CSE 4502/5717 Big Data Analytics. Fall 2022

Model Exam III Solutions

1. Let D be the input database that has n transactions. Pick a random sample S of transactions from D such that $|S| = c \log n$ for some constant c. Identify and output all the items in S whose support in S is at least $\frac{1}{8}$. This can be done as follows. Let the total number of items in all the transactions in S be $k \log n$, k being a constant. Sort all of these items together. Scan through the sorted list and identify all the items whose support in S is $\geq \frac{1}{8}$.

   Analysis: Clearly, the run time of the algorithm is $O(\log n \log \log n)$.

   Let i be an item in D whose support in D is $\geq \frac{1}{4}$. Let Q be the set of transactions in D in which item i occurs. Let $|Q| = q$. We know that $q \geq \frac{n}{4}$. Let t be any transaction in Q. Probability that t occurs in S is $\frac{s \log n}{n}$. The number m of transactions of Q that occur in S is $B\left(q, \frac{c \log n}{n}\right)$. The expected value of m is $\frac{qc \log n}{n} \geq \frac{c \log n}{4}$.

   Using Chernoff bounds, probability that $m < \frac{c \log n}{8}$ is $\leq \exp\left(-\frac{c \log n}{32}\right)$. Let the total number of items in the n transactions be jn where j is a constant. Probability that at least one of the frequent items of D does not have a support of $\geq \frac{1}{8}$ in S is $\leq jn \exp\left(-\frac{c \log n}{32}\right)$. This probability will be $\leq n^{-\alpha}$ if $c \geq 32(\alpha + 2)$.

   Let the total number of items in all the transactions in S be $k \log n$, k being a constant. Sort all of these items together in $O(\log n \log \log n)$ time. Scan through the sorted list and identify all the items whose support in S is $\geq \frac{1}{8}$. Output these items. Clearly, the run time of the algorithm is $O(\log n \log \log n)$.

2. Note that there are $\binom{d}{k} < d^k$ possible k-itemsets. We can generate all possible k-itemsets in $O(1)$ time using $\binom{d}{k}$ processors. These are the candidates. Followed by this, we count the support for each possible k-itemset. We assign $\frac{n}{\log n}$ processors for each k-itemset candidate. The support can be computed in $O(\log n)$ time using a prefix computation. Details follow.

   1) **for** each candidate c **in parallel do**
   2)   **for** each transaction t **in parallel do**
   3)   Processor $P_{c,t}$ computes $b_{c,t}$ as 1 if c is in t and zero otherwise;
   4) $\frac{n}{\log n}$ processors perform a prefix sums computation on $b_{c,1}, b_{c,2}, \ldots, b_{c,n}$.
   5)   If this sum is $\geq$ *minSupport*, output c as a frequent k-itemset;

   Analysis: All the candidates can be generated in $O(1)$ time. For a given candidate c, if we have n processors, we can complete step 3 in $O(1)$ time. Using the slow-down lemma, this can be done in $O(\log n)$ time using $\frac{n}{\log n}$ processors. Step 4 takes $O(\log n)$ time and step 5 takes $O(1)$ time. Thus the total run time is $O(\log n)$.

3. Evaluate the given polynomial at each integer in the range $[1, cn]$. If the polynomial evaluates

to zero at any v, then v is a root. We can evaluate the polynomial at cn points in $O(n \log^2 n)$ time as has been mentioned in class.

4. We first get an upper bound on d (within a factor 2) using the doubling trick. Followed by this, we use binary search to get the value of d. Once we know d, we interpolate the first d pairs in X to get and output the right polynomial.

   $k := 1$;
   repeat
   1. Interpolate the first k pairs of X to get a polynomial $f(x)$;
   2. Check if $f(r_i) = a_i$ for each $i, 1 \leq i \leq n$;
   3. If yes, **then quit else** $k := 2k$;
   forever

   The k we get from the above code is an upper bound on d such that $k \leq 2d$. Now we perform a binary search in the range $\left[\frac{k}{2}, k\right]$ to get the actual value of d in a similar manner. Once we know the value of d we do an interpolation on the first d pairs of X to get the correct polynomial $f(x)$.

   Note that one execution of step 2 above can be done in $O(n \log^2 k) = O(n \log^2 d)$ time. In the entire algorithm we perform step 2 a total of $O(\log d)$ times. Thus the total time for step 2 is $O(n \log^3 d)$. We also perform step 1 a total of $O(\log d)$ times and each execution of step 1 takes $O(k \log^3 k) = O(d \log^3 d)$ time. Thus the total time for step 1 is $O(d \log^4 d)$. Step 3 takes a total of $O(\log d)$ time.

   Thus, the run time of the entire algorithm is $O(n \log^3 d + d \log^4 d)$. If n is much larger than d, then this run time is $O(n \log^3 d)$.

5. The loss function is $L(w_1, w_2) = (w_2 - 4)^2 + (w_1 - 3)^2 + (w_1 + w_2 - 6)^2 + (2w_1 + w_2 - 10)^2$ $= 6w_1^2 + 3w_2^2 + 6w_1w_2 - 58w_1 - 40w_2 + 161$. We want to have: $\frac{\partial L}{\partial w_1} = 0$ and $\frac{\partial L}{\partial w_2} = 0$.
   $\frac{\partial L}{\partial w_1} = 0$ implies that $12w_1 + 6w_2 = 58$ and $\frac{\partial L}{\partial w_2} = 0$ implies that $6w_1 + 6w_2 = 40$. Solving these two equations, we get: $w_1 = 3$ and $w_2 = \frac{11}{3}$.

6. Here is a multilevel perceptron for realizing the Boolean function $F(x_1, x_2, x_3, x_4) = x_1\bar{x_3}x_4 + x_2\bar{x_3} + x_1x_2\bar{x_4}$:

Figure 1: A neural network for $F(x_1, x_2, x_3, x_4) = x_1\bar{x_3}x_4 + x_2\bar{x_3} + x_1x_2\bar{x_4}$.