DEIM 2020 D5-2(day2 p32)

# 属性付きグラフクラスタリングのための グラフ畳み込み非負値行列因子分解

小川　裕也† 　佐々木勇和† 　鬼塚　真†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{ogawa.yuya,sasaki,onizuka}@ist.osaka-u.ac.jp

**あらまし**　属性付きグラフクラスタリングは様々な応用がされる重要なタスクの一つである. 本稿では新しいグラフクラスタリングフレームワーク, Graph Convolutional Non-negative Matrix Factorization (GCNMF) を提案する. GCNMF は以下の 3 つの新規性を持つ. 1 つ目として, 直接クラスタ構造を捉えるための NMF とグラフ構造と属性を統合するためのグラフ畳み込みを組み合わせる. これにより属性付きグラフクラスタリングのための直接アプローチが可能となる. 2 つ目として, 様々な属性付きグラフに対応するため柔軟なグラフ畳み込みフィルタを使用する. フィルタはグラフ構造と属性の影響のバランスを制御する. 3 つ目として, GCNMF は教師なし学習の条件に従い自動でパラメータを探索する. したがって, パラメータをチューニングする必要がない. 4 つの標準的なデータセットにおける実験で, GCNMF が既存手法に対してクラスタリング精度が優れていることを示した.

**キーワード**　グラフクラスタリング, NMF, グラフ畳み込み

## 1 はじめに

ネットワークやグラフは実世界に遍在しており, 様々な関係を表す. 例として, ソーシャルネットワーク, 引用ネットワーク, タンパク質相互作用ネットワークなどがある. グラフはノードの集合であり, それらの関係はエッジで表される. グラフにおけるノードは通常属性を持ち, そのようなグラフは属性付きグラフと呼ばれる. ノード分類, グラフクラスタリング, リンク予測などのグラフ解析は長く研究が行われている. 特にグラフクラスタリングは密に関わりのある類似したノード同士を同じ集合に分割する基礎的なタスクである.

近年様々なグラフ解析に取り組むために, 多くのグラフ埋め込み法が提案されている. それらの鍵となる考えは, ノード間の隣接性を保持した低次元ベクトルを生成することである. そして, 属性付きグラフにおける課題の一つが, どのようにグラフの構造と属性を組み合わせるかである. 実際にグラフの構造と属性を組み合わせた属性付きグラフ埋め込み手法が多く提案されている. 例として, ランダムウォークベースの手法 [1], スペクトルクラスタリングベースの手法 [2], 非負値行列因子分解ベースの手法 [3] がある. 特に, グラフ畳み込みネットワークは様々なグラフ解析タスクで優れた性能を達成している [4]. 例として, graph autoencoder [5], marginalized graph autoencoder [6] 及び adversarially regularized graph autoencoder [7] などがある. グラフ畳み込みネットワークの発展が示しているように, グラフ畳み込みはグラフの構造と属性の組み合わせに効果的な解決法である. しかしながら, これらの手法はクラスタリングタスクにおいて 3 つの問題がある. 一つ目は, それらの手法のほとんどが直接クラスタを得るように設計されていないことである. すなわち, それらの手法は始めに, ノードのベクトルを学習して, その後 $k$-means などのクラスタリング手法を適用するという二段階のアプローチを要する. したがって, それらの手法はクラスタリングのために最適化されていない. 二つ目に, 実世界の属性付きグラフの多様性に対してクラスタリング手法は適応的に変化する必要があるということである. しかし, 既存のグラフ畳み込みフィルタは固定されており, グラフ構造と属性情報の影響のバランスをとることができない. 最後に, クラスタリング精度を向上させるためにパラメータを最適化する必要性があることである. 実際に, 多くの論文において [6, 7] パラメータ最適化をした際のクラスタリング結果を報告している. しかし, 実際はクラスタリングタスクにおいて訓練データを使用することができないため, クラスタリングを実行する前にパラメータを最適化する機会は存在しない.

これらの問題を解決するため, われわれは新しい属性付きグラフクラスタリングのフレームワーク Graph Convolutional Non-negative Matrix Factorization (GCNMF) を提案する. GCNMF の特徴は以下の通りである.

- GCNMF はグラフ畳み込みと NMF を統合することで属性付きグラフクラスタリングに対して直接アプローチを行う. グラフ畳み込みによってグラフの構造と属性を組み合わせて, NMF によって直接クラスタ構造を捉える.
- 既存の固定されたフィルタと異なり, われわれは柔軟なグラフ畳み込みフィルタを用いる. フィルタによって, 実世界の属性付きグラフの多様性に対処するように, 属性に対するグラフ構造の影響を適応的に制御する.
- 教師なし学習の条件に従って, GCNMF は情報エントロピーに基づいたクラスタの分離性を評価することで適切なパラメータを探索する. われわれの知る限り, GCNMF はグラフクラスタリングタスクにおいて自動パラメータ探索を行う初めてのフレームワークである.

われわれはグラフクラスタリングタスクにおいて GCNMF と 11 の既存のグラフクラスタリング手法を比較を行った. 4 つの標準的なデータセットにおいて, GCNMF が既存手法に対して優れていることを示した.

## 2 関 連 研 究

グラフ解析は古くから広く研究がされている分野である. 特に, グラフクラスタリングタスクは人気のあるトピックである. 古典的なグラフクラスタリング手法は modularity や cut, permanence などの特定の指標を最小化もしくは最大化することでクラスタを発見する [8–12]. これらの手法はグラフの構造情報のみを利用する.

近年では, 様々なグラフ解析に取り組むためのグラフ埋め込み手法が注目を集めている. グラフ埋め込み手法はグラフのノードに対して低次元表現ベクトルを見つける. DeepWalk [13] と node2vec [14] はランダムウォークを用いてノード間の近接性を保持するように表現ベクトルを得る. SDNE [15] は制限付き自己符号化器を用いて, 隣接行列を再構築することで近接性を保持する. M-NMF [16] はクラスタ構造を保持するため modularity の考えを NMF に取り込んでいる. これらの手法も古典的なグラフクラスタリング手法同様にグラフの構造情報のみを利用する. 一方で属性の情報をグラフ埋め込みに取り入れた手法も存在する. TADW [1] は DeepWalk を行列分解として扱い, 属性の情報を統合している. CDE [3] はコミュニティ構造埋め込み行列と属性行列のため損失関数をつなぎあわせている. NMF はその高い解釈性からクラスタリングに適用されており, 上記以外にもいくつかの NMF を用いたグラフクラスタリング手法が提案されている [17,18]. グラフ畳み込みはグラフ構造と属性を統合するのに効果的であるため [19], グラフ畳み込みを利用した手法が多く存在する. Graph autoencoder (GAE) と variational graph autoencoder (VGAE) [5] は 2 層のグラフ畳み込みネットワークを持ち, 自己符号化器で隣接行列に復元する. Marginalized graph autoencoder [6] は 3 層のグラフ畳み込みネットワークを持ち, 属性行列を marginalized denoising autoencoder によって復元する. Adversarially regularized graph autoencoder (ARGE) と adversarially regularized variational graph autoencoder (ARVGE) [7] は表現ベクトルが事前分布に従うように制限を課している.

## 3 提案フレームワーク

われわれは以下の既存手法の障害と実世界の属性付きグラフにおける事実を基に提案フレームワークを設計する.

a) 既存手法の障害

既存手法は 2 段階のアプローチを取る. すなわち, はじめにノードの表現ベクトルを学習して, その後 $k$-means のようなクラスタリング手法を適用する. このアプローチはクラスタの形が球状でない場合上手く働かない [20]. 別のクラスタリング手法を適用することは可能であるが, 得られたノードの表現ベクトルがクラスタリング手法に適しているかを保証することはできない.

b) 属性付きグラフの多様性

実世界の属性付きグラフは多様性を持つ. すべてのグラフはそれぞれ異なった特徴を持ち, クラスタ構造に影響を与える. われわれの実験において, 上手くクラスタ構造を捉えるためにグラフ構造と属性の効果のバランスを取り, そしてノードの属性の分布を考慮する必要があることを示している（詳細は4.3章を参照されたい）.

したがって, われわれのフレームワークは 1) 直接クラスタ構造を捉えて, 2) 適応的にグラフ構造と属性の効果のバランスを取り, そして適応的にノードの属性の事前分布を選択するべきである. しかし, ここで疑問が生じる. クラスタリングという訓練データのない状況において, どうすれば適応的にフレームワークを制御することができるだろうか.

われわれは以下のようにフレームワークを設計する. 直接的なアプローチとするため, フレームワークの根幹に NMF を使用する. NMF を用いることで直接クラスタ構造を捉えることが可能となる [17,18]. 適応的に制御するために, エントロピー最小化を導入して, それを NMF による行列分解に適用する. エントロピー最小化を行うことでフレームワークが自動でクラスタリングに適したパラメータ探索する. エントロピーを用いる背景には, 分解後の行列のエントロピーが低いときクラスタの分離度が高いと見ることができるという考えがある. 加えて, われわれはグラフ構造と属性の影響のバランスをとる柔軟なグラフ畳み込みフィルタを導入する.

この章では, 属性付きグラフのクラスタリングを定義して（3.1章）, 自動パラメータ探索の新しい考えを示し（3.2章）, そして提案フレームワークの詳細（3.3章）について述べる.

### 3.1 問 題 定 義

属性付きグラフ $G$ は, $V$ と $E$, そして $X$ から成る. このとき, $V = \{v_1, v_2, ..., v_n\}$ はノード集合, $E = \{e_{ij}\}$ はノード $i$ と $j$ 間のエッジ集合, そして $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times M}$ は属性行列である. 属性行列において, 各行ベクトルはノードと結びついており, $m$ は属性数を表す. 本稿において, 簡単化のため無向グラフのみを扱う[1]. われわれは互いに共通のノードを保有しない $K$ クラスタ $C = \{C_1, \cdots, C_K\}$ を発見することを目的とする.

### 3.2 自動パラメータ探索

われわれのフレームワークは NMF で得られた分解後の行列のエントロピーを最小化することで適したパラメータを探索する. 自動パラメータ探索のための 2 つの技術を紹介する.

a) NMF

$M$ 次元のデータベクトルが与えられたとき（属性に対応する）, そのベクトルは非負値行列 $Y \in \mathbb{R}^{N \times M}$ として表すことが可能である. このとき, $N$ はデータ数である（グラフではノード数に対応する）. NMF は行列 $Y$ を 2 つの潜在行列 $W \in \mathbb{R}^{N \times K}$ と $H \in \mathbb{R}^{K \times M}$ に分解する技術であり, 通常 $K$ は $N$ や $M$ よりも小さくなるように選ばれる.

1 : われわれのフレームワークは簡単に有向グラフにおいても使用できるように拡張可能である.

$$Y \approx WH \tag{1}$$

$W$ の列ベクトルは基底ベクトルを表し, $H$ の列ベクトルは基底ベクトル間の結びつきを表す [21]. グラフクラスタリングの文脈では基底ベクトルはクラスタを表し, 行ベクトルは各ノードのそれぞれのクラスタへの寄与度を表す.

b) エントロピー最小化

基底行列 $W$ のエントロピーを以下のようにクラスタ寄与ベクトルの平均エントロピーとして定義する.

$$\mathrm{Ent}(W) = \frac{1}{N}\sum_{i}^{N}\mathrm{ent}(\boldsymbol{w}_i) \tag{2}$$

$$= \frac{1}{NK}\sum_{i}^{N}\sum_{j}^{K} -\bar{W}_{ij}\log\bar{W}_{ij} \tag{3}$$

ここで $\bar{W} = D^{-1}W$ は行正規化基底行列, $D_{ii} = \sum_j W_{ij}$ は正規化のための対角行列である. $\mathrm{ent}(\boldsymbol{w}_i)$ は $\boldsymbol{w}_i$ の値の分布が偏っているとき, 低くなる. $\mathrm{Ent}(W)$ もまた $\boldsymbol{w}_i$ $(1 \leqq i \leqq N)$ の値の分布が平均的に偏っているとき, 低くなる. このとき, 基底ベクトル（列ベクトル）は互いに独立する傾向にある. 一度述べたように基底ベクトルはクラスタを表すため, エントロピーが低いことはクラスタの分離度が高いことを示す.

以上の考えに基づいて, われわれは提案フレームワークが $W$ のエントロピーを低くすることで教師なし学習の条件においてクラスタリングのための適切なパラメータを自動で探索するように設計する. ここで, パラメータは 1) グラフ構造と属性の影響のバランシングパラメータ（$\beta$）と 2) NMF の最適化アルゴリズムの選択の 2 つである. それぞれの詳細は以下の章で述べる.

### 3.3 Graph Convolutional Non-negative matrix Factorization

提案フレームワークである GCNMF は graph convolution step, NMF step, evaluation step から成る 3 つのステップで構成される. 以下に 3 つのステップをそれぞれ要約する.

- graph convolution step では, GCNMF が様々な属性付きグラフに対応するために適応的にグラフ構造と属性の影響を制御する柔軟なグラフ畳み込みフィルタを導入する. このステップでは隣接行列と属性行列を入力として, フィルタリングされた特徴行列を得る.
- NMF step では, フィルタリングされた特徴行列に対して NMF を適用することでクラスタ寄与ベクトルを得る. われわれは GCNMF が様々属性付きグラフに対して頑強になるように NMF に対して 2 つの最適化アルゴリズムを用いる.
- evaluation step では, クラスタ寄与ベクトルの平均エントロピーを用いて, クラスタ配属を評価する.

#### Graph convolution step

われわれはグラフ構造と属性の情報の両方を利用するため, グラフ畳み込みを用いてそれらを組み合わせる. 本稿では以下に示す柔軟なグラフ畳み込みフィルタを導入する.

$$\tilde{D}^{-1}\tilde{A} \tag{4}$$

ここで $\tilde{A} = A + \beta I$ $(\beta \geqq 1)$ は自己ループを足した隣接行列である. $\beta$ は自身の属性に対する重みパラメータである. これによって, 隣接するノードの属性に対する自身の属性の影響を制御する[2]. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij} = diag(\tilde{d}_1, \cdots, \tilde{d}_N)$ は自己ループを足した対角次数行列である. $\tilde{A}$ を正規化するために用いる. われわれは属性行列 $X$ に対するグラフ畳み込みを以下のように記す.

$$Z = \tilde{D}^{-1}\tilde{A}X. \tag{5}$$

$Z = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N]^T \in \mathbb{R}^{N\times M}$ はフィルタリングされた特徴行列と呼ぶこととする. ノード $i$ に結びついているフィルタリングされた特徴ベクトル $\boldsymbol{z}_i$ は以下のように表すことが可能である.

$$\boldsymbol{z}_i = \frac{\beta\boldsymbol{x}_i}{\tilde{d}_i} + \sum_{j\in N(i)}\frac{\boldsymbol{x}_j}{\tilde{d}_i}, \tag{6}$$

$N(i)$ はノード $i$ の隣接ノード集合である. $\beta$ を大きくすることで, 隣接ノードの属性よりも自身の属性に重みを置くことが可能である. 既存手法に従い, グラフ畳み込みフィルタを 2 回適用する [5,7,19].

#### NMF step

このステップでは NMF を適用することでフィルタリングされた特徴行列 $Z$ を分解する. より良い近似を得るために NMF における潜在行列 $W$ と $H$ の初期値は重要である. われわれは $W$ と $H$ の初期化ために Non-Negative Double Singular Value Decomposition (NNDSVD) [22] を用いる. 詳細は [22] を参照されたい.

初期化の後, クラスタリング精度を向上させることを目的として, 近似的な行列分解を見つけるため 2 つの標準的な乗法アルゴリズムを用いる. アルゴリズムの選択は入力行列の分布に依存する. また, その選択は次の evaluation step において決定する. 1 つ目のアルゴリズムはフロベニウスノルム（FN）を用いる. このアルゴリズムは入力行列がガウス分布に従うことを仮定する. FN を用いた損失関数 $L_{FN}$ は以下のように定義される.

$$L_{FN} = \min\sum_{ij}(Z_{ij} - (WH)_{ij})^2. \tag{7}$$

最適化のため以下のように潜在行列を更新する.

$$H_{ij} \leftarrow H_{ij}\frac{(W^TZ)_{ij}}{(W^TWH)_{ij}}, \tag{8}$$

$$W_{ij} \leftarrow W_{ij}\frac{(ZH^T)_{ij}}{(WHH^T)_{ij}}. \tag{9}$$

2 つ目のアルゴリズムではカルバック・ライブラー情報量（KL）を使用する. このアルゴリズムは入力行列がポワソン分布に従っていることを仮定している. KL を使用した損失関数 $L_{KL}$ は以下のように定義される.

$$L_{KL} = \min\sum_{ij}(Z_{ij}\log\frac{Z_{ij}}{(WH)_{ij}} - (Z_{ij} - (WH)_{ij})). \tag{10}$$

最適化のため以下のように潜在行列を更新する.

---

2 : $\beta$ はそれぞれのノードに対して変化させることが望ましいが, 簡単のため全てのノードに対して同じ $\beta$ を与える.

**Algorithm 1** GCNMF algorithm

**Input**: adjacency matrix $A$, feature matrix $X$

**Output**: Clusters $C$

1: set $\beta$ = 1 and $\mathrm{ent}^0 = +\infty$
2: **while do**
3: **# Graph convolution step**
4: $Z = \tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}\tilde{A}X$
5: **# NMF step**
6: initialize $W$ and $H$ by NNDSVD with $Z$
7: replace 0 elements in $W$ and $H$ with random values
8: $W^{\beta}_{FN}, H^{\beta}_{FN} \leftarrow$ NMF($Z, W, H$) with Eq. (8) and (9)
9: $W^{\beta}_{KL}, H^{\beta}_{KL} \leftarrow$ NMF($Z, W, H$) with Eq. (11) and (12)
10: **# Evaluation step**
11: **if** $\mathrm{ent}(W^{\beta}_{FN}) < \mathrm{ent}(W^{\beta}_{KL})$ **then**
12: $\mathrm{ent}^{\beta} = \mathrm{ent}(W^{\beta}_{FN}), W^{\beta} = W^{\beta}_{FN}$
13: **else**
14: $\mathrm{ent}^{\beta} = \mathrm{ent}(W^{\beta}_{KL}), W^{\beta} = W^{\beta}_{KL}$
15: **end if**
16: **if** $\mathrm{ent}^{\beta-1} < \mathrm{ent}^{\beta}$ **then**
17: break
18: **end if**
19: $\beta = \beta + 1$
20: **end while**
21: **return** $C \leftarrow argmax(W^{\beta-1})$

$$H_{ij} \leftarrow H_{ij}\frac{\sum_a W_{ai}Z_{aj}/(WH)_{aj}}{\sum_k W_{ki}} \tag{11}$$

$$W_{ij} \leftarrow W_{ij}\frac{\sum_a H_{aj}Z_{ai}/(WH)_{ia}}{\sum_k H_{jk}} \tag{12}$$

潜在行列を最適化した後, $W$ の各行ベクトルはノードごとのクラスタ寄与情報を保有する. 収束性の証明は [21] を参照されたい.

**Evaluation step**

このステップでは適応的にパラメータ（自身の属性への重み $\beta$ と NMF における損失関数 $L_{FN}$ もしくは $L_{KL}$）を選択するためエントロピー $Ent(W)$ を評価する. 簡単のため, われわれはエントロピーのはじめの極小点となる $\beta$ と損失関数を選ぶ. $\beta$ は整数の範囲で 1 から増加させて探索する.

提案フレームワーク GCNMF のアルゴリズムは algorithm 1に要約される. graph convolution step（4 行目）では, 柔軟なグラフ畳み込みフィルタを用いてグラフ構造と属性を統合する. NMF step では（6〜9 行目）, フィルタリングされた特徴行列に NMF を適用することで各ノードのクラスタ寄与ベクトルを得る. evaluation step（11〜19 行目）では, エントロピーによってクラスタ分離度を評価する. 最終的にクラスタ寄与ベクトルの最大値を持つ列のインデックスを選択することでクラスタ割り当て $C$ を得る（21 行目）.

# 4 評価実験

## 4.1 実験設定

### データセット

われわれは表 1に示すような 4 つの標準的な属性付きグラフを使用する. Cora と Citeseer, そして Pubmed は引用ネットワークであり, ノードとエッジはそれぞれ論文と引用関係を表す. Wiki はウェブページのネットワークでありノードとエッジはそれぞれウェブページとハイパーリンクを表す.

表 1: データセットの統計情報

| Dataset | ノード数 | エッジ数 | 属性数 | クラス数 |
|---|---|---|---|---|
| Cora | 2708 | 5429 | 1433 | 7 |
| Citeseer | 3312 | 4732 | 3703 | 6 |
| Pubmed | 19717 | 44338 | 500 | 3 |
| Wiki | 2405 | 17981 | 4973 | 17 |

### 評価指標

クラスタリング結果を評価するため accuracy (ACC) と normalized mutual information (NMI), そして F1 score (F1) [23] の 3 つの幅広く用いられる指標を用いる. これらの指標は正解クラスタと得られたクラスタを比較してクラスタリング結果を評価する。値が高いほどクラスタリング精度が良いことを示す.

### ベースライン手法

GCNMF のクラスタリング精度を評価するため, 11 のクラスタリング手法を用いる. それらの手法は 3 つのタイプに分けることができる. 1 つ目はグラフ構造情報のみを使用するタイプである. このタイプとして, 特異値分解を行う spectral clustering [24], NMF の一種である deep autoencoder-like non-negative matrix factorization (DANMF) [18], そしてランダムウォークを用いたグラフ埋め込み手法 DeepWalk [13] を用いる. 2 つ目は属性情報のみを使用するタイプである. このタイプとして, 基本的なクラスタリング手法である $k$-means [20], そして NMF [21] を用いる. 3 つ目はグラフ構造と属性情報両方を使用するタイプである. このタイプとして, text-associated DeepWalk (TADW) [1], graph autoencoder (GAE) と variational graph autoencoder (VGAE) [5], marginalized graph autoencoder (MGAE) [6], そして adversarially regularized graph autoencoder (ARGE) と adversarially regularized variational graph autoencoder (VAGE) [7] を用いる.

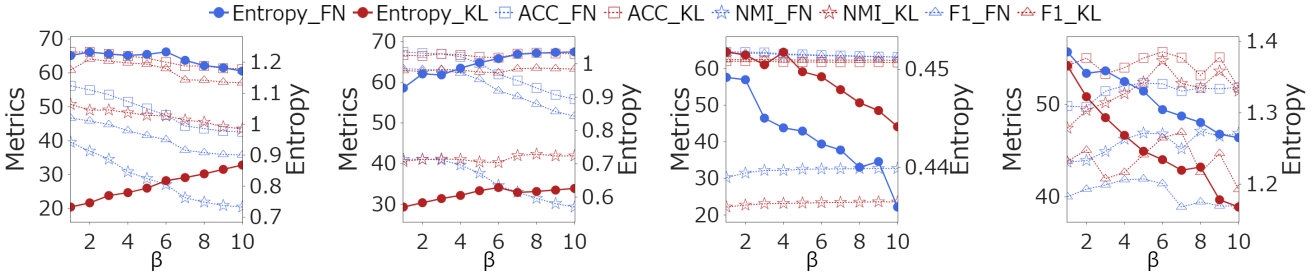
NMF と DANMF については, 提案フレームワークと同様にクラスタ割り当てを行う. その他のベースライン手法については, ノードに対する表現ベクトルに対して $k$-means を適用することでクラスタを得る.

### パラメータ設定

われわれのフレームワークはパラメータ設定を行わない. ベースライン手法のために, 以下のようにパラメータ設定を行った. DANMF では, 制限パラメータを $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$

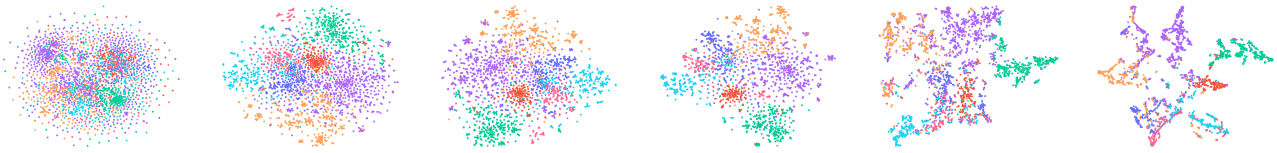
表 2: クラスタリング結果: 評価指標はパーセント表示をしている

| Method | Input | Cora | | | Citeseer | | | Pubmed | | | Wiki | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | NMI | F1 | ACC | NMI | F1 | ACC | NMI | F1 | ACC | NMI | F1 |
| spectral | S | 33.78 | 18.88 | 30.02 | 24.89 | 12.54 | 28.99 | 40.14 | 4.12 | 52.42 | 22.66 | 20.02 | 16.88 |
| DANMF | S | 53.22 | 39.18 | 38.62 | 40.32 | 16.71 | 30.12 | 61.27 | 21.18 | 46.56 | 40.33 | 33.11 | 34.45 |
| DeepWalk | S | 45.42 | 32.21 | 37.89 | 31.75 | 8.97 | 25.78 | 60.12 | 15.83 | 45.16 | 37.12 | 31.37 | 24.99 |
| k-means | A | 32.63 | 17.13 | 24.72 | 39.32 | 18.07 | 31.57 | 55.10 | 27.87 | 55.92 | 32.43 | 31.23 | 23.88 |
| NMF | A | 38.04 | 14.78 | 34.58 | 46.89 | 20.35 | 43.43 | 60.29 | 30.76 | 59.73 | 46.15 | 42.51 | 34.00 |
| TADW | S+A | 52.74 | 35.87 | 40.15 | 53.56 | 31.55 | 44.12 | 57.12 | 23.11 | 49.65 | 31.67 | 28.22 | 21.45 |
| GAE | S+A | 53.51 | 40.22 | 40.95 | 42.27 | 19.11 | 30.44 | 63.13 | 23.92 | 50.26 | 18.56 | 12.12 | 16.64 |
| VGAE | S+A | 54.89 | 38.73 | 42.10 | 46.11 | 23.46 | 32.98 | 62.48 | 21.09 | 47.95 | 29.11 | 31.54 | 21.44 |
| MGAE | S+A | 64.73 | 44.76 | 39.43 | 63.79 | 39.88 | 41.31 | 44.17 | 9.10 | 42.04 | 51.55 | 48.99 | 41.24 |
| ARGE | S+A | 64.01 | 44.89 | 61.01 | 58.10 | 35.11 | 55.89 | 60.02 | 22.89 | 58.12 | 42.32 | 39.61 | 39.09 |
| ARVGE | S+A | 63.80 | 45.55 | **62.61** | 55.35 | 27.10 | 53.02 | 59.08 | 21.52 | 23.56 | 41.75 | 41.21 | 38.22 |
| GCNMF | S+A | **66.29** | **50.74** | 60.88 | **66.64** | **40.52** | **62.84** | **63.41** | **32.81** | **63.34** | **55.76** | **54.95** | **46.45** |




(a) Cora (b) Citeseer (c) Pubmed (d) Wiki

図 1: エントロピーと 3 つの評価指標（ACC, NMI, F1): 評価指標はパーセント表示されている



(a) 元の属性 (b) $\beta = 10$ (c) $\beta = 5$ (d) $\beta = 1$ (e) NMF with FN (f) NMF with KL

図 2: Cora における GCNMF の各ステップのノード特徴の 2 次元可視化

の範囲でチューニングした. 隠れ層の数は 3 として, Cora, Citeseer, Pubmed そして Wiki において隠れ層のサイズをそれぞれ 256-64-7, 256-64-6, 512-64-3 そして 256-128-19 のように設定した. DeepWalk ではランダムウォークの回数を 10, 表現ベクトルのサイズを 128 そしてランダムウォークのパス長を 80 に設定した. TADW では表現ベクトルのサイズを 80, 制限パラメータを 0.2 とした. GAE 及び VGAE ではエンコーダの隠れ層のサイズを 32, 表現ベクトルのサイズ 16 とした. MGAE では変造パラメータ p を 0.4, エンコーダの層の数を 3, パラメータ $\lambda$ を $10^{-5}$ とした. ARGE と ARVGE では GAE と同様のエンコーダを構築して, 識別器の隠れ層を 16 と 64 とした. Cora, Citeseer そして Wiki においては識別器の訓練率を 0.001, Pubmed においては 0.008 とした.

### 4.2 クラスタリング結果と分析

クラスタリング結果を表 2に示す. 各手法 10 回クラスタリングを行いその平均をクラスタリング結果としている. 太字は各データ及び各指標おいて最も良い結果を表す. S と A, そして S+A は入力のタイプを表していて, それぞれグラフ構造のみ, 属性のみ, グラフ構造と属性両方を表す.

グラフ構造と属性両方を使用するベースライン手法と比較して, 提案フレームワーク GCNMF は Cora における F1 の値を除いてほとんどすべての場合において最も良いクラスタリング精度を示している. これは, GCNMF の優位性, つまり自動パラメータ探索, 柔軟なグラフ畳み込みフィルタ, そしてクラスタリングに対する直接アプローチが有効であることを示している.

また, GCNMF はすべてのデータセットにおいてグラフ構造もしくは属性情報のみを使用するベースライン手法に対しても優れた結果を示している. 特に, GCNMF は単純な NMF に対して大きく精度を向上させている. 例えば, Cora において GCNMF は ACC, NMI そして F1 をそれぞれ 28.25%, 35.96% そして 26.30%結果を向上させている. この結果から GCNMF が適

切にグラフ構造情報と属性情報を統合していることがわかる.

Cora 及び Citeseer においては, グラフ構造と属性情報両方を使用するベースライン手法は一方のみを使用するベースライン手法に比べ, 多くの場合において優れた結果を示している. しかし, Pubmed 及び Wiki においては, 精度が落ちている場合が多いことがわかる. これは, 既存手法が実グラフの多様性に対応できず, 適切にグラフ構造と属性情報を統合できていない可能性があることを示している.

### 4.3 自動パラメータ探索の有効性

自動パラメータ探索アルゴリズムの有効性を示すため, 図 1に $\beta$ と損失関数に対する基底行列のエントロピーとクラスタリング評価指標のプロットを示す.

全てのデータセットにおいて, 自動パラメータ探索アルゴリズムは適切な損失関数を選択できていることがわかる. つまり, そのアルゴリズムによって選択された損失関数を使用した場合のクラスタリング結果は選ばなかった方に対して同等以上の精度を示している. 例えば, Cora においてわれわれの自動パラメータ探索アルゴリズムは, 基底行列のエントロピーがより低いため, 損失関数として KL を使用した方を選択する（図 1(a) における損失関数として KL を使用したときのエントロピーを表す赤い実線は FN を使用したときのエントロピーを表す青の実線よりも低い）. 実際に, KL を使用したときのクラスタリング結果（赤い点線）が FN を使用したときのクラスタリング結果（青い点線）よりも高いことがわかる.

われわれの自動パラメータ探索アルゴリズムは適切な $\beta$ として Cora, Citeseer, Pubmed 及び Wiki においてそれぞれ 1, 1, 8 そして 7 を選択する. 図 1は多くの場合において選ばれた $\beta$ によって最高もしくはそれに近いクラスタリング精度を達成していることがわかる. 例えば, Cora において $\beta = 1$ のとき ACC（赤い点線と四角）と NMI（赤い点線と星）は最高の値, そして F1（赤い点線と三角）は最高に近い精度を達成している. しかし, Citeseer においては選ばれた $\beta$（$= 1$）よりもむしろ $\beta = 8$ のときの方がクラスタリング精度が高い. このことから, Citeseer のおける正解クラスタはクラスタ分離度があまり高くないと推測できる.

### 4.4 可 視 化

図 2では Cora におけるノードの属性もしくは特徴を可視化している. 同じ色は正解クラスタが同じノードであることを表す. われわれは t-SNE [25] を用いて属性もしくは特徴を 2 次元に圧縮している.

図 2(a) ではグラフ畳み込みフィルタを適用する前の属性行列を使いノードを可視化している. 同じ色を持ったノード同士の小さな塊が存在するが多くのノードがランダムに散在している. このことから属性情報だけではクラスタ構造を捉えることが難しいことがわかる. 図 2(b), (c) 及び (d) は $\beta$ がそれぞれ 10, 5 及び 1 としたときのグラフ畳み込みフィルタを適用した後の特徴を基にノードを可視化した図である. (b) と (a) を比較して, グラフ畳み込みフィルタがグラフ構造を取り入れたことによってクラスタがよりまとまりを持っている. $\beta$ を減少させると（(b), (c), (d) の順で）クラスタ構造がより密になっている. このことから Cora においてはグラフ構造情報がクラスタ構造にとって重要であることがわかる.

図 2(e) 及び (f) ではそれぞれ FN と KL を使用した損失関数を用いた NMF で得られたクラスタ寄与ベクトルを基にノードを可視化している. KL を使用したときの方がクラスタがより分離していることが確認できる. 実際に KL を使用したときの方がエントロピーが低く, クラスタリング精度も高い（図 1を参照）.

## 5 ま と め

本稿では新しいグラフクラスタリングフレームワーク GCNMF を提案した. 柔軟なグラフ畳み込みフィルタを高い解釈性を持った NMF を統合することで, 直接的な属性付きグラフクラスタリングフレームワークを構築した. われわれは様々な属性付きグラフに適応するため, 教師なし学習のルールに従い, 提案フレームワークが自動でクラスタの分離度が高くなるように適切なパラメータを探索するよう設計した. 自動パラメータ探索によって, われわれのフレームワークはパラメータをチューニングすることなくより良いクラスタを発見することが可能である. 実験において, 様々な既存手法に対して GCNMF が属性付きグラフクラスタリングタスクにおける優位性を持つことを示した.

## 文 献


[1] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[2] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[3] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. Community detection in attributed graphs: an embedding approach. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[5] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[6] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898. ACM, 2017.

[7] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

[8] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[9] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[10] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of ICDM*, pages 107–114, 2001.

[11] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[12] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh

Mukherjee, and Sanjukta Bhowmick. On the permanence of vertices in network communities. In *Proceedings of SIGKDD*, pages 1396–1405, 2014.

[13] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*, pages 701–710, 2014.

[14] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of SIGKDD*, pages 855–864, 2016.

[15] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of SIGKDD*, pages 1225–1234, 2016.

[16] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of AAAI*, 2017.

[17] Bing-Jie Sun, Huawei Shen, Jinhua Gao, Wentao Ouyang, and Xueqi Cheng. A non-negative symmetric encoder-decoder approach for community detection. In *Proceedings of CIKM*, pages 597–606, 2017.

[18] Fanghua Ye, Chuan Chen, and Zibin Zheng. Deep autoencoder-like nonnegative matrix factorization for community detection. In *Proceedings of CIKM*, pages 1393–1402, 2018.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[20] Pankaj K Agarwal and Nabil H Mustafa. K-means projective clustering. In *Proceedings of PODS*, pages 155–165, 2004.

[21] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[22] C. Boutsidis and E. Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350 – 1362, 2008.

[23] Charu C. Aggarwal and Chandan K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 1st edition, 2013.

[24] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.