Boolean Tuple Set Algorithm

Ohad Asor

April 15, 2018

We present an algorithm for storing and querying fixed-length vectors of booleans. We use statistical methods in order to prune the search space. We assume the reader is familiar with Least-Squares techniques. We also assume that the queries are fixed, known beforehand, and their result is always desired. Let X be our set of tuples $X = \{\boldsymbol{x}_i\}_{i=1}^n$ where $X \subset \{\pm 1\}^k$. First we assume that if $\boldsymbol{t} \in X$ then also $-\boldsymbol{t} \in X$. This is done by adding 1 as a new element in the beginning of every vector, and makes the distribution of our population to be symmetric, so all odd means (including the mean) are zero. The covariance matrix is therefore:

$$\Sigma_X = \mathbb{E}_{\boldsymbol{x} \in X}\left[\boldsymbol{x}\boldsymbol{x}^T\right]$$

Denote by X' the set of $\{\boldsymbol{x}'_i\}_{i=1}^n$ such that $\boldsymbol{x}'_i = \sqrt{\Sigma_X^{-1}}\boldsymbol{x}_i$. So X' has unit covariance and still all odd moments are zero. Suppose we'd like to find all vectors on our set given some of the bits but not all, e.g. all vectors with first and third bit set to one. Denote this query by $\boldsymbol{q}$ such that $q_i = \pm 1$ if we'd like that bit to be set/unset, or 0 if we don't care. Then

$$\left\|\sqrt{\Sigma_X^{-1}}\boldsymbol{q}\right\| \tag{1}$$

is the minimum distance of any matching vector from the mean. By that we pruned the search space if we store the vectors sorted by to their norm, $\|\boldsymbol{x}'_i\|$, as a function of the number of bits given, since the more given bits, the larger is (1). Moreover, the larger the query's norm is, the even larger the contribution to the pruning space, as the size of the sets sorted by distance goes down linearly with the norm, as Chebyshev's inequality would suggest. Observe that we couldn't possibly find a better distance function from the point of view of the first three moments, not only two.

For each new batch of bitvectors to add to X, we need to calculate X' and store its vectors sorted according to their norm, as well as resolve the queries (which we assumed fixed). This is done in a single separate pass over X. We then renormalize the queries, and with a second pass over the data we check only the tuples with high enough norm, as the queries dictate.

We're left to point how to compute $\sqrt{\Sigma_X^{-1}}$ online. This is done by rank-1 update of LDL^T decomposition (directly from the new vectors) using the algorithm by Davis&Hager (2001), implemented in libEigen3. The additive update of the square root turns to be a multiplicative update to the vectors in X':

$$\left(\sqrt{D}L + H\right)\boldsymbol{t} = \sqrt{D}L\boldsymbol{t} + \boldsymbol{h} \implies H\boldsymbol{t} = \boldsymbol{h}$$

and the norm is updated accordingly:

$$\begin{aligned}
&\left\|\left(\sqrt{D}L + H\right)\boldsymbol{t}\right\|^2 \\
&= \left[\left(\sqrt{D}L + H\right)\boldsymbol{t}\right]^T \left[\left(\sqrt{D}L + H\right)\boldsymbol{t}\right] \\
&= \boldsymbol{t}^T \left(L^T\sqrt{D} + H^T\right)\left(\sqrt{D}L + H\right)\boldsymbol{t} \\
&= \|\boldsymbol{t}\|^2 + \boldsymbol{t}^T \left(L^T\sqrt{D}H + H^T\sqrt{D}L + H^T H\right)\boldsymbol{t}
\end{aligned}$$