1 The Credit Assignment Algorithm

1.1 Notations of CLT

Let $F = \{f_1, \dots, f_k\}$ be a set of features (topics) of actions (propagated items, messages). The Content-Aware LT (CLT) model defines the activation probability of a node v as

$$P(a|v) = \sum_u p_{uv} + q_{uv}(|F_v \cap F_a|) \geq \theta_v$$

where u is in-neighbour of v; p_{uv}, q_{uv} are labels of the edge (u, v) that indicate the influence of u on v; F_v is a set of topics the node v is interested in, F_a is a set of topics that characterize the action i; and $\theta_v \in [0, 1]$ is a threshold that is randomly chosen by the node v in the beginning of propagation.

The parameters of the model are p_{uv}, q_{uv}, F_v and F_i, where the last two are discrete sets that need to be determined.

Let $t(a)$ be a timestamp of the action, and let us call an action a performed by v as *adapted* by u if both have performed the action sequentially in time

$$t_v(a) < t_u(a)$$

1.2 The Credit Assignment in the LT model

The algorithm is based on the assumption that *for each* ***successful*** *activation, all in-neighbours share the same "credit" for that activation.* For example, if two nodes have posted a message about Trump, and then their common neighbour also posted about Trump, then former two nodes are both 50% responsible for activating the third node. *It does not matter whether the ground-truth activation* ***probabilities*** *might be different from each other*, because nodes share credits only for successful actions.

For the regular LT model, the credit assignment algorithm determines influence probabilities as

$$p_{uv} = \frac{\sum_a credit_{uv}(a)}{A_v}$$

where a is an action that propagates in one diffusion instance (cascade). The credit is defined as

$$credit_{uv}(a) = \frac{1}{\sum_{w \in S} I(t_w(a) < t_u(a))}$$

where I is an indicator that an in-neighbour w of a node u has been activated with an action a earlier than u.

1.3 The Credit Assignment in the CLT model

First, we must assume that topics of actions and user preferences are predefined, i.e. must be determined *before* applying the credit assignment algorithm. This is a crucial difference to the EM algorithm.

Second, for the CLT model, we still use the same *equal credit share* assumption. That means, for each successful action an in-neighbour takes $\frac{1}{d}$ credit, where d is the total number of in-neighbours that share credit for that action.

However, instead of a single value p_{uv}, we should find coefficients p_{uv} and q_{uv}, and these coefficients are dependent on the topics of each action. We will use the *Ordinary Least Squares* (OLS) estimator. Let S be a set of in-neighbours of u. Let A_v be a set of all action performed by v, and A_{vu} be a set of actions performed by v and adapted by u. Then, for each $a \in A_v$, the equal share assumption implies that

$$p_{uv} + q_{uv}\alpha(a) = I(t_v(a) < t_u(a)) \frac{1}{\sum_{w \in S} I(t_w(a) < t_u(a))} \tag{1}$$

where the indicator function $I(t_v(a) < t_u(a))$ shows that u adopted action a at any time after v, $\sum_{w \in S} I(t_w(a) < t_u(a))$ shows the number of in-neighbours who could have influenced u if u has ever adopted a, and $\alpha(a) = |F_u \cap F_a|$ is the coefficient that shows how much topics of a are similar to preferences of u.

Let $d_u(a) = \sum_{w \in S} I(t_w(a) < t_u(a))$, and $n = |A_v|$. Then, OLS is given by

$$q_{uv} = \frac{\sum_{a \in A_{vu}} \alpha(a) \frac{1}{d_u(a)} - \frac{1}{n}(\sum_{a \in A_v} \alpha(a))(\sum_{a \in A_{vu}} \frac{1}{d_u(a)})}{\sum_{a \in A_v} \alpha^2(a) - \frac{1}{n}(\sum_{a \in A_v} \alpha(a))^2} \tag{2}$$

$$p_{uv} = \frac{1}{n} \sum_{a \in A_{vu}} \frac{1}{d_u(a)} - q_{uv} \frac{1}{n} \sum_{a \in A_v} \alpha(a) \tag{3}$$

1.4 The algorithm

Goyal et al. proposed an algorithm that calculates all necessary values for calculating p_{uv} in a minimal number of scans through data. We adapt their algorithm. In order to compute coefficients of CLT according to Eq. 3 and 2, we have to know the following statistics:

$$n_v = |A_v|$$

$$d_u(a) = \sum_{w \in S} I(t_w(a) < t_u(a))$$

$$C^1_{vu} = \sum_{a \in A_{vu}} \alpha(a) \frac{1}{d_u(a)}$$

$$C^1_v = \sum_{a \in A_v} \alpha(a)$$

$$C^2_{vu} = \sum_{a \in A_{vu}} \frac{1}{d(a)}$$

$$C^2_v = \sum_{a \in A_v} \alpha^2(a)$$

Then, coefficients are equal to

$$q_{uv} = \frac{C^1_{vu} - \frac{1}{n} C^1_v C^2_{vu}}{C^2_v - \frac{1}{n}(C^1_v)^2} \tag{4}$$

$$p_{uv} = \frac{1}{n} C^2_{vu} - q_{uv} \frac{1}{n} C^1_v \tag{5}$$

Algorithm 1 Credit Assignment

---

for all a in logs **do**
  $currentTable \leftarrow \emptyset$
  for all user tuple $< u, a, t_u >$ in chronological order **do**
    Update n_u, C_v^1, C_v^2
    $parents \leftarrow \emptyset$
    for all $v \in (v, a, t_v) \in currentTable \wedge (v, u) \in E$ **do**
      if $t_u > t_v$ **then**
        Update C_{vu}^1, C_{vu}^2
        Insert v in $parents$
    for all $v \in parents$ **do**
      update $d_v(a)$
    Add (u, a, t_u) to $currentTable$

Let E be a set of edges of a graph. Logs of actions should be sorted in chronological order.

After running Algorithm 1, coefficients for each edge are updated according to Eq. 5 and 4.

If we want to enforce a time limit within which a node can be influenced, then in all the above instead of a check $t_u > t_v$, we must check that $0 < t_u - t_v < \tau$, where τ is a time threshold.