DEIM2021 E11-3

# 半教師ありノード分類のための適応的ノード埋め込み伝搬ニューラルネットワーク

小川　裕也† 　前川　政司† 　佐々木勇和† 　藤原　靖宏†† 　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
†† NTT コミュニケーション科学基礎研究所　〒 243-0198 神奈川県厚木市森の里若宮 3–1
E-mail: †{ogawa.yuya,maekawa.seiji,sasaki,onizuka}@ist.osaka-u.ac.jp, ††yasuhiro.fujiwara.kh@hco.ntt.co.jp

**あらまし**　グラフ畳み込みニューラルネットワーク (GCNs) は少数のラベルノードを用いてノードのクラスラベルを予測する半教師ありノード分類において成功を収めている．GCNs では層の数を増加させることで，グラフ上で多ホップ離れたノード間の関係を利用し，大量のパラメータによって高い表現力を得ることが可能であるが以下の 3 つの問題を持つ．(1) 大量のパラメータを利用することによる過適合, (2) 多数のグラフ畳み込み演算によって埋め込みが似た値に収束する過平滑化, そして (3) 伝播のホップ数の選択の困難性である．本稿では，上記の GCNs の欠点を解消するために ANEPN を提案する．提案手法でははじめに，(1) 過適合を回避するために，パラメータの数を一定に保ちながら伝播のホップ数を増加させる Embedding Propagation Loss を導入する．さらに, (2) 過平滑化を回避するために，埋め込みが似た値に収束することを防ぐ Anti-Smoothness Loss を導入する．最後に，(3) 適応的に伝播のホップ数を制御するために，予測ラベルを評価する指標を導入する．3 つの標準的なグラフを用いた評価実験を行い，10 の最新の既存手法に対して分類精度向上を達成した．

**キーワード**　半教師あり学習, ノード分類, グラフ畳み込み，深層学習

## 1 はじめに

半教師ありノード分類はグラフのノードのクラスラベルを予測することを目的としており，グラフ解析において重要なタスクの一つである．グラフ畳み込みニューラルネットワーク (GCNs) [1] はニューラルネットワークの一種であり，半教師ありノード分類において成功を収めている．GCNs はグラフ上で埋め込みを近傍に伝播することで，グラフ構造とノード特徴量を組み合わせる．近年では，多くの GCNs が提案されており，自己注意機構をグラフ畳み込みに取り入れた GAT [2], 隣接ノードのサンプリング及び集約を行う GraphSage [3], ノードのプーリングを行う GraphUnet [4], そしてグラフ畳み込みに lanczos 法を利用する Lnet [5] などがある．

既存手法では多くの GCNs は 2 層構造であり，各層においてグラフ畳み込み演算をすることで埋め込みを伝播する．そのため，互いに 2 ホップより離れた (多ホップ離れた) ノード間の関係を利用することができない．GCNs では多くの層を積み重ねることで，多ホップ離れたノード間の関係を利用することが可能であり，そのノード関係は半教師ありノード分類において有効であると示されている [6]．しかしながら，多層 GCNs は以下の 3 つの問題がある．(1) 層の数が増加するにつれてパラメータの数が増加するため，多層 GCNs は訓練データに対して過適合を起こしやすい [7]．(2) 層を積み重ねることで，埋め込みが似た値に収束する過平滑化が発生する [8]．(3) 層の数，すなわち伝播のホップ数を事前に定義する必要がある．これは 入力グラフに依存する適切なホップ数を事前に知ることはできないため，GCNs が伝播のホップ数を決定することが困難であることを示している．

上記の 3 つの多層 GCNs の問題を解決するため，本稿では適応的に伝播のホップ数を増加させることで効果的に多ホップ離れたノード間の関係を利用する Adaptive Node Embedding propagation Network (ANEPN) を提案する．提案手法では，はじめに過適合を回避するために，パラメータの数を一定に保ちながら伝播のホップ数を増加させる Embedding propagation Loss (EPL) を導入する．層を積み重ねる代わりに勾配降下法の更新式によって埋め込みを伝播するように EPL を設計している．さらに，過平滑化を回避するため，Anti-Smoothness Loss (ASL) を導入する．ASL は構造的に離れたノード (直接エッジで接続されていないノード) が似た埋め込みを持たないように，それらのノードの埋め込みに罰則を科す．最後に，同じ予測ラベルを持つノードの埋め込みが近づき，異なる予測ラベルを持つノードの埋め込みが遠ざかるように予測ラベルを評価する指標を導入する．この指標に基づいて，ANEPN はモデル訓練中に伝播のホップ数を制御する．3 つの標準的なデータセットを使用した評価実験において，ANEPN が多ホップ離れたノード間の関係を利用することで，10 個の既存手法に対して高い分類精度を達成することを示した．

本稿の構成は以下のようである．2 章において事前準備と関連研究について述べる．3 章では，GCNs の欠点を克服するために ANEPN を提案する．4 章では，半教師ありノード分類における ANEPN の有効性を示すため実験を行う．最後に，5 章において本稿をまとめる．

## 2 事前準備

$N$ 個のノードから成る属性グラフ $\mathcal{G}$ は隣接行列 $\boldsymbol{A}$，特徴行列 $\boldsymbol{X}$，そしてクラス行列 $\boldsymbol{T}$ によって表される．隣接行列 $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ において，その $(i,j)$ 要素 $A_{ij}$ はノード $i$ とノード $j$ の間にエッジが存在する場合 $A_{ij}=1$，その他は $A_{ij}=0$ である．本稿ではエッジの向きを考慮しない無向グラフかつエッジの重みがないグラフを考える．$\boldsymbol{D}=diag(d_1,\cdots,d_N)\in\mathbb{R}^{N\times N}$ は隣接行列の次数行列を表す．ここで $d_i=\sum_j A_{ij}$ はノード $i$ の次数である. 正規化グラフラプラシアン行列 $\tilde{\boldsymbol{L}}$ は $\tilde{\boldsymbol{L}}=\boldsymbol{I}-\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ として定義される.. 特徴行列 $\boldsymbol{X}\in\mathbb{R}^{N\times F}$ はノードの特徴量を表し，その $i$ 番目の行ベクトル $\boldsymbol{X}_{i,:}$ はノード $i$ の $F$ 次元の特徴ベクトルである．クラス行列は $\boldsymbol{T}\in\mathbb{R}^{N\times C}$ 各ノードのクラス情報を持つ．ここで $C$ はクラス数を表す．ノードはラベルノード集合 $\mathcal{V}^L$ とラベルなしノード集合 $\mathcal{V}^U$ に分けられており ($|\mathcal{V}^L|+|\mathcal{V}^U|=N$)，$i$ 番目の行ベクトル $\boldsymbol{T}_{i,:}\in\{0,1\}^K$ はノード $i$ にクラスラベル $c$ が与えらている場合 (すなわち，ノード $i\in\mathcal{V}^L$)，$\boldsymbol{T}_{i,c}=1$ かつ $\boldsymbol{T}_{i,c'}=0\quad(c'\neq c)$ となる．ノード $i$ にラベルが与えられていない場合 (すなわち，ノード $i\in\mathcal{V}^U$)，$\boldsymbol{T}_{i,c'}=0\quad(0\leq c'\leq C-1)$ となる．半教師ありノード分類はラベルなしノード集合 $\mathcal{V}^U$ に含まれるノードのラベルを予測することを目的としている．

## 3 関連研究

グラフベース半教師あり学習は古くから研究されている．ラベル伝播法 [9] などの古典的な手法 [10–12] は隣接するノードは同じラベルを持ちやすいと仮定のもと，分類を行う．これはクラスタ仮定と呼ばれ，多くの手法がこの仮定を用いている [13]. しかしながら，これらの古典的な手法は特徴情報を使用せず，グラフ構造のみを使用する．この問題を解決するために，グラフ構造と特徴情報の両方を利用する手法が考案されてきた．例えば，SemiEmb [14] や Planetoid [15] はニューラルネットワークによって特徴量を埋め込みに変換して，正則化項を用いることでグラフ構造情報を取り入れる．

一方でグラフ畳み込みニューラルネットワークはニューラルネットワークモデルを使用して直接グラフ構造情報を埋め込みに取り入れる．GCN [1] は標準的な GCN モデルの一つである．一層の隠れ層と出力層を持ち，各層においてグラフ畳み込みを適用する．伝播のホップ数は層の数に対応しているため，GCN は各ノードn埋め込みを 2 ホップ内の隣接ノードへ伝播する．グラフ畳み込みはグラフフィルタ行列 $\boldsymbol{G}\in\mathbb{R}^{N\times N}$ と特徴行列 $\boldsymbol{X}$ (もしくはノード埋め込み) の積で定義される．グラフフィルタは特徴量内のノイズを取り除くローパスフィルタとしても知られる．Kipf と Welling ら [1] はグラフフィルタを以下のように定義している．

$$\boldsymbol{G}=\boldsymbol{I}+\tilde{\boldsymbol{L}}. \tag{1}$$

さらに彼らはそのフィルタに以下のように normalizatio trick を適用している．

$$\boldsymbol{I}+\tilde{\boldsymbol{L}}\rightarrow\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}, \tag{2}$$

ここで $\tilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$ かつ $\tilde{\boldsymbol{D}}_{ii}=\sum_j\tilde{\boldsymbol{A}}_{ij}$ である．$H$ を隠れ層の次元サイズであるとする．$\boldsymbol{W}^{(0)}\in\mathbb{R}^{F\times H}$ と $\boldsymbol{B}^{(0)}\in\mathbb{R}^{N\times H}$ はそれぞれ隠れ層の重み行列とバイアス行列であるとする．隠れ層では，GCN は以下のように埋め込み $\boldsymbol{Z}\in\mathbb{R}^{N\times H}$ を出力する．

$$\boldsymbol{Z}=\max(\boldsymbol{G}\boldsymbol{X}\boldsymbol{W}^{(0)}+\boldsymbol{B}^{(0)},0). \tag{3}$$

出力層では，GCN は以下のように予測ラベル行列 $\boldsymbol{Y}\in\mathbb{R}^{N\times C}$ を出力する．

$$\boldsymbol{Y}=\mathrm{softmax}(\boldsymbol{G}\boldsymbol{Z}\boldsymbol{W}^{(1)}+\boldsymbol{B}^{(1)}), \tag{4}$$

ここで $\boldsymbol{W}^{(1)}\in\mathbb{R}^{H\times C}$ 及び $\boldsymbol{B}^{(1)}\in\mathbb{R}^{N\times C}$ はそれぞれ出力層の重み行列とバイアス行列である．さらにある行列 $\boldsymbol{P}$ に対して $\mathrm{softmax}(\boldsymbol{P})_{ic}=\frac{\boldsymbol{P}_{ic}}{\sum_c\boldsymbol{P}_{ic}}$. 式 (4) において，GCN は以下のようにグラフフィルタ $\boldsymbol{G}$ とノード埋め込み $\boldsymbol{Z}$ の積演算によって埋め込みを隣接ノードへ伝播している．

$$(\boldsymbol{G}\boldsymbol{Z})_{ij}=\frac{\boldsymbol{Z}_{ij}}{d_i+1}+\sum_{k\in\Gamma(i)}\frac{\boldsymbol{Z}_{kj}}{\sqrt{d_i+1}\sqrt{d_k+1}}, \tag{5}$$

ここで $\Gamma(i)$ はノード $i$ の隣接ノード集合を表す．式 (5) では, 第一項はノード $i$ からノード $i$ 自身への伝播を，第二項はノード $i$ の隣接ノード集合からノード $i$ への伝播を表す．よって，係数 $\frac{1}{d_i+1}$ 及び $\frac{1}{\sqrt{d_i+1}\sqrt{d_k+1}}$ はそれぞれ自己ループ及び隣接ノード集合への伝播の重みを表す．GCN の損失関数は以下のように予測ラベルと正解ラベルを近づける cross entropy loss として定義される．

$$L_{ce}=-\sum_{i,c=1}^{N,C}\boldsymbol{T}_{ic}\log(\boldsymbol{Y}_{ic}). \tag{6}$$

パラメータ ($\boldsymbol{W}^{(0)}$, $\boldsymbol{B}^{(0)}$, $\boldsymbol{W}^{(1)}$, $\boldsymbol{B}^{(1)}$) はこの損失関数を減少するように学習される．

上記の GCN を含め，多くの GCN モデルは 2 層構造のため多ホップ離れたノード間の関係を利用しない．しかしながら，もし多ホップ離れたノード間の関係を利用するために多くの層を積み重ねると，多層 GCN は過適合，過平滑化，そして伝播ホップ数の選択困難性という 3 つの問題に悩まされる．Xie らによって提案された ALaGCN は伝播のホップ数を増加させることで多ホップ離れたノード間の関係を利用することができる．しかし，グラフ畳み込みの回数が多すぎると過平滑化が発生するため 10 ホップ以内の隣接ノードへしか埋め込みを伝播しない．一方で，提案手法では過平滑化を回避するため，10 ホップ以上 (例えば，Cora データセットでは 50 ホップ以上 (5章参照)) の隣接ノードへと埋め込みを伝播することが可能である．

## 4 提案手法

この章では，GCNs の問題に対処するために ANEPN を提案する．ANEPN の損失関数は以下のように cross entropy loss $L_{ce}$ 及び 2 つの追加の損失 Embedding Propagation Loss (EPL) $L_{ep}$ and Anti-Smoothness Loss (ASL) $L_{asm}$ から構成される．

$$L=L_{ce}+\alpha L_{ep}+\beta L_{asm}. \tag{7}$$

ここで $\alpha$ 及び $\beta$ はそれぞれ EPL と ASL の係数を表す．EPL はパラメータの数を一定に保ちながら伝播の数を増加させるため，ANEPN は過適合を回避する．ASL は全ての接続されたノードの埋め込みが似た値に収束することを防ぐため，ANEPN は過平滑化も回避する．さらに，予測ラベルを評価するための指標を導入することで入力グラフに応じて伝播の数を制御する．以下に提案手法のそれぞれの詳細について述べる．

### 4.1 Embedding Propagation Loss

過適合を回避するために，Embedding Propagation Loss (EPL) を導入する．EPL はパラメータの数を一定に保ちながら伝播の数を増加させることが可能である．EPL は勾配降下法によって埋め込みを更新することで埋め込みが伝播されるように，ノード埋め込み $\boldsymbol{Z}$ の関数として設計される．そして，$\boldsymbol{Z}$ の更新式が $\boldsymbol{Z}$ に対するグラフ畳み込みとして機能するように定式化する．更新式を反復的に適用することで，ANEPN は層の数，すなわちパラメータの数を増加をさせずにノード埋め込みを伝播する．したがって，ANEPN は過適合に悩まされることなく多ホップ内の隣接ノードへ埋め込みを伝播することが可能である．はじめに埋め込みの更新式の一般形を定義して，次にその更新式から EPL を導出する．

#### 4.1.1 埋め込みの更新式

$\frac{\partial L_{ep}}{\partial \boldsymbol{W}^{(0)}_{:,h}}$ 及び $\frac{\partial L_{ep}}{\partial \boldsymbol{B}^{(0)}_{:,h}}$ はそれぞれ $L_{ep}$ の $\boldsymbol{W}^{(0)}_{:,h}$ 及び $\boldsymbol{B}^{(0)}_{:,h}$ に対する勾配であるとする．ここで $\boldsymbol{Z} = \boldsymbol{G}\boldsymbol{X}\boldsymbol{W}^{(0)} + \boldsymbol{B}^{(0)}$.[1]であるため，$\boldsymbol{W}^{(0)}_{:,h}$ 及び $\boldsymbol{B}^{(0)}_{:,h}$ に対する勾配を得ることができる．$\eta$ を学習率であるとする．勾配降下法によって隠れ層の重み行列の $h$ 番目の列ベクトル $\boldsymbol{W}^{(0)}_{:,h}$ および隠れ層のバイアス行列 $\boldsymbol{B}^{(0)}_{:,h}$ はそれぞれ以下のように更新される．

$$\boldsymbol{W}^{(0)}_{:,h} \leftarrow \boldsymbol{W}^{(0)}_{:,h} - \eta \frac{\partial L_{ep}}{\partial \boldsymbol{W}^{(0)}_{:,h}} \tag{8}$$

$$\boldsymbol{B}^{(0)}_{:,h} \leftarrow \boldsymbol{B}^{(0)}_{:,h} - \eta \frac{\partial L_{ep}}{\partial \boldsymbol{B}^{(0)}_{:,h}}. \tag{9}$$

よって $h$ 番目の列ベクトル $\boldsymbol{Z}_{:,h} \in \mathbb{R}^N$ は勾配降下法にしたがって以下のように更新される．

$$\boldsymbol{Z}_{:,h} \leftarrow \boldsymbol{G}\boldsymbol{X}(\boldsymbol{W}^{(0)}_{:,h} - \eta \frac{\partial L_{ep}}{\partial \boldsymbol{W}^{(0)}_{:,h}}) + \boldsymbol{B}^{(0)}_{:,h} - \eta \frac{\partial L_{ep}}{\partial \boldsymbol{B}^{(0)}_{:,h}}. \tag{10}$$

$\frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}}$ は $\boldsymbol{Z}_{:,h}$ に対する $L_{ep}$ の勾配であるとする．連鎖律によって $\frac{\partial L_{ep}}{\partial \boldsymbol{W}^{(0)}_{:,h}}$ 及び $\frac{\partial L_{ep}}{\partial \boldsymbol{B}^{(0)}_{:,h}}$ はそれぞれ以下の式 (11) 及び (12) によって $\frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}}$ によって表すことが可能である．

$$\frac{\partial L_{ep}}{\partial \boldsymbol{W}^{(0)}_{:,h}} = \frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}} \frac{\partial \boldsymbol{Z}_{:,h}}{\partial \boldsymbol{W}^{(0)}_{:,h}} \tag{11}$$

$$\frac{\partial L_{ep}}{\partial \boldsymbol{B}^{(0)}_{:,h}} = \frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}} \frac{\partial \boldsymbol{Z}_{:,h}}{\partial \boldsymbol{B}^{(0)}_{:,h}} \tag{12}$$

式 (11) 及び式 (12) を用いることで，式 (10) は以下のように書き換えることができる．

$$\boldsymbol{Z}_{:,h} \leftarrow \boldsymbol{Z}_{:,h} - \eta \boldsymbol{G}\boldsymbol{X}(\boldsymbol{G}\boldsymbol{X})^T \frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}} - \eta \frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}}, \tag{13}$$

ノード埋め込み $\boldsymbol{Z}$ を伝播するために，$\frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}}$ がノード埋め込みに対するグラフ畳み込み，すなわちスカラパラメータ $\gamma$ 及び $\nu$ を持つグラフフィルタ $(\gamma\tilde{\boldsymbol{L}} - \nu\boldsymbol{I})$ とノード埋め込みの積演算を表現するように以下のように設計する．

$$\frac{\partial L_{ep}}{\partial \boldsymbol{Z}_{:,h}} = (\gamma\tilde{\boldsymbol{L}} - \nu\boldsymbol{I})\boldsymbol{Z}_{:,h}. \tag{14}$$

この式 (14) を使用することで，式 (13) の第三項はノード埋め込み $\boldsymbol{Z}$ を以下のように更新する．

$$\boldsymbol{Z}_{ih} \leftarrow (\nu - \gamma)\boldsymbol{Z}_{ih} + \gamma \textstyle\sum_{j\in\Gamma(i)} \frac{\boldsymbol{Z}_{jh}}{\sqrt{d_i}\sqrt{d_j}} \tag{15}$$

更新式 (15) は 1 回のノード埋め込み $\boldsymbol{Z}$ の伝播に対応する．更新式の第一項はあるノード $i$ からそのノード $i$ 自身への伝播，第二項はあるノード $i$ の隣接ノードからノード $i$ 自身への伝播として機能する．したがって，ANEPN は重み行列のようなパラメータ含む多くの層を積み重ねる代わりに更新式を反復的に適用することで複数ホップ内の隣接ノードへ埋め込みを伝播する．結果として，ANEPN はパラメータの数を一定に保ちながら反復回数 $t$ を増加させることで多ホップ離れたノード間の関係を利用することが可能である．

#### 4.1.2 EPL の導出

以下のように式 (14) の両辺を積分することで EPL $L_{ep}$ を導出する．

$$L_{ep}(\gamma, \nu) = \textstyle\int (\gamma\tilde{\boldsymbol{L}} - \nu\boldsymbol{I})\boldsymbol{Z}_{:,h} d\boldsymbol{Z}_{:,h}. \tag{16}$$

式 (16) はグラフ畳み込みにおける伝播の重み ($\gamma$ 及び $\nu$) によってパラメータ化された EPL の一般形を示している．EPL が隣接ノード間の埋め込みの smoothness [16, 17] を測るようにその重みを設定する．具体的には $\nu$ を $\frac{2\boldsymbol{Z}^T_{:,h}\boldsymbol{Z}_{:,h}}{H(\boldsymbol{Z}^T_{:,h}\boldsymbol{Z}_{:,h})^2}$ に，そして $\gamma$ を $\frac{2\boldsymbol{Z}^T_{:,h}\tilde{\boldsymbol{L}}\boldsymbol{Z}_{:,h}}{H(\boldsymbol{Z}^T_{:,h}\boldsymbol{Z}_{:,h})^2}$ に割り当てることで，以下の式を得る．

$$L_{ep} = \tfrac{1}{2H} \textstyle\sum_{h,i,j=1}^{H,N,N} \boldsymbol{A}_{ij}(\frac{\hat{\boldsymbol{Z}}_{ih}}{\sqrt{d_i}} - \frac{\hat{\boldsymbol{Z}}_{jh}}{\sqrt{d_j}})^2, \tag{17}$$

where $\hat{\boldsymbol{Z}}_{:,h} = \frac{\boldsymbol{Z}_{:,h}}{||\boldsymbol{Z}_{:,h}||}$. EPL $L_{ep}$ を最小化することで，ノード埋め込みはより滑らかになる，すなわちグラフ上で構造的に近いノードは似た埋め込みを持つようになる．

### 4.2 Anti-Smoothness Loss

ノード埋め込みの過平滑化を回避するために，Anti-Smoothness Loss (ASL) を導入する．ASL は構造的に離れたノードが似ていない埋め込みを持つようにそれらのノード間の埋め込みの smoothness を増加させる．本稿では構造的に離れたノード間の埋め込みの smoothness を anti-smoothness と呼ぶ．それは構造的に離れたノード間の埋め込み間の距離の合計として以下の式で表される．

$$\tfrac{1}{2H} \textstyle\sum_{h,i,j=1}^{H,N,N} \bar{\boldsymbol{A}}_{ij}(\frac{\hat{\boldsymbol{Z}}_{ih}}{\sqrt{\bar{d}_i}} - \frac{\hat{\boldsymbol{Z}}_{jh}}{\sqrt{\bar{d}_j}})^2 \tag{18}$$

ここで $\bar{d}_i = \sum_j \bar{\boldsymbol{A}}_{ij}$ である．式 (18) において，$i = j$ のとき $\boldsymbol{J}_{ij} = 0$，その他は $\boldsymbol{J}_{ij} = 1$ である行列 $\boldsymbol{J}$ を使用して $\bar{\boldsymbol{A}} = \boldsymbol{J} - \boldsymbol{A}$

1：max(·,0) は後に取り除くため無視している．(4.4) 章参照

**Algorithm 1** ANEPN

**Input:** attributed graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{T}\}$, number of negative edge $\bar{E}$, increasing step $\Delta\alpha$, interval of increasing $T_{int}$, iteration to pre-train $T_p$, maximum iteration $T_{max}$, patience $q$

**Output:** predicted class labels $\boldsymbol{Y}$

**### Parameter initialization and Pre-process ###**
$\alpha \leftarrow 0$ ▷ Regularization coefficient
$p \leftarrow 0$ ▷ Patience count for training stop
$\bar{\boldsymbol{X}} \leftarrow \boldsymbol{G}^2\boldsymbol{X}$ ▷ pre-smoothing
$\bar{\boldsymbol{A}}^{sam} \leftarrow$ Negative edge sampling($\boldsymbol{A}$,$\bar{E}$)
**### Pre-train ###**
**for** $t$ from 1 to $T_p$ **do**
  $\boldsymbol{Z} \leftarrow \bar{\boldsymbol{X}}\boldsymbol{W}_{\boldsymbol{Z}} + \boldsymbol{b}_{\boldsymbol{Z}}$
  $\boldsymbol{Y} \leftarrow \mathrm{softmax}(\boldsymbol{Z}\boldsymbol{W}_{\boldsymbol{Y}} + \boldsymbol{b}_{\boldsymbol{Y}})$
  $loss \leftarrow L_{ce}$
  Update network parameters ($\boldsymbol{W}_{\boldsymbol{Z}}, \boldsymbol{b}_{\boldsymbol{Z}}, \boldsymbol{W}_{\boldsymbol{Y}}, \boldsymbol{b}_{\boldsymbol{Y}}$)
**end for**
**### Main-train ###**
**for** $t$ from $T_p$ to $T_{max}$ **do**
  $\boldsymbol{Z} \leftarrow \bar{\boldsymbol{X}}\boldsymbol{W}_{\boldsymbol{Z}} + \boldsymbol{b}_{\boldsymbol{Z}}$
  $\boldsymbol{Y} \leftarrow \mathrm{softmax}(\boldsymbol{Z}\boldsymbol{W}_{\boldsymbol{Y}} + \boldsymbol{b}_{\boldsymbol{Y}})$
  **if** $t\%T_{int} == 0$ **then**
    $\alpha += \Delta\alpha$
  **end if**
  **if** $\mathrm{VR}^{(t)}(\boldsymbol{Z}, \boldsymbol{Y}) \leq \mathrm{VR}^{(t-1)}(\boldsymbol{Z}, \boldsymbol{Y})$ **then**
    $p += 1$
  **end if**
  **if** $q == p$ **then**
    break
  **end if**
  $loss \leftarrow L_{ce} + \alpha L_{sm} + \alpha L_{asm}$
  Update network parameters ($\boldsymbol{W}_{\boldsymbol{Z}}, \boldsymbol{b}_{\boldsymbol{Z}}, \boldsymbol{W}_{\boldsymbol{Y}}, \boldsymbol{b}_{\boldsymbol{Y}}$)
**end for**
**return** $\boldsymbol{Y}$

である．$\bar{\boldsymbol{A}}$ は negative edge matrix であり，構造的に離れたノード間の仮想的なエッジで表される．$\bar{\boldsymbol{A}}$ 内の negative edge 数が膨大であるため，[18] にしたがい $2E$ 本 ($E$ は隣接行列 $\boldsymbol{A}$ 内のエッジの数) の negative edge を $\bar{\boldsymbol{A}}$ からサンプリングすることで計算コストを削減する．簡単化のためランダムに一様サンプリングを行う[2]．

$\bar{\boldsymbol{A}}$ を sampled negative edge matrix $\bar{\boldsymbol{A}}^{sam}$ で置き換えることにより，以下のように ASL を定式化する．

$$L_{asm} = \max(\mu - \frac{1}{2H}\sum_{h,i,j=1}^{H,N,N}\bar{\boldsymbol{A}}_{ij}^{sam}(\frac{\hat{\boldsymbol{Z}}_{ih}}{\sqrt{\hat{d}_i}} - \frac{\hat{\boldsymbol{Z}}_{jh}}{\sqrt{\hat{d}_j}})^2, 0) \quad (19)$$

contrasitive loss [21] で使用される考えを利用することで，式 (19) において ASL が正の値をとるように anti-smoothness の程度を表すマージンパラメータ $\mu$ を導入して，さらに $\max(\cdot, 0)$ を使用する．EPL と ASL が互いに敵対するように作用するため [22]，$\beta = \alpha$ (式 (7) 参照) を置くことで EPL と ASL に同じ係数 $\alpha$ を使用する．

2：1 つ以上の共通の隣接ノードを持つノードは構造的に近いとみなされるため，サンプリングから除外する [19,20].

### 4.3 適応伝播制御

反復の数 $t$，すなわち伝播ホップ数を制御するために，予測ラベルを評価する指標を導入する．その指標に基づいて入力グラフに応じて $t$ を制御することが可能である．指標を導入するために，Siamese Network [23] の考えを利用する．Siamese Network はノード埋め込みのクラス内分散 (同じクラスラベルを持つノードの埋め込み間の距離の合計) を最小化して，ノード埋め込みのクラス間分散 (異なるクラスラベルを持つノードの埋め込み間の距離の合計) を最大化する．$\boldsymbol{m} = \frac{1}{N}\sum_i \boldsymbol{Z}_{i,:}$ は全ノード埋め込みの平均ベクトルとする．さらに，$\boldsymbol{m}_q = \frac{1}{N_q}\sum_{i\in C_q}\boldsymbol{Z}_{i,:}$ はクラス $C_q$ 内のノードの埋め込みの平均ベクトルを表すとする．ここで $N_q$ はクラス $C_q$ 内のノードの数である．ノード埋め込み $\boldsymbol{Z}$ と予測ラベル $\boldsymbol{Y}$ を使用して，クラス内分散をクラス内分散行列 $\boldsymbol{W}_c \in \mathbb{R}^{N\times N}$ のトレース $\mathrm{Tr}(\boldsymbol{W}_c)$ として以下のように定義する．

$$\mathrm{Tr}(\boldsymbol{W}_c) = \mathrm{Tr}(\sum_{q=1}^{C}\sum_{i\in C_q}(\boldsymbol{Z}_{i,:} - \boldsymbol{m}_q)^T(\boldsymbol{Z}_{i,:} - \boldsymbol{m}_q)). \quad (20)$$

同様に，クラス間分散をクラス間分散行列 $\boldsymbol{B}_c \in \mathbb{R}^{N\times N}$ のトレース $\mathrm{Tr}(\boldsymbol{B}_c)$ として以下のように定義する．

$$\mathrm{Tr}(\boldsymbol{B}_c) = \mathrm{Tr}(\sum_{q=1}^{C} N_q(\boldsymbol{m}_q - \boldsymbol{m})^T(\boldsymbol{m}_q - \boldsymbol{m})). \quad (21)$$

最後に，クラスタリング結果の指標である Calinski-Harabasz index [24] を拡張することで，上記 2 つの分散の比として以下のように予測ラベルを評価する指標 Variance Ratio (VR) を定義する．

$$\mathrm{VR}(\boldsymbol{Z}, \boldsymbol{Y}) = \frac{\mathrm{Tr}(\boldsymbol{B}_c)}{\mathrm{Tr}(\boldsymbol{W}_c)} \times \frac{N-C}{C-1} \quad (22)$$

この $\mathrm{VR}(\boldsymbol{Z}, \boldsymbol{Y})$ の値が高くなるように反復の回数 $t$ を適応的に制御する．分類精度を向上させるために，EPL と ASL の係数である $\alpha$ も制御する．提案手法では，$t$ と $\alpha$ を反復的に増加させて，各 $t$ における $\mathrm{VR}^{(t)}(\boldsymbol{Z}, \boldsymbol{Y})$ を計算する．そして，$\mathrm{VR}^{(t)}(\boldsymbol{Z}, \boldsymbol{Y})$ が減少すると ($\mathrm{VR}^{(t+1)}(\boldsymbol{Z}, \boldsymbol{Y}) < \mathrm{VR}^{(t)}(\boldsymbol{Z}, \boldsymbol{Y})$ のとき) 反復を停止する．しかし，VR が偶然減少してしまったときに反復を停止すると適切な $t$ を選択できない．よって，ニューラルネットワークの学習における early stopping と同様に $\mathrm{VR}^{(t)}(\boldsymbol{Z}, \boldsymbol{Y})$ の減少を許す回数 patience $q$ を設定する．もし VR が前の反復時点での VR よりも小さくなった場合，すぐに反復を停止せずに patience count $p$ を 1 増加させる．そして，与えられた patience $q$ と等しくなったとき反復を停止する．

### 4.4 ANEPN のアーキテクチャ

分類精度を向上させるために，2で述べた二層 GCN のアーキテクチャを変更する．一つ目に，$\max(\cdot, 0)$ は過平滑化を引き起こす可能性があるため [25] $\max(\cdot, 0)$ を隠れ層 (式 (3) 参照) から取り除く．二つ目に，EPL によってノード埋め込みの伝播がされているため，以下のように出力層 (式 (4) 参照) からグラフフィルターを取り除く．

$$\boldsymbol{Z} = \boldsymbol{G}^2\boldsymbol{X}\boldsymbol{W}^{(0)} + \boldsymbol{B}^{(0)}, \quad (23)$$

$$\boldsymbol{Y} = \mathrm{softmax}(\boldsymbol{Z}\boldsymbol{W}^{(1)} + \boldsymbol{B}^{(1)}). \quad (24)$$

三つ目に，GCN で使用されるグラフフィルター $\boldsymbol{G} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ は適切にノイズを取り除くことができない可能性があるため [17]，特徴行列 $\boldsymbol{X}$ のノイズを取り除くために，適切なローパスフィルタとして AGC [16] で使用されるグラフフィルタ $\boldsymbol{G} = \boldsymbol{I} - \frac{1}{2}\tilde{\boldsymbol{L}}$ を使用する．

Algorithm 1に提案手法 ANEPN のアルゴリズムを示す．はじめに，パラメータを初期化して，特徴行列をフィルタリングし，そして sampled negative edge matrix を取得する (1-5 行目). ANEPN がランダムに初期化された埋め込みを使用することを回避するために，cross entropy loss $L_{ce}$ のみを使用してパラメータ $\{\boldsymbol{W}^{(0)}, \boldsymbol{B}^{(0)}, \boldsymbol{W}^{(1)}, \boldsymbol{B}^{(1)}\}$ を事前学習する (6-11 行目)．事前学習後，すべての損失を利用してパラメータを学習することに加え (14-15 行目 及び 22-23 行目)，係数 $\alpha$ を $T_{int}$ の間隔で増加させて (16-17 行目)，VR を計算及び評価する (18-19 行目). 最後に，ANEPN は予測クラスラベルを出力する (24 行目).

Algorithm 1において，特徴行列 $\boldsymbol{X}$ のフィルタリングの計算量は $\mathcal{O}(EF)$ となる．negative edge のサンプリングの計算量は $\mathcal{O}(Ed)$ となる．ここで $d$ は全ノードの平均次数を表す. ANEPN の訓練において，ニューラルネットワークの順伝播の計算量は $\mathcal{O}(FHC)$ となる．加えて，ANEPN は EPL と ASL 及び VR の計算が必要となる．それらの計算量はそれぞれ $\mathcal{O}(EH)$, $\mathcal{O}(EH)$， そして $\mathcal{O}(NH)$ となる．したがって，ANEPN の全体の計算量は $\mathcal{O}(t(FHC + EH) + EF)$ となる．ここで実世界のグラフにおいて広くエッジの数 $E$ はノードの数 $N$ よりも多いことを利用している．GCN の計算量は $\mathcal{O}(tEFHC)$ であるため [1]，ANEPN はより効率的であることがわかる．

## 5 評価実験

この章では提案手法 ANEPN の評価を行う．この章は以下の 5 つの質問に答えるように設計されている．

- **Q1**: ANEPN は分類精度の観点で既存手法を上回るか?
- **Q2**: ANEPN は効率性の観点で既存手法を上回るか?
- **Q3**: EPL 及び ASL は分類精度，VR，ノード埋め込みにどのような影響を与えるか?
- **Q4**: ANEPN は入力グラフに対して適切に伝播ホップ数を制御することができるか?
- **Q5**: マージンパラメータ $\mu$ は分類精度と VR にどのような影響を与えるか?

### 5.1 設　　定

文献 [26] で使用される設定にしたがって，3 つの属性付きグラフのデータセット，Cora，Citeseer, そして Pubmed を使用する. 表 1に 3 つのデータセットの統計情報をまとめる．Cora, Citeseer そして Pubmed は引用ネットワークである[3]．よって，それらのネットワーク中のノード及びエッジはそれぞれ論文及び引用関係を表す．属性は論文の文章の bag-of-words 表現によって表される．論文はカテゴリごとに分けられており，これがクラスラベルとなる．

表 1: データセットの統計情報

| データセット | ノード数 | エッジ数 | 特徴量の次元 | クラス数 |
|---|---|---|---|---|
| Cora | 2708 | 5429 | 1433 | 7 |
| Citeseer | 3312 | 4732 | 3703 | 6 |
| Pubmed | 19717 | 44338 | 500 | 3 |

本稿では提案手法と以下の 10 の既存手法との比較を行う．

- **Label propagation (LP)** [12] はグラフ上でラベルを伝播することでクラスラベルを予測する．ノードの特徴量は使用せず，グラフ構造情報のみを使用する．
- **Graph Convolutional Network (GCN)** [1] は標準的な 2 層 GCN モデルである．グラフ畳み込みによって，ノード特徴量とグラフ構造を組み合わせる．
- **Graph Attention Network (GAT)** [2] は自己注意機構をグラフ畳み込みに組み込むことで伝播の重みを制御することが可能な 2 層 GCN モデルである．
- **Self-training** [8] は GCN が予測したラベルの中で確信度が高いものを疑似ラベルとして訓練用ラベルノードに追加した後，再び GCN を学習させる手法．この手法では半教師あり学習における自己訓練 [27] の考えを利用している．
- **Co-training** [8] はラベル伝播法の亜種 [28] が予測したラベルの中で確信度が高いものを疑似ラベルとして訓練用ラベルノードに追加した後，GCN を学習させる手法．この手法では半教師あり学習における共訓練 [29] の考えを利用している．
- **Union** [8] は上記 Self-training と Co-training を組み合わせた手法．それぞれが疑似ラベルを付けたノード集合の和集合を訓練用ラベルノードに追加して，GCN を訓練させる．
- **Intersection** [8] 上記 Self-training と Co-training を組み合わせた手法．それぞれが疑似ラベルを付けたノード集合の積集合を訓練用ラベルノードに追加して，GCN を訓練させる．
- **Multi-Stage Self-Supervised Training Algorithm (M3S)** [26] は自己教師あり学習の一種である DeepCluser [30] と GCN を統合した手法である．
- **ALaGCN** 及び **ALaGAT** [6] はそれぞれ GCN と GAT を拡張した手法である．伝播ホップ数を増加させることで多ホップ離れたノード間の関係を利用する．

ANEPN では文献 [2] で使用される設定にしたがって，学習率を 0.01 とした Adam optimizer [31] を使用して，隠れ層の次元を 64，weight decay を 5e-4 に設定する．他のパラメータについては，マージンパラメータ $\mu, \alpha$ の増加幅，増加の間隔，事前訓練の反復回数，最大反復回数，そして patience $q$ をそれぞれ 1, 0.05, 10, 50, 500，そして 10 に設定する．これらのパラメータは VR が最も高くなるときに訓練を停止するように調整されている．実装は Pytorch[4] 1.7.0 を使用した. 比較手法のパラメータについてはそれぞれの論文を参照されたい．

3：これらのデータセットはhttps://github.com/tkipf/gcn/tree/master/gcnで入手可能である

4：https://pytorch.org/

表 2: テストデータに対する分類精度 (%) 太字は最も高い結果を表す． Gain-GCN 及び Gain-SOTA は ANEPN と GCN との差分及び ANEPN と既存手法の中で最も高い値との差分を表す.

| | Cora | | | | | Citeseer | | | | | Pubmed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Label rate | 0.5% | 1% | 2% | 3% | 4% | 0.5% | 1% | 2% | 3% | 4% | 0.03% | 0.05% | 0.1% |
| LP | 54.3 | 60.1 | 64.0 | 65.3 | 66.5 | 37.7 | 42.0 | 44.2 | 45.7 | 46.3 | 58.6 | 61.9 | 66.9 |
| Self-training | 55.4 | 62.5 | 73.0 | 76.4 | 79.1 | 48.4 | 59.5 | 65.4 | 66.0 | 70.2 | 58.7 | 59.2 | 66.6 |
| Co-training | 50.1 | 60.3 | 69.5 | 76.2 | 77.8 | 39.5 | 53.2 | 63.5 | 66.6 | 69.8 | 53.3 | 59.2 | 63.4 |
| Union | 45.7 | 57.3 | 72.5 | 76.3 | 77.2 | 41.2 | 52.9 | 62.7 | 65.6 | 68.1 | 47.2 | 59.1 | 66.3 |
| Intersection | 48.7 | 60.9 | 73.0 | 77.3 | 79.8 | 49.1 | 60.1 | 63.7 | 68.3 | 69.4 | 49.2 | 54.1 | 69.7 |
| M3S | 59.9 | 66.7 | 75.8 | 77.4 | 79.2 | 54.2 | 62.7 | 66.2 | 69.8 | 70.4 | 57.0 | 62.9 | 68.4 |
| GCN | 44.5 | 59.8 | 68.7 | 74.4 | 77.0 | 43.6 | 47.4 | 61.7 | 66.8 | 68.6 | 45.6 | 55.0 | 64.9 |
| GAT | 41.1 | 50.2 | 54.2 | 60.3 | 77.0 | 40.1 | 46.2 | 62.8 | 67.0 | 68.7 | 50.2 | 53.0 | 60.5 |
| ALaGCN | 57.9 | 66.7 | 73.7 | 74.6 | 78.5 | 41.0 | 49.7 | 59.3 | 63.5 | 67.2 | 57.1 | 63.0 | **71.4** |
| ALaGAT | 48.2 | 62.4 | 73.5 | 75.0 | 77.3 | 38.4 | 52.3 | 58.6 | 66.7 | 68.4 | 56.8 | 62.4 | 69.3 |
| ANEPN (ours) | **66.1** | **73.2** | **77.6** | **78.3** | **79.9** | **60.5** | **64.8** | **68.8** | **70.5** | **71.0** | **60.8** | **69.5** | **71.4** |
| Gain-GCN | +21.6 | +13.4 | +8.9 | +3.9 | +2.9 | +16.9 | +17.4 | +7.1 | +3.7 | +2.4 | +15.2 | +14.5 | +6.5 |
| Gain-SOTA | +6.2 | +6.5 | +1.8 | +0.9 | +0.1 | +6.3 | +2.1 | +2.6 | +0.7 | +0.6 | +2.1 | +6.5 | +0.0 |

表 3: 訓練時間 (秒)

| Dataset | Cora | Citeseer | Pubmed |
|---|---|---|---|
| LP | 0.0063 | 0.0063 | 0.028 |
| Self-training | 1.51 | 1.54 | 1.66 |
| Co-training | 1.71 | 2.46 | 290.27 |
| Union | 2.37 | 3.25 | 291.11 |
| Intersection | 2.42 | 3.23 | 291.18 |
| M3S | 3.62 | 3.73 | 4.13 |
| GCN | 0.78 | 0.80 | 0.86 |
| GAT | 2.11 | 2.22 | 38.35 |
| ALaGCN | 126.34 | 89.80 | 241.82 |
| ALaGAT | 50.67 | 150.39 | 309.17 |
| ANEPN | 0.74 | 0.69 | 0.70 |

実行ごとに，ノードは訓練用の小さな集合とテスト用の 1000 サンプルに分割する．文献 [26] の設定にしたがい，Cora では全体に対して 0.5%, 1%, 2%, 3%, 4%，Citeseer では 0.5%, 1%, 2%, 3%, 4% ，Pubmed では 0.03%, 0.05%, 0.1%の訓練用のラベルノードを使用してモデルを学習する．

表 4: Cora における ANEPN の各損失の分類精度への影響. “w/o ASL”, “w/o EPL”, “w/o EPL or ASL” は ANEPN から EPL または ASL もしくはその両方を取り除いた場合の結果を表す.

| | Cora | | | | |
|---|---|---|---|---|---|
| Label rate | 0.5% | 1% | 2% | 3% | 4% |
| ANEPN | **66.1** | **73.2** | **77.6** | **78.3** | **79.9** |
| w/o ASL | 39.1 | 60.8 | 68.2 | 75.3 | 78.6 |
| w/o EPL | 52.6 | 63.5 | 71.5 | 73.0 | 73.3 |
| w/o EPL or ASL | 49.4 | 62.6 | 70.0 | 74.4 | 76.2 |

表 5: Cora における提案手法の各損失の VR への影響．“w/o ASL”, “w/o EPL”, “w/o EPL or ASL” はそれぞれ ANEPN から ASL または EPL もしくはその両方を取り除いた場合を表す.

| | Cora | | | | |
|---|---|---|---|---|---|
| Label rate | 0.5% | 1% | 2% | 3% | 4% |
| ANEPN | **596** | **722** | **821** | **851** | **915** |
| w/o ASL | 308 | 353 | 566 | 642 | 706 |
| w/o EPL | 343 | 464 | 587 | 654 | 727 |
| w/o EPL or ASL | 321 | 493 | 602 | 665 | 739 |

### 5.2 結　　果

#### 5.2.1 半教師ありノード分類 (Q1)

表 2は 10 回平均の分類精度 (accuracy) の結果を示している．ANEPN はすべての場合において既存手法を上回る性能を示している．これは大域的にノード埋め込みを伝播することで多ホップ離れたノード間の関係を利用することが半教師ありノード分類において効果的であることを示唆している．ALaGCN 及び ALaGAT は多ホップ離れたノード間の関係を利用するが，ANEPN はより離れたノード間の関係を利用することができるためそれらの手法よりも高い分類精度を達成している．特にラベル率が低い場合において ANEPN は大幅に精度向上させている (表 2の Gain-GCN 及び Gain-SOTA を参照されたい)．例えば，Cora データセット上でラベル率が 0.5%の場合において GCN と比較して 21.6%，既存手法で最も高い値と比較して 6.2% の向上を達成している．この結果はラベルノードが少ないほどより多くのラベルノードとの関係を利用するために，多ホップ離れたノード間の関係を利用することが必要になることを示している．

LP はノード特徴量を使用していないのにも関わらず，ノード特徴量を使用する GCN に対して高い分類精度を達成する場合

(a) 初期値 ($t = 0$) (b) ANEPN ($t = 100$) (c) w/o ASL ($t = 100$) (d) w/o EPL ($t = 100$) (e) w/o ASL or EPL ($t = 100$)

図 1: Cora におけるノード埋め込みの可視化. 各点はノード埋め込みを表し，正解クラスによって色分けされている．

がある (例えば, Cora データセット上でラベル率が 0.5%の場合や Pubmed での結果)．これは，LP では予測ラベルが変化しなくなるまでラベル情報をグラフ上で伝播させることで，多ホップ離れたノードのラベル情報を利用することが可能であるためである．その結果，クラスラベルが構造に強く依存するデータセットの場合，すなわち隣り合うノード同士は同じラベルを持つデータセットの場合では LP の分類精度が GCN を上回ることがある．Self-training, Co-training, Union, Intersection, M3S は予測ラベルの内確信度が高いものをラベルセットに加えることで，間接的に多ホップ離れたノード間の関係を利用する．しかし，予測ラベルは間違っている可能性があるため，直接的に多ホップ離れたノード間の関係を利用する ANEPN に比べ分類精度が下がる．

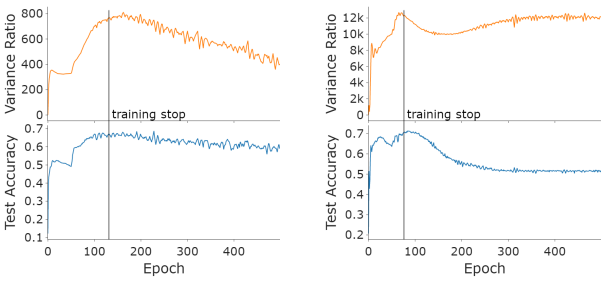
**5.2.2** 訓練時間 (Q2)

表 3に, Tesla V100 GPU with 16GB RAM 上での各手法の訓練時間の結果を示す．ANEPN は高い分類精度を達成しながら他の GCN モデルよりも高速に学習を終了している．LP はすべての手法の中で最も早く訓練を終了しているが，その分類精度は表 2に示すように提案手法よりも大幅に低い．Self-training, Co-training, Union, Intersection, M3S は GCN の訓練を何度も繰り返すため，GCN よりも訓練に時間を要する．特に，Co-training, Union, Intersection では逆行列演算を含むため大幅に訓練時間が長い．GAT では自己注意機構を用いたグラフ畳み込み演算の計算が GCN のグラフ畳み込み演算の計算に比べ計算量が多いため，GCN よりも訓練時間が長い．そして，ALaGCN 及び ALaGAT は多層構造であるためニューラルネットワークの順伝播の計算に時間を要するため，訓練時間が長い．

**5.2.3** 各損失の影響 (Q3)

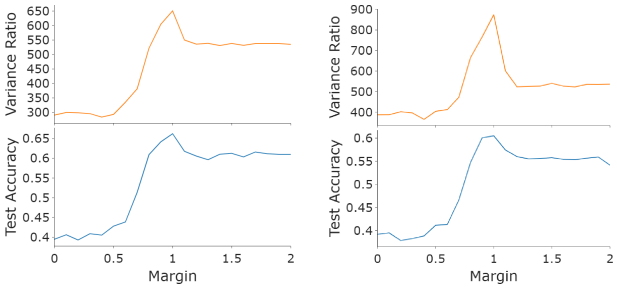
ここでは EPL と ASL の分類精度，VR，埋め込みへの影響を評価する．表 4に ANEPN と ANEPN から EPL または ASL もしくはその両方を取り除いた場合 (それぞれ “w/o ASL”, “w/o EPL”, “w/o EPL or ASL”) の Cora における分類精度の結果をまとめる．表に示されるように，ANEPN は損失を取り除いた場合よりも分類精度が高い．これは EPL と ASL が分類精度の向上に効果的であることを示している．特に ASL を取り除いた場合の “w/o ASL”と比較して，ASL を導入することで過平滑化を回避して，その結果分類精度の向上につながったことを示唆する．

表 5に ANEPN と ANEPN から EPL または ASL もしくはその両方を取り除いた場合 (それぞれ “w/o ASL”, “w/o EPL”, “w/o EPL or ASL”) の Cora における VR の結果をまとめる．表に示




(a) Cora (b) Pubmed

図 2: 反復の回数に対する分類精度 (test accuracy) と VR の値




(a) Cora (b) Citeseer

図 3: マージンに対する分類精度 (test accuracy) と VR の値

されるように，ANEPN は損失を取り除いた場合よりも VR の値が高い．この結果は ASL と EPL による作用によって引き起こされている．すなわち，(1) ASL はグラフ上で離れたノード間の埋め込みを遠ざけることで，異なるクラス間でノード埋め込みを分離して, (2) EPL は埋め込みの伝播をすることで構造的に近いノード同士の埋め込みを近づけることで，同じクラス内のノードの埋め込みの距離が近づけるためである..

図 1に ANEPN または ANEPN から EPL または ASL もしくはその両方を取り除いた場合の Cora におけるノード埋め込みの可視化を示す．埋め込みの可視化のために tSNE [32] を用いて 64 次元 (=隠れ層の次元 $H$) を 2 次元に圧縮している．各点はノード埋め込みを表し，正解クラスによって色分けされている．初期値 (図 1 (a)) では異なるクラス間の境界が近く分離されておらず，同じクラス内のノード埋め込みが離れている．しかし，ANEPN による訓練後，(図 1 (b)) 異なるクラス間でノード埋め込みが分離され，同じクラス内のノード埋め込みが近づいている．一方で，EPL または ASL もしくはその両方を取り除いた場合 (図 1 (c), (d), (e)) においてノード埋め込みが異なるクラス間で分離されていない．

**5.2.4** 伝播ホップ数の制御 (Q4)

ANEPN が適切に伝播ホップ数を制御しているかを検証する．

図 2 は反復回数に対するテストデータの分類精度 (test accuracy) 及び VR の値を示している．ここで図中の “training stop” は ANEPN が訓練を停止した時点での反復回数を示している．なお，この実験においては評価のために “training stop”の後に訓練を続けている．Citeseer における “training stop”が Cora における結果と近いため，Citeseer における結果は省略している．図 2 は test accuracy がそれぞれのグラフにおける最も高い値に近いため ANEPN が適切に訓練を停止している，すなわち適切に伝播ホップ数を制御していることを示している．

**5.2.5** マージンパラメータの影響 (Q5)

図 3に，マージンパラメータ $\mu$ は分類精度と VR にどのような影響を与えるかを検証するためにマージンパラメータに対するそれらの値を示している．なお Pubmed においても同様の結果を得たため省略している．図 3に示すように，マージンが 0 付近のとき test accuracy 及び VR の値は非常に低い．これは ASL がほとんど作用せず，過平滑化を引き起こしてしまうためである．一方，ASL が過平滑化を回避するように機能するため，マージンが 1 付近に近づくにつれて test accuracy 及び VR の値が上昇する．しかしながら，マージンが比較的大きい場合 (図 3中で 1.5 より大きい場合)，test accuracy 及び VR の値は減少する．これは，ASL の働きを強くなりすぎてしまい，EPL によるノード埋め込みの伝播を妨げるように働くためであると考えられる．

## 6 ま と め

本稿では多くの GCN モデルが多ホップ離れたノード間の関係を利用できないことを議論した．多層 GCNs は多ホップ離れたノード間の関係を利用することができるが，過適合，過平滑化，そして適切な伝播ホップ数の選択が困難という 3 つの問題に悩まされる．上記の問題を解決するために，本稿では多ホップ離れたノード間の関係を効果的に利用する ANEPN を提案した．EPL と ASL を利用することで，ANEPN は過適合と過平滑化に悩まされることなく多ホップ内の隣接ノードへ埋め込みを伝播することができる．さらに，ANEPN は予測ラベルを評価する VR に基づいて伝播ホップ数を適応的に制御する．評価実験によって，ANEPN の半教師ありノード分類への有効性を示した．

**謝辞**

本研究は JSPS 科研費 JP20H00583 の助成を受けたものです．

### 文　献

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[4] Hongyang Gao and S. Ji. Graph u-nets. In *ICML*, 2019.

[5] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. Lanczosnet: Multi-scale deep graph convo-lutional networks. In *ICLR*, 2019.

[6] Y. Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. When do gnns work: Understanding and improving neighborhood aggregation. In *IJCAI*, 2020.

[7] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *ICLR*, 2020.

[8] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pages 3538–3545, 2018.

[9] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[10] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, 2001.

[11] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. label propagation and quadratic criterion. In *semi-supervised learning*, pages 193–216, 2006.

[12] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, pages 321–328, 2004.

[13] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *AISTATS*, 2005.

[14] J. Weston, F. Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *ICML*, 2008.

[15] Z. Yang, William W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

[16] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. In *IJCAI*, pages 4327–4333. AAAI Press, 2019.

[17] Ganqu Cui, J. Zhou, Cheng Yang, and Zhiyuan Liu. Adaptive graph encoder for attributed graph embedding. *KDD*, 2020.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[19] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

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

[21] Raia Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. *CVPR*, 2:1735–1742, 2006.

[22] Ian J. Goodfellow, Jean Pouget-Abadie, M. Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[23] S. Chopra, Raia Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *CVPR*, 1:539–546 vol. 1, 2005.

[24] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[25] K. Sun, Zhouchen Lin, and Zhanxing Zhu. Adagcn: Adaboosting graph convolutional networks into deep models. *ArXiv*, abs/1908.05081, 2019.

[26] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labels. In *AAAI*, 2020.

[27] Xiaojin Zhu. Semi-supervised learning literature survey. volume 2, page 4.

[28] Xiao ming Wu, Zhenguo Li, Anthony M. So, John Wright, and Shih fu Chang. Learning with partially absorbing random walks. In *NIPS*, pages 3077–3085, 2012.

[29] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98*, 1998.

[30] M. Caron, P. Bojanowski, Armand Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[32] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.