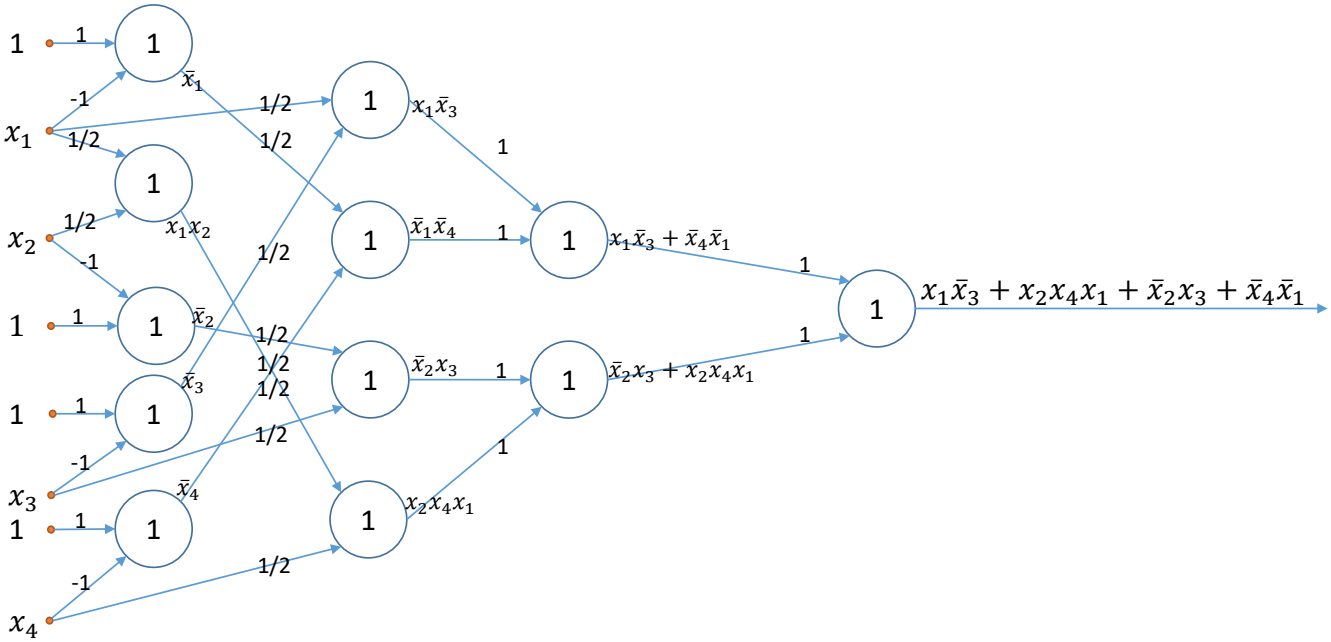
CSE 4502/5717 Big Data Analytics
Fall 2019; Homework 3 Solutions

1. The loss function is $L(w_1, w_2) = (w_2 - 2)^2 + (w_1 - 4)^2 + (w_1 + w_2 - 4)^2 = 2w_1^2 + 2w_2^2 + 2w_1w_2 - 16w_1 - 12w_2 + 28$. 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 $w_2 = \frac{4}{3}$.

2. Here is one possible solution:



3. Let the activation values of the nodes in level A be o_i^A, $1 \leq i \leq m$ and let the activation values of the nodes in level B be o_j^B, $1 \leq j \leq n$. Let the weight of the edge from node i of level A to the node j of level B be w_{ji}^B, $1 \leq i \leq m$, $1 \leq j \leq n$. Also, let the weighted input at node j of level B be $I_j^B = \sum_{i=1}^{m} w_{ji}^B o_i^A$, for $1 \leq j \leq n$. Note that $o_j^B = \sigma(I_j^B)$, for $1 \leq j \leq n$, where $\sigma(.)$ is the activation function. Thus, if we can compute the weighted input values we can compute the activation values in an additional $O(1)$ time using n CREW PRAM processors.

   If $O^A = (o_1^A \quad o_2^A \quad \ldots \quad o_m^A)^T$ and $I^B = (I_1^B \quad I_2^B \quad \ldots \quad I_n^B)^T$, then, we realize that

$I^B = W\ O^A$, where

$$W = \begin{bmatrix} w_{11}^B & w_{12}^B & \dots & w_{1m}^B \\ w_{21}^B & w_{22}^B & \dots & w_{2m}^B \\ \dots & & & \\ w_{n1}^B & w_{n2}^B & \dots & w_{nm}^B \end{bmatrix}.$$

Therefore, computing the weighted sums for the nodes of level B can be done with a matrix-vector multiplication, where the matrix is of size $n \times m$ and the vector is of size $m \times 1$. Consider the problem of computing the dot product between the ith row of W and O^A, for any $i,\ 1 \leq i \leq n$. This can be done in $O(\log m)$ time using $\frac{m}{\log m}$ CREW PRAM processors. We can employ the optimal prefix computation algorithm here.

As a result, we can compute I^B in $O(\log m)$ time using $\frac{nm}{\log m}$ CREW PRAM processors. □

4. (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 $\sum_{k=2}^{d} \binom{d}{k}(2^k - 2) = \sum_{k=2}^{d} \binom{d}{k} 2^k - 2\sum_{k=2}^{d} \binom{d}{k}$ $= 3^d - 2d - 1 - 2(2^d - d - 1) = 3^d - 2^{d+1} + 1$.