動的グラフ畳み込みへのカプセルネットワークの適用

Evolving Graph Capsule Convolutional Networks for Dynamic Graphs

島田研太 [1]　尾崎知伸 [2]
Kenta Shimada [1]　Tomonobu Ozaki[2]

[1] 日本大学大学院総合基礎科学研究科
[1] Graduate School of Integrated Basic Sciences, Nihon University
[2] 日本大学文理学部
[2] Callege of Humanities and Science,Nihon University

Abstract: A dynamic graph is a graph in which vertices and edges increase and decrease with time. The most current neural network models for handling dynamic graphs are not sufficient to hold and express rich characteristics on the changes of graph structures since they propagate only scalar values in the model. To alleviate this limitation by replacing the scalars to the vectors, we propose to take the concept of capsule networks into account for the graph convolutional neural networks. The obtained model is evaluated using real world dynamic graphs.

1　はじめに

グラフデータは，個体とその関係を表す汎用的なデータ構造である．近年では，ニューラルネットワークを中心に，グラフデータを対象とした分析技術が盛んに研究されている．グラフ畳み込みニューラルネットワーク（Graph Convolutional Neural Networks, GCN）は，グラフデータに対する畳み込み演算を定義することで，効率的なグラフ埋め込み表現を獲得でき，高精度な分類や予測を実現している．一方，静的なグラフに加え，時間と共に構造が変化する動的グラフに対しても，いくつかのニューラルネットワークモデルが提案されている．例えば文献 [1] では，GCN とリカレントニューラルネットワーク（Recurrent Neural Network, RNN）を融合したモデルが提案されている．また文献 [2] では，GCN のパラメタを RNN から計算することにより，時間軸全体のノード情報を必要としない GCN である EvolveGCN が提案されている．

動的グラフを対象としたこれらの既存モデルでは，一般的なニューラルネットワーク同様，各ノード（ニューロン）間でスカラー値を伝播させることで対象を表現する．しかし動的グラフにおいては，単にノードの値が変化するだけでなく，変化の前後間の関係性も考慮する必要があり，ノード表現をスカラー単位で運用した場合，必ずしも十分な表現能力を有するとは言えない．

本研究では，動的グラフを対象としたニューラルネットワークモデルの能力向上を目指し，カプセルグラフニューラルネットワーク（Capsule Graph Neural Network）[3] のアイディアを援用することを提案する．すなわち，各ノードの出力をベクトルに置き換えることで，より多くの周辺ノードの情報や，変化するノードの前後関係を捉えることが可能な動的グラフに対する GCN を提案する．具体的には動的な GCN のモデルに対し，畳み込み演算後のノードの出力をベクトルへと変換する．加えて，入力値をベクトルとしたグラフ畳み込み演算と，自己注意（self-attention）による正規化を用いて高品質なグラフ埋め込みを表現する．また，実データを用いた辺予測タスクを通じ，提案モデルの評価を行う．

本論文の構成は以下の通りである．2 章では動的グラフ畳み込み手法，また 3 章ではカプセルグラフニューラルネットワークの概要をそれぞれ示す．4 章で動的グラフ畳み込みに対するカプセルグラフニューラルネットワークの適用を提案する．5 章で実験と評価を行い，最後に 6 章でまとめと今後の課題を述べる．

2　動的グラフ畳み込み

本章ではまず，文献 [5] に従い，グラフ畳み込み演算を形式的な定義を与える．その後，文献 [2] に従い，動的グラフへの畳み込み演算の適用について説明する．

2.1 グラフ畳み込み

時刻 t における第 l 層への入力であるノード特徴行列を $H_t^{(l)}$ と表記する．A_t をグラフの隣接行列，$W_t^{(l)}$ を訓練可能な重み行列としたとき，$H_t^{(l)}$ のグラフ畳み込みを下式で与える．

$$
\begin{aligned}
H_t^{(l+1)} &= \mathrm{GCONV}(\, A_t,\, H_t^{(l)},\, W_t^{(l)}\,) \\
&:= \sigma(\, \hat{A}_t\, H_t^{(l)}\, W_t^{(l)}\,) \\
&where \\
\widehat{A} &= \widetilde{D}^{-\frac{1}{2}}\, \widetilde{A}\, \widetilde{D}^{-\frac{1}{2}}, \\
\widetilde{A} &= A + I, \\
\widetilde{D} &= diag\left(\sum_j \widetilde{A}_{ij} \right)
\end{aligned}
$$

ここで $\widehat{A}_t$ と $\widetilde{D}$ は，それぞれ正規化グラフラプラシアン行列および $\widetilde{A}$ の次数行列である．一方，σ は活性化関数であり，一般的には ReLU 関数を用いる．また，初期のノード特徴行列 $H_t^{(0)}$ をグラフのノード特徴量とし，畳み込み演算を L 回繰り返すことで得られる特徴行列 $H_t^{(L)}$ をグラフの埋め込み表現（分散表現）とする．

2.2 EvolveGCN

動的グラフを対象としたモデルとして，これまでに，GCN と RNN を組み合わせたモデルがいくつか提案されている [1, 6]．これらのモデルでは，GCN を特徴抽出器として利用し，抽出された特徴を RNN へと渡すことで分類や予測を行う構成となっている．しかし基本的に，事前知識としてすべてのノード情報が必要であり，新ノードの出現や既存ノードの消失が頻繁に繰り返されるような不規則な振る舞いを示す動的グラフに対しては，RNN での学習が困難であり，必ずしも性能が期待できないといった弱点が知られている．

これに対し，EvolveGCN[2] として，パラメタである GCN の重み行列を時間ごとに更新するモデルが提案されている．具体的には，過去の重み行列と現在のグラフ埋め込み表現を GRU または LSTM に与え，GCN の重み行列を更新する．また GCN は，更新された重み行列を用いて現在のグラフに対して畳み込み処理を行い，グラフ埋め込み表現を更新する．時間ごとにこれらの処理を繰り返すことで，過去に情報を持たない新たなノードを含むグラフに対する頑健性の向上が期待できる．

図 1 に，GRU と LSTM のそれぞれを用いた進化グラフ畳み込みユニット（Evolving Graph Convolutional Unit, EGCU）を示す．なお，GRU を用いて重み更新

EGCU-H

$[H_t^{(l+1)}, W_t^{(l)}] = \mathrm{EGCU\text{-}H}(A_t, H_t^{(l)}, W_{t-1}^{(l)})$
　$X_t = summarize(H^{(l)}, \#col(W_{t-1}^{(l)}))^T$
　$W_t^{(l)} = \mathrm{GRU}(X_t, W_{t-1}^{(l)})$
　$H_t^{(l+1)} = \mathrm{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$

$H_t = \mathrm{GRU}(X_t, H_{t-1})$
　$Z_t = \mathrm{sigmoid}(W_Z X_t + U_Z H_{t-1} + B_Z)$
　$R_t = \mathrm{sigmoid}(W_R X_t + U_R H_{t-1} + B_R)$
　$\tilde{H}_t = \tanh(W_H X_t + U_H (R_t \circ H_{t-1}) + B_H)$
　$H_t = (1 - Z_t) \circ H_{t-1} + Z_t \circ \tilde{H}_t$

$Z_t = \mathrm{summarize}(X_t, k)$
　$y_t = X_t p / ||p||$
　$i_t = \text{top-indiecs}(y_t, k)$
　$Z_t = [X_t \circ \tanh(y_t)] i_t$

EGCU-O

$[H_t^{(l+1)}, W_t^{(l)}] = \mathrm{EGCU\text{-}O}(A_t, H_t^{(l)}, W_{t-1}^{(l)})$
　$W_t^{(l)} = \mathrm{LSTM}(W_{t-1}^{(l)})$
　$H_t^{(l+1)} = \mathrm{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$

$H_t = LSTM(X_t)$
　$F_t = \mathrm{sigmoid}(W_F X_t + U_F H_{t-1} + B_F)$
　$I_t = \mathrm{simoid}(W_I X_t + U_I H_{t-1} + B_I)$
　$O_t = \mathrm{sigmoid}(W_O X_t + U_O H_{t-1} + B_O)$
　$\tilde{C}_t = \tanh(W_C X_t + U_C H_{t-1} + B_C)$
　$C_t = F_t \circ C_{t-1} + I_t \circ \tilde{C}_t$
　$H_t = O_t \circ \tanh(C_t)$

図 1: Evolving Graph Convolutional Unit（[2] より）

を行うユニットを EGCU-H，LSTM を用いて重み更新を行うユニットを EGCU-O と呼ぶ．また，GRU に入力する各グラフはノードの個数を一定以下に要約する必要があり，これには関数 `summarize` を用いている．これらの要約では，時間に依存しないパラメータ p を用いて各ノードの重みづけを行い，上位 k ノードが選択される．詳細は文献 [7, 8] を参照されたい．

3 Capsule Graph Neural Network

一般的な GCN から出力される各ノードの表現はベクトルではなく，スカラーとして表現される [5]．しかしこれはグラフの特性を効率的に保持するのに必ずしも十分ではない．高品質なグラフ埋め込みを表現するには，各ノードの周辺ノードの特徴や構造等だけでは

なく，位置や方向，接続等の詳細な特性を保持することが重要となる．

これに対し，グラフの特性情報をより効率的に保持するために，各ノードの GCN 中での表現をカプセルと呼ばれるベクトルに拡張したモデルである Capusule Graph Neural Network (CapsGNN)[3] が提案されている．CapsGNN は，主に画像処理のために考案されたカプセルネットワーク [4] と呼ばれる手法を GCN に拡張・適用したものであり，大きくノードカプセル抽出ブロック，グラフカプセル抽出ブロック，グラフ分類ブロックの 3 つから構成される．なおカプセル（ベクトル）は，その長さが特性の存在確率を，角度が特性の詳細を表しており，スカラーを利用する場合に比べて多くの情報が表現可能であることが分かる．

CapsGNN の第一ブロックであるノードカプセル抽出ブロックでは，抽出器として GCN[5] を利用し，基本的なノード特徴の抽出を行う．具体的には，GCN の中間層の値の総和を各ノードの特徴とし，そこからカプセル表現を特定する．また，各ノードに対して複数のカプセルを準備することで，それぞれ異なる側面からノードを表現する．

ノードカプセルの数は入力されるグラフサイズに依存する．そのため，第二のブロックであるグラフカプセル抽出ブロックでは，まず注意構造（attention mechanism）を用いて正規化処理を行い，その後，動的ルーティング（Dynamic routing）を用いてグラフカプセルを出力する．具体的には，以下に示す 2 層の全結合ニューラルネットワークから算出される注目度尺度（Attention measure）を用いた正規化を行う．

$$scaled(\mathbf{s}_{(n,i)}) = \frac{F_{attn}(\tilde{\mathbf{s}_n})_i}{\sum_n F_{attn}(\tilde{\mathbf{s}_n})_i}\mathbf{s}_{(n,i)}$$

ここで $\tilde{\mathbf{s}_n} \in \mathbb{R}^{1\times C_{all}d}$ と $\mathbf{s}_{(n,i)} \in \mathbb{R}^{1\times d}$ は，それぞれノード n のすべてのカプセルの総和と，ノード n の第 i 番目のカプセルである．また，$F_{atten}(\tilde{\mathbf{S}_n}) \in \mathbb{R}^{1\times C_{all}}$ は生成された注目度尺度である．

第三のブロックであるグラフ分類ブロックでは，再度，動的ルーティングを用いてグラフカプセルからクラスカプセルを生成する．その際，下記に示す margin loss[4] を分類の損失関数として採用している．

$$\begin{aligned} L_k &= \sum \{T_k \max(0, m^+ - ||\mathbf{v}_k||)^2 \\ &\quad + \lambda(1-T_k)\max(0, ||\mathbf{V}_k|| - m^-)^2\} \end{aligned}$$

ここで $m^+ = 0.9, m^- = 0.1$ であり，また T_k はグラフのクラスが k のときに 1 を取る指示関数である．

EGCU-H′

1 : $[H_t^{(l+1)}, W_t^{(l)}] = \text{EGCU-H}'(A_t, H_t^{(l)}, W_{t-1}^{(l)})$

2 : $X_t = summarize(H^{(l)}, \#col(W_{t-1}^{(l)}))^T$

3 : $W_t^{(l)} = \text{GRU}(X_t, W_{t-1}^{(l)})$

4 : $H_t^{(l+1)} = \text{CapsGNN}(A_t, H_t^{(l)}, W_t^{(l)})$

図 2: 提案モデル

4 ベクトル表現を用いた動的グラフ畳み込みの拡張

既に述べた通り，時間によって各ノードやエッジが変化する動的グラフを対象とした場合，時間情報に加え，ノード間の関係性やグラフ構造の変化を適切かつ十分に表現できるモデルが望まれる．その様なモデルの一つとして，本論文では，EvolveGCN における GCONV を CapsGNN に置き換えたモデルを提案する．すなわち，スカラー値を対象としたグラフ畳み込みをカプセル（ベクトル）へと置き換え，長さが存在確率，向きが特徴の内容を表すというカプセルの高い表現能力を利用することを考える．

EvolveGCN の中間層の値をカプセルに置き換えることで，ノードが頻繁に変化するような動的グラフの中で，ノードの詳細な特性を保持することが可能となり，高品質なグラフ埋め込みと，その結果としての動的グラフに対する GCN の高精度化が期待できる．また，カプセル（ベクトル）を利用する更なる利点として，内積を用いた特性間の比較による，層間での値伝播の効率化があげられる．具体的には，ベクトル内積に基づき各ノード間の関連度合いを算出し，関連度が高い親ノードからはより多くの情報を伝播させることで，より効率的にノード情報を伝えることが可能となると考えている．

図 2 に提案モデルを示す．なお今回は，ノードの情報価値が高い場合は GRU を用いるモデルの方が効率的な場合があるという知見 [2] に基づき，GRU を用いた EGCU-H を基にしている．

ところで，GRU の特性上，GRU から算出される $W_t^{(l)}$ は $H^{(l)}$ とデータのサイズが等しくなる．そのため，CapsGNN の重みのサイズと GRU から算出される $W_t^{(l)}$ のサイズとを揃える必要が生じる．そのための手法として，本研究では，1 層の全結合層を用いる手法と最大値をとる手法の 2 通りを考える．全結合層ではデータの整形も学習対象とさせることで，各データに合った整形となることが予測される．また最大値をとる場合は各値は一意に定まるのでデータに依存しない整形となる．

5　実験と評価

Python を用いて提案モデルを実装し，Bitcoin Alpha[9, 10] データ（プラットフォーム上で取引をするビットコインのユーザ同士の評価ネットワーク）を対象に評価実験を行った．

ビットコインは匿名で利用するため，各利用者の信頼性が重要となるが，各ユーザは，他のユーザから-10〜10 の範囲でその信頼性が評価される形となっている．Bitcoin Alpha データでは，各ユーザが各ノードに対応し，あるユーザが別のユーザを評価した際,（その時間とともに）エッジが付与される．また，ノード総数は 3783，エッジ総数は 24186 である．

今回は，以下の手順に従い，リンク予測タスクを対象に実験を行った．すなわち，まずグラフを訓練用とテスト用にそれぞれ分け，次いで，時刻 t におけるグラフの埋め込み表現を GRU と CapsGNN を用いて獲得する．その後，ランダムに選択された時刻 t における 2 つのノードを，先ほど獲得したグラフ埋め込みを用いてベクトル化し，2 つのノードのベクトルを全結合層の入力としてリンクの有無を学習する．

実験の結果を表 1 に示す．表中において，Linear は全結合層を用いた整形手法を採用した場合を，Max は最大値を用いた整形手法を採用した場合をそれぞれ表す．また，EvolveGCN[2] は従来手法であり，スカラー値に基づく動的グラフ畳み込みを採用している．

実験結果より，最大値による整形と比べ，全結合層による整形の方がわずかではあるが再現率と適合率がともに高いことが分かる．一方，従来手法と比べ，提案手法の適合率がわずかに高いこと，再現率が非常に低いことが分かる．この要因として，提案手法におけるモデルの増加が考えられる．既存手法では，GRU が学習する GCN の重みは単一の行列である．これに対し提案手法では，GRU の学習対象である CapsGNN の重み行列は複数存在し，それぞれに対し複数の GRU を用いて重みを計算する必要がある．すなわち，モデルの増加に伴い最適化する重みパラメタが増加し，それぞれの最適化が十分に出来ず，結果として再現率が低い値となったと考えられる．

表 1: 実験結果:各モデルの再現率と適合率

	再現率	適合率
Linear	0.0013	0.0432
Max	0.0010	0.0422
Evolve-GCN	0.9284	0.0301

6　まとめと今後の課題

本研究では，動的グラフに対するグラフ畳み込み処理の高品質化を目的に，カプセルネットワークを用いた EvolveGCN の拡張モデルを提案した．しかし，実データを用いた実験においては，その効果は確認ができなかった．

今後の課題として，低再現率の根本原因の解明と，複数のデータセットを用いた評価があげられる．

参考文献

[1] Y. Seo, M. Defferrard : Structured sequence modeling with graph convolutional recurrent networks, *Advances in Neural Information Processing Systems 31*, pp.362-373, 2018

[2] A. Pareja, G. Domeniconi, J. Chen, et al. : EvolveGCN: Evolving graph convolutional networks for dynamic graphs, *Association for the Advancement of Artificial Intelligence 2020*, 2020

[3] Z. Xinyi, L. Chen. : Capsule graph neural network, *International Conference on Learning Representations 2019*, 2019

[4] S. Sabour, N. Frosst, G. E. Hinton. : Dynamic Routing Between Capsules, *Advances in Neural Information Processing Systems 30*, pp.3856-3866, 2017

[5] T. N. Kipf, M. Welling. : Semi-supervised classification with graph convolutional networks, *International Conference on Learning Representations 2017*, 2017

[6] A. Narayan, P. Roe. : Learning graph dynamics using deep neural networks, *IFAC-PapersOnLine 51*, pp.433-438, 2018

[7] C. Cangea, P. Velickovic. : Towards sparse hierarchical graph classifiers, *NIPS Workshop on Relational Representation Learning*, 2018

[8] H. Gao, S. Ji. : Graph U-Nets, *In Proceedings of the 36th International Conference on Machine Learning*, 2018

[9] S.Kumar, F. Spezzano : Edge Weight Prediction in Weighted Signed Networks, *IEEE International Conference on Data Mining*, 2016

[10] S. Kumar, B. Hooi : REV2: Fraudulent User Prediction in Rating Platforms, *11th ACM International Conference on Web Search and Data Mining*, 2018.