Technical Perspective: Model Counting Meets Distinct Elements in a Data Stream

David P. Woodruff
Carnegie Mellon University
dwoodruf@cs.cmu.edu

Model counting is the problem of approximately counting the number $|\mathrm{Sol}(\phi)|$ of satisfying assignments to a given model ϕ, which could, for example, be a formula in conjunctive normal form (CNF) or a formula in disjunctive normal form (DNF). One is interested in an (ϵ, δ)-approximation scheme, which is a randomized algorithm which outputs a number X for which $(1-\epsilon)|\mathrm{Sol}(\phi)| \leq X \leq (1+\epsilon)|\mathrm{Sol}(\phi)|$ with probability at least $1-\delta$. There are fully polynomial-time (ϵ, δ)-approximation schemes for DNFs, while for CNFs it is NP-hard to devise such a scheme. Nevertheless, such a scheme is possible for CNFs with a small number $(\mathrm{poly}(|\phi|, 1/\epsilon, 1/\delta))$ of calls to an NP-oracle. This often suffices in practice using efficient SAT solvers to replace the oracle calls.

Seemingly unrelated to model counting is the problem of computing over data streams. Here an algorithm is given a stream $a_1, a_2, \ldots, a_m$ of m elements, each drawn from a universe $\{0,1\}^n$ of $N = 2^n$ items, and the goal is to estimate statistics of the stream using low memory with a single pass over the data stream. A statistic of particular importance is the number F_0 of distinct elements. Computing F_0 exactly requires $\Omega(\min(m, 2^n))$ memory. However, there are very low memory (ϵ, δ)-approximation schemes for F_0.

The notion of efficiency for model counting is very different than that for estimating the number of distinct elements in a data stream. In model counting one wants to minimize the *time complexity*, or if the problem is NP-hard, one seeks to minimize the number of NP-oracle calls. In contrast, in the data stream model often the main goal is to minimize the *space complexity*, i.e., the memory required, to estimate F_0. It is not at all clear that these two models are related.

The paper "Model Counting Meets F_0 Estimation" [4] by A. Pavan, N. V. Vinodchandran, A. Bhattacharyya, and K.S. Meel is a beautiful work which gives a generic transformation of data stream algorithms for F_0-estimation to algorithms for model counting. The authors also show a converse to their generic transformation. Namely, by, framing F_0-estimation as a special case of DNF model counting, the authors obtain a generic algorithm for tasks such as F_0-estimation over affine spaces and DNF sets.

The starting point of the above paper is the observation that a hashing-based technique of Stockmeyer for model counting for CNFs [5], extended by Chakraborty, Meel, and Vardi to DNFs [2], actually uses the same techniques as an F_0-estimation data stream algorithm of Gibbons and Tirthapura [3].

The idea for model counting for CNFs is to hash all possible solutions (assignments to variables) using a hash function h to the range $\{0,1\}^m$. Invoking an NP oracle multiple times, one can find the smallest value of m for which $h^{-1}(0^m)$ contains a number t between $c\epsilon^{-2}$ solutions and $2c\epsilon^{-2}$ solutions, for a constant $c > 0$, at which point one can estimate the total number of solutions as $t \cdot 2^m$, which can be shown to give an (ϵ, δ)-approximation scheme.

The idea for F_0-estimation in a data stream is to use a hash function h to map the universe $\{0,1\}^n$ to $\{0,1\}^n$ but to first restrict the range to $\{0,1\}^m$ for an $m \leq n$. Then one stores the set $h^{-1}(0^m)$ of preimages, and if this set size exceeds $2c\epsilon^{-2}$, one increases m by 1 and only retains those items in the current set that are also in the set $h^{-1}(0^{m+1})$ of preimages. In this way, one eventually finds an m for which the set $h^{-1}(0^m)$ of preimages contains a number t between $c\epsilon^{-2}$ distinct items and $2c\epsilon^{-2}$ distinct items, and can estimate F_0 as $t \cdot 2^m$, which like the algorithm for model counting, can be shown to given an (ϵ, δ)-approximation scheme.

Inspired by this striking similarity, Pavan et al. [4] give a generic transformation which captures all three distinct element algorithms in [1]. One can show that implementing the various distinct element algorithms efficiently in the context of model counting boils down to finding the minimum value that a hash function takes over a structured set of values, such as the solutions to a CNF or DNF formula. This can be done efficiently by choosing certain linear algebraic hash functions (affine maps with a Toeplitz structure for efficiency) and performing Gaussian elimination.

1. REFERENCES

[1] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In RANDOM, 2002.
[2] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In IJCAI/AAAI, 2016.
[3] Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In SPAA, 2001.
[4] Aduri Pavan, N. V. Vinodchandran, Arnab Bhattacharya, and Kuldeep S. Meel. Model counting meets f_0 estimation. In PODS, 2021.
[5] Larry J. Stockmeyer. The complexity of approximate counting. In STOC, 1983.