A Critique of Network Distance Prediction using Matrix Factorization

Ashwin Verma(13166), Saksham Agarwal(13603), Varun Gupta(13771)
Mentor - Prof. Ketan Rajawat

Abstract—The paper provides a critical analysis for the work done in [6]. After introducing and formulating the problem, the algorithm used in the same is briefly discussed. We reproduce the simulation results and verify the arguments in the above paper. We also provide simulations for added scenarios which were left out by the authors in [6].

I. INTRODUCTION

Network distance prediction is crucial for many Internet applications like routing and file transfer, etc. Hence, prediction of unmeasured node distances pose an important problem to be solved. To solve this issue, the concept of Network Coordinate System (NCS) was introduced. GNP[1] is an NCS based algorithm which measures the node distances with reference to some predefined landmarks. Vivaldi[3], an another NCS based algorithm does not require the use landmarks and is therefore, fully decentralised. An inherent issue with NCS approach is that network distance space, unlike the metric space is not symmetric in nature and need not follow triangle inequality rule. So, the above mentioned algorithms fail to work in general networks. This led to the development of algorithms like IDES[5] which dropped the metric space assumption. It relied on the technique of distance prediction by factorizing the distance matrix into small matrices. However, IDES still required reference landmark nodes for its implementation making it prone to failure and overloading[5]. Hence to solve this problem in a fully decentralized manner using non metric space, DMF algorithm was proposed in [6]. To eliminate the need for landmarks, the node distances are iteratively updated using randomly selected nodes. The stability and competitiveness of DMF performance was measured experimentally.

II. DMF ALGORITHM

As described in [6], given a matrix $X$ of size $N \times N$, the idea is to approximately factorize this matrix as a product of two latent factor matrices $U$ and $V$ with dimension as $N \times L$ where $L \ll N$.

$$X \approx UV^T$$  \hspace{1cm} (1)

Each row of $U$ and $V$ represents a $L$-dimensional latent factor $u_n$ and $v_n$. In this way each element in the $m^{th}$ row and $n^{th}$ column of the distance matrix $X$ can be written as:

$$X_{mn} = u_n^T v_m$$  \hspace{1cm} (2)

Our aim is to achieve dimensionality reduction or a low-rank factorization of the matrix $X$. This kind of factorization can be used to predict some unknown internode distances between any two nodes. We would solve this problem by using optimization methods such as Gradient Descent to estimate the unknown parameters $\{u_n\}_{n=1}^N$ and $\{v_n\}_{n=1}^N$. Another method to solve this problem could be to use Stochastic Gradient Descent (SGD)[7]. The loss function for this problem can be defined as

$$L = \sum_{(n,m)} (X_{nm} - u_n^T v_m)^2$$  \hspace{1cm} (3)

The solution in accordance with the above loss function becomes computationally expensive as we have to consider all $N - 1$ neighbors for a given node (row). The problem was simplified by the authors of GNP [1] and IDES [5] who considered some nodes as landmarks nodes. This kind of approach creates voids in the distance matrix such that the distance between some of the nodes in the network is not known. The formulation of the cost function when some of the elements are missing is modelled as:

$$L = \|W \ast (X - UV^T)\|^2$$

$$= \sum_{(n,m)} w_{nm} \times (X_{nm} - u_n^T v_m)^2,$$ \hspace{1cm} (4)

where $w_{nm}$ is 1 if the distance between $m^{th}$ and $n^{th}$ node is known and 0 otherwise.

We introduce notation $\Omega = (n, m)$ such that $X_{nm}$ is known, $\Omega_{rn}$ denoting the column indices in the $n^{th}$ row of $X$ and $\Omega_{en}$ representing the row indices of the entries in the $m^{th}$ column of $X$ whose measurements are known. To get a stable solution, we introduce $L_2$ regularizers in the loss function.

$$L = \sum_{(n,m) \in \Omega} (X_{nm} - u_n^T v_m)^2$$

$$+ \frac{\lambda}{2} \sum_{n=1}^N \|u_n\|^2 + \frac{\lambda}{2} \sum_{m=1}^N \|v_m\|^2$$ \hspace{1cm} (5)

The above problem can be solved by alternately optimizing for $u_n$ and $v_n$. First step involves keeping $v_m$ fixed and optimizing on $u_n$ as a Ridge Regression problem whose solution is given as:

$$u_n = \left( \sum_{m \in \Omega_{rn}} v_m v_m^T + \lambda I \right)^{-1} \left( \sum_{m \in \Omega_{rn}} X_{nm} v_m \right)$$ \hspace{1cm} (6)

Similarly, the second step involves a similar process except that we have to keep $u_n$ as fixed and $v_m$ as variable.
\[ v_m = \left( \sum_{n \in \Omega_{cm}} u_n u_n^T + \lambda I_k \right)^{-1} \left( \sum_{n \in \Omega_{cm}} X_{nm} u_n \right) \] (7)

The author has randomly selected \( k \) neighbors from the network for a given node such that the distance from the neighboring nodes is exactly known. And the distance to non-neighboring nodes is taken to be unknown. These \( k \) random neighbors for \( n^{th} \) node (row) constitute \( \Omega_{cm} \). And, since we consider all the rows one by one, \( \Omega_{cm} \) represents all the \( N \) rows. The approach of the author in [6] is different from IDES in a sense that the author chooses the neighbors for each row randomly while IDES chooses a global landmarks considered to be neighbors for every node. The randomization in choosing the neighbor helps the algorithm to outperform the IDES algorithm.

III. SIMULATION AND REPRODUCTION OF RESULTS

The performance of the algorithm in [6] was checked using two datasets - P2psim [2] and Meridian [4]. We, however, verify and present our arguments with reference to only P2psim. Both the datasets contain only symmetric distances. The authors used the following criteria for performance evaluation:

\[ R_{\text{EE}} = \frac{|\hat{d}_{i,j} - d_{i,j}|}{d_{i,j}} \] (8)

\[ \text{stress} = \sqrt{\frac{\sum_{i,j}(d_{i,j} - \hat{d}_{i,j})^2}{\sum_{i,j}d_{i,j}^2}} \] (9)

\[ \text{MAEE} = \text{median}_{i,j}(|d_{i,j} - \hat{d}_{i,j}|) \] (10)

We verify the experimental results shown by the author. The author presented various performance curves obtained after varying a parameter and keeping others constant. We reproduce these results and critically analyze the arguments present in [6] which do not hold strong due to the lack of evidence or reasoning.

A. Dimension Size (\( l \))

The author suggests that the performance of DMF is independent to increasing the value of \( l \geq 3 \), and says the observation was confirmed by the Vivaldi algorithm. However, the argument here is flawed - there is no logical justification given for this result. And generalizing this result for any network seems counter intuitive, as we can have different optimal \( l \) values for different data sets. \( l \) value denotes the approximate rank of the matrix, hence a dataset containing most of the nodes far apart from each other will tend to form many sets of small clusters, thus possibly requiring a larger value of \( l \).

In Fig (1), (2) and (3), we reproduce the results shown in [6] showing the performance of DMF for various values of \( l \). In addition to the original values, we also check for the values \( l = 15 \) and \( 20 \).

B. Neighbor Size (\( k \))

To test the effect of number of neighbors selected at every iteration on the performance of DMF, the authors simulate the algorithm for different \( k \) values keeping the dimension size \( l = 10 \) and \( \lambda = 50 \). We have carried out a similar simulation while increasing the value of \( k \) from 15 to 250 in Fig (4), (5) and (6). From the results observed, we can corroborate the observation provided in the paper, that as \( k \) increases the performance of the algorithm improves.

C. Regularization Parameter(\( \lambda \))

As the author suggested in [6], we also observe high value of error metrics at large and small values of \( \lambda \) in Fig (7), (8) and (9) and \( \lambda = 50 \) seems to be a good choice.

D. Initialization value

As shown in [6], we also see that in Fig (10), (11) and (12) the DMF performance is independent of random initialization of \( x_i \) and \( y_i \).
E. Iteration value

In order to observe how the values change with iterations, a 2D matrix factorization is carried out in [6], 2D so that we can plot the points, and the coordinates are observed to get a feel for how the stabilization occurs. Apart from the coordinate plot for 2D factorization (Fig (13) - (16)), a histogram representing the distribution of the difference between the predicted distances at 20th and 100th iteration is provided (Fig (17)) which helps in understanding the distribution of change which takes place while iterating.

F. Asymmetric Data

An important contribution of [6] is the development of algorithm for non-metric distances. However, the authors do not provide evidence for their claim that the algorithm gives an acceptable performance for non-asymmetric data as well. We create asymmetric data of varying degrees of asymmetry and then test the algorithm for the artificially created data. To create the asymmetric data, we carry out the following operation:

\[ x_{ji} = (1 + 0.1 \times n \times d)x_{ij} \]  

(11)
where $d$ denotes the degree of asymmetry with $d = 0$ for the symmetric case, and $n \sim \mathcal{N}(0, 1)$.

We have carried out the test for $d = 1, 3$ and $5$.

From the results obtained, we observed that REE (Fig (18)) increases as the degree is increased. Similar results are obtained for MAE (Fig (20)) and stress (Fig (19)). The stress plot for different $d$ indicates that the magnitude of error keeps increasing as the asymmetry level increases. Therefore, we cannot make any inference about DMF working for asymmetric data, as the error keeps increasing. Hence, the authors claim regarding the DMF working for asymmetric data has a significant chance of being incorrect and the algorithm may face some challenges when the distances in the network are asymmetric in nature, which usually is the case.

IV. Future Work and Conclusion

In this paper, we reproduced and critically analyzed the algorithm provided by [6] for network distance prediction of distances not following the metric assumption. We have been successful in analyzing the shortcomings and pitfalls of the paper. We have also been able to work on the future challenges presented by the paper such as the performance of the proposed algorithm for asymmetric dataset. In this process of thorough deliberation, we found that Stochastic Gradient Descent for non-negative matrix factorization (NNMF) can further help in cutting down the computational cost. Also, k-means based longest distance heuristic can be further implemented to choose the neighbors wisely to achieve optimal accuracy with minimal computational cost. These are the areas for further explorations.
REFERENCES


