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 of subgraph feature sets in graph classification

白川　稜 [1*]　　岡崎　文哉 [2]　　瀧川　一学 [2,3]
Ryo Shirakawa[1]　Fumiya Okazaki[2]　Ichigaku Takigawa2,3

[1] 北海道大学 工学部
[1] School of Engineering, Hokkaido University
[2] 北海道大学 大学院 情報科学研究科
[2] Graduate School of Information Science and Technology, Hokkaido University
[3] 科学技術振興機構, さきがけ
[3] JST, PRESTO

Abstract: In the graph classification problem, the presence of a subgraph is often used as a feature. The total number of subgraphs in an input graph can be huge, and complete enumeration is practically intractable. An existing method, gBoost, can efficiently perform feature search and classifier learning at the same time to consider all subgraphs as feature candidates without explicitly enumerating them. However, the exact search for relevant subgraphs requires still large computational cost. In this paper, we propose three types of probabilistic search for subgraphs as an alternative to the exact search to improve computational efficiency for graph classification problem. We also empirically evaluate them by using real datasets.

1　はじめに

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量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量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，探索における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．

2　既存手法

本章では，既存手法である gBoost 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，グラフの線形分類モデルから始め，既存手法の説明をする．

*連絡先： 北海道大学 工学部
〒 060-0814 札幌市北区北 14 条西 9 丁目
E-mail: sira@art.ist.hokudai.ac.jp

2.1 グラフ線形分類モデル

まず，グラフに対する線形分類モデルについて説明する．グラフ分類問題に用いる特徴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として用いる．

入力として ℓ 個のグラフとクラスラベル $\{G_n, y_n\}_{n=1}^{\ell}, y_n \in \{+1, -1\}$ 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 0-1 特徴量 $x_t = \mathbb{I}(t \subseteq G_n), t \in \mathcal{T}$ を用いて $|\mathcal{T}|$ 次元のベクトル $\boldsymbol{x}_n$ として表現できる．ここで $\mathbb{I}(\cdot)$ は括弧内が真なら 1，偽なら 0 を返す関数，$t \subseteq G_n$ はグラフ t, G_n の部分グラフ同型を表す．この時，各部分グラフ特徴 t に対して弱学習器 h を以下で定義する．

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

ここで $\omega \in \Omega = \{-1, 1\}$ はパラメタである．この弱学習器を用いて，gBoost では以下の加法モデルを構築する．上記の h の定義より変数 x_t に関して線形のモデルになる．

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

$\alpha_{t,\omega}$ 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．2 値分類（+1 or -1）の場合，$f(x) \geq 0$ の時に 1，$f(x) \leq 0$ の時に-1 と予測する．

2.2 gBoost

gBoost では，LPBoost で定式化された線形計画問題を以下の双対問題に変換し，列生成法を用いて解くことで，先ほど定義した線形モデルの重み $\alpha_{t,\omega}$ の値を導く．

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

列生成法はこの双対問題においては，式 (1) における制約条件がゼロの状態から始め，以下で定義される gain の値が最大の特徴を順次制約に加えていくことで効率的に最適解を求める手法である．

$$g(t, \omega) = \sum_{n=1}^{\ell} \lambda_n^{(k)} y_n h(\boldsymbol{x}_n; t, \omega). \quad (2)$$

$\{\lambda_n^{(k)}\}_{n=1}^{\ell}$ は制約条件を k 個追加した場合の最適な $\boldsymbol{\lambda} = \{\lambda_n\}_{n=1}^{\ell}$ の値である．

gBoost における上記の (2) 式が最大となる部分グラフ特徴 t の探索を考える．特徴探索は gSpan[9] と呼ばれる頻出部分グラフ列挙法に基づいており，重複無しに全部分グラフを列挙した探索空間である DFS コード木を用いる．DFS コード木は，全部分グラフ指示子を持つ他に，子が親のスーパーグラフとなるという性質を持つ．この探索木を深さ優先的にたどり，各ノードにおいて gain を計算することで，最適な特徴を探索する．子が親のスーパーグラフとなる性質を利用すると，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 の値 $b(t)$ は，現探索部分グラフを t，そのスーパーグラフを t' とすると，以下の式で計算できる．

$$b(t) = \max\{2 \sum_{\{n|y_n=+1, t\subseteq G_n\}} \lambda_n^{(k)} - \sum_{n=1}^{\ell} y_n \lambda_n^{(k)},$$
$$2 \sum_{\{n|y_n=-1, t\subseteq G_n\}} \lambda_n^{(k)} - \sum_{n=1}^{\ell} y_n \lambda_n^{(k)}\}. \quad (3)$$

また，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．

$$g^* > b(t).$$

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．

3 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，本稿では，厳密探索ではなく，確率的に探索を行う手法を提案する．

3.1 探索空間

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探索空間を用いる．つまり，提案手法は，親ノードに辺を 1 本追加した部分グラフが全て子ノードとなるような，DFS コード木を拡張した探索木を利用し，特徴探索を行う．この探索空間の変更の詳細は 3.3 において言及する．

3.2 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 パス探索を定義する．

定義 1 (探索パス)**.** 根ノードを始点とし, 子ノードを列挙する. 列挙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. 子ノードの列挙, 選択の動作を葉ノードに達するまで繰り返す. 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.

定義 2 (K パス探索)**.** K 本の探索パスによってカバーされるノード集合における特徴探索を K パス探索と定義する.

以下では K パス探索を用いた，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 x と対応する部分グラフを同一視する.

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:     x ← 根ノード
4:     Grow(x, K)
5: end procedure
6: function Grow(x, K)
7:     x に対する gain g(x), bound b(x) を計算
8:     if b(x) < g* then
9:         return
10:    end if
11:    if g* < g(x) then
12:        t* ← x, ω* ← ω, g* ← g(x)
13:    end if
14:    if x が葉ノード then
15:        return
16:    end if
17:    C = {x 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} ← 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
```

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) および (3) で計算 (探索) し, 既存手法と同様にして bound の値で探索の枝刈りを行う. gain の値がこれまでの最良値である場合には保存する. その後, 渡されたノードが葉ノードではない場合には子ノードを列挙し (C), 列挙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 つの確率設定手法を提案する.

3.2.1 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.

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

3.2.2 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ほう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.

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

3.2.3 提案手法 3：bound の情報を加えた確率設定

本節では，式 (3) 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 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.

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

本手法は bound の値を用いるため，予め全ての子ノードを探索する必要がある．したがって，提案手法 1,2 に比べて 1 本の探索パスを伸ばすときの探索ノード数が多くなることが予想される.

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 1: 使用したデータセット

3.3 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．当アルゴリズムにおいて辞書順最小のものは出現場所に偏りがあるため，DFS コード木は木全体の形に偏りが生じるという特徴を持つ．

ここで問題となるのが，DFS コード木を確率的に探索する場合，探索木の形が偏っているという理由から，提案手法 1 のように一様に確率を設定する場合においても同階層のノードへの探索確率に大きな差が生じてしまうということである．

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，木全体の偏りを緩和し，以上のような探索確率の差が軽減するよう試みる．

4 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 2 つを使用した．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．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 (1) 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 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，

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 2: 実験 2

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 2 に示す．横軸は K，縦軸は探索ノード数とある．K の刻み幅は，前述したものと同じ値とする．

最後に，前述した 2 つの実験から，各提案手法の探索ノード数に対して，目的関数の最終値の推移を図 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 10，100，1000，10000 とし，乱数に 10 個のシード値を与え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 2 に結果を示す.

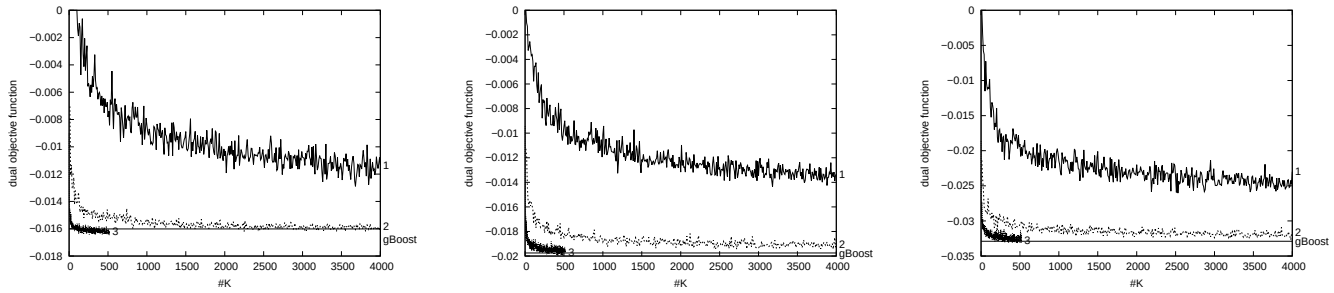
4.4 実験 3：モデルの予測精度に関する実験

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．既存手法の精度が 10 分割交差検証の正答率の値で測られるため，各提案手法では 10 分割交差検証を 10 回行い，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．パラメータ ν の設定に関しては gBoost における最良値 (CPDB:0.01,Mutag:0.2) を利用する．横軸を探索ノード数，縦軸を正答率の値とし，図 4 に結果を示す.

5 考察

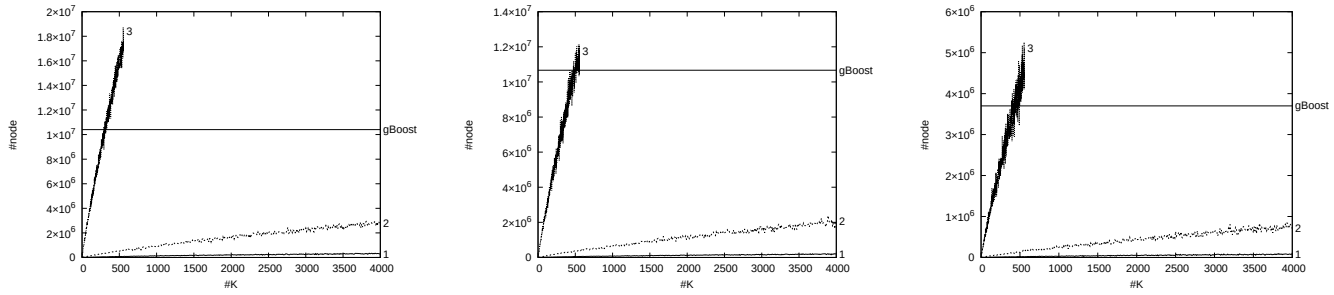
5.1 実験 1

図 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



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

図 1: 実験 1-1（K - 目的関数）



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

図 2: 実験 1-2（K - 探索ノード数）

なった．この理由としては，図 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 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．

図 1，図 2 より全ての提案手法において，既存手法よりも効率的に探索を行うことが可能である．しかし，提案手法 1 に関しては，目的関数の最終値が既存手法の値と比べて少し離れる結果となった．

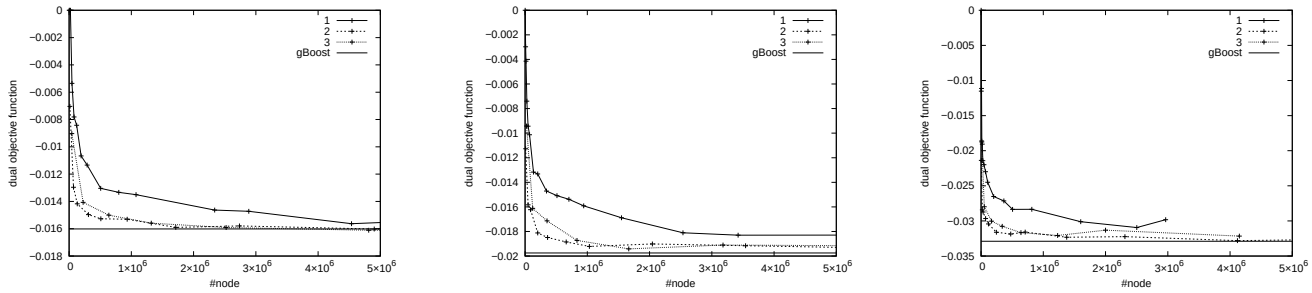
次に各提案手法の優位性を図 3 より考える．探索の効率として，探索ノード数に応じた目的関数の最終値を考えると，提案手法 1 が他と比べて少し劣る結果となった．これには 2 つの理由が予想される．1 つ目は，提案手法 1 は，2 と 3 に比べて探索パスに対する探索ノード数が少ない (図 2)．したがって，他の手法と同程度のノード数を探索するためには，探索パスの数を増やす必要がある．ここで問題であるのが，本探索空間は同型判定を除外しているため，同じ部分グラフを表現するノードが複数存在しうることである．探索パスの数が多い時は少ない時に比べて，この同じ部分グラフを表現する異なるノードにたどり着く確率が大きくなる．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．2 つ目は，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という点が考えられる．以上の理由から，提案手法 1 が他の手法より劣る結果となったと考える．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

探索ノード数を基準として分散を考える．表 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えられる．

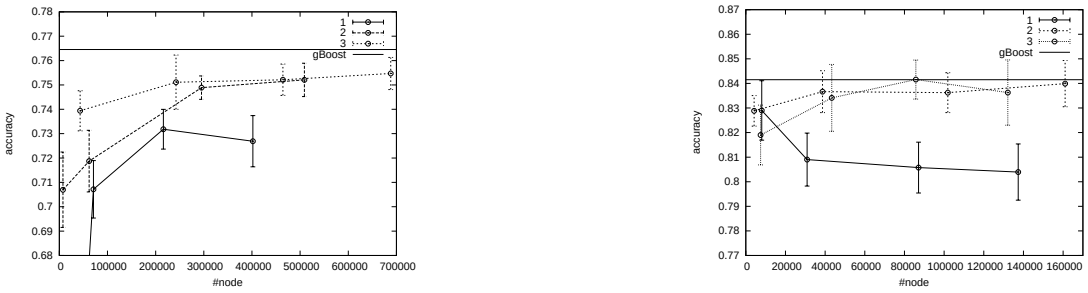
5.3 実験 3

図 4 より各手法の予測精度を比較する．既存手法の 10 分割交差検証での探索ノード数平均値が，CPDB の場合 17,834,167.0，Mutag の場合 2,270,063.5 であるため，各手法共に，大きく探索を削減しつつ，大幅な精度の減少なしに予測モデルを構築することができている．各手法の比較としては，提案手法 1 が他の手法に比べ少し劣る結果となった．これに関しては，実験 1 の結果をもとに，提案手法 1 が他の手法よりうまく探索を行えなかったことが考え



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

図 3: 実験 1-3（探索ノード数 - 目的関数）



(a) CPDB (b) Mutag

図 4: 実験 3（探索ノード数 - 正答率）

られる．また，予測精度の分散値を比較すると，実験 2 と同様に提案手法 2 が他より少し小さい値をとった．

6 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，特徴探索の効率化手法を提案した．その結果，既存手法と比較して，大幅な特徴量探索数の削減を達成した．ここで，探索パスのパラメータ K が大きくなると gBoost で探索した特徴数を超える場合がある．これは，同型判定除外における探索空間の増大が影響しているためである．今後の展望として，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 状の探索空間に拡張することで解決したい．また，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．

謝辞

本研究は JSPS 科研費 17H01783，17K19953 および JST さきがけの助成を受けたものです．

参考文献

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

[2] Y. Karklin, R. F. Meraz, and S. 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] I. Takigawa and H. Mamitsuka. Graph mining: procedure, application to drug discovery and recent advances. *Drug Discovery Today*, Vol. 18, No. 1âĂŞ2, pp. 50 – 57, 2013.

[4] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, Vol. 12, pp. 2539–2561, 2011.

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

[6] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. gboost: a mathematical programming approach to graph classification and regression. *Machine Learning*, Vol. 75, No. 1, pp. 69–89, 2009.

[7] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. P. Liebana, S. Samothrakis, and S. 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] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, Vol. 46, No. 1-3, pp. 225–254, 2002.

[9] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pp. 721–724, 2002.

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

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