社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

LSTM と条件付き VAE を用いた 構造的特徴を指定可能なグラフ生成モデル

中沢　昇平† 　佐藤　良紀† 　中川　健治† 　津川　　翔†† 　渡部　康平†

† 長岡技術科学大学 大学院工学研究科　〒940–2188 新潟県長岡市上富岡町 1603–1
†† 筑波大学 システム情報系　〒305–8573 茨城県つくば市天王台 1–1–1
E-mail: †{s173160, s171039}@stn.nagaokaut.ac.jp, †{nakagawa, k_watabe}@vos.nagaokaut.ac.jp,
††s-tugawa@cs.tsukuba.ac.jp

あらまし　近年, グラフを用いたアプリケーション, シミュレーションの重要性が高まっている. グラフには様々な特徴があり, 特徴が異なると, シミュレーションの結果は異なる. そのため, 様々な特徴を有したグラフを用意するための生成手法が研究されている. 古典的には, リンクやノードの確率分布を与えて生成するモデルが研究されてきたが, 近年では, 機械学習を用いることで, 実際のグラフデータから特徴を学習し, そのグラフを模倣したグラフを生成する手法が提案されてきている. しかし従来の機械学習を用いた研究では, データから特徴を学習できるが, 特徴を指定して, 任意の特徴のグラフの生成を行うことはできない. 本稿では, 機械学習により, データからグラフを学習しつつ, 特定の特徴のグラフを指定し, 生成可能なモデルを提案する. また, 提案した生成モデルの有効性を検証するために, 古典的な従来手法の一種によって生成した様々な特徴のグラフを学習し, 生成時に特徴を指定して, 生成が可能であることを示す.
キーワード　ネットワーク, グラフ, 生成, 条件付き VAE, 機械学習

A Study of a Tunable Generative Model for Graph Data Using Machine Learning

Shohei NAKAZAWA†, Yoshiki SATO†, Kenji NAKAGAWA†, Sho TSUGAWA††, and Kohei WATABE†

† Graduate School of Engineering Nagaoka University of Technology,
Kamitomiokamachi 1603–1, Nagaoka, Niigata 940–2188, Japan
†† Faculty of Engineering, Information and Systems, University of Tsukuba,
1-1-1 Tennohdai, Tsukuba, Ibaraki 305–8573, Japan
E-mail: †{s173160, s171039}@stn.nagaokaut.ac.jp, †{nakagawa, k_watabe}@vos.nagaokaut.ac.jp,
††s-tugawa@cs.tsukuba.ac.jp

Abstract　In recent years, applications and simulations using graphs are becoming more important. The graph has various features, and the simulation results differ depending on the features. Therefore,In recent years, applications and simulations using graphs are becoming more important. A graph has various features, and the simulation results differ depending on the features. Therefore, a graph generation method for preparing a graph with various features has been studied. Classically, a model that generates using a pre-defined probability distribution of edges and nodes has been studied. In recent years, a method of generating a graph that imitates the learned graph by learning features from actual graph data using machine learning has been studied. However, in conventional research using machine learning, features can be learned from data, but it is not possible to specify features and generate graphs of arbitrary features. In this paper, we propose a model that learns graphs from data and can generate a specified graph by specifying a value of a feature. With the proposed model that learned graphs of various features generated by the conventional method, we verified whether the graphs of arbitrary features could be generated by specifying the features.
Key words　network, graph, generation, conditional VAE, machine learning

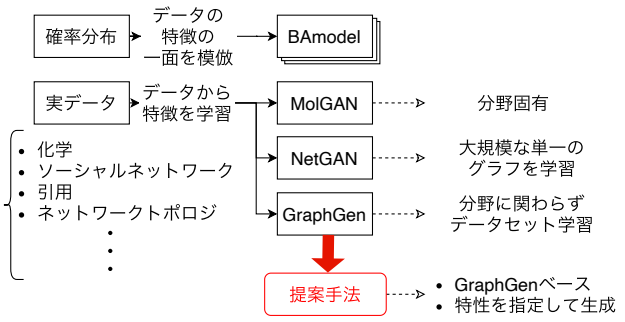

図 1　提案手法の位置付け

1. はじめに

近年, 様々な分野でグラフデータを用いたアプリケーションが増加しており, それに伴い, グラフ生成モデルの需要が高まってきている. ソーシャルネットワークや分子構造, 通信ネットワークなど, 世の中の様々なデータはグラフとして表現可能であり, それらの分野で, グラフ, ノードの分類や異常検知, 推薦など多くのアプリケーションが提案されてきている. 同時にグラフを用いたシミュレーションの重要性も高まってきている. 通信ネットワークであればトラヒックの均一性や QoS 指標に関するシミュレーションを行うことができ, 人と人のつながりを表現したグラフであれば, 感染病の伝染などのシミュレーションを行うことが可能である. 現在までに現実のグラフは様々な特徴を有していることが確認されており, 現実の環境を想定したアプリケーション, シミュレーションの場合, それらの特徴に従うようなグラフの生成が生成モデルに求められる. また, シミュレーションの結果はグラフの構造的な特徴によって大きく変動することが直感的に理解できる. 例えば, 通信ネットワークのトラヒックの均一性に関するシミュレーションであれば, 通信ネットワークにおけるハブがどれほど存在するか, ハブがグラフのどこに存在するかなどで, トラヒックの偏り方が異なる. よって, グラフのシミュレーションの観点から述べると, 生成モデルにより, それらの特徴を滑らかに変化させたグラフを生成し, 結果がどのように変化するかのシミュレーションを行えることが好ましい.

グラフ生成モデルには, 古典的な統計的手法を用いたものが存在し, 実在するグラフの特徴の一面に似たグラフを生成可能だが, 特定の実在するグラフに似たものの生成は難しい. 統計的手法を用いたグラフ生成モデルは, 実在するグラフの構造的特徴に着目して, 事前にノード, エッジなどの確率分布を定義し, 生成を行う. 現実に存在するグラフは, スケールフリー, スモールワールドなど, なんらかの特徴に従っていることが多く, 統計的な手法では, それらの特徴に従うグラフを生成できるように確率分布を定義し, 生成を行う. 統計的な生成手法で生成したグラフは, 特徴の一面は, 上述した実在するグラフの特徴に倣っているが, 特定の実在するグラフの特徴すべてを捉えたグラフの生成は難しい.

グラフ生成モデルには, 機械学習を用いたものも存在し, データからグラフの特徴を学習し, その特徴に則ったグラフを生成可能であるが, 特徴量の値を指定して生成できない. 深層生成モデルの一種である Generative Adversarial Networks (GAN) [1] や Variational Auto Encoder (VAE) [2], ノード, エッジの存在確率を時系列として捉えて学習する Long Short Term Memory (LSTM) [3] を用いたモデルが存在する. 近年までのグラフ生成モデルには, 新薬の発見など化学の分野に特化した MolGAN [4] のように, 分野固有の生成モデルや単一の大規模なグラフを生成する NetGAN [5] が存在している. また, 分野にとらわれず, ラベル付きグラフを学習可能な生成モデル GraphGen [6] も存在する. いずれのモデルも, 実在するグラフデータより, グラフの構造的特徴などを学習するモデルであるが, 現在までに, 学習したグラフの構造的特徴量の値を指定し, 生成可能とするようなグラフ生成モデルは存在していない.

本稿では, 分野に依存せず, スケーラブルなモデルである従来手法の GraphGen と条件付き VAE である CVAE (Conditional VAE) により, グラフの埋め込みと条件によるグラフの構造的特徴量の値の指定を可能とした生成手法を提案する. 本稿で提案するモデルと統計, 機械学習を用いた生成手法との位置付けは, 図 1 に示す通りである. 本稿で提案するモデルは, 統計的な生成手法と異なり, 実在するデータより, グラフの構造的特徴の学習を行い, 近年までの機械学習を用いた生成手法と異なり, グラフの構造的特徴量の値を指定して生成可能なモデルである.

本稿の構成は以下の通りである. 第 2 章において, 本研究で取り扱う問題を定式化する. 次に第 3 章において, 提案法のベースとなっている GraphGen に触れた後に, 第 4 章で, 提案法を説明する. 第 5 章では, 提案法でいくつかの特徴のグラフのデータセットを学習し, 特徴量の値を指定して生成可能であるかを検証し, 提案法の有効性を確かめる. 第 6 章で評価結果を示す. 最後に第 7 章で, 本稿のまとめと今後の展望を述べる.

2. 問題の定式化

本稿では, グラフの構造的特徴をベクトルとして抽出する写像と構造的特徴を表すベクトルより対応したグラフを再構築する逆写像によって問題の定式化を行う. 本章では, 問題の定式化について述べる.

本稿で扱うグラフは全て無向グラフである. 無向グラフ $G = (V, E)$ は, ノード集合 $V = \{v_1, v_2, \ldots, v_n\}$, エッジ集合 $E = \{(v_i, v_j) | v_i, v_j \in V \wedge v_i \neq v_j\}$ で定義される. グラフのノード, エッジはラベル付けされており, $L_n : V \to \boldsymbol{V}$, $V_e : E \to \boldsymbol{E}$ により, すべてのノード, エッジはラベル集合にマッピングされる. $\boldsymbol{V}, \boldsymbol{E}$ はそれぞれノード, エッジのラベル集合を表している. $L(v), L(e)$ は任意のノード v, エッジ e のラベルを表しているものとする. また, グラフはすべて連結グラフであり, 自己ループが存在しないものと仮定する.

存在しうる全てのグラフ集合を $\boldsymbol{\Omega} = \{G_1, G_2, \ldots\}$ とする. グラフ G_i は, 写像 $F(\cdot)$ によって, $F(G_i) = A_{G_i} = [\alpha^1_{G_i} \alpha^2_{G_i} \cdots \alpha^k_{G_i}]^T$ のベクトルに射影される. ベクトル $A_{G_i} \in \mathbb{R}^k$ は, 入力されたグラフ G_i の特徴を表現したベクトルになっており, ベクトルの各次元の値がそれぞれ, 別個の構造的特徴量 (グラフのノード数, グラフのエッジ数, 次数分布のべき指数, クラスタ係数等) の値

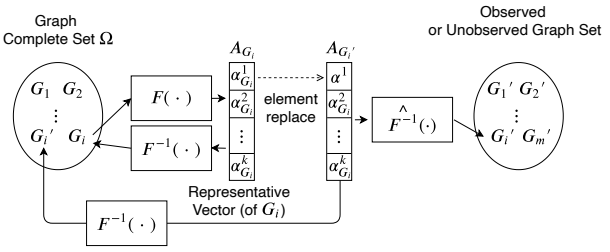

図 2　問題の定式化

になっている. ここで, 逆写像 $F^{-1}(\cdot)$ を定義し, 逆写像により, $F^{-1}(A_{G_i}) = G_i$ であるとする.

本稿では, 図 2 のように, 存在しうる全てのグラフ集合の部分集合である $\boldsymbol{S} = \{G_1, \ldots, G_m\} \subset \boldsymbol{\Omega}$ より, $F(\cdot), F^{-1}(\cdot)$ を近似した $\hat{F}(\cdot), \hat{F}^{-1}(\cdot)$ を, 機械学習を用いて学習する問題を取り扱う. この問題を解くことにより, ベクトル A_{G_i} の任意の次元を任意の値に変更したベクトル $A_{G_i'}$ に従う G_i' を構築することが可能である. 例えば, ベクトル A_{G_i} における 1 次元目の要素を α^1 に変更した $A_{G_i'} = [\alpha^1 \alpha^2_{G_i} \cdots \alpha^k_{G_i}]^T$ より, $\hat{F}^{-1}(A_{G_i'}) = G_i'$ を達成することが可能である. 1 次元目の要素がグラフのクラスタ係数を表現していた場合, G_i' は, クラスタ係数は指定したクラスタ係数, その他のグラフの構造的特徴量の値はグラフ G の A_G に従ったグラフとなる.

3. GraphGen

3.1. DFS コード

GraphGen では, グラフを DFS コードによってグラフ上を深さ優先探索 (DFS) した軌跡を表すシーケンスにエンコードし, 学習を行う. DFS コードは, グラフ上を深さ優先探索した際のエッジの探索順序のシーケンスにエンコードする手法である. 深さ優先探索は, グラフのどのノードから始めても良い.

DFS コードでは, エッジのシーケンスにエンコードするが, それに先立ち, ノードを深さ優先探索で順番付け, 0 から順番にタイムスタンプを付与する. エッジ (u, v) に訪れた際, ノード u, v にそれぞれタイムスタンプ t_u, t_v を割り振る. 図 3(a) のようなグラフが存在した際に, 深さ優先探索の順序が $(v_0, v_1), (v_1, v_2), (v_1, v_3)$ であった際のタイムスタンプは図 3(b) のように示される.

$t_u < t_v$, すなわち, ノードの順番付けと同じ方向につながるエッジを forward edge, $t_u > t_v$, すなわちノードの順番付けを遡る方向のエッジを backward edge と呼ぶ. エッジ $e = (u, v)$ はタイムスタンプを定義したことにより, 5-tuple である $(t_u, t_v, L(u), L(e), L(v))$ で記述することが可能である.

DFS コードは, ノードのタイムスタンプの順番を保ちつつ, 極力隣接したエッジのシーケンスになるようにグラフのエンコードを行う. DFS コードのエンコードは下記の制約に従うように行う.

- 任意の backward edge (u, u') は, (u, v) の全て forward edge の前に順序づける
- 任意の backward edge (u, u') は, (w, u) の全て forward edge の後に順序づける
- 同じノード u からの backward edge である $(u, u'), (u, u'')$ は, もし $t_{u''} < t_{u'}$ であれば, (u, u'') が優先される

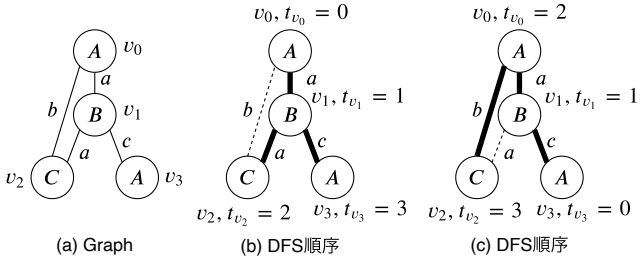

図 3　深さ優先探索順序

最終的に DFS コードにより, グラフは, 5-tuple である $(t_u, t_v, L(u), L(e), L(v))$ からなるシーケンスのリストとして表現される.

上述の制約を満たす 5-tuple のシーケンスは, 無数に存在するが, GraphGen では, 5-tuple の辞書式順序に従い, 優先順序を決める. 例えば, 図 3(a) のようなグラフにおいて, 図 3(b), (c) のように深さ優先探索する最初のノードがそれぞれ v_0, v_3 の時, その 5-tuple はそれぞれ, $(0, 1, A, a, B), (0, 1, A, c, B)$ である. この時, 辞書式順序に従えば, 図 3(b) が最小である. グラフ G の最小 DFS コードによるシーケンスは $F_{\text{minDFS}}(G) = S = [s_1, \cdots, s_m]$ で表現する. $m = |E|$ である. また s は, 5-tuple である. 探索の最初の 5-tuple はグラフ内の辞書式順序において最小のものであり, 複数の未探索エッジが存在した際も, 辞書式順序が最小の 5-tuple から優先に探索していく.

3.2. モデル化

グラフのデータセット $\boldsymbol{G} = \{G_1, \cdots, G_m\}$ は, 最小 DFS コードにより, シーケンスデータセット $\boldsymbol{S} = \{F_{\text{minDFS}}(G_1), \cdots, F_{\text{minDFS}}(G_m)\}$ に変換可能である. GraphGen の目的は, シーケンス S の確率 $p(S)$ をデータから学習することであり, $p(S)$ は, i 番目の 5-tuple を $s_i = (t_u, t_v, L_u, L_e, L_v)$ とし, i 番目までの 5-tuple を $s_{<i}$ とした時, 次式で表される.

$$p(S) = \prod_{i=1}^{m+1} p(s_i | s_1, \cdots, s_{i-1}) = \prod_{i=1}^{m+1} p(s_i | s_{<i}) \tag{1}$$

GraphGen は, 状態推移関数 f_{trans}, 埋め込み関数 f_{emb} と 5-tuple $s_i = (t_u, t_v, L_u, L_e, L_v)$ のそれぞれの要素のための出力関数で構成される. $vstack$ はベクトルの結合を表す. $\sim_M$ は多項分布 θ からのサンプリングを表す. GraphGen は, 再帰的に 5-tuple を入力し, シーケンスの生成を行なっていく. シーケンスの i step の 5-tuple s_i の生成プロセスは次式で表される.

$$h_i = f_{\text{trans}}(h_{i-1}, f_{\text{emb}}(s_i - 1)) \tag{2}$$

$$t_u \sim_M \theta_{t_u} = f_{t_u}(h_i) \tag{3}$$

$$t_v \sim_M \theta_{t_v} = f_{t_v}(h_i) \tag{4}$$

$$L_u \sim_M \theta_{L_u} = f_{L_u}(h_i) \tag{5}$$

$$L_e \sim_M \theta_{L_e} = f_{L_e}(h_i) \tag{6}$$

$$L_v \sim_M \theta_{L_v} = f_{L_v}(h_i) \tag{7}$$

$$\tilde{s}_i = \text{vstack}(t_u, t_v, L_u, L_e, L_v) \tag{8}$$

s_i は 5-tuple のそれぞれの要素を one-hot ベクトルで表現した

ものを結合したベクトルである．(2) 式では，f_{emb} によりスパースベクトル s_i を連続ベクトルに埋め込んだ連続ベクトルと前回のステップの 5-tuple s_{i-1} を f_{trans} を用いて，状態ベクトル h_i に変換する．与えられた状態ベクトル h_i より，関数 $f_{t_u}(h_i), f_{t_v}(h_i), f_{L_u}(h_i), f_{L_e}(h_i), f_{L_v}(h_i)$(それぞれパラメータが $\theta_{t_u}, \theta_{t_v}, \theta_{L_u}, \theta_{L_e}, \theta_{L_v}$) の出力は，エッジ tuple のそれぞれの要素の多項確率分布を表現している．GraphGen では，f_{trans} は多層 LSTM の cell, $f_{t_u}, f_{t_v}, f_{L_u}, f_{L_e}, f_{L_v}$ は dropout を用いた完全結合多層パーセプトロンであり，f_{emb} は学習可能な線形埋め込み関数である．

上記のデザインの重要な要素は，多項分布の近似であり，これを LSTM を用いて実現する．GraphGen は，グラフのデータセット $\boldsymbol{G}$ からシーケンスデータセット $\boldsymbol{S}$ に変換を行い，$\boldsymbol{S}$ より多項式分布の近似を行う．それぞれのシーケンス $S \in \boldsymbol{S}$ の，それぞれのエッジ s_i と (2)~(8) 式によって生成された 5-tuple $\tilde{s}_i$ が近くなるよう binary crossentropy である BCE$(*)$ により，最適化を行う．

$$\begin{aligned}\text{BCE}(\tilde{s}_i, s_i) = -\sum_c (s_i[c]^T \log \tilde{s}_i[c] \\ +(1 - s_i[c])^T \log(1 - \tilde{s}_i[c]))\end{aligned} \tag{9}$$

またシーケンスの最初に SOS (Start Of Sequence) を表現したベクトルと，シーケンスの最後に EOS (End Of Sequence) を表現したベクトルを vstack し，学習することで，生成の開始，終了タイミングも同時に学習させる．

4. 提 案 法

提案モデルでは，GraphGen を Conditional Vartiational Auto Encoder (CVAE) で拡張し，グラフの構造的特徴の値を条件としたベクトルに従ったグラフを生成可能なモデルを提案する．本章では，提案法のモデルの概要を示す．

4.1. 前 処 理

提案モデルでは，GraphGen と同様にグラフ $\boldsymbol{G}$ を DFS コードによって変換したシーケンスデータセット $\boldsymbol{S} = F_{\text{minDFS}}(\boldsymbol{G})$ を学習する．DFS コードのシーケンスの最後の要素は，シーケンスの終了 EOS を表すベクトルとなっている．本稿では，グラフデータはノード，エッジ共にラベルを有していないものとし，前処理で，ノードに次数の逆数をラベリングしている．グラフの次数分布のべき指数を計算する関数を $F_{\text{degree}}(\cdot)$，グラフのクラスタ係数を計算する関数を $F_{\text{cluster}}(\cdot)$ とした時，これらの関数を用いて，グラフのデータセットの集合に対応した条件ベクトルの集合 $\boldsymbol{C} = \{c_1, \ldots, c_m\} = \{[F_{\text{degree}}(G_1)\ F_{\text{cluster}}(G_1)]^T, \ldots, [F_{\text{degree}}(G_m)\ F_{\text{cluster}}(G_m)]^T\}$ を作成する．

4.2. モデル構成

提案モデルは，Encoder と Decoder の二種類の LSTM から成る VAE であり，図 4 のように表される．前処理を行った DFS コードによるシーケンスデータセット $\boldsymbol{S}$ と条件ベクトルの集合 $\boldsymbol{C}$ を用いて学習を行う．

Encoder Encoder は LSTM を用いており，DFS コードによるシーケンスを学習し，その特徴に応じた多変量正規分布にマッピングを行い，その分布から潜在変数 z をサンプリングする． グラフ G_i の DFS コードによるシーケンスを $F_{\text{minDFS}}(G_i) = S_i = \{s_1, s_2, \ldots, s_k\}$, 条件ベクトルを c_i とする．Encoder を f_{enc}, 埋め込み関数を f_{emb} とした時，次式に示すように，学習時は，Encoder に，シーケンスの各要素に対して c_i を vstack したベクトルを再帰的に入力する．$k(= |S_i|)$ 回のイテレーションの後に，LSTM の出力である h_k を，線形層 f_μ, f_σ を用いて，多変量正規分布のパラメータに変換し，reparameterization trick によって，グラフ G_i の潜在変数である z に変換する．c_i と s_{j-1} を Encoder に対してともに入力することで，Encoder に対して，現在の入力 s_{j-1} の条件 c_i であることを学習させる．

$$h_j = f_{\text{enc}}(h_{j-1}, f_{\text{emb}}(\text{vstack}(\text{s}_{\text{j-1}}, \text{c}_\text{i}))) \tag{10}$$

$$z \sim N(f_\mu(h_k), f_\sigma(h_k)) \tag{11}$$

Decoder Decoder は LSTM を用いており，潜在ベクトル z と条件ベクトル c_i を最初の入力とし，その二つのベクトルの値に応じた，特徴のシーケンスを生成する．提案モデルの Decoder は，シーケンスの学習を行う f_{dec}, 埋め込み関数 f_{emb}, 潜在ベクトル z を所定のサイズに変更する関数 f_z, 隠れ層の出力をそれぞれのシーケンスの 5-tuple の要素 $s_i = (t_u, t_v, L_u, L_e, L_v)$ に変換する関数で構成される．t_u, t_v, L_u, L_e, L_v はそれぞれ，Softmax 関数によって表されたカテゴリカル分布からのサンプリングした onehot vector を表す．s_i は，onehot vector 同士を vstack したものとなっており，5-tuple を表す．s_0 は，シーケンスの SOS を表現するベクトルとなる．学習時に，デコーダに対して s_{j-1}, c_i を入力することで，現在のシーケンスが条件 c_i のシーケンスであることを学習させる．生成時には，c_i を元に，c_i に従った特徴のシーケンスを生成させる．

$$s_0 = f_z(z) \tag{12}$$

$$h_j = f_{\text{dec}}(h_{j-1}, f_{\text{emb}}(\text{vstack}(\text{s}_{\text{j-1}}, \text{c}_\text{i}))) \tag{13}$$

$$t_u \sim \theta_{t_u} = \text{Softmax}(f_{t_u}(h_j)) \tag{14}$$

$$t_v \sim \theta_{t_v} = \text{Softmax}(f_{t_v}(h_j)) \tag{15}$$

$$L_u \sim \theta_{L_u} = \text{Softmax}(f_{L_u}(h_j)) \tag{16}$$

$$L_e \sim \theta_{L_e} = \text{Softmax}(f_{L_e}(h_j)) \tag{17}$$

$$L_v \sim \theta_{t_v} = \text{Softmax}(f_{L_v}(h_j)) \tag{18}$$

$$s_i = \text{vstack}(t_u, t_v, L_u, L_e, L_v) \tag{19}$$

4.3. 学習と生成

推論時，提案法の学習は Algorithm 1 に示す手順で行う．まず，グラフのデータセットを $\boldsymbol{G}$, 最小 DFS コードによりシーケンスデータセット $\boldsymbol{S}$ に変換する (1 行目)．その後，全ての学習されていないニューラルネットのパラメータの初期化を行う (2 行目)．シーケンスの各要素 s_{j-1} に対して c_i を vstack したものを f_{emb} によって埋め込んだベクトルと，前回のステップでの LSTM の出力 h_{j-1} を再帰的に，f_{enc} に入力する (8-10 行目)．シーケンス S を，Encoder に対して入力し終えたのち，最後の出力である h_k を，f_μ, f_σ に入力することで，パラメータ μ, σ へ変換する (11

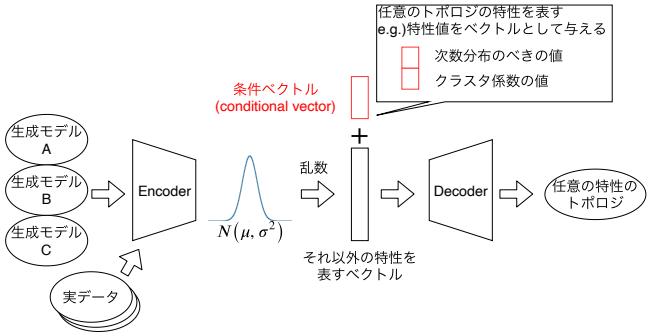

図 4　提案モデル

行目). reparameterization trick により, μ, σ を用いて, 潜在変数 z をサンプリングする (12 行目). $\text{Criterion}_{\text{enc}}$ により, μ, σ より, Encoder の誤差を計算し, 変数 $Loss$ に加算する (13 行目). ベクトル μ, σ の次元が L である時, $Criterion_{\text{enc}}$ は次式で表される. μ_i, σ_i は, ベクトル μ, σ の i 次元目の要素を表す.

$$\text{Criterion}_{\text{enc}}(\mu, \sigma) = \frac{1}{2}\sum_{l=1}^{L}(1+\log((\sigma_l)^2)-(\mu_l)^2-(\sigma_l)^2) \quad (20)$$

f_z により, 潜在変数 z を Decoder への最初の入力 (SOS) である s_0 に変換する (14 行目). Encoder と同様に, シーケンスの各要素 s_{j-1} に対して c_i を vstack したベクトルを f_{emb} によって埋め込んだベクトルと, 前回のステップでの LSTM の出力 h_{j-1} を再帰的に, f_{dec} に入力する (17-18 行目). LSTM の出力 h_j は, 5 つの線形層により, $c \in \{t_u, t_v, L_u, L_e, L_v\}$ の予測分布 θ_c をそれぞれ計算し, 全てを vstack したものを $\tilde{s}_j$ とする (19-23 行目). その後, Decoder の誤差を, $\text{Criterion}_{\text{dec}}$ を用いて計算し, 変数 $Loss$ に加算する (24 行目). 誤差関数 $\text{Criterion}_{\text{dec}}$ を次式に示す. $s[c]$ は, ベクトル s における, $c \in \{t_u, t_v, L_u, L_e, L_v\}$ に対応した次元を抜き出す操作に当たる. また log はベクトルの各要素に対して, 計算を行う.

$$\text{Critrion}_{\text{dec}}(\tilde{s}_j, s_j) = -\sum_{c} s_j[c]\log(\tilde{s}_j[c]) \quad (21)$$

計算した $Loss$ を用いて, 勾配情報の更新を行う (28 行目).

生成時は Algoritm 2 に示すように, 生成した 5-tuple を再帰的に入力し, 生成を行う. まず入力として, 生成したい特徴量の値を要素としたベクトルである C を受け取る. 標準正規分布 N より, 潜在変数 z をサンプリングする (2 行目). 潜在変数 z を関数 f_z によって, Decoder への最初の入力である s_0 に変換する (4 行目). 前回のステップで生成したシーケンスの要素 $\tilde{s}_{i-1}$ に対して C を vstack したものを f_{emb} によって埋め込んだベクトルと, 前回のステップでの LSTM の出力 h_{i-1} を再帰的に, f_{dec} に入力し, 現在のステップの出力である h_i を得る (7 行目). LSTM の出力 h_i は, 15-19 式に示す 5 種類の線形層により, $c \in \{t_u, t_v, L_u, L_e, L_v\}$ の予測分布 θ_c をそれぞれ計算し, 全てを vstack したものを $\tilde{s}_i$ とする (10-12 行目). シーケンスのリストに対して, hstack を用いて新たな 5-tuple を追加する.(14 行目) この生成プロセスを EOS が生成されるまで繰り返す (16 行目).

5. 実　　験

提案モデルにより, データからグラフを学習し, 特徴量の値

Algorithm 1 train VAE

Input :Dataset of Graphs $\boldsymbol{G} = \{G_1, G_2, \ldots, G_m\}$, Condition of Graphs $\boldsymbol{C} = \{C_1, C_2, \ldots, C_m\}$

Output :Learned functions $f_{enc}, f_{t_u}, f_{t_v}, f_{L_u}, f_{L_e}, f_{L_v}, f_{dec}, f_{mu}, f_{sigma}, f_z$

1: $\boldsymbol{S} = \{S = F_{\text{minDFS}}(G) | \forall G \in \boldsymbol{G}\}$
2: Initialize $f_{\text{enc}}, f_{t_u}, f_{t_v}, f_{L_u}, f_{L_e}, f_{L_v}, f_{\text{dec}}$
3: **repeat**
4: **for** i from 1 to m **do**
5: $S = S_i = [s_1, s_2, \ldots, s_k]$
6: $loss \leftarrow 0$
7: Initialize h_0
8: **for** j from 1 to k **do**
9: $h_j \leftarrow f_{\text{enc}}(h_{j-1}, f_{\text{emb}}(\text{vstack}(s_{j-1}, C_i)))$
10: **end for**
11: $\mu = f_\mu(h_k)$; $\sigma = f_\sigma(h_k)$
12: $z \sim N(\mu, \sigma^2)$
13: $loss = loss + \text{Criterion}_{\text{enc}}(\mu, \sigma)$
14: $s_0 = f_z(z)$
15: Initialize h_0
16: **for** j from 1 to k **do**
17: $h_{j-1} = \text{vstack}(h_{j-1}, z)$
18: $h_j = f_{\text{dec}}(h_{j-1}, f_{\text{emb}}(\text{vstack}(s_{j-1}, C_i)))$
19: $\tilde{s}_j \leftarrow []$
20: **for** $c \in (t_u, t_v, L_u, L_e, L_v)$ **do**
21: $\theta_c \leftarrow \text{Softmax}(f_c(h_j))$
22: $\tilde{s}_j \leftarrow \text{vstack}(\tilde{s}_j, \theta_c)$
23: **end for**
24: $loss = loss + \text{Criterion}_{\text{dec}}(\tilde{s}_j, s_j)$
25: **end for**
26: Back-propagate loss and upate weights
27: **end for**
28: **until** stopping criteria

Algorithm 2 generate conditional graph

Input : Desired Condition C

Output : Graph G

1: $S \leftarrow []$
2: $z \sim N(0, I^2)$
3: $i \leftarrow 1$
4: $\hat{s}_0 = f_z(z)$
5: Initialize h_0
6: **repeat**
7: $h_i = f_{\text{dec}}(h_{i-1}, f_{\text{emb}}(vstack(\hat{s}_{i-1}, C)))$
8: $\hat{s}_i \leftarrow []$
9: **for** $c \in (t_u, t_v, L_u, L_e, L_v)$ **do**
10: $\theta_c \leftarrow \text{Softmax}(f_c(h_i))$
11: $c_i \sim \theta_c$
12: $\hat{s}_i = \text{vstack}(\hat{s}_i, c_i)$
13: **end for**
14: $S = \text{hstack}(S, \hat{s}_i)$
15: $i \leftarrow i + 1$
16: **until** $\exists c_i, c_i = \text{EOS}$

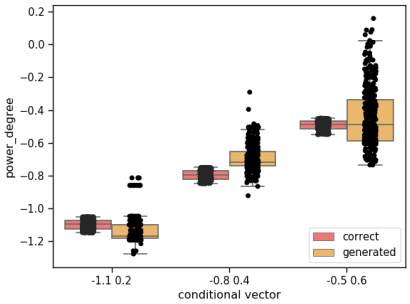

図 5　次数分布のべき指数

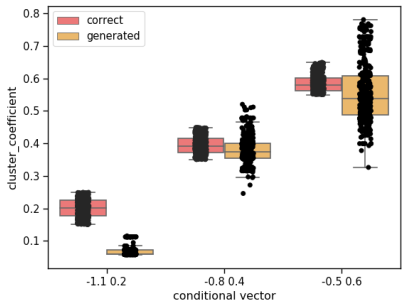

図 6　クラスタ係数

を指定して生成可能であるかの検証を行った．データには, 実データの代替として, グラフの統計的生成モデルの一つである Connecting Nearest Neighbor モデルを用いて生成を行ったグラフの集合より, 特定の特徴の値のグラフをサンプリングして, データセットとした．データセットの内訳は Table 1 に示す．Table 1 における条件ベクトルは,『次数分布のべき指数 _ クラスタ係数』の次数分布のべき指数とクラスタ係数の組み合わせであり, データセットの部分集合に対する id である．データセットのグラフのノード数は全て 25 である．

表 1　学習データセットの内訳

id	次数分布のべき指数	クラスタ係数	データ数
−0.5_0.6	-0.5	0.6	400
−0.8_0.4	-0.8	0.4	400
−1.1_0.2	-1.1	0.2	400

提案法のモデルのパラメータは, Encoder, Decoder の LSTM の隠れ層の次元は 256, 埋め込み層の次元は 128, Encoder の正規分布のパラメータに変換を行う線形層は 20 次元である．なお最適化アルゴリズムは Adam を用いており, 初期学習係数は 0.001, weight decay は 0.01, 重みの clip 閾値は 0.015 である．ミニバッチ数は 60, エポック数は 400 である．

提案法によって, 学習で与えた特徴量 (データセットに含まれる特徴量) の値を指定し, 指定した特徴量の値のグラフを生成可能であるか評価を行った．

6. 結　　果

本節では, 学習時に与えた特徴量 (データセットに含まれる特徴量) の値を指定し, 指定した特徴量の値グラフを生成可能であるかを提案法を用いて検証した．

学習で与えた特徴量を条件ベクトルとした時の生成結果の次数分布のべき指数, クラスタ係数の分布をそれぞれ図 5, 6 に示す．データ数は条件ごとに 100 である．横軸は, 与えた条件ベクトルの種類であり, Table 1 の表記に準拠している．縦軸はそれぞれの特徴量を表している．また赤色の box は, 訓練データの特徴量の分布を表し, 橙色のボックスは特徴量を指定して生成したグラフの特徴量の分布を表している．またデータ点は, 一つ一つのグラフの特徴量を表している．図 5, 6 より, 橙色のボックスが与えた条件ベクトルに応じて変化していることから, 条件ベクトルを与えることで, 生成するグラフの特徴の変動していることが確認できる．また, 赤色と橙色のボックスが近くなっていることから, 概ね, 生成したグラフが条件ベクトルで与えた特徴量に近いものになっていると言える．

7. おわりに

本稿では, GraphGen を conditional VAE で拡張することで, 特徴量を指定してグラフを生成する手法を提案した．

いくつかの実験を通じて, 提案手法の妥当性を検証した．学習データセットに含まれる特徴量のグラフであれば, その特徴量を指定して生成可能であることを検証した．

本稿での検証を踏まえた上で, 提案手法には, いくつかの展望が考えられる．提案手法では, 学習データセットに含まれない未知の特徴量のグラフであっても, 一部特徴量を指定して生成可能であった．よって今後, 本稿では用いなかった生成モデルによって生成したグラフもデータセットに含め, 学習を行うことで, 従来の統計的な生成モデルでは生成し得なかったグラフも特徴を指定して生成可能になるのではないかと考えれられる．またいくつかの実データを学習することで, 複数の実データを混合した特徴を有したグラフも生成可能ではないかと考えられる．

謝　　辞

本研究の一部は科研費 20H04172 の助成を受けたものである．

文　　献

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[2] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of 2nd International Conference on Learning Representations (ICLR 2014)*, Banff, Canada, 2014.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[5] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating Graphs via Random Walks. In *Proceedings of Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 609–618, Stockholm, Sweden, 2018.

[6] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of The Web Conference 2020*, pages 1253–1263, 2020.