smaller $x1$ $|x1| = |x2|$, with $x1 \in S$

The variance takes maximum value for a left compressed and down set S.

3.4 Rashtchian's Inequality

For a left-compressed and down set S, $c_S(w) \leq 2 \cdot \left(\frac{8e\sqrt{n\cdot\ell}}{w}\right)^w \cdot |S|$ where $\ell = \lceil \log |S| \rceil$.

Substituting this back to the original expression, we get that
for any $S \subseteq \{0,1\}^n$
$\frac{\sigma^2[\mathsf{Cnt}_{\langle S,m\rangle}]}{\mathsf{E}[\mathsf{Cnt}_{\langle S,m\rangle}]} \leq \sum_{w=0}^{\ell} 2 \cdot \left(\frac{8e\sqrt{n\cdot\ell}}{w}\right)^w r(w,m)$ with $\ell = \lceil \log |S| \rceil$

The expression on the right can be evaluated numerically once ℓ is fixed. This value is ρ in concentrated hashing. This means that ρ and k are directly related to each other.

Algorithm 1 $\mathsf{ApproxMC5Core}(F, \rho, \mathsf{thresh})$

```
Choose $h$ at random from $\mathcal{H}_\rho(n, n)$;
Choose $\alpha$ at random from $\{0, 1\}^n$;
$Y \leftarrow \mathsf{BoundedCount}(F \wedge (h^{(n)})^{-1}(\alpha^{(n)}), \mathsf{thresh})$;
if ($|Y| \geq \mathsf{thresh}$) then return $2^n$
end if
$m \leftarrow \mathsf{LogSATSearch}(F, h, \alpha, \mathsf{thresh})$;
$\mathsf{Cnt}_{\langle F,m \rangle} \leftarrow \mathsf{BoundedCount}(F \wedge (h^{(m)})^{-1}(\alpha^{(m)}), \mathsf{thresh})$;
return $(2^m \times \mathsf{Cnt}_{\langle F,m \rangle})$;
```

Algorithm 2 $\mathsf{ApproxMC5}(F, \varepsilon, \delta, \rho, \mathsf{qs})$

```
$\mathsf{thresh} \leftarrow 1 + 9.84 \cdot \rho \cdot \left(1 + \frac{\varepsilon}{1+\varepsilon}\right)\left(1 + \frac{1}{\varepsilon}\right)^2$;
$\mathsf{iniThresh} \leftarrow \mathsf{thresh} * 2^{\mathsf{qs}+3}$
$Y \leftarrow \mathsf{BoundedCount}(F, \mathsf{iniThresh})$;
if ($Y < \mathsf{iniThresh}$) then return $|Y|$;
end if
$t \leftarrow \lceil 17 \log_2(3/\delta) \rceil$;
$\mathsf{nCells} \leftarrow 2$; $C \leftarrow \mathsf{emptyList}$; $\mathsf{iter} \leftarrow 0$;
repeat
    $\mathsf{iter} \leftarrow \mathsf{iter} + 1$;
    $\mathsf{nSols} \leftarrow \mathsf{ApproxMC5Core}(F, \rho, \mathsf{thresh})$;
    $\mathsf{AddToList}(C, \mathsf{nSols})$;
until ($\mathsf{iter} < t$);
$\mathrm{finalEstimate} \leftarrow \mathsf{FindMedian}(C)$;
return $\mathrm{finalEstimate}$
```

4 The magical $\mathcal{H}_{Rennes}$ Hashing family

Let $k, n \in \mathbb{N}$ and let $H^{-1} : [0, 1] \rightarrow [0, \frac{1}{2}]$ be the inverse binary entropy function restricting its domain to $[0, \frac{1}{2}]$ so that the inverse is well defined. We then define $\mathcal{H}^k_{Rennes}(n, n) \triangleq \{h : \{0, 1\}^n \rightarrow \{0, 1\}^n\}$ to be the family of functions of the form $h(x) = \boldsymbol{A}x + \boldsymbol{b}$ with $\boldsymbol{A} \in \mathbb{F}_2^{n \times n}$ and $\boldsymbol{b} \in \mathbb{F}_2^{n \times 1}$ where the entries of $\boldsymbol{A}[i]$ (for $1 \leq i \leq n$) and $\boldsymbol{b}$ are independently generated according to Bern(p_i) and Bern($\frac{1}{2}$) respectively, where $p_i \geq min(\frac{1}{2}, \frac{16}{H^{-1}(\delta)} \cdot \frac{\log_2 i}{i})$ for $\delta = \frac{i}{i+\log_2 k}$, and for $1 \leq i \leq n-1, p_i \geq p_{i+1}, p_i \in (0, \frac{1}{2}]$.

$\mathcal{H}^k_{Rennes}$ **is prefix-**(ρ, qs, k)**-concentrated.**

When we prove this theorem, qs turns out to be dependent on ρ and k both. qsis the smallest value of m for which the family is concentrated.

5 The ApproxMC Algorithm

5.1 Subroutines

BoundedCount(F**,**thresh**)**

Counts the solutions of F until thresh number of solutions have been counted and returns the count.

LogSATSearch(F**,** ρ**,** α**,** thresh**)**

Returns the appropriate m such that the divided cell is small enough to have thresh or less solutions.

5.2 ApproxMC5Core

algorithm 1 above

5.3 ApproxMC

algorithm 2 below