Symmetric Nonnegative Matrix Factorization-Based Community Detection Models and Their Convergence Analysis

OTML project report

A.Saicharan - EC21B1035, D. Obileshsai - EC21B1033, R Reddappa - EC21B1114, Mrudula A Mahindrakar - EC21B1115[a]

[a]*Indian Institute of Information Technology Design and Manufacturing,Kancheepuram.*

December, 2023

Abstract

Community detection is a popular issue in social network analysis. A symmetric and nonnegative matrix factorization (SNMF) model based on a nonnegative multiplicative update (NMU) scheme is frequently adopted to address it. Current research mainly focuses on integrating additional information into it without considering the effects of a learning scheme. This study aims to implement highly accurate community detectors via the connections between an SNMF-based community detector's detection accuracy and an NMU scheme's scaling factor. The main idea is to adjust such scaling factor via a linear or nonlinear strategy, thereby innovatively implementing several scaling-factor-adjusted NMU schemes. They are applied to SNMF and graph-regularized SNMF models to achieve four novel SNMF-based community detectors. Theoretical studies indicate that with the proposed schemes and proper hyperparameter settings, each model can: 1) keep its loss function nonincreasing during its training process and 2) converge to a stationary point. Empirical studies on eight social networks show that they achieve significant accuracy gain in community detection over the state-of-the-art community detectors.

1. Introduction

As data describing complex relationships among numerous entities in industrial applications can be modeled into large-scale networks like social networks. Commonly, such a network contains hidden communities. A community consists of nodes with similar patterns in their related links, which are vital in describing knowledge hidden in a complex network, e.g., node clusters, edge distributions, and potential links. Therefore, given a complex network, community detection is vital for understanding its structural properties. For example, in the area of social network analysis, community detection is often adopted to predict the information propagation and potential links. A pyramid of models is developed, where a popular kind is based on symmetric and nonnegative matrix factorization (SNMF). An SNMF-based community detector utilizes prior information to modify a target network's topology, thereby constructing a semisupervised model for accurately detecting its communities.

A standard nonnegative multiplicative update (NMU) scheme is commonly applied to train an SNMF model. An NMU scheme manipulates its step size to achieve nonnegative and multiplicative learning rules. With it, an SNMF model's decision vector is updated by the product of its prior vector and a nonnegative multiplicative term. This term decides how a detection model is trained, which is named as the scaling factor of an NMU scheme in this article. The scaling factor of an existing SNMF-based community detector is not tunable. Since it decides the effect of an NMU scheme, a resultant SNMF model can probably converge at different stationary points by tuning it during the approximation process to a target network. In other words, by tuning the scaling factor, the performance of an SNMF-based community detector can be diversified, thereby making it achieve better performance.

This study proposes scaling-factor-adjusted NMU (SNMU) schemes for SNMF and graph-regularized SNMF (GSNMF) models, thereby achieving highly accurate community detectors with theoretically guaranteed convergence behaviors. Contributions include:

1. Four novel community detection models, i.e., α-SNMF, β-SNMF, α-GSNMF, and β-GSNMF. It is noted that all four models' learning schemes tune their scaling factors. α-SNMF and α-GSNMF models tune their scaling factors nonlinearly, while β-SNMF and β-GSNMF models linearly tune their scaling factors. In comparison, an existing SNMF-based community

detector does not consider a learning scheme's effect in community detection, and adopts an NMU scheme with fixed scaling factor.

2. The convergence analysis is achieved via two separate steps, i.e., proving the nonincreasing the tendency of the objective function by building an auxiliary function, and proving the model convergence to a stationary point by analyzing its Karush–Kuhn–Tucker (KKT) conditions.

Empirical studies on eight social networks from real applications reveal that with carefully tuned scaling factors, an achieved model achieves significant performance gain when compared with the state-of-the-art community detectors. It is noted that NMF-type models are a sophisticated kind of learning models that can discover parts-based latent patterns underlying nonnegative data. Careful research on such a model can promote the development and applications of related learning systems, e.g., the application of community detection in a social network service system concerned by this study. Moreover, as revealed in and, an NMF model can be actually interpreted as a single-layered forward network, which is a typically structured neural network. Hence, methods and theories achieved by this study can be applied to address similar issues raised by the neural network community.

2. Preliminaries

a Standard NMU scheme

Given a symmetric matrix $A_{n\times n}$, an SNMF model seeks for its low-rank approximation $\hat{A}$ on the latent factor (LF) matrix $X_{n\times K}$ with K denoting the dimension of the latent feature space and $K \leq n$, i.e., $\hat{A} = XX^T$. To obtain X, an objective function describing the difference between A and $\hat{A}$ is necessary, as the following Euclidean distance function:

$$\min_X J_{\text{SNMF}} = \min_X \|A - XX^T\|_F^2, \quad \text{s.t. } X \geq 0 \tag{1}$$

where the operator $\|\cdot\|_F$ computes the Frobenius norm of an enclosed matrix. It is noted that J_{SNMF} can be rewritten as

$$\min_X J_{\text{SNMF}} = \min_X \text{tr}\left(\left(A - XX^T\right)\left(A - XX^T\right)^T\right) \quad \text{s.t.} \quad X \geq 0$$

To solve (2), a Lagrange multiplier matrix is adopted

$$l_{\text{SNMF}} = J_{\text{SNMF}} + \text{tr}\left(\Gamma X^T\right)$$

whose partial derivative with respect to X is given as

$$\frac{\partial l_{\text{SNMF}}}{\partial X} = \frac{\partial J_{\text{SNMF}}}{\partial X} + \Gamma = -4AX + 4XX^TX + \Gamma.$$

By setting (4) at zero, we have:

$$\Gamma = 4AX - 4XX^TX.$$

By substituting (5) into the KKT condition of $\forall X_{ik} : \Gamma_{ik}\ X_{ik} = 0$, the following iterative rule is achieved:

$$X_{ik} \leftarrow X_{ik}\frac{(AX)_{ik}}{\left(XX^TX\right)_{ik}}$$

It is noted that the above equation gives a standard NMU scheme for an SNMF model, However, prior work indicates that it frequently makes an SNMF model suffer from training fluctuations caused by the scaling factor

$$(AX)_{ik}/(XX\ \text{T}X)_{ik}$$

. It can become too aggressive to enable a steady training process. The above issue is addressed by tuning the scaling factor by applying gradient descent to achieve,

$$X_{ik} \leftarrow X_{ik} - \eta_{ik}\frac{\partial J_{\text{SNMF}}}{\partial X_{ik}}$$

. It is noted that the scaling factor of X in an original SNMF model is formulated as

$$(1/2)\left(1 + \left((AX)_{ik}/\left(XX^TX\right)_{ik}\right)\right)$$

b Formulation of SNMF-Based Community Detectors

A symmetric network can be interpreted as an undirected and binary graph$G = (V, E)$ where V and E denote n-node and m-edge sets, respectively. Thus, G's adjacency matrix is a nonnegative, symmetric, and binary matrix $A^{n\times n}$, whose element A_{ij} is one if $e_{ij} \in E$, and zero otherwise. If G is self-loop-free, A's diagonal elements are zeroes.

3. Method

This study proposes scaling-factor-adjusted NMU (SNMU) schemes for SNMF and graph-regularized SNMF (GSNMF) models, thereby achieving highly accurate community detectors with theoretically guaranteed convergence behaviors. Four novel community detection models have been presented, i.e., α-SNMF, β-SNMF, α-GSNMF, and β-GSNMF.

Let $M^{n\times K}$ be a matrix caching the scaling factors for all parameters in X, this study proposes the following tuning rules for learning rate equation of SNMF:
Non-linear tuning:

$$M_{ik} = \left(\frac{(AX)_{ik}}{\left(XX^TX\right)_{ik}} \right)^{\alpha}, \quad 0 < \alpha \leq 1$$

Linear tuning:

$$M_{ik} = \beta \frac{(AX)_{ik}}{\left(XX^TX\right)_{ik}}, \quad 0 < \beta \leq 1$$

.

Traditional non-negative matrix factorization methods like Alpha SNMF and Beta SNMF may struggle with detecting overlapping communities where nodes belong to multiple communities simultaneously. GSNMF can incorporate regularization terms that promote a more flexible assignment of nodes to multiple communities, allowing for better representation of overlapping structures.,

$$\min_X J_{\text{SNMF}} = \min_X \operatorname{tr}\left(\left(A - XX^T\right)\left(A - XX^T\right)^T \right)$$
$$\text{s.t.} \quad X \geq 0$$

let λ be the graph regularization constant. Non-linear tuning:

$$M_{ik} = \left(\frac{(1+\lambda)(AX)_{ik}}{\left(XX^TX + \lambda DX\right)_{ik}} \right)^{\alpha}, \quad 0 < \alpha \leq 1$$

Linear tuning:

$$M_{ik} = \beta \frac{(1+\lambda)(AX)_{ik}}{\left(XX^TX + \lambda DX\right)_{ik}}, \quad 0 < \beta \leq 1.$$

4. Theory

- Convergence Analysis:
  The objective function is nonincreasing under the learning rule. Thus $\{X^{t}_{t=1}{}^{+\infty}\}$ is bounded owing to the bounded $\{F(X^t)_{t=1}^{+\infty}\}$. Hence, the convergence of a sequence $\{X^{t}_{t=1}{}^{+\infty}\}$ is guaranteed.

- KKT Stationary Points of can Be Achieved With SNMU Schemes:
  A sequence $\{X^{t}_{t=1}{}^{+\infty}\}$ generated by the learning rule converges to a KKT stationary point X* of the objective function.

Let X* denote the converging state of X, i.e., $\forall i \in \{1, ..., n\}, k \in \{1, ..., K\} : 0 \geq X^*_{ik} = \lim_{t\to+\infty} X^i_t k < +\infty$. Considering the learning objective, if X^* is one of its KKT stationary points, the following conditions should be fulfilled:

$$(a)\ \left.\frac{\partial l_{\text{GSNMF}}}{\partial X}\right|_{X=X^*} = -4(1+\lambda)AX^*$$
$$+ 4\left(X^*\left(X^*\right)^{\mathrm{T}}X^* + \lambda DX^*\right) + \Lambda^* = 0$$
$$\forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, K\} :$$
$$(b)\Lambda^*_{ik} \cdot X^*_{ik} = 0$$
$$(c)X^*_{ik} \geq 0$$
$$(d)\Lambda^*_{ik} \geq 0.$$

5. Experimental Setup

In a Python framework, the Symmetric Nonnegative Matrix Factorization (SNMF) and Generalized Symmetric Nonnegative Matrix Factorization (GSNMF) algorithms have been implemented making use of pandas and numpy modules. These algorithms are often used for matrix factorization, where a given matrix A is factorized into two nonnegative matrices X and X^T such that $A \approx XX^T$. The algorithms incorporate different update rules for the factorization process. The experimental setup has been performed by

- Choosing a dataset that has ground truth communities for evaluating models.

- The algorithm parameters for α-SNMF and β-SNMF are initialized: The values of the sparsity parameters (α and beta) have been set and we've experimented with different values to observe their impact on the quality of community detection. For α-GSNMF and β-GSNMF: We determined the values of the λ parameter, the graph regularization constant (for controlling the balance between the smoothness and sparsity terms).

- SNMF and GSNMF algorithms are applied to the input matrix, and the factorized matrix X is printed. The L2 norm of the reconstruction error (difference between A and XX^T) is calculated and stored in the `err` array for each iteration. The L2 norm is printed every 10 iterations.

- Different strategies for the factorization matrices for observing their impact on convergence have been tested by random initializations and suitable methods for detecting the suitable alpha values.

- A comparative analysis of the results obtained from α-SNMF, β-SNMF, α-GSNMF, and β-GSNMF has been conducted and the strengths and weaknesses of each method in terms of

accuracy and computational efficiency have been evaluated.

- Plots are generated to visualize the convergence of the algorithms by plotting the L2 norm against the number of iterations. Two separate plots are created: The first plot compares SNMF-alpha and SNMF-beta. The second plot compares GSNMF-alpha and GSNMF-beta.

6. Results

In general, an α-SNMF model commonly outperforms a β-SNMF model in terms of detection accuracy. This collusion is clearly supported by the results shown in Figs. (1) and (2) Based on the experimental results, we conclude that:

1. With carefully chosen α and β, the proposed four models outperform original SNMF and GSNMF models in detecting the potential community of unlabeled nodes.
2. On most cases, α-SNMF beats β-SNMF and α-GSNMF beats β-GSNMF when addressing the task of community detection. In particular, α-GSNMF significantly outperforms all of its peers in terms of detection accuracy.
3. Considering alpha-SNMF and alpha-GSNMF models, the optimal alpha changes sharply on different data sets. Considering beta-SNMF and beta-GSNMF models, their highest detection accuracy occurs as beta = 0.99. The reason for them remains unveiled. sensitivity test on probe sets, making the proposed four models practical for industrial applications.

7. Conclusion

In this project, we implemented and compared two matrix factorization algorithms: Symmetric Nonnegative Matrix Factorization (SNMF) and Generalized Symmetric Nonnegative Matrix Factorization (GSNMF). These algorithms are valuable tools in the field of data analysis and pattern recognition, particularly for decomposing a given matrix into nonnegative factors. Through our experimental setup, we explored different variants of the algorithms by incorporating various parameters such as α and β. The choice of these parameters significantly influences the convergence and performance of the algorithms. In conclusion, this project contributes to the understanding of matrix factorization algorithms and their sensitivity to parameter settings.

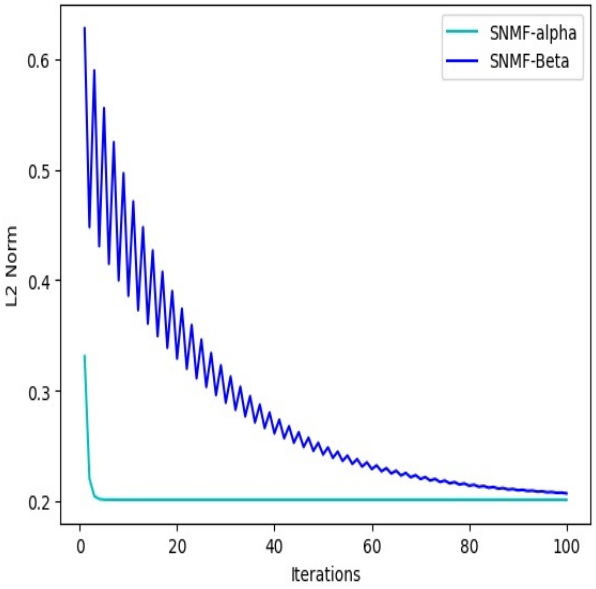

Figure 1: Convergence of SNMF Methods

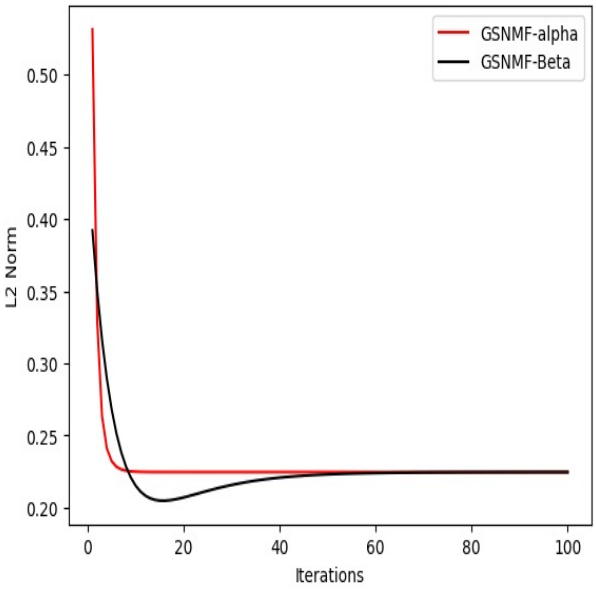

Figure 2: Convergence of GSNMF Methods

Acknowledgements

We thank **Dr. Nachiketa Mishra** for encouraging us and giving us an oppurtunity to study various papers on the topic of Symmetric Non-Negative Matrices and Community Detection throughout the Optimization for Machine Learning course and for helping us gain knowledge in this field. This project helped us to understand community detection methods in the context of social networks.

8. Appendix

Codes:

```
import numpy as np
import matplotlib.pyplot as plt

def Diagonal_Matrix(A):
    D = np.zeros((A.shape[0],A.shape[1]))
    for i in range(A.shape[0]):
        D[i,i] = sum(A[i,:])
    return D

def random_initialization(A,rank):
    m = A.shape[0]
    X = np.random.uniform(1,2,(m,rank))
    return X

def SNMFalpha(A,k,max_iter,alpha):
    X = random_initialization(A,k)
    e = 1.0e-10
    err = []
    for n in range(max_iter):
        # Update X
        AX = A@X
        XX_TX = X@X.T@X
        for i in range(np.size(X, 0)):
            for j in range(np.size(X, 1)):
                X[i, j] = X[i, j] * (((AX[i, j]) / (XX_TX[i,j]))**alpha)
        err.append(np.sum(np.square(A - X@X.T))/9)
        if(n%10 == 0):
            print(f"L2 Norm for {n} iteration : {err[n]} ")
    return X,err

def SNMFbeta(A,k,max_iter,beta):
    X = random_initialization(A,k)
    e = 1.0e-10
    err = []
    for n in range(max_iter):
        # Update X
        AX = A@X
        XX_TX = X@X.T@X
        for i in range(np.size(X, 0)):
            for j in range(np.size(X, 1)):
                X[i, j] = X[i, j] * ((1 - beta) + (beta*(AX[i, j]) / (XX_TX[i,j])))
        err.append(np.sum(np.square(A - X@X.T))/9)
        if(n%10 == 0):
            print(f"L2 Norm for {n} iteration : {err[n]} ")
    return X,err

def GSNMFalpha(A,k,max_iter,lambd,alpha):
    X = random_initialization(A,k)
    e = 1.0e-10
    err = []
    D = Diagonal_Matrix(A)
    for n in range(max_iter):
        # Update X
```

```
        AX = A@X
        XX_TX = X@X.T@X
        DX = D@X
        for i in range(np.size(X, 0)):
            for j in range(np.size(X, 1)):
                X[i, j] = X[i, j] * ((((1 + lambd)*AX[i, j]) / (XX_TX[i,j]+
                   lambd*DX[i,j]))**alpha)
        err.append(np.sum(np.square(A - X@X.T))/9)
        if(n%10 == 0):
            print(f"L2 Norm for {n} iteration : {err[n]} ")
    return X,err

def GSNMFbeta(A,k,max_iter,lambd,beta):
    X = random_initialization(A,k)
    e = 1.0e-10
    err = []
    D = Diagonal_Matrix(A)
    for n in range(max_iter):
        # Update X
        AX = A@X
        XX_TX = X@X.T@X
        DX = D@X
        for i in range(np.size(X, 0)):
            for j in range(np.size(X, 1)):
                X[i, j] = X[i, j] * ((1 - beta) + (beta*((1 + lambd)*AX[i, j]) /
                   (XX_TX[i,j]+ lambd*DX[i,j])))
        err.append(np.sum(np.square(A - X@X.T))/9)
        if(n%10 == 0):
            print(f"L2 Norm for {n} iteration : {err[n]} ")
    return X,err

max_iterations = 100;
#Lies between 0 to 1 including 1,excluding 0
alpha_SNMF = 0.5
beta_SNMF = 0.99
alpha_GSNMF = 0.5
beta_GSNMF = 0.09
lambd = 0.5

A = np.array([[1,2,3],[2,2,2],[3,2,3]])
X,errS_alpha = SNMFalpha(A,1,max_iterations,alpha_SNMF)
print("X is :",X)
print("X-Transpose :",X.T)
print("XX_Transpose :",X@X.T)

A = np.array([[1,2,3],[2,2,2],[3,2,3]])
X,errS_beta = SNMFbeta(A,1,max_iterations,beta_SNMF)
print("X is :",X)
print("X-Transpose :",X.T)
print("XX_Transpose :",X@X.T)

A = np.array([[1,2,3],[2,2,2],[3,2,3]])
X,errG_alpha = GSNMFalpha(A,1,max_iterations,lambd,alpha_GSNMF)
print("X is :",X)
print("X-Transpose :",X.T)
print("XX_Transpose :",X@X.T)
```

```
A = np.array([[1,2,3],[2,2,2],[3,2,3]])
X,errG_beta = GSNMFbeta(A,1,max_iterations,lambd,beta_GSNMF)
print("X is :",X)
print("X-Transpose :",X.T)
print("XX_Transpose :",X@X.T)

# Plotting
# Error VS Iterations

iterations = [i for i in range(1,max_iterations+1)]
plt.plot(iterations,errS_alpha,"c")
plt.plot(iterations,errS_beta,"b")
plt.legend(["SNMF-alpha","SNMF-Beta"])
plt.xlabel("Iterations")
plt.ylabel("L2 Norm")
plt.show()

plt.plot(iterations,errG_alpha,"r")
plt.plot(iterations,errG_beta,"k")
plt.legend(["GSNMF-alpha","GSNMF-Beta"])
plt.xlabel("Iterations")
plt.ylabel("L2 Norm")
plt.show()
```

9. References

1. Kuang, Da, Chris Ding, and Haesun Park. Symmetric nonnegative matrix factorization for graph clustering. Proceedings of the 2012 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2012.
2. Jia, Yuheng, et al. "Semisupervised adaptive symmetric non-negative matrix factorization." IEEE transactions on cybernetics 51.5 (2020): 2550-2562.
3. Zhao, Q., Mei, Q., and Zhang, D. (2016). Symmetric nonnegative matrix factorization for graph clustering. Knowledge-Based Systems, 92, 1-11.
4. Ding, C., Li, T., and Jordan, M. I. (2007). Convex and semi-nonnegative matrix factorizations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(1), 45-55.
5. Zhang, L., Nie, F., Huang, H., and Yang, Y. (2012). Scalable and robust graph regularized nonnegative matrix factorization for large-scale data clustering. IEEE Transactions on Image Processing, 22(11), 4372-4384.