グラフ分類における部分グラフ特徴集合の確率的探索

“Probabilistic Search for Subgraph Feature Sets in Graph Classification”

白川　稜

平成30年2月

北海道大学工学部情報エレクトロニクス学科

情報理工学コース

目次

要 旨

グラフ分類問題は，生命科学や物質科学，創薬の分野などで広く応用されている問題である．グラフ分類問題では部分グラフの有無を特徴量とすることが多い．一般に入力グラフにおける部分グラフ数は膨大であり，全列挙は現実的ではない．これに対して既存手法 gBoost では，特徴探索と分類器の学習を同時に行うことで．部分グラフを全列挙せずに全部分グラフを候補にした学習を可能にする．しかし，gBoost ではある目的関数を最大にする部分グラフを厳密に探索するため．特徴探索の面で多大なコストを要する．ここで，部分グラフ探索空間には厳密解と同程度の目的関数を達成する解が多数存在しており，厳密に探索を行う必要がない場合もあると考えられる．したがって本研究では，グラフ分類問題に対して，確率的に探索空間を削減し，部分グラフ特徴探索を効率化する手法を提案する．

第1章

はじめに

1.1 背景

グラフは，自然言語処理 [1]，RNA 二次構造 [2]，低分子化合物の構造式 [3] などの知識処理に幅広く用いられる重要なデータ構造である．近年こうした科学分野のグラフデータが公共データベースに蓄積，整備されるようになり，有効な利活用が課題となっている．特に，グラフデータからの教師付き学習は，生命科学や物質科学における構造活性相関や構造物性相関の定量的モデルとして機械学習分野で研究されており，より高精度で効率的な手法が求められている．

グラフデータからの教師付き学習では，特徴量として部分グラフの有無 (部分グラフ指示子) を用いることが多い．この場合，検査する部分グラフ特徴の選択が学習モデルの予測精度を左右する．しかし，適切な特徴集合はデータによって異なる．一方で，与えられたグラフデータに生起する全ての部分グラフの列挙は現実的ではない．したがって，グラフカーネル法 [4] や ECFP 法 [5] など，対象となる部分グラフのクラスを予め発見的に制限する手法や，gBoost 法 [6] や Adaboost に基づく手法 [1] など，必要な部分グラフのみを効率的に探索しながら学習を行う方法が提案されている．後者のアプローチは候補集合から必要な部分グラフのみを探索でき，広い種類のグラフデータに対する汎用性が期待できる．一方で，必要な部分グラフを厳密に何度も探索するため，与えられたグラフデータのグラフ数及び含まれるグラフの大きさによっては現実的な時間でモデル構築をするのが困難になる場合もある．

そこで，本研究では，部分グラフ指示子の探索を効率化するために，確率的アルゴリ

ズムを用いた特徴探索手法を提案する．確率的アルゴリズムには，強化学習やモンテカルロ木探索法 [7] などがあり，扱う探索空間の規模が大きいとより効果が期待できることが知られている．したがって，本研究で扱うグラフ分類問題において，既存手法におけるモデル構築のボトルネックである特徴探索の効率化を行うために確率的アルゴリズムを取り入れる．

提案手法では，探索木におけるパスを複数本サンプリングし，サンプリングしたパス内に探索を制限することにより，探索空間の縮小と探索コストの削減を図る．また，パスのサンプリングにおいて確率の振り方を三種類定め，それぞれの確率を利用した三種類の確率的探索手法を考える．各手法の評価を，実データを用いた実験により比較する．

1.2 構成

本論文では，以降の章で以下の通り詳述する．

- 第 2 章では，扱う問題設定，既存手法について述べる．
- 第 3 章では，提案手法の枠組みと，それを用いた三種類の探索手法について述べる．
- 第 4 章では，提案手法を用いた実験とその結果を述べる．
- 第 5 章では，実験結果に対する考察を述べる．
- 第 6 章では，まとめと今後の展望を述べる．

第2章

既存手法

本章では，既存手法である gBoost[6] について説明する．gBoost は線形計画問題で定式化されたブースティング手法である LPBoost[8] と，部分グラフの探索アルゴリズムを組み合わせたグラフの分類，回帰モデルである．本研究ではグラフの分類問題を扱うため，以下では，問題設定，既存手法と順を追って説明する．

2.1 グラフ分類問題

まずはじめに，今回扱う問題であるグラフ分類問題について説明する．グラフ分類問題とは，教師データとしてグラフとクラスラベルのペアの集合 $\{G_n, y_n\}_{n=1}^{\ell}$ を与えられた際，未知のグラフデータに対するクラスラベルを予測する予測器 f を学習する問題である．扱うグラフは，連結であるものとする．本研究では，クラスラベル y に関して，$(y \in \{+1, -1\})$ となるような二値分類を対象として扱う．

2.2 gBoost

一般的に機械学習の線形モデルは，扱う特徴に対してその線型結合の値を出力値として予測に用いる．2 値分類（+1 or -1）の場合，出力値の値が 0 以上の時に 1，0 未満の時に-1 と予測する．グラフ分類問題に用いる特徴量は様々なものが考えられるが，gBoost ではグラフデータに含まれる部分グラフの有無を特徴量として用いる．

入力として ℓ 個のグラフとクラスラベルのペアの集合 $\{G_n, y_n\}_{n=1}^{\ell}(y_n \in \{+1, -1\})$ が与えられた時，入力データに少なくとも 1 回は出現する全ての部分グラフを $\mathcal{T}$ とする．こ

の時グラフ G_n は $|\mathcal{T}|$ 次元のベクトル $\boldsymbol{x}_n(x_{n,t} = \mathbb{I}(t \subseteq G_n), \forall t \in \mathcal{T})$ として表現することができる．$\mathbb{I}(\cdot)$ は括弧内が真なら 1，偽なら 0 を返す指示関数である．

この時，各特徴に対して仮説（hypothesis）と呼ばれる，弱学習器を以下で定義する．

$$h(\boldsymbol{x}; t, \omega) = \omega(2x_t - 1).$$

$\omega \in \Omega = \{-1, 1\}$ はパラメータである．この弱学習器を用いて，gBoost では以下の線形分類モデルを構築する．

$$f(\boldsymbol{x}) = \sum_{(t,\omega)\in\mathcal{T}\times\Omega} \alpha_{t,\omega} h(\boldsymbol{x}; t, \omega). \tag{2.1}$$

$\alpha_{t,\omega}$ は各弱学習器の重みであり，$\sum_{(t,\omega)\in\mathcal{T}\times\Omega} \alpha_{t,\omega} = 1, \alpha_{t,\omega} \geq 0$ を満たす．

gBoost では，LPBoost で定式化された以下の線形計画問題を主問題として解くことで，式 (2.1) の線形モデルの重み $\alpha_{t,\omega}$ の値を導く．

$$\begin{aligned}
&\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\rho} -\rho + D\sum_{n=1}^{\ell} \xi_i \\
&\text{s.t.} \begin{cases} \displaystyle\sum_{(t,\omega)\in\mathcal{T}\times\Omega} y_n \alpha_{t,\omega} h(\boldsymbol{x_n}; t, \omega) + \xi_n \geq \rho, \xi_n \geq 0, n = 1, \ldots, \ell, \\ \displaystyle\sum_{(t,\omega)\in\mathcal{T}\times\Omega} \alpha_{t,\omega} = 1, \alpha_{t,\omega} \geq 0. \end{cases}
\end{aligned}$$

ここで，ρ はソフトマージン，$D = \frac{1}{\nu\ell}, \nu \in (0, 1)$ であり， ν は誤分類に対するコスト制御のパラメータである．一般に，変数 $\boldsymbol{\alpha}$ の数は部分グラフの総数であるため膨大であり，主問題を解くことは困難である．したがって，主問題と等価な以下の双対問題を扱う．

$$\begin{aligned}
&\min_{\boldsymbol{\lambda},\gamma} \gamma \\
&\text{s.t.} \begin{cases} \displaystyle\sum_{n=1}^{\ell} \lambda_n y_n h(\boldsymbol{x}_n; t, \omega) \leq \gamma, \forall (t, \omega) \in \mathcal{T} \times \Omega \\ \displaystyle\sum_{n=1}^{\ell} \lambda_n = 1,\ 0 \leq \lambda_n \leq D,\ i = 1, \ldots, \ell. \end{cases}
\end{aligned} \tag{2.2}$$

双対問題は，主問題に比べ制約式の数は多くなるが列生成法を用いることで解を得ることが可能となる．列生成法は，制約なしの状態から始め，現制約を最も違反する制約

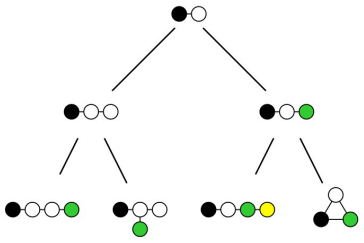


図 2.1: DFS コード木

を追加し，最適化問題を解くという過程を繰り返すことで解を得る手法である．最も違反する制約を見つける問題は以下の最大化問題と等価である．

$$(t^*, \omega^*) = \arg\max_{t \in \mathcal{T}, \omega \in \Omega} \sum_{n=1}^{\ell} \lambda_n^{(k)} y_n h(\boldsymbol{x_n}; t, \omega).$$

これに対して最大化する目的関数を gain と呼び以下の式で定義する．

$$g(t, \omega) = \sum_{n=1}^{\ell} \lambda_n^{(k)} y_n h(\boldsymbol{x_n}; t, \omega). \tag{2.3}$$

もし以下を満たすような弱学習器が存在しない場合，現在の解が双対問題の最適解となる．

$$\sum_{n=1}^{\ell} \lambda_n^{(k)} y_n h(\boldsymbol{x_n}; t, \omega) > \gamma^{(k)}.$$

経験的に終了付近の制約追加による重みの更新は僅かであるため，緩和項 ϵ を用いて以下のように終了条件を緩和することで，計算を効率化する．

$$\sum_{n=1}^{\ell} \lambda_n^{(k)} y_n h(\boldsymbol{x_n}; t, \omega) > \gamma^{(k)} + \epsilon.$$

双対問題の終了条件緩和により得られる主問題の解は，V^* を緩和なしでの最適解とすると，$V \leq V^* + \epsilon$ を満たす．

次に gBoost における特徴探索を考える．特徴探索では重複無しに全部分グラフを列挙した探索空間である DFS コード木 [9] （図 2.1）を用いる．DFS コード木は，全部分グラフをノードに含む他に，子が親の拡大グラフとなるという特徴を持つ．この探索木を

深さ優先順にたどり，各ノードにおいて gain を計算することで，最適な部分グラフを探索する．一般に，部分グラフの総数は膨大となるため全てのノードを探索するためには相当なコストを要する．しかし，子が親の拡大グラフとなる特徴を利用すると，bound（子孫ノードでの gain の上限値）を計算することができるため，安全な枝刈りを行うことが可能である．bound の計算，枝刈り可能条件を以下に示す．

現探索部分グラフを t，その拡大グラフを t' とすると，bound の値 $b(t)$ は以下の式で計算できる．

$$b(t) = \max\{2 \sum_{\{n|y_n=+1, t\subseteq G_n\}} \lambda_n^{(k)} - \sum_{n=1}^{\ell} y_n \lambda_n^{(k)}, 2 \sum_{\{n|y_n=-1, t\subseteq G_n\}} \lambda_n^{(k)} - \sum_{n=1}^{\ell} y_n \lambda_n^{(k)}\}. \tag{2.4}$$

枝刈り可能条件は，現探索までの gain の最大値を g^* とすると以下となる．

$$g^* > b(t). \tag{2.5}$$

この枝刈りが有効に働くため，gBoost アルゴリズムは探索空間に対して効率よくモデルを構築することができる．

第3章

提案手法

既存手法の gBoost では探索木上を深さ優先的に厳密探索する．しかし，入力グラフ集合に対して，個々のグラフの大きさや全体数の増加に伴い，存在しうる部分グラフの総数は組合せ的に増加するため，厳密探索におけるコストは莫大なものである．問題によってモデル構築が困難なものも存在する．したがって，本研究では，厳密探索ではなく，確率的に探索を行う手法を提案する．

既存手法では，探索空間として gSpan[9] の DFS コード木を利用していたが，提案手法では，DFS コード木から同型判定を除外した，つまり，同じ部分グラフを表すノードが複数箇所に出現することを許容した探索空間を利用し，特徴探索を行う．この探索空間の変更の詳細は 3.2 において言及する．

3.1 確率的探索

まず初めに，本提案手法で用いる探索パスおよび K パス空間を定義する．

定義 1 (探索パス)**.** 根ノードを始点とし，子ノードを展開する．展開した子ノードの中からある確率分布 P に従い一つの子ノードを選択する．子ノードの展開，選択の動作を終了条件を満たすまで繰り返す．ここで，終了条件とは，子を持たないノード，もしくは予め設定する部分グラフの最大サイズ（*maxpat*）のノードのことを指す．以上の動作によって選択された根ノードから葉ノードまでのパスを探索パスと定義する．(図 *3.1*)

定義 2 (K パス空間)**.** K 本の探索パスをサンプリングすることによってカバーされるノード集合を K パス空間と定義する．(図 *3.2*)

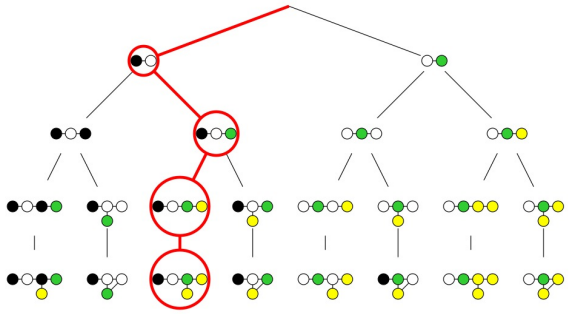


図 3.1: 探索パス

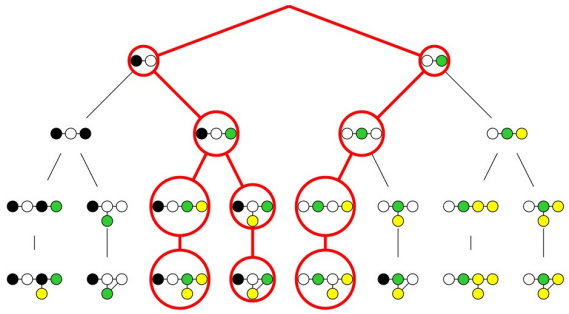


図 3.2: K パス空間

以下では K パス空間を探索空間として利用した，提案手法の枠組みを説明する．Algorithm1 に探索アルゴリズムの擬似コードを示す．ここでは探索木のノード t と対応する部分グラフを同一視する．

Algorithm1 では，K パス探索において同一ノードの探索を複数回行わないために，探索とパスの成長を同時に行う．まず，根ノードを始点に設定し，探索パスの本数 K とともに GROW 関数へ渡す．GROW 関数では引数として渡されたノードにおける gain と bound の値を式 (2.3) および (2.4) で計算し，gain の値がこれまでの最良値である場合には更新する．また，既存手法と同様にして式 (2.5) を用いて枝刈りを行う．その後，終了条件を満たさないならば，渡されたノードの子ノードを展開し（C），展開した子ノード集合上に選択確率分布 P を設定する．SAMPLECOUNT 関数内で，設定した確率分布 P に基づきサンプリングを行い， K を各子ノードに割り振る．各子ノード c に対応する整数 k_c が 0 より大きい場合，GROW 関数にそれぞれを引渡し再帰的に探索を行う．以上のアルゴリズムにより深さ優先的な K パス探索を達成する．

本研究では探索手順における，子ノードの選択確率分布 P に関して，以下に示す 3 つ

Algorithm 1 確率的特徴探索

```
1: procedure RandomSearch(k)
2:     グローバル変数：t*, ω*, g*
3:     t ← 根ノード
4:     Grow(t, K)
5: end procedure
6: function Grow(t, K)
7:     t に対する gain g(t), bound b(t) を計算
8:     if b(t) < g* then
9:         return
10:    end if
11:    if g* < g(t) then
12:        t* ← t, ω* ← ω, g* ← g(t)
13:    end if
14:    if t が終了条件を満たす then
15:        return
16:    end if
17:    C = {t の子ノード}
18:    集合 C 上の確率分布 P を計算                          ▷ 提案手法 1〜3
19:    {k_c | c ∈ C} ← SampleCount(C, K, P)
20:    for all c ∈ C do
21:        if k_c > 0 then
22:            Grow(c, k_c)
23:        end if
24:    end for
25: end function
26: function SampleCount(C, K, P)
27:    k_c ← 0 for all c ∈ C                          ▷ 子ノード c の選択回数を保存
28:    for i = 1 to K do
29:        c ← P に基づき子ノード一つをサンプリング
30:        k_c ← k_c + 1
31:    end for
32:    return {k_c}
33: end function
```

の確率設定手法を提案する．

3.1.1 提案手法 1：一様確率設定

提案手法 1 では，子ノードに対して一様に確率を設定する．現探索ノードの子ノード集合を C とすると，子ノード c の選択確率を以下に示す．

$$P(c) = 1/|C|.$$

3.1.2 提案手法 2：支持度（support）の情報を加えた確率設定

提案手法 1 は，入力グラフの特徴を利用しないナイーブな手法である．本節では，入力グラフに対する支持度（support）の情報を利用した手法を提案する．ある部分グラフ $x \in \mathcal{T}$ の支持度とは，x が ℓ 個の全入力グラフの内いくつのグラフで現れるかを表す値であり以下の式で定義される．

$$\sigma(x) = \sum_{n=1}^{\ell} \mathbb{I}(x \subseteq G_n).$$

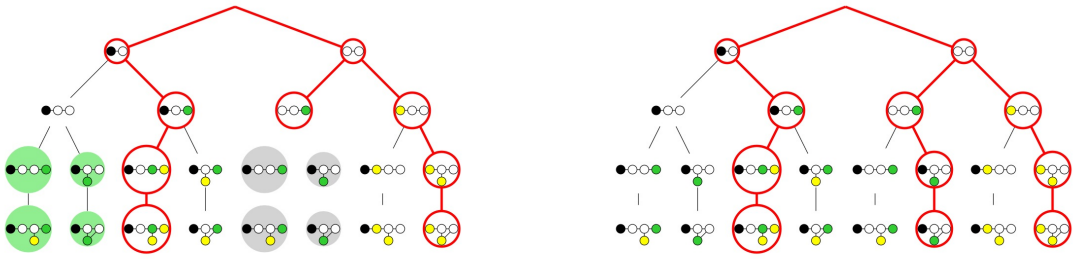
ℓ は入力グラフの総数，$\mathcal{T}$ は入力グラフにおける部分グラフ集合である．

探索木の性質上，支持度が大きい部分グラフの方が小さいものに比べて複数の入力グラフ上での拡大グラフを考えることができるため，子孫空間が広いことを期待できる．子孫空間が狭いノードに多くの探索パスを伸ばす場合，探索が広がらず無駄になる可能性が大きい．したがって，本手法では支持度が大きい子ノードが選択されやすくなるよう確率を定める．現探索ノードの子ノード集合を C とすると，子ノード c の選択確率を以下に示す．

$$P(c) = \sigma(c) / \sum_{t \in C} \sigma(t).$$

3.1.3 提案手法 3：bound の情報を加えた確率設定

本節では，式 (2.4) の bound $b(t)$ の情報を利用した手法を提案する．$b(t)$ は t の子孫ノードで得ることのできる gain の上限値を表すため，bound の大きな子ノードへの探索を行うことで，より大きな gain の値を得ることが期待できる．したがって，本手法では bound



(a) 既存手法での探索空間　　(b) 提案手法での探索空間

図 3.3: 探索空間の変更

の値が大きい子ノードが選択されやすくなるよう確率を定める．現探索ノードの子ノード集合を C とすると，子ノード c の選択確率を以下に示す．

$$P(c) = b(c) / \sum_{t \in C} b(t).$$

3.2 探索空間の同型判定除外

既存手法では，gSpan[9] アルゴリズムで構築される DFS コード木を探索空間として用いている．gSpan アルゴリズムでは，グラフ同型問題を解き，同型である場合には当アルゴリズムで定義される辞書順を用いて最小のものだけを木に採用する．つまり，辞書順最小でないノードが現れた際には，そのノードを木に含まない．

ここで問題となるのが，DFS コード木に探索パスを流す際，探索パスが辞書順最小ではないノードに向かうと途中でパスの成長が止まってしまうという点である．このようなパスの滞りが多く発生する場合，深い所のノード、つまり大きな部分グラフを表すノードの探索をうまく行うことが出来ない．

したがって本研究では，gSpan アルゴリズムでのグラフ同型判定を行わずに，辞書順最小でないノードも木に追加することで，探索パスの滞りを防ぎ，各深さでの探索に偏りが生じないよう探索空間を変更する．(図 3.3)

第4章

実験と結果

4.1 データセット

本稿では，gBoost の元論文 [6] で用いられた化合物データセットのうち，CPDB と Mutag の 2 つを使用した．これら 2 つは，細胞に化合物をふりかけた際の突然変異誘発性の有無を示すデータである．ただし，元論文 [6] と異なり，化合物のグラフ表現は，水素を明示的に付与し，Sybyl Mol2 形式で頂点ラベル，辺ラベルを付与した分子グラフである．表 4.1 に各データセットのグラフ数，平均ノード・エッジ数，正例と負例の数を示す．

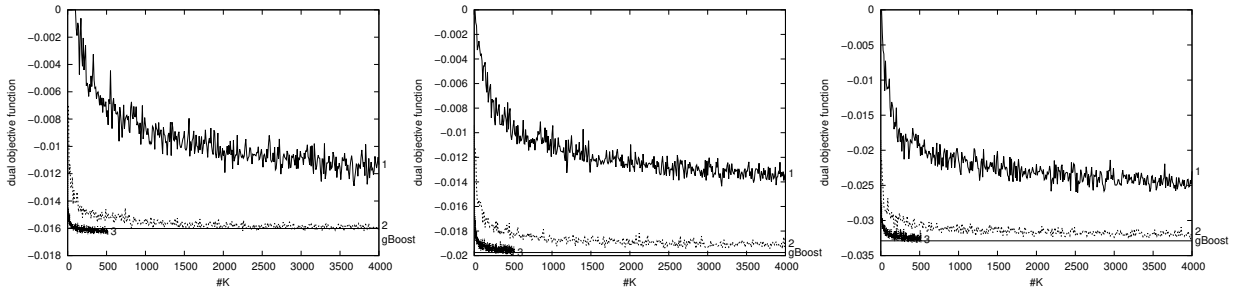
4.2 実験 1：探索の精度，効率に関する実験

実験 1 では，CPDB データに対して各提案手法がどれだけ精度よく探索できているか，どれだけ既存手法より効率化できているかを計る実験を行う．本稿では，式 (2.2) の双対問題における目的関数の最終値を探索精度の指標に，モデル構築までに gain と bound の値を計算したノードの総数を探索ノード数とし，効率化の指標に用いる．以下では，誤分類に対するコスト制御のパラメータ ν の値が 0.1，0.3，0.5 の 3 つの場合に対して実験を行う．(その他のパラメータ設定に関しては，gBoost における固定値 maxpat=10，ϵ=0.01 を用いる．)

初めに，探索パスの本数パラメータ K を変化させた時の目的関数の最終値の推移を図 4.1 に示す．横軸は K，縦軸は目的関数の最終値である．K の刻み幅は，提案手法 1，2 に対しては 10 とする．提案手法 3 は，アルゴリズム上，子ノードの gain と bound を全て計算する必要があるため，提案手法 1，2 と比べて探索ノード数が多くなる．したがっ

データ名	グラフ数	平均ノード数	平均エッジ数	正例	負例
CPDB	684	25.2	25.6	341	343
Mutag	188	26.3	28.1	125	63

表 4.1: 使用したデータセット



(a) $\nu = 0.1$ (b) $\nu = 0.3$ (c) $\nu = 0.5$

図 4.1: 実験 1-1（K - 目的関数）

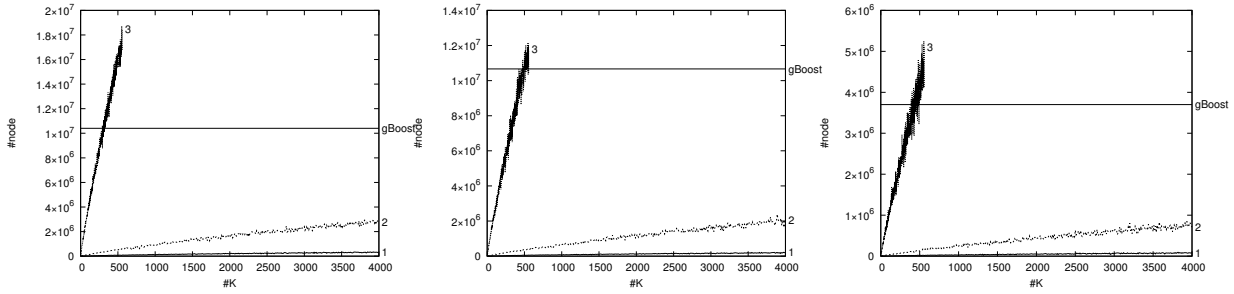
て，K の刻み幅は 1 とする．

次に，K の数を変化させた時の探索ノード数の推移を図 4.2 に示す．横軸は K，縦軸は探索ノード数である．K の刻み幅は，前述したものと同じ値とする．

最後に，K に対する目的関数と探索ノード数の実験から，各提案手法の探索ノード数に対する目的関数の最終値の推移を図 4.3 に示す．横軸は探索ノード数，縦軸は目的関数の最終値である．

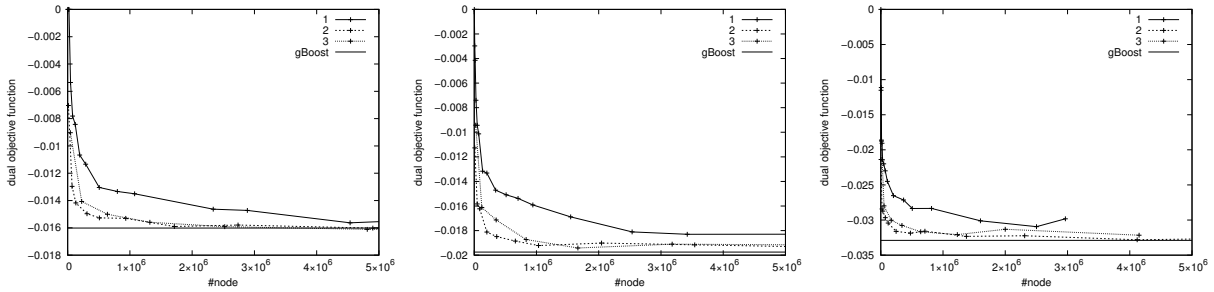
4.3 実験 2：乱数の影響に関する実験

提案手法では，乱数を使用しているため，結果に分散が生じる．したがって，実験 2 では CPDB データに対して各提案手法が乱数にどれだけ影響を受けるのかを調べる．K の数を 10，100，1000 とし，異なるシード値を与えて 10 回実行した時の各提案手法での目的関数の最終値の平均，分散，探索ノード数の平均値を比較する． ν の値を 0.3 に固定し，表 4.2 に結果を示す．



(a) $\nu = 0.1$ (b) $\nu = 0.3$ (c) $\nu = 0.5$

図 4.2: 実験 1-2（K - 探索ノード数）



(a) $\nu = 0.1$ (b) $\nu = 0.3$ (c) $\nu = 0.5$

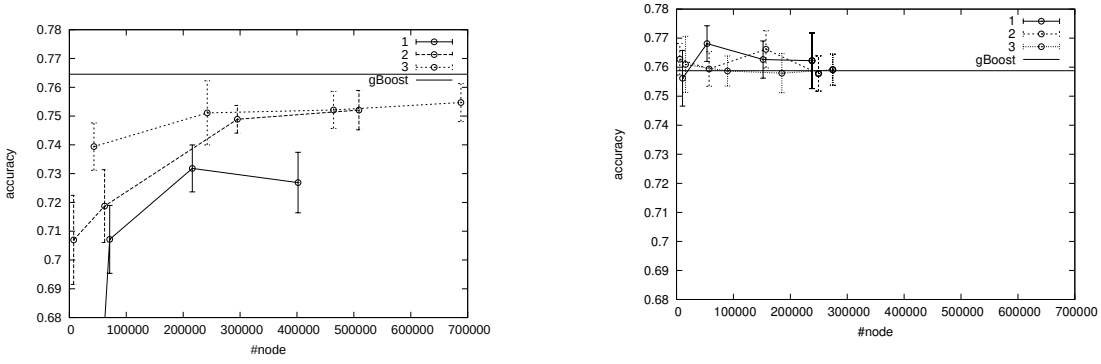
図 4.3: 実験 1-3（探索ノード数 - 目的関数）

4.4 実験 3：モデルの予測精度に関する実験

実験 3 では，既存手法と 3 つの提案手法を用いて構築された予測モデルの実際の予測精度を比較する．データには CPDB，Mutag の 2 つのデータを利用する．既存手法では精度が 10 分割交差検証の正答率の値で測られる．したがって，確率的探索を行う各提案手法では 10 分割交差検証を 10 回行うことで，正答率の平均値，分散値，探索ノード数の平均値を求め，各手法の精度を比較する． ν の値が，0.01，0.1，0.2，0.3，0.4，0.5，0.6 の場合に対して実験を行なう． 横軸を探索ノード数，縦軸を正答率の値とし，図 4.4，図 4.5 に結果を示す．

K 10	目的関数平均	分散	探索ノード数平均
提案手法 1	0	0	1625.0
提案手法 2	-0.011543	3.0385e-04	9000.8
提案手法 3	-0.017377	1.7890e-04	277280.6
K 100	目的関数平均	分散	探索ノード数平均
提案手法 1	-0.004994	2.6913e-04	16253.6
提案手法 2	-0.016711	9.4498e-05	86857.9
提案手法 3	-0.019121	6.4197e-05	2456806.0
K 1000	目的関数平均	分散	探索ノード数平均
提案手法 1	-0.011567	2.8003e-04	84839.9
提案手法 2	-0.018501	8.2502e-05	647241.9
提案手法 3	-0.019604	6.2834e-05	19284961.9

表 4.2: 実験 2

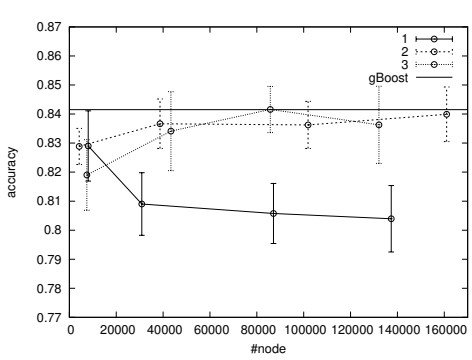


(a) gBoost の最良 ν パラメータを用いた場合　(b) 各手法で ν パラメータのチューニングを行なった場合

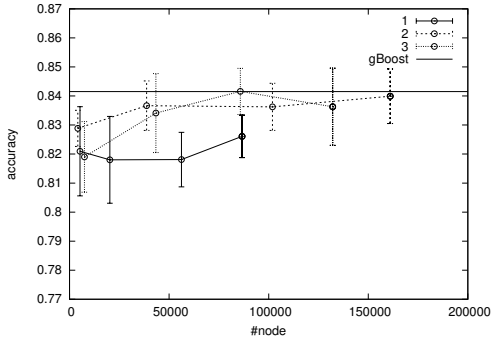
図 4.4: 実験 3（探索ノード数 - 正答率）CPDB

4.5 実験 4：探索空間の変更に関する実験

提案手法では，各階層での探索の偏りを防ぐために，既存手法で利用する探索木を拡大した探索木を用いている．実験 4 では CPDB データに対して，探索木を変更したことによる探索精度の影響を調べる． ν の値を 0.3 に固定し，探索ノード数に対する目的関数の結果を，数既存手法で用いる探索木を利用した場合と，提案手法で変更した探索木を利用した場合とで比較する．横軸を探索ノード数，縦軸を目的関数の最終値とし，図 4.6 に結果を示す.

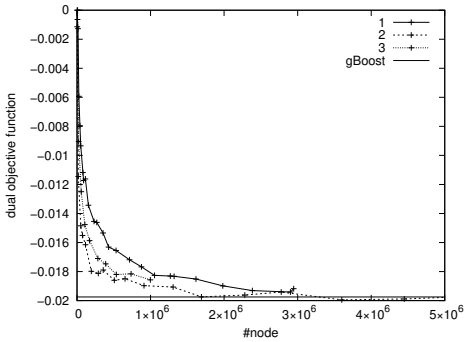


(a) gBoost の最良 ν パラメータを用いた場合

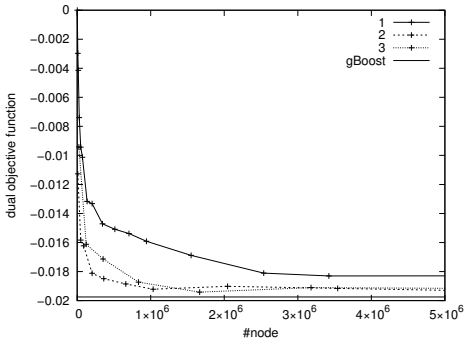


(b) 各手法で ν パラメータのチューニングを行なった場合

図 4.5: 実験 3（探索ノード数 - 正答率）Mutag



(a) 既存の探索木 (DFS コード木) を用いた場合



(b) 変更した探索木を用いた場合

図 4.6: 実験 4 探索木の変更による影響

第5章

考察

5.1　実験1：探索の精度，効率に関する実験

図4.1から見て取れるように，同数の探索パスを伸ばす場合には，提案手法3，2，1の順に目的関数の値が良い値となった．この理由としては，図4.2のように，ランダムに探索パスを伸ばすよりもグラフの特徴を利用した方が広く探索を行えるからであると考えられる．図4.2において，提案手法3のノード数が他の手法より莫大に多くなるのは，前述したとおり子ノードのboundを利用して確率を振るためには予め全ての子ノードを探索する必要があるためである．また，手法3が既存手法の探索ノード数を上回る理由としては，既存手法の探索木から同型判定を除外したため，既存手法よりも探索空間が広くなっているためであると考えられる．

次に各提案手法の優位性を図4.3より考える．探索の効率として，探索ノード数に応じた目的関数の最終値を考えると，提案手法1が他と比べて少し劣る結果となった．これには2つの理由が予想される．1つ目は，提案手法1は，2と3に比べて探索パスに対する探索ノード数が少ない（図4.2）．したがって，他の手法と同程度のノード数を探索するためには，探索パスの数を増やす必要がある．ここで問題であるのが，本探索空間は同型判定を除外しているため，同じ部分グラフを表現するノードが複数存在しうることである．探索パスの数が多い時は少ない時に比べて，この同じ部分グラフを表現する異なるノードにたどり着く確率が大きくなる．これらのノードは子孫の空間が同じであるため，探索が無駄になる可能性が生じる．したがって，同探索ノード数での目的関数の値が他の手法より劣る結果になったのではないかと予想される．2つ目は，提案手法2，

3 に関しては，訓練データ集合の特徴を利用した探索手法であるのに対し，提案手法 1 ではグラフの特徴を利用しないナイーブな手法であるという点が挙げられる．この点より，提案手法 1 の探索空間よりも，他 2 つの探索空間のほうが目的関数に寄与する特徴を含みやすいのではないかと考えられる．以上の 2 つの理由から，提案手法 1 が他の手法より劣る結果となったと考る．提案手法 2 と 3 に関しては，同程度の探索効率であることが見て取れる．

5.2 実験 2：乱数の影響に関する実験

探索ノード数を基準として分散を考える．表 4.2 より，$K100$ の提案手法 2 と $K1000$ の提案手法 1 を比較すると，同程度の探索ノード数であるが，分散値は提案手法 2 のほうが小さい（$K10$ 場合の提案手法 1 は目的関数の最終値が 0 であるため学習が行われなかったものとし除外する）．また，$K10$ の提案手法 3 と $K100$ の提案手法 2 を比較すると，提案手法 2 のほうが探索ノード数が少ないにもかかわらず，小さい分散値をとる．以上より，提案手法 2 が最も分散の少ない手法であることがわかる．提案手法 2 は他の手法と比べて，毎回の特徴探索に同じ確率を用いているのに加え確率に偏りがあることから，探索ノードに差が生じにくい．したがって，乱数に大きく影響を受けなかったと考る．

5.3 実験 3：モデルの予測精度に関する実験

図 4.4，図 4.5 より各手法の予測精度を比較する．既存手法の最良 ν パラメータを用いた結果と各種法で ν パラメータのチューニングを行なった結果を比較すると，正答率に差が生じることから，既存手法と提案手法の最良の ν は異なる値をとることが分かった．また，既存手法での探索ノード数の平均値が，CPDB の場合 17,834,167.0，Mutag の場合 2,270,063.5 であることを考慮すると，各手法共に，大きく探索を削減しつつ，既存手法と同等以上の精度を出すことが分かる．精度の上昇に関しては，探索を確率的に行なうことで汎化性能が出たものと考える．

5.4　実験 4：探索空間の変更に関する実験

図 4.6 で二つのグラフにおいて同探索ノード数での目的関数の値を比較すると，既存の探索木を用いる場合の方が良い値となった．この理由としては，各深さでの探索の偏りを防ぐことよりも、探索木を変更することによって生じる同じ部分グラフを複数回探索する恐れがあるという問題点の方が，探索の精度に関して大きく影響を与えたためだと考えられる．

第6章

結論と今後の展望

本研究では，確率的に部分グラフ特徴空間を探索することで，特徴探索を効率化する手法を提案した．その結果，既存手法と比較して，精度の低下なしに大幅な特徴量探索数の削減を達成した．しかし，用いた探索木の問題点として，同一部分グラフを表現するノードが複数存在するという点が挙げられる．したがって，今後の展望としては，探索木構築の際に同型判定を取り除かずに，gSpan の辞書順最小ではないノードが出現した際には辞書順最小のノードにエッジを繋ぐ，つまり既存手法の探索木を DAG 状の探索空間（図 6.1）に拡張することで解決したい．また，既存手法と提案手法ではハイパーパラメータチューニングに関して差が生じたため，各手法のハイパーパラメータに関する特徴も考察したい．最後に，本研究では学習モデルとして gBoost を拡張したが，確率的探索を行う場合は Random Forest [10] や Extra Trees [11] などのアンサンブルモデルにおいても，本提案手法の探索が利用できると考えられるため，他のモデルへの応用も検討する．

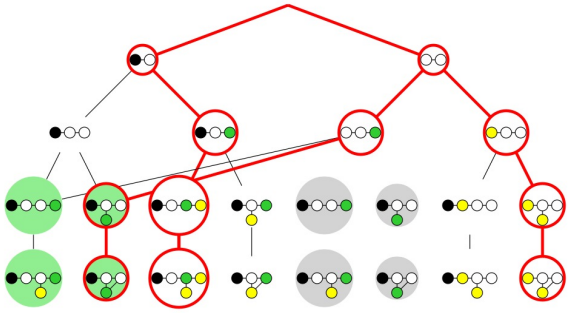


図 6.1: DAG 状の探索空間

謝辞

本研究を行うにあたり，基礎知識や研究アイデア，方向性など全ての面でご指導いただきました瀧川一学准教授，また，研究面のみならず幅広い知識や学生としての姿勢などをご教授いたいただきました湊真一教授には，心より感謝申し上げます．ありがとうございます．加えて，研究にあたって様々な知識やアイデアをいただいた皆様，研究や日常生活を含め様々な面で支えていただいた先輩方，研究室の皆様にも深く感謝いたします．最後に，日常生活を支え，研究に集中できる環境を作っていただいた家族に感謝いたします．

参考文献

[1] Taku Kudo, Eisaku Maeda, and Yuji Matsumoto. An application of boosting to graph classification. In *NIPS*, pp. 729–736, 2004.

[2] Yan Karklin, Richard F. Meraz, and Stephen R. Holbrook. Classification of non-coding RNA using graph representations of secondary structure. In *Pacific Symposium on Biocomputing*, pp. 5–16. World Scientific, 2005.

[3] Ichigaku Takigawa and Hiroshi Mamitsuka. Graph mining: procedure, application to drug discovery and recent advances. *Drug Discovery Today*, Vol. 18, No. 1âĂŞ2, pp. 50 – 57, 2013.

[4] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, Vol. 12, pp. 2539–2561, 2011.

[5] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, Vol. 50, No. 5, pp. 742–754, 2010.

[6] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. gboost: a mathematical programming approach to graph classification and regression. *Machine Learning*, Vol. 75, No. 1, pp. 69–89, 2009.

[7] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, Vol. 4, No. 1, pp. 1–43, 2012.

[8] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, Vol. 46, No. 1-3, pp. 225–254, 2002.

[9] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *ICDM*, pp. 721–724. IEEE Computer Society, 2002.

[10] Leo Breiman. Random forests. *Machine Learning*, Vol. 45, No. 1, pp. 5–32, 2001.

[11] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, Vol. 63, No. 1, pp. 3–42, Apr 2006.