

修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院 情報理工学研究科 情報・ネットワーク工学専攻 博士前期課程		
氏 名	CHEN HONG	学籍番号	1831097
論 文 題 目	エッジ特徴によるグラフ分散表現の改良		
<p>要 旨</p> <p>グラフは複雑なシステムを記述する強力なツールとして、ソーシャルネットワーク分析、生物情報科学、化学情報科学などの多くの分野で幅広く使用されている。例えば、分子の生物活性を予測するなどのパターン認識がグラフ分類問題に帰着できる。しかし、グラフに適用する数学ツールが少ないため、グラフ分類問題は伝統的な機械学習のベクトル分類問題より難しい。そこで、グラフデータを既存の機械学習アルゴリズムに適用するために、グラフデータを数値特徴ベクトルに変換する技術が注目されている。特に近年、ニューラルネットワークを用いたグラフの特徴ベクトル(分散表現とも呼ばれる)を学習する課題が盛んである。</p> <p>近年提案されたニューラルネットワークを用いて、グラフの分散表現を獲得する手法ではノード特徴のみを活用し、エッジ特徴を扱えない手法が多い。そこで、本研究では、エッジ特徴を用いてグラフの特徴ベクトルを改良することを目指す。提案手法では、まずライングラフを用いて、元のグラフのエッジをノードに変換する。この操作により、元のグラフのエッジ特徴がライングラフのノード特徴となるため、ノード特徴しか取り扱えない手法を利用して、元のグラフのエッジ特徴を反映したグラフ分散表現を獲得する。</p> <p>この方針に基づき、ノード特徴のみを使う既存の2つのグラフ分散表現学習手法を改良した。1つ目は Graph2vec である。Graph2vec はノードラベル付きのグラフ集合から、教師なしでグラフの分散表現を獲得する手法である。本論文では、まず従来手法 Graph2vec の(1)エッジラベル情報を扱えないことと、(2)グラフ間構造の類似性が適切に表現できないという欠点を指摘して、それを対処するため、ライングラフを利用して補う手法 GL2vec を提案した。</p> <p>2つ目は GIN(Graph Isomorphism Network)である。GIN はノード属性付きのグラフ集合から、教師ありでグラフの分散表現を獲得する。GIN はノード間で情報伝搬することによって、End-to-End で特徴ベクトル及び分類器を一気に学習させる。しかし、GIN もエッジの属性情報を使用していない。本論文では、ライングラフを用いて、GIN が生成するグラフの特徴ベクトルにエッジの特徴を補完できる手法 GLIN を提案した。</p> <p>実験によって、エッジ特徴とノード特徴両方とも考慮した提案法が従来法よりもグラフ分類性能を改善することを示した。上記の二つ改良例から、ノード属性しか扱えないグラフ分散表現学習のモデルの一族に対し、ライングラフはエッジ属性を活用できる有効なツールであることを確認できた。なお、エッジ特徴がグラフ分類タスクに役に立つことも確認できた。</p>			

令和元年度 修士論文

エッジ特徴によるグラフ分散表現の改良

電気通信大学大学院 情報理工学研究科
情報・ネットワーク工学専攻

1831097 陳宏 (CHEN HONG)

指導教員

古賀 久志 准教授

南 泰浩 教授

令和2年1月27日

目次

第1章 はじめに	1
1.1 研究背景	1
1.2 研究目的	3
1.3 本論文の構成	4
第2章 先行研究	5
2.1 順伝播型ニューラルネットワーク	5
2.1.1 構造	5
2.1.2 出力層の設計と誤差関数	6
2.1.3 パラメータの学習	7
2.2 Graph2vec	8
2.2.1 根付き部分グラフの抽出	10
2.2.2 分散表現の学習	12
2.3 GIN	13
2.3.1 Graph Neural Network	13
2.3.2 Graph Isomorphism Network	17
第3章 Graph2vec の改善	19
3.1 Graph2vec の欠点	19
3.2 提案手法:GL2vec	21
3.2.1 ライングラフ	21
3.2.2 GL2vec のフレームワーク	22
3.3 実験	24
3.3.1 グラフデータセット	24
3.3.2 実験の設定	27
3.3.3 実験結果	27




3.4 まとめ	28
第4章 GINの改善	30
4.1 GINの欠点	30
4.2 提案手法:GLIN	31
4.3 実験	32
4.3.1 グラフデータセット	32
4.3.2 実験の設定	33
4.3.3 実験結果	34
4.4 まとめ	35
第5章 関連研究	36
第6章 結論	37
謝辞	38
参考文献	39
図一覧	42
表一覧	43
業績リスト	44

第1章

はじめに

1.1 研究背景

グラフはノード (頂点) の集合とエッジ (辺) の集合で構成されるデータ構造である。ノードはオブジェクトを表し、エッジはオブジェクト間の関係を表すことができる。グラフは複雑なシステムを記述する強力なツールとして、ソーシャルネットワーク分析、生物情報科学、化学情報科学、コンピュータビジョン、自然言語処理などの多くの分野で幅広く使用されている。

例えば、図  のような既に実測された分子の構造情報と生物活性データがある。図  について、各分子の構造情報の下にある数字 “+1” と “-1” が各分子の乳癌細胞株に対する生物活性のクラスを表す。化学情報分野では、創薬の効率を向上するために、このような実測されたデータをコンピュータに学習させ、多種多様な未知の分子が持つ生物活性を予測することによって、有用な候補分子を発見する研究課題がある。この課題は分子をグラフとしてモデル化することで、分子が持つ生物活性の推定をグラフ分類問題に帰着することができる。例を図  に示す。ここで、ノードが原子を表し、エッジが化学結合を表す。なお、ノードは原子種類によってラベルつけされている。エッジは化学結合のタイプによってラベル (色) をつけられている。

グラフ分類の問題とは、クラスラベル付きのグラフの集合 $D = \{(G_1, y_1), (G_2, y_2), \dots, (G_N, y_N)\}$ が与えられたときに、ここで $y_i \in \mathcal{Y}$ はグラフ $G_i \in \mathcal{G}$ に対応するクラスのラベルであり、グラフ分類タスクの目的はグラフの空間からクラスの空間へのマッピング $\mathcal{F}: \mathcal{G} \rightarrow \mathcal{Y}$ を構築すること。

機械学習における伝統的なベクトル分類問題と比べると、グラフ分類問題はか

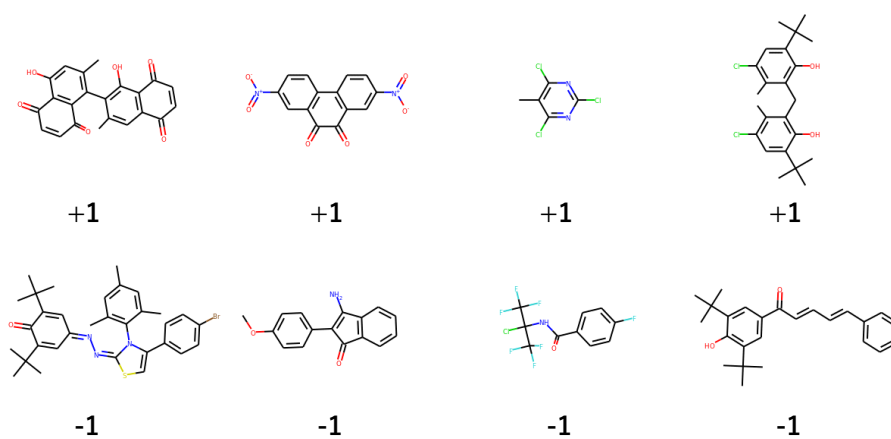


図 1-1: 分子構造の乳癌細胞株に対する生物活性

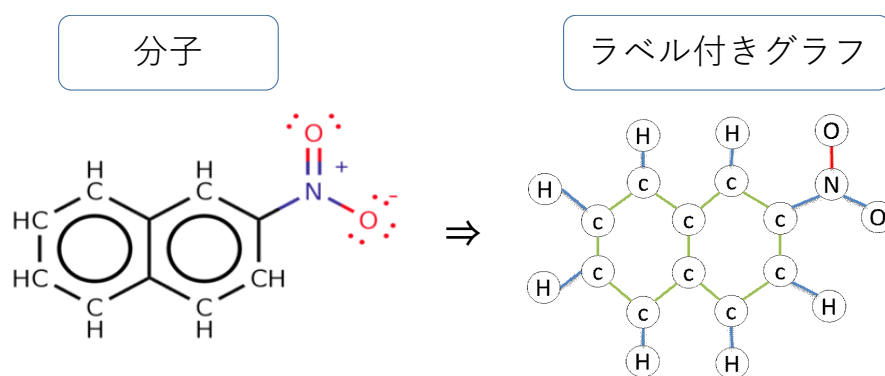


図 1-2: 分子のグラフ表現

なり難しい．以下の二つ原因がある．

- グラフが構造を持つため，ベクトルより複雑である
- グラフに適用する数学ツールが少ない

グラフデータを既存の機械学習アルゴリズムに適用するために，グラフデータを数値特徴ベクトルに変換する技術が注目されている．特に近年，ニューラルネットワークを用いたグラフの特徴ベクトルを学習アプローチが盛んに成っている．また，ニューラルネットワークで学習されたグラフの特徴ベクトルをグラフの分散表現とも呼ぶ．

1.2 研究目的

近年提案されたニューラルネットワークを用いて，グラフの分散表現を獲得する手法ではノード特徴のみを使用し，エッジ特徴を扱えない手法が多い．そこで，本研究では，エッジ特徴を用いてグラフの特徴ベクトルを改良することを目指す．提案手法では，まずライングラフを用いて，元のグラフのエッジをノードに変換する．この操作により，元のグラフのエッジ特徴がライングラフのノード特徴となるため，ノード特徴しか取り扱えない手法を利用して，元のグラフのエッジ特徴を反映したグラフ分散表現を提案手法で獲得する．2つノード特徴のみを使用する既存のグラフ分散表現学習手法を改良した．

1つ目は Graph2vec[3] である．Graph2vec はノードラベル付きのグラフ集合から，教師なしで，個々のグラフに対する特徴ベクトルを獲得する手法である．ここでの教師なしと言うのはグラフの特徴ベクトルを学習する時点で，分類タスク用のクラスラベルを使わないことである．そして，得られた特徴ベクトルをクラスラベルと一緒に SVM(サポートベクターマシン) などの分類器に学習させることによって，グラフ分類器を作るのに用いることができる．Graph2vec はノードの属性情報を利用できるだが，エッジ属性があっても扱えない．なお，抽出されたグラフの特徴ベクトルには構造情報が適切に反映できない場合がある．なので，本論文ではこの二つ課題に対処するため，元のグラフ G のエッジ双対グラフ LG (line graph)[4] を活用することを提案する．もとのグラフの特徴量で反映されないエッジ属性特徴と構造特徴をライングラフの特徴ベクトルで補完．実験により，提案法が従来法よりも，分類性能を向上できることを示す．

2つ目はGIN[4]である。GINはノード属性付きのグラフ集合から、教師ありで、個々のグラフに対する特徴ベクトルを獲得する手法である。ここでの教師ありと言うのはグラフの特徴ベクトルを学習する時点で、分類タスク用のクラスラベルを使うことである。GINはGNN(グラフニューラルネットワーク)[5]の一員である。End-to-End(特徴抽出器と分類器が一つのモデル内にある)のアーキテクチャで、グラフに対し、特徴ベクトルと分類器を一気に学習させる。しかし、GINもエッジの属性情報を使用していない。本論文では、ライングラフを用いて、GINで学習されたグラフの特徴ベクトルにエッジの特徴を補完することで、GINを改良する。実験によって、提案法が従来法よりも、分類性能を向上できることが可能であることを示す。

上記の2つ改良例から、ノード属性しか扱えないグラフ分散表現学習のモデルの一族に対し、ライングラフはエッジ属性を活用できる有効なツールであることを示す。

1.3 本論文の構成

以下に本論文の構成をまとめる。

第2章では、まず、ニューラルネットワークの基礎知識を述べる。その後、2つの特徴抽出手法 Graph2vec と GIN を説明する。

第3章では、まず、Graph2vec の欠点を指摘する。その後、ライングラフを用いた Graph2vec への拡張モデル GL2vec について説明する。最後、提案法の実験結果および考察について述べる。

第4章では、まず、GIN の欠点を指摘する。その後、ライングラフを用いた GIN への拡張モデル GLIN について説明する。最後、提案法の実験結果および考察について述べる。第5章では、ライングラフを活用した関連研究について述べる。第6章では結論および本研究の今後の課題についてまとめる。

第2章

先行研究

本章では、ニューラルネットワークを用いたグラフの特徴ベクトルを学習する先行研究について説明する。まず、2.1節では、ニューラルネットワークについて説明を行う。2.2節では、言語処理分野におけるニューラル言語モデル Doc2vec[8]に基づくグラフの特徴学習モデルである Graph2vec について説明を行う。また、2.3.2節では、グラフニューラルネットワークに所属する GIN について述べる。

2.1 順伝播型ニューラルネットワーク

本章では、順伝播型ニューラルネットワークについて説明する。

2.1.1 構造

順伝播型ネットワーク (feedforward neural network) とは、情報が入力側から出力側に一方向にのみ伝搬していくネットワークであり、多層パーセプトロン MLP (Multilayer perceptron) とも呼ばれる。構造を図 2.1 に示す。最初の層は入力層である。最後の層は出力層である。入力層と出力層間の層は隠れ層と呼ばれる。図中の丸がニューロンユニットを表し、各層はいくつかのニューロンで構成される。各ニューロンは入力を受け取ると活性化関数 $f(\cdot)$ により出力を決定し、次の層へと値を出力する。活性化関数として、式 2.1 で定義されるシグモイド関数 $\sigma(\cdot)$ と、式 2.2 で定義される ReLU 関数 $ReLU(\cdot)$ がよく用いられる。

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (2.1)$$

$$ReLU(x) = \max(0, x) \quad (2.2)$$

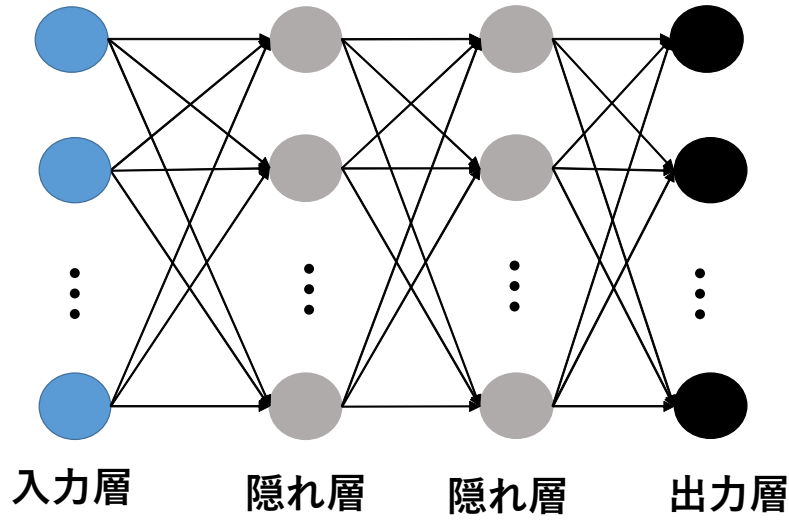


図 2-1: 二つの隠れ層を持つ順伝播型ニューラルネットワーク

入力層を0層で定義し，各層の入力を $u^{(l)}$ ，出力を $z^{(l)}$ で表すと，入出力は次のように計算される．

$$\begin{aligned} u^{(l)} &= W^{(l)} z^{(l-1)} + b^{(l)} \\ z^{(l)} &= f(u^{(l)}) \end{aligned}$$

ここで， $W^{(l)}$ は l 層と $l-1$ 層の間の結合重みを表し， l 層のニューロン数が $d^{(l)}$ のとき， $W^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$ である．また， $b^{(l)}$ は l 層のバイアス項を表し， $b^{(l)} \in \mathbb{R}^{d^{(l)}}$ である．

このように，ネットワークは与えられた入力 x に対して，入力層から出力層へと上の計算を繰り返し，MLP の全ての層のパラメータ W と b をまとめて θ で表すと，MLP はパラメータ θ もつ関数 $y = y(x; \theta)$ と表現することができる．MLP はパラメータを変えることで，様々な関数を表現することができる [22] ．

2.1.2 出力層の設計と誤差関数

与えられたデータの集合 $D = \{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$ に対し，MLP を用いて $\mathcal{F}: \mathcal{X} \rightarrow \mathcal{Y}$ のようなマッピングを構築するために，扱う問題の種類に応じて，誤差関数（損失関数とも呼ばれる）と出力層の活性化関数を適切に設定する必要がある．ここで，誤差関数はモデルの出力 y と正解 \hat{y} 間の差 $E(y, \hat{y})$ を測る．

回帰問題の場合，出力層の活性化関数は恒等関数となる．そして，誤差関数は式(2.3)のような二乗誤差を採用するのが一般的である．

$$E(\theta) = \frac{1}{N} \sum_{n=1}^N E^n(\theta) = \frac{1}{N} \sum_{n=1}^N \|\hat{y}^n - y(x^n; \theta)\|_2^2 \quad (2.3)$$

ここで， $E^n(\theta)$ は n 番目のデータに関する誤差値である．

分類問題の場合，入力に対応するクラスラベルを，one-hot ベクトルを用いて表現する．したがって，出力層の i 番目のニューロンの出力を下記のようなソフトマックス関数を用いて計算する．

$$y_i \equiv z_i^{(L)} = \frac{\exp(u_i^{(L)})}{\sum_{j=1}^{d^{(L)}} \exp(u_j^{(L)})} \quad (2.4)$$

ここで， $u_i^{(L)}$ は出力層 (L 番目の層) の i 番目のニューロンが隠れ層 $L-1$ から受取る入力である．

誤差関数は下記のような交差エントロピー誤差を使用するのが一般的である．

$$E(\theta) = \frac{1}{N} \sum_{n=1}^N E^n(\theta) = -\frac{1}{N} \sum_{n=1}^N \sum_{i=1}^{d^{(L)}} \hat{y}_i^n \log y_i(x^n; \theta) \quad (2.5)$$

ここで， \hat{y}_i^n は n 番目の学習データの正解ラベル \hat{y}^n における i 番目要素である．

2.1.3 パラメータの学習

誤差関数の値をなるべく小さくするようにネットワークの重みを調整する操作が学習と呼ばれる．パラメータの更新にはバッチ勾配降下法が用いられる．下記のような更新式を使うのが一般的である．

$$\theta \leftarrow \theta - \eta \frac{1}{N} \sum_{n=1}^N \frac{\partial E^n(\theta)}{\partial \theta} \quad (2.6)$$

ここで， η は学習率と呼ばれ，小さな正の値である． N はバッチ内のデータ数．

これからの式で使う符号 E は $E^n(\theta)$ を代表する．すなわち，1つの学習データに対するの誤差である．

さて，隠れ層における第 $l-1$ 層の i 番目のニューロンから第 l 層の j 番目のニューロンへの結合重み $w_{ij}^{(l)}$ に着目すると，誤差 E が $w_{ij}^{(l)}$ に関する偏微分は下記の式で

算出できる．

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^{(l)}} &= \frac{\partial u_j^{(l)}}{\partial w_{ij}^{(l)}} \times \frac{\partial z_j^{(l)}}{\partial u_j^{(l)}} \times \frac{\partial E}{\partial z_j^{(l)}} \\ &= z_i^{(l-1)} \times \frac{\partial z_j^{(l)}}{\partial u_j^{(l)}} \times \frac{\partial E}{\partial z_j^{(l)}} \\ &= z_i^{(l-1)} \times f'(u_j^{(l)}) \times \frac{\partial E}{\partial z_j^{(l)}}\end{aligned}$$

ここで， $f'(\cdot)$ は活性化関数 f の導関数である．例えば，シグモイド関数 $\sigma(\cdot)$ の導関数は $\sigma(\cdot) \times (1 - \sigma(\cdot))$ である．そして， $\frac{\partial E}{\partial z_j^{(l)}}$ の値が下記の式で算出できる．

$$\begin{aligned}\frac{\partial E}{\partial z_j^{(l)}} &= \sum_q^{d^{(l+1)}} \frac{\partial u_q^{(l+1)}}{\partial z_j^{(l)}} \times \frac{\partial z_j^{(l+1)}}{\partial u_q^{(l+1)}} \times \frac{\partial E}{\partial z_q^{(l+1)}} \\ &= \sum_q^{d^{(l+1)}} w_{jq} \times f'(u_q^{(l+1)}) \times \frac{\partial E}{\partial z_q^{(l+1)}}\end{aligned}$$

ここで， $d^{(l+1)}$ は $l+1$ 層のニューロン数である．上記の式から見ると，出力層における各ニューロンに関する偏微分値があれば，出力層から入力層まですべての偏微分値が算出できる．例えば，誤算関数が式 2.3 で定義された二乗誤差の場合，誤差 E が出力層における k 番目のニューロンに関する偏微分 $\frac{\partial E}{\partial z_k^{(L)}}$ を下記の式で算出できる．

$$\frac{\partial E}{\partial z_k^{(L)}} = -2(\hat{y}_k - z_k^{(L)}) \quad (2.7)$$

ここで， \hat{y}_k は正解データの k 番目の要素である．

以上の計算流れが，誤差逆伝搬法 [23] と呼ばれる．また，ネットワークの構造がさらに複雑になると，上記のような手動微分 (Manual Differentiation) は不現実である．なので，pytorch[27] などの深層学習向けのライブラリでは，ネットワークの計算過程を計算グラフ (computational graph) に登録し，計算グラフを基に自動微分 (Automatic Differentiation) アルゴリズムを使うことが一般的である．

2.2 Graph2vec

本節では Graph2vec を説明する．Graph2vec はノードラベル付きグラフを対象にする．ここで，ノードラベル付きグラフが $\{V, E, \lambda\}$ で表す． V はノードの集合

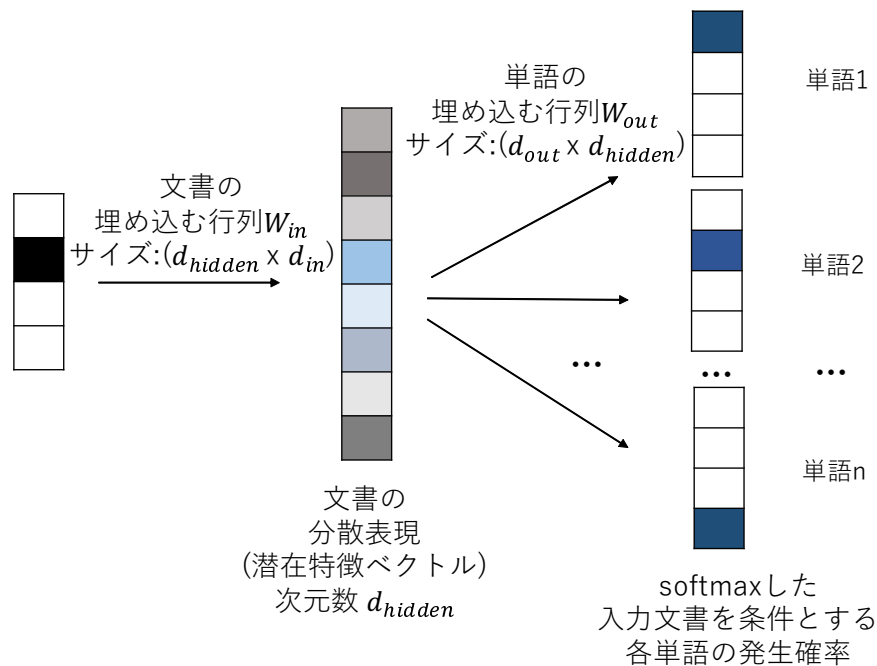


図 2-2: DBOW の仕組み

であり, $E \subset (V \times V)$ はエッジの集合である. λ は, 関数 $\lambda: V \rightarrow \mathcal{L}$ で, アルファベット \mathcal{L} からすべてのノード $v \in V$ に一意のラベルを割り当てる.

グラフの集合 $\{G_1, G_2, \dots, G_N\}$ と正整数 δ が与えられて, Graph2vec は $\{G_1, G_2, \dots, G_N\}$ から特徴ベクトルの集合 $\{\vec{G}_1, \vec{G}_2, \dots, \vec{G}_N\}$ への写像を学習する. ここで, G_i の特徴ベクトルである \vec{G}_i の次元数は δ である.

Graph2vec は自然言語処理における文章の特徴ベクトルを学習するモデルである Doc2vec[8] を基にしている. Doc2vec ではドキュメント (文書) を単語集合として表し, それを入力として, ニューラル言語モデル skip-gram[4] を拡張した言語モデル DBOW[8] を用いて文書の分散表現を学習する. DBOW の仕組みは図 2-2 に示す. 文書中に出現する単語を予測するよう学習し, 入力層と中間層の間の重み (文書の埋め込み行列) を更新する. また, 文章の埋め込み行列における各列が各文書の分散表現と見なす. この様に, 可変長の文書を固定長の特徴ベクトルに変換する.

Doc2vec から獲得した文章の特徴ベクトルが重要な性質がある. それは 2 つ文書が意味的に類似していれば, 特徴ベクトル空間において近い位置にマッピングされることである. また, 文書間意味的に類似とは, 共通の単語が多いというこ

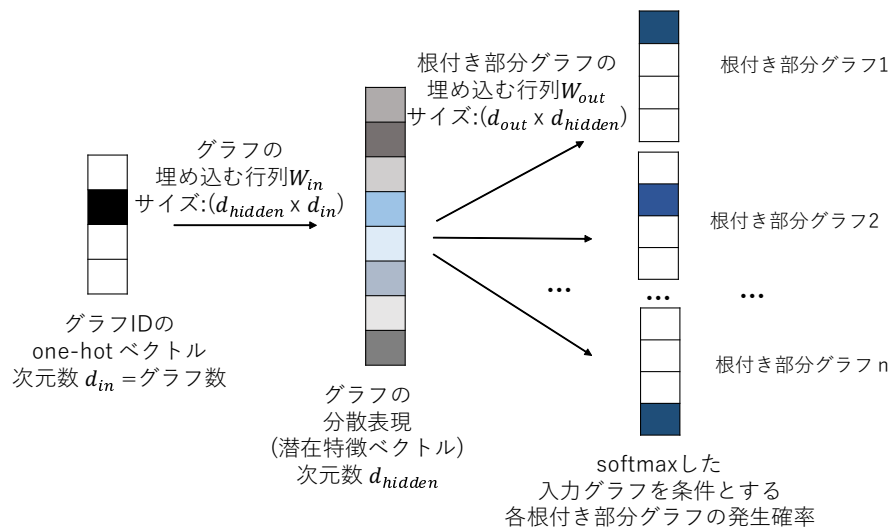


図 2-3: グラフに適用した DBOW の仕組み

とである。

Graph2vec では、1 つのグラフを局所的な根付き部分グラフの集合として表す。各グラフを根付き部分グラフの集合で表現した後、グラフを文書とし、根付き部分グラフを単語として、Doc2vec で使われた言語モデル DBOW に入力、各グラフの特徴ベクトルを学習する。グラフに適用した DBOW が図 2-3 で示すようになる。

特に、Graph2vec で学習された特徴ベクトルも以下の性質がある。 G_i と G_j が意味的に類似していれば、 \vec{G}_i と \vec{G}_j も特徴ベクトル空間内において近い。グラフ同士間意味的に類似とは、共通の根付き部分グラフが多いということである。

本節残りの部分は次のように構成されている。第 2.2.1 節では、グラフから根付き部分グラフ集合への作り方を述べる。第 2.2.2 節では、構築した根付き部分グラフの集合からグラフの特徴ベクトルを学習する方法について説明する。

2.2.1 根付き部分グラフの抽出

Graph2vec では、まず個々のグラフ G を根付き部分グラフの集合 $c(G)$ と表現する。事前に根付き部分グラフの最大深さ H を設定し、グラフ内のすべてのノード v に対し、各 v を根とする $(H+1)$ 個の根付き部分グラフを生成して、式 2.8 のような $n(H+1)$ 個根付き部分グラフから構成された集合 $c(G)$ を作る。ここで、 $sg_i^{(t)}$ は i 番目のノード v_i を根とする、深さ t の根付き部分グラフを表す。 n はグラフ内の

ノード数である．

$$c(G) = \{sg_1^{(0)}, sg_2^{(0)}, \dots, sg_n^{(0)}, sg_1^{(1)}, sg_2^{(1)}, \dots, sg_n^{(1)}, \dots, sg_1^{(H)}, sg_2^{(H)}, \dots, sg_n^{(H)}\} \quad (2.8)$$

上記の各根付き部分グラフが WL(Weisfeiler-Lehman) 再ラベル戦略 [6] で識別 ID にマッピングされる．

以下に、すべてのノードに対し、深さ t の根付き部分グラフの生成と識別 ID への量子化するための t 回目での WL 再ラベル操作の手順を説明する．以下のアルゴリズムでは、 $\lambda^t(v)$ が t 回操作後ノード v のラベルであり、 v を根とする深さ t の根付き部分グラフ $sg_v^{(t)}$ の識別 ID と表す．

W-L(Weisfeiler-Lehman) 再ラベル操作が下記の四つのステップで構成される．

1. $\forall v \in V$, ノード v に隣接する近傍ノードのラベルを集めて、多重集合 $M^t(v) = \{\lambda^{t-1}(u) | u \in Neighbors(v)\}$
2. 多重集合 $M^t(v)$ の要素を昇順でソートした文字列 $S^t(v)$ を作る．その後、 $S^t(v)$ の先頭に根ノードのラベル $\lambda^t(v)$ を加える．
3. 全単射能力を持つハッシュ関数を用いて $S^t(v)$ を識別 ID にマッピングする． $Hash(S^t(v)) = Hash(S^t(w)) \text{ iff } S^t(v) = S^t(w)$ ．このハッシュ関数は異なる文字列を異なる識別 ID に写像することが要求されるが、基数ソートを使用すれば簡単に実装できる．
4. 識別 ID を v の新しいラベルとする． $\lambda^t(v) = Hash(S^t(v))$

ここで、ステップ2では、 v の隣接ノードの深さ $t-1$ の根付き部分グラフと v 自身の深さ $t-1$ の根付き部分グラフから、 v の深さ t の根付き部分グラフ $sg_v^{(t)}$ を合成し、ステップ2で根付き部分グラフを識別 ID である $\lambda^t(v)$ へ量子化している．再ラベル操作の実行例を図 2-4 に示す．例えば、 v_5 を根とする部分グラフ $sg_{v_5}^{(1)}$ が “2-1,1,2” で表し、ここで、先頭の “2” は根ノード v_5 のラベル、“1,1,2” は v_5 に隣接するノード v_1, v_3, v_4 のラベル．そして、 $sg_{v_5}^{(1)}$ を代表する “2-1,1,2” が識別 ID “6” にマッピングする．このような一回目操作後、グラフ G から抽出された根付き部分グラフの集合 $c(G) = \{sg_{v_1}^{(0)}, sg_{v_2}^{(0)}, sg_{v_3}^{(0)}, sg_{v_4}^{(0)}, sg_{v_5}^{(0)}, sg_{v_1}^{(1)}, sg_{v_2}^{(1)}, sg_{v_3}^{(1)}, sg_{v_4}^{(1)}, sg_{v_5}^{(1)}\}$ は識別 ID の多重集合 $c(G) = \{1, 1, 1, 2, 2, 4, 5, 6, 5, 6\}$ に代表される．また、 H 回繰り返すの操作により、深さ H までの $n(H+1)$ 個の根付き部分グラフの集合 (識別 ID の多重集合) が作れる．

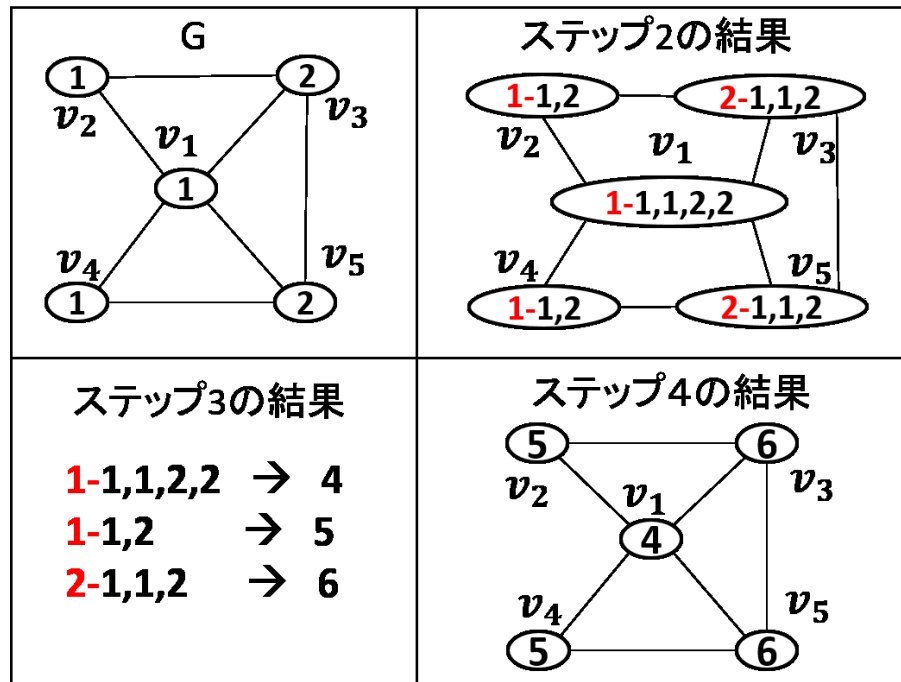


図 2-4: グラフ G に対する 1 回 W-L 再ラベル操作

2.2.2 分散表現の学習

すべてのグラフから根付き部分グラフを抽出した後, Graph2vec は DBOW モデルを用いて, グラフの特徴ベクトルを学習する. DBOW は自然言語処理における文書の特徴ベクトルを抽出する言語モデルであり, 一つ隠れ層を持つ順伝搬ニューラルネットワークで構築される. 入力層では, 各グラフの ID を one-hot ベクトルの形で表現し, モデルに入力する. 出力層では, 根付き部分グラフがグラフから抽出されたと言う条件を基にして, 各根付き部分グラフの条件付き確率分布を出力する. 学習により, 中間層から各グラフの特徴ベクトル獲得できる.

グラフの集合 $\{G_1, G_2, \dots, G_N\}$ と対応する部分グラフの集合 $S = \{c(G_1), c(G_2), \dots, c(G_N)\}$ が与えられたときに, Graph2vec は次元数 δ の各グラフ G_i の特徴ベクトル \vec{G}_i と, $(1 \leq i \leq n_i(H+1))$, S の要素である各部分グラフの特徴ベクトル \vec{sg}_j を学習する. 特に, Graph2vec には, \vec{sg}_j が $c(G_i)$ からサンプリングされたことに基づき, 式 2.9 のような対数尤度関数の最大化を目指し, ネットワークのパラメータを学習する.

$$\sum_{j=1}^{n_i(H+1)} \log P_r(sg_j|G_i) \quad (2.9)$$

ここで, n_i は G_i のノード数, また, 確率 $\log P_r(sg_j|G_i)$ は式 2.10 のように定義される.

$$\frac{\exp(\vec{G}_i \cdot s\vec{g}_j)}{\sum_{sg \in V_{oc}} \exp(\vec{G}_i \cdot s\vec{g})} \quad (2.10)$$

ここで, V_{oc} はすべてのグラフに渡るすべての根付き部分グラフの集合である. また, グラフの特徴ベクトル \vec{G}_i が式 2.11 で定義される.

$$\vec{G}_i = W_{in} \cdot \text{onehot}(G_i) \quad (2.11)$$

つまり, \vec{G}_i は W_{in} の i 列目である. また, 根付き部分グラフの特徴ベクトル $s\vec{g}_j$ が W_{out} の j 行目である. ここで, W_{in} と W_{out} は図 2.3 に示す通り, それぞれが入力層と中間層, 中間層と出力層の間の重みであり, グラフと根付き部分グラフの埋め込み (特徴ベクトル) 行列と見なす. 学習によって, 特徴ベクトルを更新する.

また, 学習完了後, 多く共通の根付き部分グラフを持つグラフ同士が, 潜在特徴ベクトル空間内に近い, つまり, 類似な特徴ベクトルへマッピングされる.

2.3 GIN

本節では GIN(Graph Isomorphism Network)[4] を説明する. GIN はノード属性付きのグラフ集合から, 教師ありで, 個々のグラフに対する特徴ベクトルを獲得する手法である. ここでの教師ありと言うのはグラフの特徴ベクトルを学習する時点で, 分類タスク用のクラスラベルを使うことである. GIN は GNN(Graph Neural Network)[5] の一員である. End-to-End(特徴抽出器と分類器が一つのモデル内にある)のアーキテクチャでグラフに対し, 特徴ベクトルの抽出から分類まで一気に学習させる.

本節残りの部分は次のように構成されている. 第 2.3.1 節では, GIN が所属する GNN モデルの概念を説明する. 第 2.3.2 節では, GIN の仕組みを述べる.

2.3.1 Graph Neural Network

グラフ $G = (V, E)$ における, 各ノード $v \in V$ が初期特徴ベクトル x_v を持つ, これらの初期特徴はノードの属性を表す. GNN の目標は, グラフのノード属性情報

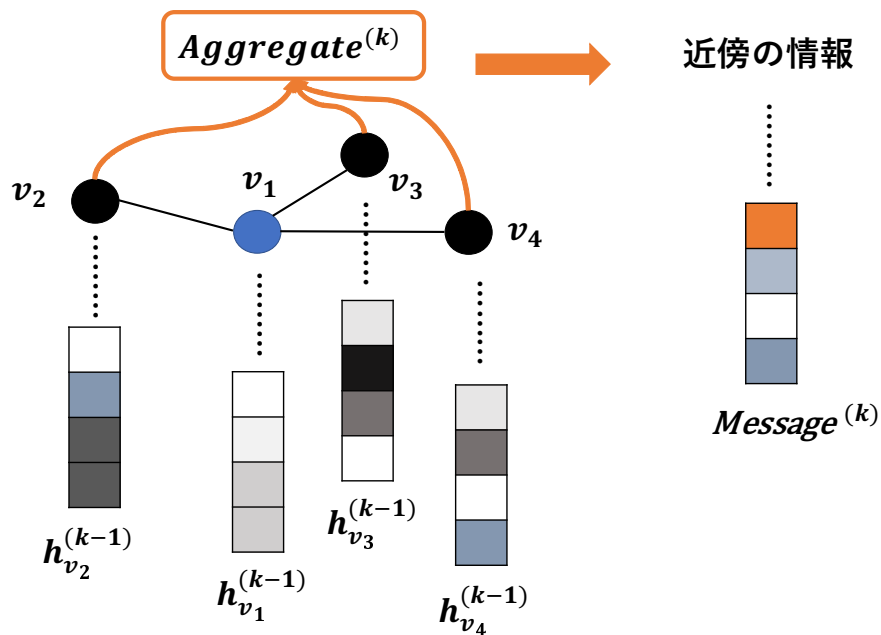


図 2-5: 近傍の情報を集約

とグラフの構造情報（隣接行列など）からノードの潜在特徴ベクトル h_v およびグラフの潜在特徴ベクトル h_G を学習する．ノードの潜在特徴ベクトルはノード分類，コミュニティ検出，およびリンク推定などのノードレベルのタスクで使われる．一方，グラフの潜在特徴ベクトルがグラフ分類とグラフクラスタリングなどのタスクで使われる．

近年，様々な GNN が提案された，代表的な手法としては GCN[9]，GraphSAGE[10]，GIN[4] 等がある．これらの共通点はグラフにおけるノード情報の伝搬によるノードの潜在特徴ベクトルを更新することである．

k 回目の情報伝搬の手順について説明する．具体的に，まず各ノードを根として，近傍ノードの特徴ベクトルを集約する．例を図 2-5 に示す．ここで，青いノード v_1 を根ノードとする場合には， v_1 と隣接している近傍ノード群 $\{v_2, v_3, v_4\}$ の特徴ベクトル $\{h_{v_2}^{(k-1)}, h_{v_3}^{(k-1)}, h_{v_4}^{(k-1)}\}$ を集約関数 $Aggregate^{(k)}$ に入力して，近傍情報を代表するベクトル $Message^{(k)}$ を生成する．次では，近傍情報を根ノードの特徴を結合し，根ノードの潜在特徴ベクトルを更新すること．例を図 2-6 に示す．ここで，得られた近傍の情報 $Message^{(k)}$ と根ノード v_1 の現在の特徴ベクトル $h_{v_1}^{(k-1)}$ を結合関数 $Combine^{(k)}$ に入力して，出力を v_1 の新しい特徴ベクトル $h_{v_1}^{(k)}$ と見なして， v_1 の状態を更新する．このような情報伝搬の操作を繰り返すことによって，

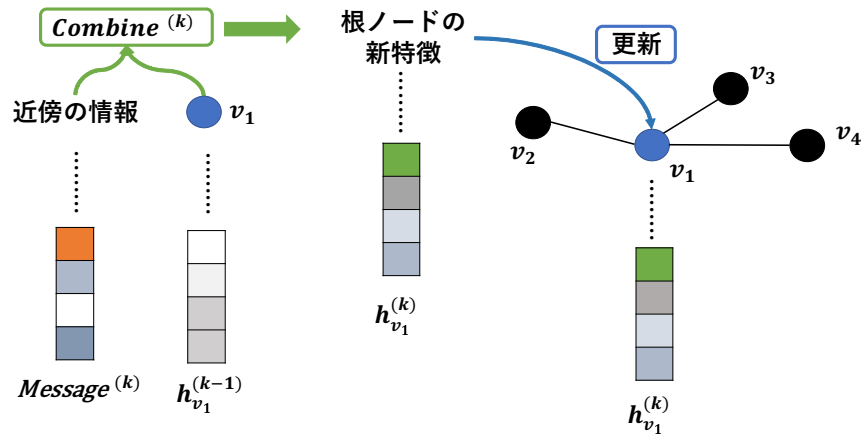


図 2-6: 近傍情報と根ノードの現在の特徴ベクトルを結合し，根ノードの特徴量を更新

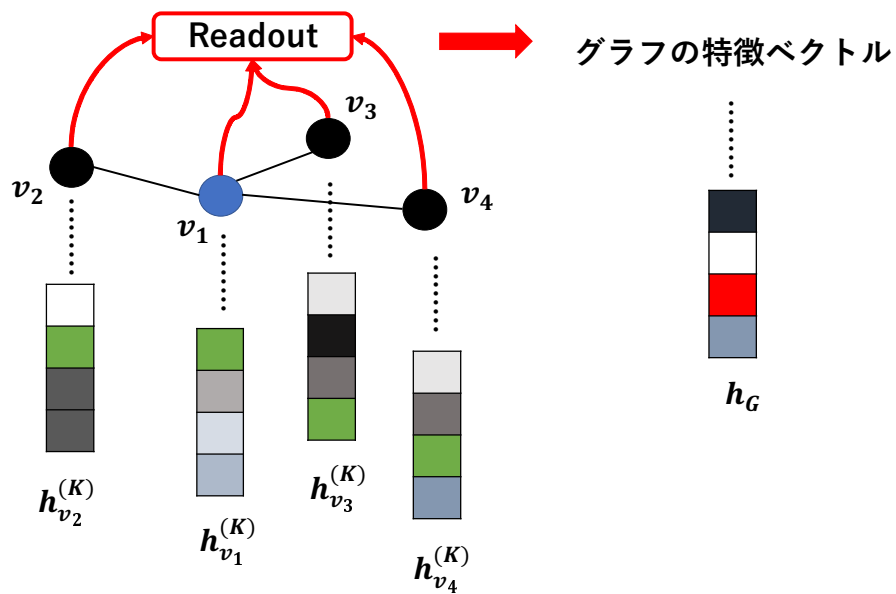


図 2-7: 全ノードの特徴ベクトルからグラフの特徴ベクトルを生成

より広い範囲の周辺ノードまで考慮した潜在特徴ベクトルが得られる．GNNのレイヤー数は潜在特徴ベクトルの更新回数と等しい． k レイヤーの式が下記のようになる．

$$Message_v^{(k)} = Aggregate^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\}) \quad (2.12)$$

$$h_v^{(k)} = Combine^{(k)}(h_v^{(k-1)}, Message_v^{(k)}) \quad (2.13)$$

ここで、 $h_v^{(k)}$ は k レイヤーにてノード v の潜在特徴ベクトルである．また、レイヤー0での特徴量 $h_v^{(0)}$ がノードの初期特徴 x_v である．そして、 $\mathcal{N}(v)$ は根ノード v に隣接している近傍ノードの集合である．なお、集約関数 $Aggregate^{(k)}(\cdot)$ と結合/更新関数 $Combine^{(k)}(\cdot)$ の設計がGNNにとって非常に肝心なことである．GraphSAGEの場合集約関数は式(2.14)のように定義される．

$$Message_v^{(k)} = MAX(\{ReLU(W \cdot h_u^{(k-1)}) : u \in \mathcal{N}(v)\}) \quad (2.14)$$

ここで、 W は学習可能なニューラルネットワークのパラメーター、ここで、 MAX 関数は複数のベクトルを入力として、次元ごとの最大値を取ったベクトルを出力する．また、結合/更新関数は式(2.15)の通りである．

$$h_v^{(k)} = W \cdot [h_v^{(k-1)}, Message_v^{(k)}] \quad (2.15)$$

ここで、 $[\cdot]$ は結合(concatenation)の操作である．

GCNの場合集約関数と結合関数を合わせて式(2.16)の通りである．

$$h_v^{(k)} = ReLU(W \cdot MEAN\{h_u^{(k-1)} : u \in \mathcal{N} \cup \{v\}\}) \quad (2.16)$$

ここで、 $MEAN$ 関数は複数のベクトルを入力として、次元ごとの平均値を取ったベクトルを出力する．

ラストレイヤー K で生成したノードの潜在特徴ベクトル $h_v^{(K)}$ はノードレベルのタスクに適用できる．なお、グラフレベルのタスクに適用するために、グラフの潜在特徴ベクトル h_G を生成する必要がある．その時、ノードの特徴からグラフの特徴への読み出す関数 $Readout(\cdot)$ を使用する．例を図(2.7)に示す．ここで、得られた全ノードの特徴ベクトル $\{h_{v_1}^{(K)}, h_{v_2}^{(K)}, h_{v_3}^{(K)}, h_{v_4}^{(K)}\}$ を読み出す関数 $Readout(\cdot)$ に入力して、グラフの特徴ベクトル h_G 出力する．

$$h_G = Readout(\{h_v^{(K)} | v \in G\}) \quad (2.17)$$

ここで、読み出す関数 $Readout(\cdot)$ は足し算など簡単な置換不変(permutation invariant)の操作がよく使われる．それ以外、もっと複雑な操作[11][12]も提案された．

2.3.2 Graph Isomorphism Network

GIN には、2.3.1 節で説明した GCN, GraphSAGE より、グラフに対する識別能力が高い GNN モデルと知られている。これから、GIN の仕組みを説明する。GIN も情報伝搬フレームワークに基づいてノードの特徴ベクトルを更新する。GIN は GNN と W-L 同型テスト [6] の類似性に着目して提案された。W-L 同型テストについて説明する。これは、二つのグラフ G_1 と G_2 が同型であるか判定するテストである。W-L 同型テスト操作の手順が下記のようにまとめる。

1. 各ノードを根として、根ノードと近傍ノードのラベルを集める。
2. ラベルの多重集合を全単射ハッシュ関数で識別 ID へマッピングする。
3. 生成した識別 ID を根ノードの新しいラベルと見なす。
4. 二つグラフにて、ノードラベルが不一致の場合、不同型と判定する。

設定した回数までに、1 から 4 までの操作を繰り返し、不同型と判定されなかった場合は同型と判定する。

GNN と W-L テストは下記のような類似性がある。

- 近傍情報を集約する
- 近傍情報に識別する
- 近傍情報と根ノード情報を結合することにより、根ノードの状態を更新する

グラフに対し、GNN が W-L テストと同じ識別能力になるために、以下の条件を満たさなければならない。

条件: 近傍集約関数 $Aggregate^{(k)}(\cdot)$ 、結合/更新関数 $Combine^{(k)}(\cdot)$ 、および読み出す関数 $Readout(\cdot)$ がそれぞれ多重集合に対する全単射能力を持つ操作のことが要求される。

GIN では、理論により、前述の条件を満たす式 2.18 と式 2.19 を使用する。

$$h_v^{(k)} = MLP^k((1 + \epsilon^{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}) \quad (2.18)$$

$$h_G = CONCAT(SUM(\{h_v^{(K)} | v \in G\}) | k = 0, 1, \dots, K) \quad (2.19)$$

ここで、式 2.18 について、MLP は多層パーセプトロンであり、学習によって、異なるノード特徴ベクトルの多重集合を異なる潜在特徴表現にマッピングすることができる。また、 ϵ も学習可能なパラメータである。そして、式 2.19 での CONCAT は結合 (concatenation) 操作である。このような 0 レイヤーから K レイヤーまで特徴を CONCAT する操作により、グラフに対し、W-L グラフカーネル [6] という手法と同じレベルのグラフ識別能力を持つ。

ネットワークのパラメータを学習するときには、読み出すステップで出力したグラフの潜在特徴ベクトル h_G を 2 層の全結合ニューラルネットワーク (分類器) に入力し、分類の誤差信号の逆伝搬により、ネットワークのパラメータを更新する。このような特徴ベクトルの生成モデルと分類モデルと一緒に学習させるの仕組みが end-to-end モデルと言われる。

第3章

Graph2vecの改善

[2.2](#) 節で述べた Graph2vec はノードラベル付きグラフを対象に，抽出された根付き部分グラフのコーパスにより，言語モデル DBOW を用いて，グラフの特徴ベクトルを学習する．そのようなアプローチが実に2つの欠点がある．本章では，Graph2vec の欠点を改善する手法 GL2vec を提案する．

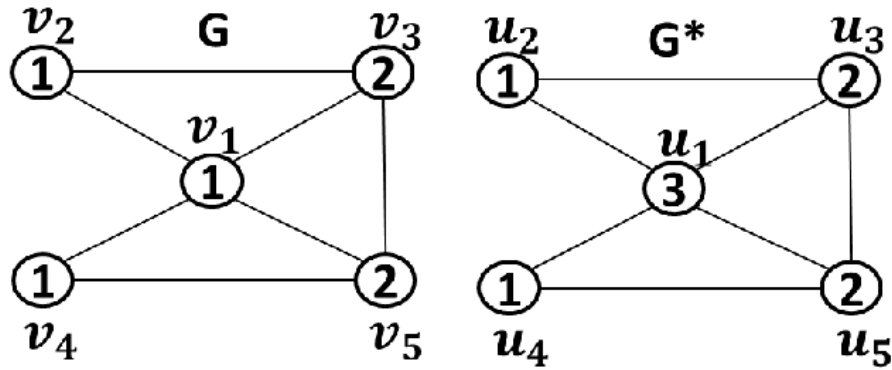
まず，[3.1](#) 節で，Graph2vec の2つの欠点を指摘する．そして，[3.2](#) 節で，その課題を克服する手法を提案する．[3.3](#) 節は実験．[3.4](#) 節はまとめ．

3.1 Graph2vec の欠点

本節で，Graph2vec が持つ2つの欠点を述べる．

1つ目の欠点はエッジラベルを扱えないということ．たとえ分類したいグラフデータにはエッジラベルが付与されていても，Graph2vec はそれを利用できない．なぜかと言うと，[2.2.1](#) 節で説明した通り，根付き部分グラフを抽出する時点で，Graph2vec は W-L 再ラベル戦略を基にしている．ノードを対象に，ラベルの集約と更新を行う．その結果，エッジの属性情報があっても，完全に捨てられてしまう．つまり，作った根付き部分グラフのコーパスがエッジ情報に反映されない．そのようなコーパスを用いて学習したグラフの特徴ベクトルもエッジの属性情報が含まれない．しかし，エッジの属性情報（ノード間の関係を表す情報）がかなり重要な情報である．それを活用できれば，エッジ属性が役に立つ分類などのタスクでさらに一步より高い認識精度の向上が期待される．

2つ目の欠点は抽出した根付き部分グラフを識別 ID へマッピングする際に，ノードラベルとグラフ構造を同時に畳み込むため，構造の類似性を適切に表現できな

図 3-1: 形状が同一で中央ノードのラベルだけが異なる 2 つのグラフ G, G^*

$c(G)$	$sg_{v_1^{(0)}}$	$sg_{v_2^{(0)}}$	$sg_{v_3^{(0)}}$	$sg_{v_4^{(0)}}$	$sg_{v_5^{(0)}}$	$sg_{v_1^{(1)}}$	$sg_{v_2^{(1)}}$	$sg_{v_3^{(1)}}$	$sg_{v_4^{(1)}}$	$sg_{v_5^{(1)}}$
	1	1	2	1	2	4 (1-1,1,2,2)	5 (1-1,2)	6 (2-1,1,2)	5 (1-1,2)	6 (2-1,1,2)
$c(G^*)$	$sg_{u_1^{(0)}}$	$sg_{u_2^{(0)}}$	$sg_{u_3^{(0)}}$	$sg_{u_4^{(0)}}$	$sg_{u_5^{(0)}}$	$sg_{u_1^{(1)}}$	$sg_{u_2^{(1)}}$	$sg_{u_3^{(1)}}$	$sg_{u_4^{(1)}}$	$sg_{u_5^{(1)}}$
	3	1	2	1	2	9 (3-1,1,2,2)	7 (1-2,3)	8 (2-1,2,3)	7 (1-2,3)	8 (2-1,2,3)

図 3-2: 深さ $H = 1$ までの根付きの部分グラフの集合 (識別 ID の多重集合) $c(G)$ と $c(G^*)$

い場合があることである．マッピングされた根付き部分グラフの集合 (識別 ID の多重集合) がグラフ間の構造的な類似性も適切に反映できない．一般に，ノードラベル付きグラフの類似性は (1) ノードラベルの類似性と (2) 構造つまりグラフ形状の類似性の両者で決定される．さて，Graph2vec における根付き部分グラフを識別 ID へのマッピングはノードラベルとグラフの構造を同時に畳み込んでいる．しかし，この方式ではノードラベルが一致している条件でのみ構造の同一性を評価するので，二つのグラフの構造が類似していることが判別できない．この問題はかなり複雑なので，これから，図 3-1 に示す二つグラフの例を使って説明する．

形状が同一で中央ノードのラベルだけが異なる 2 つのグラフ G, G^* を対象に，深さ $H = 1$ までの根付きの部分グラフの集合 (識別 ID の多重集合) $c(G)$ と $c(G^*)$ を作ってみる．結果を図 3-2 に示す．この二つ識別 ID の多重集合 $c(G)$ と $c(G^*)$ は半分以上の要素が異なる．例えば，グラフ G における v_5 を根とする部分グラフ $sg_{v_5}^{(1)}$ が “2-1,1,2” で，マッピングされた識別 ID が “6” である．一方，グラフ G^* におけ

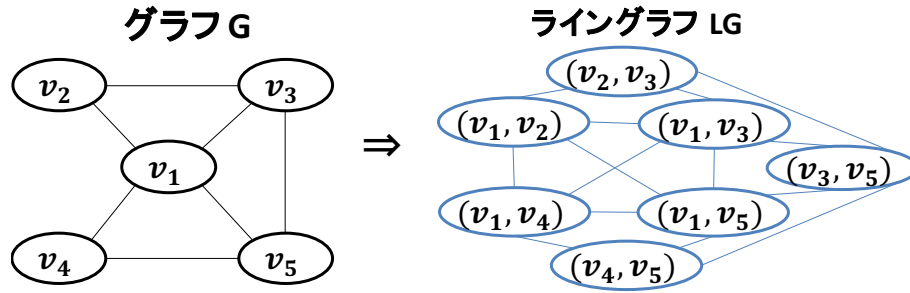


図 3-3: グラフ G と対応するライングラフ LG

る u_5 を根とする部分グラフ $sg_{u_5}^{(1)}$ が “2-1,2,3” で，マッピングされた識別 ID が “8” である．形が同じだが，識別 ID が異なる． $c(G)$ と $c(G^*)$ から「G と G^* はノードラベルが一致するノード $[1,1,2,2]$ を 4 つ持つ」ことしか判別できない．G と G^* の形状が同一であることはおろか，ノードラベルが一致するノードの次数が一緒であることすら判別困難である．これは，形状が同じ根付き部分グラフでもノードラベルが 1 つでも違えば異なる識別 ID になり，形状が同一であるという情報は捨てられるためである．

前述した Graph2vec の 2 つの欠点への改良手法が次の節で述べる．

3.2 提案手法:GL2vec

本節では，[3.1](#) 節で指摘した Graph2vec の 2 つの欠点を改善するために，ライングラフ [\[4\]](#) を用いた Graph2vec への拡張するフレームワークである GL2vec を提案する．[3.2.2](#) 節での提案手法 GL2vec を説明する前に，まず [3.2.1](#) 節でライングラフについて説明を行う．

3.2.1 ライングラフ

グラフ $G = (V, E)$ を与えられて，対応するライングラフ $LG = (LV, LE)$ とは， G におけるエッジの隣接関係を表すグラフである．エッジからノードへの双対グラフとも呼ばれる．ライングラフの定義として，まず，式 [3.1](#) で， LG の各ノードが G のエッジから変換される．

$$LV = \{v(e) | e \in E\} \quad (3.1)$$

また，エッジの集合 LE が式 3.2 のルールで構築される．

$$LE = \{(v(e_i), v(e_j)) | e_i \text{ と } e_j \text{ が } G \text{ において端点を共有する} \} \quad (3.2)$$

例を図 3.3 に示す．ここで，例えば，グラフ G におけるエッジ (v_1, v_2) とエッジ (v_1, v_5) は端点 v_1 を共有している．この場合，対応するライングラフ LG におけるノード (v_1, v_2) とノード (v_1, v_5) も連結している．また，グラフ理論では，ノード v と隣接しているノードの数がノード v の次数と定義され， $\deg(v)$ で記述される．同様に，エッジ e と隣接しているエッジの数もエッジ e の次数で定義され， $\deg(e)$ で表す．グラフにおけるエッジ次数とノード次数との関係が式 3.3 に示す．

$$\deg(e) = \deg(v_a) + \deg(v_b) - 2 \quad (3.3)$$

ここで，ノード v_a とノード v_b はエッジ e の二つの端点である．

また，ライングラフにおけるノード $v(e)$ の次数はグラフにおけるエッジ e の次数と同じである．つまり， $\deg(v(e)) = \deg(e)$ ．

一つ説明必要なのは，グラフがライングラフに変換するした後，元のグラフに対し，どれぐらいの構造情報に残るのか? 答えとしては，ほぼ全部の構造情報がそのまま反映できる．なぜかと言うと，Whitney graph isomorphism theorem[2] により，二つ連結グラフが同型であれば，対応するライングラフも同型はずである．ただし，一つだけの例外がある: 完全グラフ K_3 と完全二部グラフ $K_{1,3}$ のライングラフは同じで K_3 である．

3.2.2 GL2vec のフレームワーク

3.1 節で指摘した Graph2vec で対処するのが困難なグラフ G の (1) エッジラベルと (2) 構造情報を, G に対するライングラフ LG を利用して補う手法を提案する．提案手法を GL2vec(graph and line to vector) と呼ぶ．本論文でライングラフに着目した理由は以下の 2 つである．

- G のエッジ e が LG のノード $v(e)$ に変換されるので, G のエッジラベルを LG のノードラベルとして扱える．
- LG は G のノードラベル情報を持たないので, 構造の類似性をノードラベルとは独立に評価するのに適している．

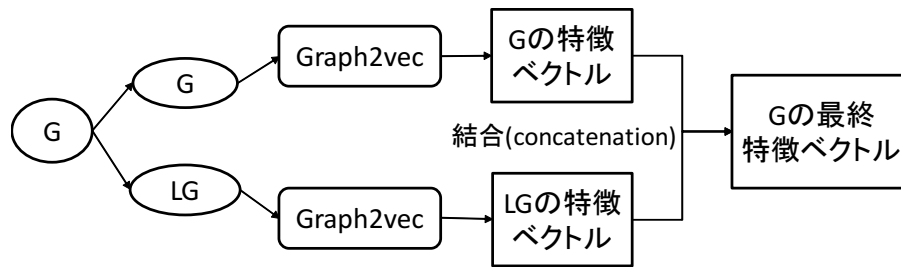


図 3-4: 提案手法 GL2vec のフレームワーク

GL2vec では、元のグラフ G のエッジラベルと構造情報の片方をユーザが選択し、 LG の特徴ベクトルに反映する。典型的には、 G がエッジラベルを持つ場合、 G のエッジラベルを LG のノードラベルとして与え、 G のエッジラベルに基づいた LG の特徴ベクトルを獲得する。 G がエッジラベルを持たない場合は、ライングラフ LG においてノード次数をノードラベルとすることで G のノードラベルとは独立に G の構造情報を LG の特徴ベクトルに入れる。ここで、強調すべきのは、 LG のノード次数 (G のエッジ次数) はグラフ構造に依存する特徴である。

ここで、なぜわざとライングラフを経由して、元のグラフ G の構造特徴を抽出するのか？適当に元のグラフに対し、ノードラベルにノード次数を入れ替えれば、グラフの構造特徴も獲得可能ではないか？答えとして、確かにノード次数の入れ替えことで、グラフ G から構造特徴を抽出できるだが、説明必要なのは、基本的に、図 3-3 に示すように、変換したライングラフ LG が元のグラフ G より構造的に複雑になる。なので、もと細かい構造の特徴を学習できることが期待される。

LG の特徴ベクトルのみを生成すると、逆に G のノードラベルが無視されてしまうので、 G の特徴ベクトルも同時に生成する。提案手法 GL2vec のフレームワークが図 3-4 に示す。また、手順は下記のように構成される。

1. 与えられたグラフの集合 $\{G_1, G_2, \dots, G_N\}$ に対し、ライングラフの集合 $\{LG_1, LG_2, \dots, LG_N\}$ を作成。さらに、 G_i のエッジ情報を LG_i へ渡すときに、データセットにより、二つ場合を分ける。
 - G_i がエッジラベルを持つ場合、 G_i のエッジラベルを LG_i のノードのラベルに持たせる。
 - G_i がエッジラベルを持たない場合、 G_i のエッジ次数を LG_i のノードのラベルに持たせる。

2. $\{G_1, G_2, \dots, G_N\}$ に対して, Graph2vec モデルを適用する. その結果, G_i に対して δ 次元の特徴ベクトル \vec{G}_i が得られる.
3. $\{LG_1, LG_2, \dots, LG_N\}$ に対して, Graph2vec モデルを適用する. その結果, LG_i に対して δ 次元の特徴ベクトル $L\vec{G}_i$ が得られる.
4. G_i に対して, \vec{G}_i と $L\vec{G}_i$ を結合した 2δ 次元のベクトルを G_i に対する最終的な特徴ベクトルとする.

なお, エッジラベルとノードラベル両方ともないデータセットに対し, Graph2vec の論文ではノード次数をノードラベルとして, 構造特徴を学習する. それについて, 前文で述べたとおり, ライングラフを経由し, エッジ次数を利用することで, もとのグラフより一步細かい構造特徴を抽出可能である. G_i と LG_i の連結により, 粗い構造特徴と細かい構造特徴を組み込むことも可能になる.

3.3 実験

本節では, 提案手法の優位性を検証するために, ソーシャルネットワーク, 生物情報分野と化学情報分野分野におけるいくつかのベンチマークデータセットを用いて, グラフ分類 (2 クラス) タスクを行う. 補助特徴量としてのライングラフの特徴ベクトルが元のグラフの特徴ベクトルと連結する形で, 分類タスクの正解率を向上させるのかどうかを検証する.

3.3.1 グラフデータセット

本論文では, 11 個のグラフ分類ためのデータセットを用意し, 二種類に分ける: (1) エッジラベル無しの六つのデータセット; (2) エッジラベル有りの五つのデータセット. すべてのデータセットの情報 (サンプル数, 平均ノード数, ノードラベル種類数, エッジラベル種類数) を表 3-1 に示す. また, これから, 各データセットの説明を行う.

エッジラベルなしデータセット

種類 1 に所属する 6 つのデータセットは, MUTAG[3], PTC[3], PROTEINS[3], NCI1[3], NCI109[3], および IMDB-B[14] である. 最初の 5 つは, [3] で Graph2vec

表 3-1: データセットの統計量

データセット 名	サンプル 数	ノード 数 (平均)	ノードラベル 種類数	エッジラベル 種類数
MUTAG	188	17.9	7	-
PTC	344	25.5	19	-
PROTEINS	1113	39.1	3	-
NCI1	4110	29.8	37	-
NCI109	4127	29.6	38	-
IMDB-B	1000	19.8	-	-
MUTAG*	188	17.9	7	4
NCI33	2843	30.2	29	4
NCI81	4030	29.6	31	4
NCI83	3867	29.5	28	4
DBLP	19456	100.5	41324	3

の評価にすでに使用されていた。

MUTAG, PTC, NCI1, NCI109 これらのデータセットは化学情報学の分野からのデータである。化学データをグラフに変換したときには、ノードが原子を表し(水素は省略され)、エッジが化学結合を表す。なお、ノードは原子タイプによってラベル付けされている。MUTAG データセットは、細菌に対する変異原性の影響に応じて2クラスに分類された188個の化合物で構成される。PTC データセットは344 個の化合物から成り、ネズミにおける発がん性を示している。NCI1 と NCI109 はそれぞれ非小細胞肺癌細胞株と卵巣癌細胞株に対する活性についてスクリーニングされた化合物データセットからバランスを考慮してサンプリングしたサブセットであり、4110 個と 4127 個のグラフで構成される。

PROTEINS はタンパク質に関するデータセットで、ノードが2次構造要素を表し、エッジがアミノ酸配列または3D空間の近傍を表す1113 個のグラフで構成される。

IMDB-B はグラフにノードラベルもエッジラベルもない、特別なグラフデータセットである。このデータセットには、映画コラボレーションデータベースから変換されたグラフが格納される。より具体的には、2つのジャンル「アクション」と「ロマンス」の2つの映画コラボレーションネットワークが構築される。ノードは俳優/女優を表し、同じ映画に登場する場合はそれらの間にエッジがある。次に、この二つのネットワークから各俳優/女優の ego-network を抽出しグラフデータとする。そして、各 ego-network を代表するグラフが2つのジャンルのどちらに属するかを判断するタスクを行う。

エッジラベルありデータセット

種類2に所属する5つのデータセットは、MUTAG*[24]、NCI33[13]、NCI81[13]、NCI83[13]、および DBLP[13] である。

MUTAG*、NCI33、NCI81、NCI83 これらのデータセットも化学情報学の分野からのデータである。MUTAG*データセットはMUTAGと同じで、さらにエッジが化学結合タイプでラベル付けされている。NCI33、NCI81 および NCI83 データセットはそれぞれ黒色腫の癌細胞株、大腸癌細胞株および乳癌細胞株に対する活性についてスクリーニングされた化合物データセットからバランスを考慮してサンプリングしたサブセットであり、2843 個、4030 個および 3867 個のグラフで構成される。さらにエッジは化学結合タイプでラベル付けされている。

DBLP データセットは、コンピュータサイエンスの参考文献データから変換された 19456 個のグラフで構成されている。各論文は下記のルールでグラフへ変換する。(1) 各論文は論文ノードになる。(2) 論文タイトルに含まれるキーワードはキーワードノードになり、論文ノードと接続される。また、同じ論文ノードに接続されたキーワードノード群は全結合する。(3) 論文の間に引用関係があれば、対応する論文ノードペアにエッジを張る。論文ノードは論文番号を、キーワードノードはキーワードをノードラベルとして持つ。また、エッジには両端点が論文ノードであるかキーワードノードであるかを表すラベルが付与される。このデータセットに対するタスクは、各論文の分野が「データベースとデータマイニング」或は「コンピュータビジョンとパターン認識」のどちらであるかを推定する2クラス分類である。

3.3.2 実験の設定

GL2vec の実装では，既存の Graph2vec ソフトウェアパッケージをサブルーチンとして呼び出し，Graph2vec のパラメーターについては，根付き部分グラフの最大の深さ H を 3 に設定し，Graph2vec で生成した特徴ベクトルの次元を $\delta = 1024$ に設定する．そして，GL2vec はグラフの特徴ベクトルとライングラフの特徴ベクトルを連結するので，最終の特徴ベクトルが 2048 次元になる．

生成した各グラフの (特徴ベクトル, 所属するクラス) ペアを線形 SVM 分類器に任せて，グラフ分類タスクを行う．データセットごとに，サンプルの 90 % を学習データとしてランダムに選択し，残りの 10 % のサンプルをテストデータとして使用する．SVM のハイパーパラメーターは 5 分割交差検証によって調整され，実験を 20 回繰り返し，テストデータでの平均分類精度を報告する．

3.3.3 実験結果

表 3-2: エッジラベル無しグラフ分類の正解率 (平均 \pm 標準偏差)%

データセット名	MUTAG	PTC	PROTEINS	NCI1	NCI109	IMDB-B
Graph2vec	83.68 \pm 7.02	61.00 \pm 5.58	72.50 \pm 6.16	75.82 \pm 2.72	75.87 \pm 2.27	72.80 \pm 3.42
GL2vec	86.58\pm5.78	60.57 \pm 4.41	70.09 \pm 5.52	77.77\pm2.34	79.69\pm2.04	74.10\pm4.44

表 3-3: エッジラベル有りグラフ分類の正解率 (平均 \pm 標準偏差)%

データセット名	MUTAG*	NCI33	NCI81	NCI83	DBLP
Graph2vec	83.68 \pm 7.02	78.95 \pm 1.82	77.77 \pm 2.24	75.90 \pm 1.66	90.63 \pm 0.59
GL2vec	87.63\pm7.50	81.30\pm2.17	79.60\pm2.09	77.29\pm1.31	92.27\pm0.62

種類 1 に所属する 6 つのエッジラベル無しのデータセットに対し，従来手法 Graph2vec および提案手法 GL2vec で生成した特徴ベクトルを用いたグラフ分類の正解率が表 3-2 で報告される．これらのデータセットのうち 5 つは [3] で Graph2vec の評価に既に使用されているが，本論文における Graph2vec の実験では，もとの論文とほぼ同じ精度が再現できた．表 3-2 は，GL2vec が 4 つのデータセット (MUTAG, NCI1, NCI109, および IMDB-B) で Graph2vec よりも優れていることを示

している．提案手法が従来手法よりそれぞれ 2.90%, 1.95%, 3.82% で分類精度が上がっている．そして PTC に対して，提案手法と従来手法が同じぐらい精度が出る．なお，PROTEINS に対して，正解率が 2.41% で下がっている．これらの結果から，ノードラベルの影響を受けずにライングラフの特徴ベクトルで構造情報を補完する GL2vec のアプローチは有望であると主張しているが，まだ決定的ではない．

IMDB-B データセットに関して，元々ノードラベルがないため，Graph2vec はノードの次数を元のグラフのノードラベルとして使用する．3.2.2 節の最後の説明によると，この実験結果は次のように解釈される．GL2vec は，元のグラフとライングラフを用いての 2 つの異なるレベル (粗い構造特徴と細かい構造特徴) で構造的類似性を測る，単一レベルの構造的な類似性に基づく Graph2vec よりも優れている．

種類 2 に所属する 5 つのエッジラベル有りのデータセットに対し，従来手法 Graph2vec および提案手法 GL2vec で生成した特徴ベクトルを用いたグラフ分類の正解率が表 3-3 で報告される．表 3-3 は，すべてのデータセットに対し，提案手法が従来手法より分類精度が上回っていることを示している．これらの結果は，GL2vec が Graph2vec でのエッジラベルを処理できないことを適切に解決し，エッジラベル情報をライングラフの特徴ベクトルで補完することを証明している．また，エッジラベルがグラフ分類タスクに役立つことも明らかになった．

全体的に，GL2vec は 11 個のデータセットのうち 9 個で Graph2vec よりもグラフ分類の正解率が上回る．なので，ライングラフを用いて，元のグラフの特徴を補完できることが確認できた．

3.4 まとめ

本章では，まず，Graph2vec の二つ欠点を指摘した．一つ目はエッジラベルを扱えないこと，二つ目は抽出した根付き部分グラフを識別 ID へマッピングする際に，ノードラベルとグラフ構造を同時に畳み込むため，構造の類似性を適切に表現できない場合があることである．そして，この二つ欠点を改良するために，Graph2vec への拡張する手法 GL2vec を提案した．提案手法 GL2vec では，ライングラフを活用することで，元のグラフにおけるエッジラベルおよびエッジ次数がライングラフのノードに渡すことが出来る．ライングラフの特徴ベクトルにより，元のグラフ

の特徴ベクトルで含まないエッジ属性情報と構造情報を補完できる．実験で，ライングラフと元のグラフの特徴ベクトルを連結することで，GL2vec は Graph2vec よりもグラフ分類タスクの性能を改善することを示した．

第4章

GINの改善

2.3.1 節で述べた GIN では、ノード属性付きグラフを対象に、ノード間で情報伝搬することで、ノードの潜在特徴ベクトルを更新する。また、全ノードの特徴ベクトル合成してグラフ分類用のグラフ全体に対する特徴ベクトルを生成する。また、end-to-end で、特徴抽出器と分類器と一緒に学習させる。このようなアプローチも欠点がある。本章では、GIN の不足を改良する手法 GLIN を提案する。

まず、4.1 節で、GIN の欠点を指摘する。そして、4.2 節で、その課題を克服する手法を提案する。4.3 節は実験。4.4 節はまとめ。

4.1 GIN の欠点

2.3.1 節で説明した通り、GIN は GNN の一員であり、情報伝搬を行う。特に、各ノード v を根として、集約した近傍ノードの特徴ベクトルと根ノード現在の特徴ベクトルを結合することで、根ノードの特徴ベクトルを更新する。このように情報伝搬で得られるノードの潜在特徴ベクトルが下記四つの条件で決定される。

1. 入力したノードの属性情報
2. グラフの構造、特に隣接ノード関係
3. モデルの構成: 潜在特徴ベクトルの次元数やネットワークの層数など
4. 学習されたネットワークの重み

上記の条件から見ると、明らかに、GIN ではエッジの属性情報を使用していない。グラフ上で、いくらノード間でノード属性情報を伝搬しても、生成したノードの

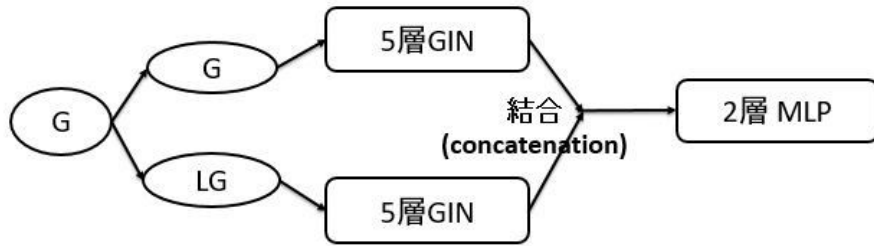


図 4-1: 提案手法 GLIN の仕組み

潜在特徴ベクトルにはエッジの情報が一切含まれない．この結果，全ノードの特徴ベクトルから計算されるグラフの特徴ベクトルもエッジの情報に一切反映していない．3章の実験で示した通り，エッジの属性情報（ノード間の関係を表す情報）はグラフ分類にとってかなり重要な情報である．それを活用できれば，一歩より高いグラフ分類精度が得られると期待される．

4.2 提案手法:GLIN

本節では，4.1節で指摘した GIN の欠点を改善するために，GIN への拡張するフレームワークである GLIN(Graph and Line graph Isomorphism Network) を提案する．

提案手法のフレームワークが図 4-1 に示す．3章で提案した GL2vec と同様に，GLIN もライングラフを用いて，グラフの特徴ベクトルをエッジ特徴で補完する．手順が下記のように構成される．

1. 与えられたグラフの集合 $\{G_1, G_2, \dots, G_N\}$ に対し，ライングラフの集合 $\{LG_1, LG_2, \dots, LG_N\}$ を作成． G_i のエッジ属性を LG_i のノード属性として使う．
2. G_i を GIN 層に入力し， G_i に対して δ 次元の特徴ベクトル \vec{G}_i が得られる．
3. LG_i を GIN 層に入力し， LG_i に対して δ 次元の特徴ベクトル \vec{LG}_i が得られる．
4. \vec{G}_i と \vec{LG}_i を結合した 2δ 次元のベクトルを MLP に入力し，分類を行う．

具体的に，元のグラフ G を GIN に入力際に，式 2.18 と式 2.19 により，ノード領域における近傍ノード間の情報伝搬で，各ノード特徴ベクトル h_v を生成（更新）し，

それを用いてグラフの特徴ベクトル h_G を合成する． h_G にはノード属性情報しか持っていない．なので，提案手法では h_G にエッジの属性情報を補完させるために，ライングラフを活用する．ライングラフ LG を GIN に入力すると，[4.1](#) と式 [4.1](#) により，エッジ領域における近傍エッジ間の情報伝搬で，各エッジ特徴ベクトル h_e を生成 (更新) し，それを用いてグラフの特徴ベクトル h_{LG} を合成する，合成したライングラフの特徴ベクトル h_{LG} がエッジの属性情報を持つ．また， h_G と h_{LG} を結合 (concatenate) したものがノード特徴とエッジ特徴両方とも反映できるグラフの特徴ベクトルになる．

$$h_e^{(k)} = MLP^k((1 + \epsilon^{(k)}) \cdot h_e^{(k-1)} + \sum_{i \in \mathcal{N}(e)} h_i^{(k-1)}) \quad (4.1)$$

$$h_{LG} = CONCAT(SUM(\{h_e^{(K)} | e \in G\}) | k = 0, 1, \dots, K) \quad (4.2)$$

4.3 実験

本節では，提案手法の優位性を検証するために，生物化学情報分野分野における四つのデータセットを用いて，グラフ分類 (2 クラス) タスクを行う．単一の h_G より， h_G と h_{LG} を結合した特徴ベクトルが分類タスクの正解率を向上させるのかどうかを検証する．

4.3.1 グラフデータセット

本実験で使った NCI(National Cancer Institute) データシリーズは，アメリカ国立がん研究所からの抗がん活性予測データである．化学データをグラフに変換したときには，ノードが原子を表し (水素は省略され)，エッジが化学結合を表す．なお，ノードは原子タイプによってラベル付けされている．エッジは化学結合タイプでラベル付けされている．本実験では NCI33，NCI81，NCI83，および NCI123 この四つデータセットを使用する．それぞれ黒色腫の癌細胞株，大腸癌細胞株，乳癌細胞株，および白血病細胞株に対する活性についてスクリーニングされた化合物データセットからバランスを考慮してサンプリングしたサブセットであり，2843 個，4030 個，3867 個，および 5260 個のグラフで構成される．すべてのデータセットの情報 (サンプル数，平均ノード数，ノードラベル種類数，エッジラベル種類数) を表 [4.1](#) に示す．

表 4-1: NCI データセットの統計量

データセット 名	サンプル 数	ノード 数 (平均)	エッジ 数 (平均)	ノードラベル 種類数	エッジラベル 種類数
NCI33	2843	30.2	32.9	29	4
NCI81	4030	29.6	32.3	31	4
NCI83	3867	29.5	32.2	28	4
NCI123	5260	28.8	31.4	33	4

4.3.2 実験の設定

ハイパーパラメータの設定について GIN の論文 [4] での最適なハイパーパラメータ設定を参照し、設定は以下通り。

- モデルのアーキテクチャ

- ◇ GIN の階層数:4
- ◇ 各 GIN 層内にある MLP の階層数:2
- ◇ 分類器としての MLP の階層数:2
- ◇ 各 MLP における隠れ層のニューロン数:32
- ◇ ϵ :0
- ◇ 各隠れ層に batch normalization[17] を使用する。

- パラメータの学習

- ◇ 最適化アルゴリズム:Adam[15]
- ◇ 学習率:初期値は 0.005 であり、50 エポックごとに 0.5 の減衰率で学習率を調整する。
- ◇ エポック数:300
- ◇ batch-size:128
- ◇ 分類器での dropout[16] 率:0.5

実験で使った python のライブラリについて `rdkit`[25], `networkx`[26], `pytorch-geometric`[18], および `pytorch`[27], この四つのライブラリを用いて実験を行う。ライブラリの役割は以下通り。

- `rdkit`: 化合物データ (.sdf ファイル) を読み取ると分析する。
- `networkx`: グラフの生成と, グラフからライングラフへ変換などのグラフに対する操作を行なう。
- `pytorch-geometric`: GIN レイヤーを構築する。
- `pytorch`: MLP レイヤーの構築とネットワークの学習などの操作を行う。

評価基準について 各データセットを 80%, 10%, 10% の割合で, 学習, 検証, テストの三つのグループにランダムに分ける。過学習を抑えるために, 学習データを使って学習するときには, 検証データに最も正確に分類できるモデル (ネットワークの重み) を選択する。そして, そのモデルによりテストデータのグラフ分類実験を行う。モデルの生成は (1) 学習データ, 検証データ, テストデータをランダムに選択すると, (2) ネットワークの重みをランダムに初期化するというように乱数を 2 箇所で行っている。グループ分けのランダムシードを五つと, 重み初期化のランダムシードを三つ用意して, 合わせて 15 回の実験 (15 パターン) を行って, テストデータでの分類正解率 (平均と標準偏差) を報告する。

4.3.3 実験結果

表 4-2: グラフ分類の正解率 (平均 \pm 標準偏差)%

データセット名	NCI33	NCI81	NCI83	NCI123
GIN	80.56 \pm 2.96	77.06 \pm 1.22	75.35 \pm 1.84	74.47 \pm 2.29
GLIN	80.92\pm2.26	79.21\pm1.86	76.39\pm1.91	75.02\pm1.48

実験結果を表 4-2 に示す。すべてのデータセットに対し, 提案手法 GLIN が, 従来手法 GIN よりもグラフ分類の性能を向上させていることがわかる。なお, 提案手法では, エッジ領域における情報伝搬で学習したライングラフの特徴ベクトルと, ノード領域における情報伝搬で学習した元のグラフの特徴ベクトルを結合す

ることで、ノード属性情報とエッジ属性情報両方を考慮したグラフの特徴ベクトルを適切に表現することができたと考えられる。3.3 節の実験結果と同様に、本実験の結果も、エッジ属性情報がグラフ分類タスクに役に立つことを示した。

4.4 まとめ

本章では、まず、GIN の欠点を指摘した。その後、GIN への拡張するフレームワーク GLIN について述べた。そして、実験によって、ライングラフを用いて、元のグラフの特徴ベクトルにエッジ属性特徴を補完することが可能であることを示した。また、提案法 GLIN は従来法である GIN よりもグラフ分類タスクの性能を改善することを示した。

第5章

関連研究

本章では、グラフに基づくパターン認識分野におけるライングラフを用いた二つの関連手法について説明する。

Bai et al.[19] は、一致したエッジペアの数から 2 つのグラフ間の類似性を測定するエッジベースのマッチンググラフカーネル EMBK を提案した。2 つのエッジが一致するかどうかを判断するために、彼らはライングラフを用いて各エッジの特徴ベクトルを生成する。特に、グラフ G における各エッジ e の特徴ベクトルの i 番目の座標値は、ライングラフにおけるノード $v(e)$ の i ホップ周囲の周辺情報から決定される。彼らは、エッジベースのマッチングカーネルが、元のグラフで定義されるノードベースのマッチングカーネルよりも優れていることを示している。本研究で提案した手法と EMBK の違いは下記の 2 点である。(1) EMBK はグラフマッチングカーネルであり、グラフの特徴ベクトルを直接に生成しない。一方、提案法は、任意のベクトルベースの機械学習アルゴリズムに適用できるグラフの特徴ベクトルを生成する。(2) EMBK は元のグラフを無視して、線グラフのみを使用するが、提案法は元のグラフとライングラフ両方とも考慮したグラフの特徴ベクトルを生成する。

DPGCNN[20] は、GAT(Graph Attention Networks)[21] を拡張した双対グラフ畳み込みネットワークある。GAT はノード特徴を用いてアテンションスコアを計算するが、DGCNN はエッジ特徴を基にしてアテンションスコアを計算する。提案法と類似しているのは、DPGCNN もライングラフのノード特徴を元のグラフのエッジ特徴をとして扱うこと。なお、違いとして、DGCNN がノードとエッジの特徴量を生成し、それをノード分類、リンク推定などのタスクに適用する。一方、提案法はグラフの特徴ベクトルを生成して、グラフ分類などのタスクに適用する。

第6章

結論

本論文では、ニューラルネットワークを用いたグラフの分散表現 (特徴ベクトル) を学習する課題を着目し、エッジ特徴により、グラフの分散表現を改良する手法を提案した。具体的に、まず、2章で、近年提案された様々なグラフの分散表現の学習手法から二つの代表的な手法である Graph2vec と GIN について説明した。そして、3章で、従来手法 Graph2vec の二つ欠点 (1) エッジラベル情報を扱えないことと、(2) グラフ間構造の類似性が適切に表現できないことを指摘した。それに対処するため、グラフ G のエッジ双対グラフ LG (ライングラフ) を利用して補う手法 $GL2vec$ を提案した。ライングラフ LG を用いて、グラフ G のエッジ特徴を表現できることを示した。実験によって、エッジ特徴 (属性特徴、構造特徴) とノード特徴両方を考慮した提案法 $GL2vec$ が従来法 Graph2vec よりもグラフ分類性能を改善することを示した。また、4章で、エッジ属性付きグラフに適用できない GIN への拡張モデル $GLIN$ を提案した。ライングラフを用いて、エッジ特徴を表現できることを再び示した。実験によって、エッジ特徴を補完することで、提案法 $GLIN$ は従来法 GIN よりもグラフ分類の性能を改善することを示した。

上記の二つ改良例から、ノード属性しか扱えないグラフ分散表現学習のモデルの一族に対し、ライングラフはエッジ属性を活用できる有効なツールであることが考えられる。エッジ特徴がグラフ分類タスクに役に立つことも明かになった。

今後の課題について、1つの興味深い研究問題は、大きいかつ密なグラフの処理に関するものである。元のグラフが大きいかつ密な場合、ライングラフにおけるノードの数が元のグラフより2乗まで増加する。したがって、計算リソースがかかる。このような場合にライングラフを用いたモデルへの拡張する方法は、検討する価値がある。

謝辞

本研究を行うにあたり，研究の場を与えていただき，なおかつ多くのご指導とご助言をいただいた古賀久志准教授，南泰浩教授，戸田貴久准教授，ならびに中鹿亘助教に心より深く感謝いたします．ご多忙の中，多くのご助言，ご協力をくださった柳生智彦客員准教授，岸田拓也研究員に深く感謝いたします．そして，研究室での生活や研究の様々な場面でご助言，ご協力をいただきました南・古賀・戸田・中鹿研究室の学生の皆さま，すでにご卒業された先輩方に心から感謝いたします．

令和2年1月27日

参考文献

- [1] Harary. F.: Graph Theory. Massachusetts: Addison-Wesley, pp. 7183. (1972)
- [2] Whitney, H.: Congruent graphs and the connectivity of graphs. Hassler Whitney Collected Papers. Birkhuser Boston, pp. 61-79.(1992)
- [3] Narayanan, A., et al.: graph2vec: Learning distributed representations of graphs. In: 13th International Workshop on Mining and Learning with Graphs. (2017).
- [4] Xu, K., Hu, W., Leskovec, J., and Jegelka, S.:How powerful are graph neural networks. In: Proceedings of the 7th International Conference on Learning Representations (2019)
- [5] Wu, Z., et al.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019).
- [6] Shervashidze, N., et al.: WeisfeilerLehman Graph Kernels. Journal of Machine Learning Research 12, pp. 25392561 (2011)
- [7] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26, pp. 3111-3119 (2013)
- [8] Le, Q., and Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2): pp. 1188-1196 (2014)
- [9] Kipf, T. N., and Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations (2017)
- [10] Hamilton, W., Ying, Z., and Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems pp.1024-1034 (2017)

- [11] Ying, Z., et al.: Hierarchical graph representation learning with differentiable pooling. In: *Advances in Neural Information Processing Systems* pp. 4800-4810 (2018)
- [12] Zhang, M., Cui, Z., Neumann, M., and Chen, Y.: An end-to-end deep learning architecture for graph classification. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* pp. 4438–4445 (2018)
- [13] Zhu, X., Zhang, C., Pan, S., and Yu, P.: Graph stream classification using labeled and unlabeled graphs. In: *Proceedings of the IEEE 29th International Conference on Data Engineering*, pp. 398-409 (2013)
- [14] Yanardag, P., and Vishwanathan, S. V. N.: Deep graph kernels. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 1365-1374 (2015)
- [15] Diederik, P., Kingma, and Jimmy Ba.: Adam: A method for stochastic optimization. In: *Proceedings of the 3rd International Conference on Learning Representations*. (2015)
- [16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. In: *The journal of machine learning research*, 15(1), pp. 1929-1958 (2014)
- [17] Ioffe, S., and Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on Machine Learning* pp. 448-456 (2015)
- [18] Fey, M., and Lenssen, J. E.: Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
- [19] Bai, L., Zhang, Z., Wang, C., and Hancock, E. R.: An edge-based matching kernel for graphs through the directed line graphs. In: *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, pp. 85-95 (2015)

- [20] Monti, F., Shchur, O., Bojchevski, A., Litany, O., Gnnemann, S., and Bronstein, M. M.: Dual-primal graph convolutional networks. arXiv preprint arXiv:1806.00770. (2018)
- [21] Velickovi, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y.: Graph attention networks. In: Proceedings of the 6th International Conference on Learning Representations. (2018)
- [22] Hornik, K.: Approximation capabilities of multilayer feedforward networks. In: Neural networks, 4(2), pp: 251-257 (1991)
- [23] Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: Learning representations by back-propagating errors. In: nature, 323(6088), pp: 533-536 (1986)
- [24] “ Benchmark Data Sets for Graph Kernels ”, <http://graphkernels.cs.tu-dortmund.de>
- [25] “ rdkit ”, <https://www.rdkit.org/>
- [26] “ networkx ”, <https://networkx.github.io/documentation/stable/tutorial.html>
- [27] “ pytorch ”, <https://pytorch.org/>

図一覧

1-1 分子構造の乳癌細胞株に対する生物活性	2
1-2 分子のグラフ表現	2
2-1 二つの隠れ層を持つ順伝播型ニューラルネットワーク	6
2-2 DBOW の仕組み	9
2-3 グラフに適用した DBOW の仕組み	10
2-4 グラフ G に対する 1 回 W-L 再ラベル操作	12
2-5 近傍の情報を集約	14
2-6 近傍情報と根ノードの現在の特徴ベクトルを結合し、根ノードの特徴量を更新	15
2-7 全ノードの特徴ベクトルからグラフの特徴ベクトルを生成	15
3-1 形状が同一で中央ノードのラベルだけが異なる 2 つのグラフ G, G^*	20
3-2 深さ $H = 1$ までの根付きの部分グラフの集合 (識別 ID の多重集合) $c(G)$ と $c(G^*)$	20
3-3 グラフ G と対応するライングラフ LG	21
3-4 提案手法 $GL2vec$ のフレームワーク	23
4-1 提案手法 $GLIN$ の仕組み	31

表一覧

3-1 データセットの統計量	25
3-2 エッジラベル無しグラフ分類の正解率 (平均 ± 標準偏差)%	27
3-3 エッジラベル有りグラフ分類の正解率 (平均 ± 標準偏差)%	27
4-1 NCIデータセットの統計量	33
4-2 グラフ分類の正解率 (平均 ± 標準偏差)%	34

業績リスト

査読なし国内学会発表

Cheng Hong, 古賀久志: “双対グラフを用いたグラフの分散表現学習”, 第 18 回 情報科学技術フォーラム (FIT2019), (2019)

査読付き国際学会発表

Hong Chen and Hisashi Koga: “GL2vec: Graph Embedding Enriched by Line Graphs with Edge Features.” In: Proceedings of 26th International Conference on Neural Information Processing. Part 3, pp.3-14 (2019)