CSE 4502/5717 Big Data Analytics
Fall 2022; Homework 3 Solutions

1. (a) Note that for a pair of items to be frequent, it has to occur in at least one transaction. Thus the only two-itemsets we have to generate as candidates are the pairs of items we can generate from each transaction. As a result, the number of candidates is $O(n)$. For each such candidate we have to compute the support.

   Support for the candidates can be computed as described in class. In particular, we use a hash tree to store all the candidates. The expected size of each leaf in the hash tree is $O(1)$. Then, we do the following:

   for each transaction T in DB **do**
       for each pair (i, i') of items in T do
           Use the hash tree to increment of the support of (i, i') by 1.
   Output all the pairs of items that have a support of $\geq minSupport$.

   It is clear that the above algorithm has an expected run time of $O(n)$.

   (b) Note that if X is an itemset with k items, then we can form $2^k - 2$ association rules using X. This in turn means that the total number of association rules we can form from a set I of d items is $\sum_{k=2}^{d} \binom{d}{k}(2^k - 2) = \sum_{k=2}^{d} \binom{d}{k} 2^k - 2\sum_{k=2}^{d} \binom{d}{k}$ $= 3^d - 2d - 1 - 2(2^d - d - 1) = 3^d - 2^{d+1} + 1$.

2. Let the transactions in the database be $t_1, t_2, \ldots, t_q$. Note that any item in the database can be represented as an integer in the range $[1, n^c]$.

   S is an empty sequence to begin with;
   for $i = 1$ **to** q **do**
       for every item $a \in t_i$ **do**
           Add a to the sequence S;
   Sort S using the integer sorting algorithm;
   Scan through the sorted sequence to count the support for each item and output those that have enough support.

   Clearly, the above algorithm runs in $O(n)$ time.

3. The coefficients of the polynomial are given by $\binom{n}{i}$, $i = 0, 1, \ldots, n$.
   Since $\binom{n}{i} = \binom{n}{i-1}$ $(n - i + 1)/i$, the coefficients can be computed in time $O(n)$.
   FFT can be used to multiply two nth degree polynomials in $O(n \log n)$ time. We can compute the coefficients of $(1+x)^n$ by multiplying $(1 + x)^{n/2}$ and $(1 + x)^{n/2}$. If $T(n)$ is

the time needed to compute $(1+x)^n$, then, $T(n) = T(n/2) + O(n \log n)$, which solves to $O(n \log n)$.

4. Let A be a *Toeplitz* matrix and B be an $n \times 1$ vector. Let's consider the multiplication of the lower triangular part of A (including the main diagonal elements) with B.

   Let the elements of A be the following:

   $a_{n,n} = a_{n-1,n-1} = a_{n-2,n-2} = \ldots = a_{2,2} = a_{1,1} = a_1$

   $a_{n,n-1} = a_{n-1,n-2} = a_{n-2,n-3} = \ldots = a_{2,1} = a_2$

   $a_{n,n-2} = a_{n-1,n-3} = a_{n-2,n-4} = \ldots = a_{3,1} = a_3$

   $\vdots$

   $a_{n,1} = a_n$

   Let the elements of B be the following:

   $b_{1,1} = b_1, b_{2,1} = b_2, \ldots, b_{n,1} = b_n$

   Multiplication of the lower triangular part of A with B gives the following:

$$\begin{bmatrix} a_1 b_1 \\ a_2 b_1 + a_1 b_2 \\ a_3 b_1 + a_2 b_2 + a_1 b_3 \\ \vdots \end{bmatrix}$$

   We can notice that the above is nothing but the multiplication of two polynomials $(a_1 + a_2 x + a_3 x^2 + \ldots)$ and $(b_1 + b_2 x + b_3 x^2 + \ldots)$.

   Since the plolynomials can be multiplied in $O(n \log n)$ time, the matrices can also be multiplied in $O(n \log n)$ time. Multiplication of the upper triangular elements of A with B is symmetrical to the above and it would not affect the asymptotic complexity.

5. The loss function is $L(w_1, w_2) = (w_2 - 2)^2 + (w_1 - 4)^2 + (w_1 + w_2 - 4)^2 = 2w_1^2 + 2w_2^2 + 2w_1 w_2 - 16w_1 - 12w_2 + 28$. 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 $2w_1 + w_2 = 8$ and $\frac{\partial L}{\partial w_2} = 0$ implies that $w_1 + 2w_2 = 6$. Solving these two equations, we get: $w_1 = \frac{10}{3}$ and $w_2 = \frac{4}{3}$.