I. Context

We are given hypergraph snapshots over time. We are trying to find out the ranking of nodes similar to a given initial node (in the current snapshot) in the hypergraph. We are doing label propagation over hypergraph snapshots over time using bringing in temporal regularization (between the vertices of different snapshots based upon domain knoweldge). For this we have come up with a cost function in which we are trying to use alternate minimization (AM) for the ranks of vertices in different snapshots one at a time. The question is that wether this over all approach of using AM is right way as right now my code is not converging for it.

II. Problem

Our aim is minimize a cost function $Q(\mathbf{f})$ where $\mathbf{f}$ is a complete vector consisting of k smaller vectors $\{\mathbf{f}_{t_1}, \mathbf{f}_{t_2}, ..., \mathbf{f}_{t_m}\}$ each of which is contains the rank of vertices in the hypergraph in t_k^{th} snapshot. The total cost function (this comes from the equation (7) which we are skipping to be terse and the longer longer second detailed pdf can be looked for it) is:

$$Q(\mathbf{f}) = \sum_{k=1}^{m} Q(\mathbf{f_{t_k}}) \tag{1}$$

where,

$$\begin{aligned} Q(\mathbf{f}_{t_k}) = \mathbf{f}_{t_k}^T \mathbf{\Delta}^{'}_{\mathbf{t_k}} \mathbf{f}_{t_k} + \mu(\mathbf{f}_{t_k} - \mathbf{y}_{t_k})^T(\mathbf{f}_{t_k} - \mathbf{y}_{t_k}) \\ +\gamma(\mathbf{f}_{t_k} - \mathbf{f}_{t_{k-1}} - \mathbf{p}\mathbf{J}_{user})^T(\mathbf{f}_{t_k} - \mathbf{f}_{t_{k-1}} - \mathbf{p}\mathbf{J}_{user}), \end{aligned} \tag{2}$$

represents the cost for the snapshot at $t = t_k$.

and where,

$$\mathbf{\Delta}^{'}_{\mathbf{t_k}} = (\mathbf{D_v})^{\mathbf{-1/2}}_{\mathbf{t_k}} \mathbf{H}_{\mathbf{t_k}} \mathbf{W}_{\mathbf{t_k}} (\mathbf{D_e})^{\mathbf{-1}}_{\mathbf{t_k}} \mathbf{H}^{\mathbf{T}}_{\mathbf{t_k}} (\mathbf{D_v})_{\mathbf{t_k}}{}^{-1/2} \tag{3}$$

is the hypergraph laplacian.

$\mathbf{J}_{user}$ is a diagonal matrix with ones in diagonal where it is a vertex of a particular kind and rest all zeros.

The above is a convex function with a natural split of optimization variables into k sets of $\{\mathbf{f}_{t_1}, \mathbf{f}_{t_2}, ..., \mathbf{f}_{t_m}\}$ and $\mathbf{p}$ being the $k+1$ variable. Therefore, until convergence we alternately optimize (Algorithm 1) these variables by calculating value of one variable by taking the values of the other variables as constants from the previous iterations. The value of p we find out using gradient descent updates in this convergence loop.

Now, Taking gradient with respect to $\mathbf{f}_{t_k}$,

$$\begin{aligned} \frac{dQ}{d\mathbf{f}_{t_k}} = (\mathbf{I} - \Delta^{'}_{t_k})\mathbf{f}_{t_k} + \mu(\mathbf{f}_{t_k} - \mathbf{y}_{t_k}) \\ +\gamma(\mathbf{f}_{t_k} - \mathbf{f}_{t_{k-1}} - \mathbf{p}\mathbf{J}_{user}) = 0 \end{aligned} \tag{4}$$

The solution to the above is a solution to the linear equation,

$$\mathbf{f_{t_k}} = ((1+\mu+\gamma)\mathbf{I} - \Delta^{'}_{t_k})^{-1}(\mu\mathbf{y}_{t_k} + \gamma\mathbf{y}_{t_{k-1}} + \mathbf{p}\mathbf{J}_{user}) \tag{5}$$

with the recursive iteration for the above equation being:

$$\mathbf{f_{t_k}} = (\Delta^{'}_{t_k} - (\mu+\gamma)\mathbf{I})\mathbf{f_{t_k}} + \mu\mathbf{y}_{t_k} + \gamma\mathbf{y}_{t_{k-1}} + \mathbf{p}\mathbf{J}_{user} \tag{6}$$

We do a gradient descent for $\mathbf{p}$ and the gradient will only involve the third term of equation (2) as rest all shall be constant in our alternate minimization process. The algorithm for the proposed optimization is shown below. For every k we have $\mathbf{y}_{t_k}$ initialized to 1 at the index of the current query user and zero otherwise.

$$\begin{aligned} \text{Q(f)} = & \sum_{k=1}^{m} \left(\frac{1}{2} \sum_{i,j=1}^{n} \sum_{e \in E} \frac{1}{\delta(e,t_k)} \right. \\ & \left. \sum_{\{v_i(t_k), v_j(t_k)\} \subseteq w(e,t_k)} \left\| \frac{f_i(t_i)}{\sqrt{d(v_i,t_k)}} - \frac{f_j(t_i)}{\sqrt{d(v_j,t_k)}} \right\|^2 \right) \\ & + \sum_{k=1}^{m} \left(\mu \sum_{i=1}^{n} \|f_i(t_k) - y_i(t_k)\|^2 \right) \\ & + \sum_{k=1}^{m} \left(\gamma \sum_{i=1}^{n} \|f_i(t_k) - f_i(t_{k-1}) - p_i \mathbf{J_{user}}(i,i)\|^2 \right) \end{aligned} \tag{7}$$

Algorithm 1 HYPER-TEMPORAL-RECOM ($H_{t_1}, ...H_{t_m}$, $W_{t_1}, ...W_{t_m}$, $\mathbf{J}_{user}$, $\mathbf{y}_{t_k}$)

Calculate $(\mathbf{D_v})_{\mathbf{t_k}}$ and $(\mathbf{D_e})_{\mathbf{t_k}}$.
$z = 0$, initialize $\mathbf{f_{t_k}}$ and $\mathbf{p}$ randomly.
repeat
$z = z + 1$
for $k \in \{1, 2, ..., m\}$ **do**

- Find $\mathbf{f_{t_k}}$ using the equation (5) indirectly using methods like jacobi iterations for inverse calculation using the iteration (6). The values for $\mathbf{f_{t_g}}$ for $g \neq k$ should be taken from the $(z-1)^{th}$ iteration.

end for

- Use gradient descent for $\mathbf{p}$ using the $(z-1)^{th}$ iteration values for any other variables.
- Calculate the current cost q_z using the equation (2).

until $(q_z - q_{z-1}) > \epsilon$
return $\mathbf{f_{t_m}}$
=0