SIG-SWO-037-03

RDF グラフの冗長な特徴表現に対するカーネル関数

A Kernel Function for Redundant Features from RDF Graphs

荒井 大地 [1*]　兼岩 憲 [2]
Daichi Arai[1]　Ken Kaneiwa[2]

[1] 電気通信大学 情報理工学部 情報・通信工学科
[1] Department of Communication Engineering and Informatics,
Faculty of Informatics and Engineering, The University of Electro-Communications
[2] 電気通信大学大学院 情報理工学研究科 情報・通信工学専攻
[2] Department of Communication Engineering and Informatics,
Graduate School of Informatics and Engineering, The University of Electro-Communications

Abstract: Machine learning on RDF data has become important in the field of the Semantic Web. However, RDF graph structures are redundantly represented by noisy and incomplete data on the Web. In order to apply SVMs to such RDF data, we propose a kernel function to compute the similarity between resources on RDF graphs. This kernel function is defined by selected features on RDF paths that eliminate the redundancy on RDF graphs. Our experiments show the performance of the proposed kernel with SVMs on binary classification tasks for RDF resources.

1　はじめに

近年，Web 上のデータの増加に伴い，それらのデータを用いた機械学習が盛んに研究されている．機械学習アルゴリズムには，決定木，ベイズ学習，ニューラルネットワーク，SVM 等がある．特にデータ点を線形分離するための学習器である SVM（サポートベクターマシン）は，汎化性能や計算効率の観点で高く評価されている．さらに SVM は，適切なカーネル関数を与えることで非線形分離問題にも対応できる．SVM を効率良く解くためのアルゴリズムは既に示されており，実質的な問題はカーネル関数の設計と言える．

セマンティック Web 技術の 1 つとして，Web 上の構造化されたデータを記述する枠組みに，RDF（Resource Description Framework）がある．これは Web 上のリソース間の関係を表現する，グラフ構造をもつ機械可読なデータである．これにより，Web 上の膨大な情報がコンピュータでも解読できるようになる．特に RDF グラフを用いた検索，推論や学習に関しては，セマンティック Web の分野で広く研究されている [1–3]．

RDF のようなグラフ構造データに対して SVM を適用する場合，カーネル関数が必要となる．Web のリンク構造に対して，木構造カーネルを用いた SVM [4] が提案されている．しかし，RDF グラフは膨大な情報をもつので，単純な木構造データに対するカーネル関数では不十分である．また，RDF グラフは既存のデータベースから低コストで大量に自動生成できる代わりに，誤ったデータを含む，あるいはあるべきデータが生成されないことがあり得る．このように冗長で誤りを含んだ RDF グラフに対して，機械学習を適用するのは容易ではない．

本研究では，RDF で記述されたリソースから冗長な特徴を削減するカーネル関数を設計する．これを SVM に適用することで，RDF グラフ上のリソースを様々な性質の有無で分類する SVM の精度と計算効率を向上できる．この分類器は，自動生成に対する欠落データの補完や誤データの修正，手動生成の支援に応用できると考えられる．

2　準備

2.1　RDF グラフ

RDF トリプルは，主語 s（リソース），述語 p（プロパティ），目的語 o（プロパティ値）の三つ組 (s, p, o) である．RDF グラフは RDF トリプルの有限集合であり，主語と目的語を頂点，述語を主語から目的語への有向辺とした有向グラフである．これにより概念間の関係性を記述でき，構造化されたデータを形成する．

*連絡先：電気通信大学 情報理工学部 情報・通信工学科
〒 182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: arai@sw.cei.uec.ac.jp

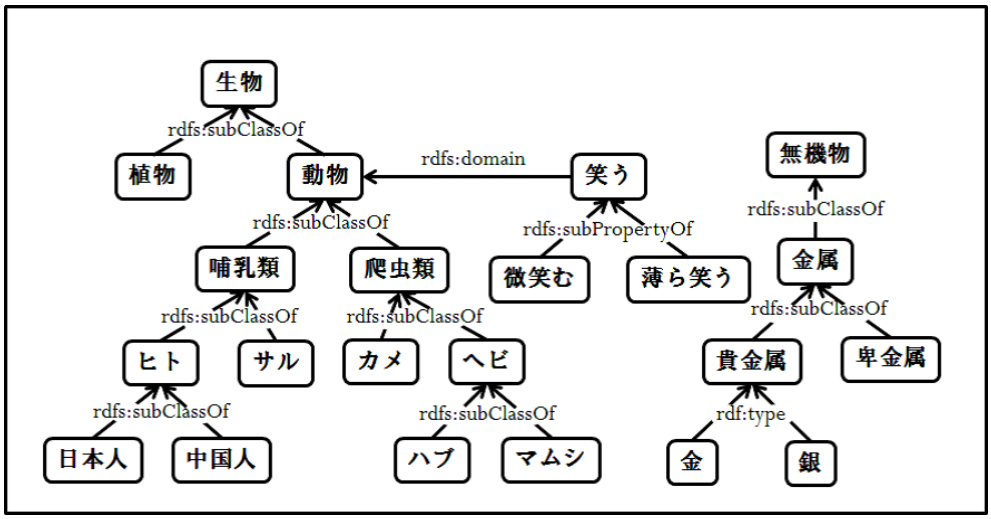

図 1: RDF グラフの例

RDF トリプル (s,p,o) に対し，プロパティ p の逆プロパティを表す述語を p^{-1} として，(o,p^{-1},s) を逆トリプルと呼ぶ．ある RDF グラフ G が与えられたとき，G の逆トリプルの集合を $G^{-1}=\{(o,p^{-1},s)\mid(s,p,o)\in G\}$ で表す．本研究では，RDF グラフ G は暗黙に G^{-1} を含むとする．プロパティ p または逆プロパティ p^{-1} を p^* で表す．ここで，$s,r_1,\ldots,r_{n-1},o$ をリソース，$p_1^*,\ldots,p_n^*$ をプロパティまたは逆プロパティとして RDF グラフ上の長さ n のパスを $(s,p_1^*,r_1,p_2^*,r_2,p_3^*,\ldots,r_{n-1},p_n^*,o)$ で表し，RDF パスと呼ぶ．このとき，RDF グラフは $(s,p_1^*,r_1),(r_1,p_2^*,r_2),\ldots,(r_{n-1},p_n^*,o)$ を含む．特に，長さ 0 の RDF パスは (s,s) となる．

図 1 の RDF グラフにおいて,「ヒト」を始点とする RDF パスの例を以下に示す．

長さ 0 : (ヒト, ヒト)
長さ 1 : (ヒト, sc, 哺乳類)
長さ 2 : (ヒト, sc, 哺乳類, sc, 動物)
長さ 3 : (ヒト, sc, 哺乳類, sc, 動物, sc, 生物)
長さ 3 : (ヒト, sc, 哺乳類, sc, 動物, sc^{-1}, 爬虫類)

但し，sc は rdfs:subClassOf の略記である．

2.2 カーネル関数

高次元空間における内積にカーネル関数を利用した機械学習の手法をカーネル法と呼ぶ．カーネル関数とは，与えられたデータ点の高次元な内積を返す関数である．すなわち，データ点を $\boldsymbol{x},\boldsymbol{x'}\in X$，データ点に対する高次元空間への写像を ϕ とすれば，カーネル関数 $K:X\times X\to\mathbb{R}$ は以下のように表される．

$$K(\boldsymbol{x},\boldsymbol{x'})=\phi(\boldsymbol{x})\cdot\phi(\boldsymbol{x'})$$

一般的には高次元空間への写像を明示せず，直接カーネル値を計算できるように設計する．これにより，データ点の内積の形をもつ学習の最適化問題に対し，高次元空間における内積を高速に取得できる．また，内積を類似度としても設計できる．

2.3 非線形 SVM

SVM は与えられたデータ点の集合に対し，マージンを最大化する分離超平面を求める線形分離器である．データ点とクラスラベルを $(\boldsymbol{x}^{(i)},y^{(i)})$，分離超平面を $f(\boldsymbol{x})=\boldsymbol{w}\cdot\boldsymbol{x}-b$，各訓練データの誤分類の許容度を $C\xi_i$ とすれば，以下の二次計画問題で表される [5].

$$\begin{aligned}\min.\quad & \frac{1}{2}\boldsymbol{w}^2+C\sum_i\xi_i\\ \text{s.t.}\quad & y^{(i)}(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}-b)\geq 1-\xi_i,\\ & \xi_i\geq 0.\end{aligned}$$

但し，実際は $f(\boldsymbol{x})=\sum_i\alpha_i y^{(i)}\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}-b$ として，以下の双対問題によって解決する．

$$\begin{aligned}\max.\quad & -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(j)}+\sum_i\alpha_i\\ \text{s.t.}\quad & \sum_i\alpha_i y^{(i)}=0,\ 0\leq\alpha_i\leq C\end{aligned}$$

ここで，あるカーネル関数 K を導入すれば，$f(\boldsymbol{x})=\sum_i\alpha_i y^{(i)}K(\boldsymbol{x}^{(i)},\boldsymbol{x})-b$ となり，最適化問題は以下のよ

うになる．

$$\max. \quad -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}K(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})+\sum_i \alpha_i$$
$$\text{s.t.} \quad \sum_i \alpha_i y^{(i)}=0,\ 0\le\alpha_i\le C$$

これにより，高次元空間でデータ点を線形分離することになり，元の次元において非線形分離できる．

3　RDF のカーネル関数

本研究では，学習の対象を RDF グラフに含まれるリソースの集合とする．そのリソースに対応するクラスラベルを与えて学習することで，新たなリソースに対する未知のクラスラベルを出力する．RDF データによるグラフ構造に対して SVM を適用するために，リソースに対するカーネル関数を設計する．

3.1　RDF パスからの特徴抽出

リソースに対するカーネル関数を設計するために，リソース間の類似度について考える．ここでは類似度の基準となる特徴として，リソースを始点とする RDF パスを用いる．しかし単純な比較だと，リソース自体の一致でしか類似度を測れないため，カーネル関数として不適切である．

本研究では，RDF パスから冗長なデータを取り除く方法を提案する．特徴に始点を含めると，それが一致するか否かを中心に見ることになり，リソースの特徴をうまく比較できない．特徴に終点と最後の有向辺を含めないと，その長さのパスを取る必要性がなくなる．したがって，特徴の最低限満たすべき制約は，始点を含まず終点または最後の有向辺を含むことである．この制約に基づき，以下の特徴抽出を考える（図 2）．

（１）RDF パスから始点を除く

（２）RDF パスから始点と終点を除く

（３）RDF パスから始点と内部頂点を除く

（４）RDF パスから頂点を除く

（５）RDF パスから始点と有向辺を除く

（６）RDF パスから始点と途中の有向辺と終点を除く

（７）RDF パスから終点以外を除く

（８）RDF パスから最後の有向辺以外を除く

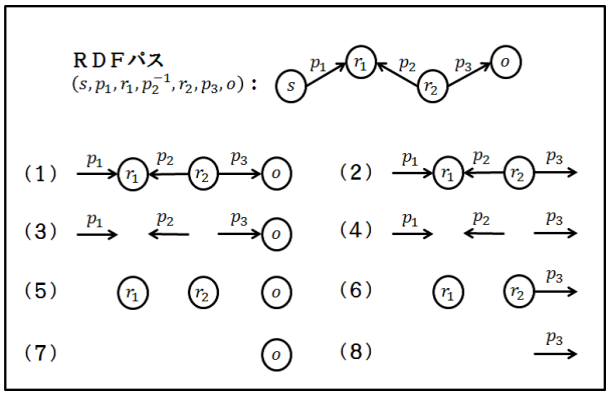

図 2: RDF パスから抽出される特徴

図 1 を例に，RDF パスから抽出された 8 つの特徴を説明する．学習対象のリソースを「ヒト」,「サル」,「カメ」とし，胎生か否かで分類する場合,「ヒト」と「サル」のみが「哺乳類」という性質で一致していることが分かれば良い．その際,（１）～（３）,（５）～（７）は問題ないが,（４）,（８）はそもそもリソースの違いを判別できない．学習対象のリソースを「ヒト」,「サル」,「カメ」,「貴金属」とし，呼吸をするか否かで分類する場合,「ヒト」,「サル」,「カメ」のみが「動物」という性質で一致していることが分かれば良い．その際,（３）,（７）は問題ないが,（１）,（２）,（５）,（６）は「哺乳類」と「爬虫類」の違いにより,「ヒト」と「サル」に対して「カメ」を全く異なるリソースとして捉えてしまう．学習対象のリソースを「ヒト」,「サル」,「カメ」,「微笑む」とし，呼吸をするか否かで分類する場合,「微笑む」が rdfs:subClassOf ではない述語によって「動物」へ到達することが分かれば良い．その際,（３）は問題ないが,（７）は述語の区別がつかず「カメ」と「微笑む」を同じものとして捉えてしまう．以上の考察から,（３）のみがリソースの妥当な特徴と言える．

3.2　PRO カーネル

前節の特徴抽出を採用して，カーネル関数を設計する．RDF パス $(s,p_1^*,r_1,p_2^*,r_2,p_3^*,\ldots,r_{n-1},p_n^*,o)$ から始点と内部頂点を除いたとき，$(p_1^*,\ldots,p_n^*,o)$ と表す．これを，リソース s を始点とする長さ n の PRO（もしくはリソース s の PRO）と呼ぶ．本研究では，2 つのリソース r, r' を始点とする PRO の一致数に応じて r と r' の類似度を決定する．

最初に，PRO に対するカーネル関数を設計する．RDF グラフ G 上の 2 つの PRO f, f' に対する PRO カーネ

ルを以下に示す．

$$K_{pro}(f, f') = \begin{cases} 1 & \text{if } f = f' \\ 0 & \text{otherwise} \end{cases}$$

2 つの PRO が等しければ 1，そうでなければ 0 を返す．

3.3 リソースカーネル

PRO の長さを i に指定したリソースカーネルを設計する．RDF グラフ G 上の 2 つのリソース r, r' の長さ i の PRO のみを特徴として，リソースカーネルを以下に定める．但し，$F_n(G, r)$ は RDF グラフ G 上のあるリソース r を始点とする長さ n の PRO 集合である．

$$\begin{aligned} K^i_{res}(r, r') &= \frac{1}{C_1} \sum_{\substack{f \in F_i(G,r) \\ f' \in F_i(G,r')}} K_{pro}(f, f') \\ C_1 &= \max\{|F_i(G, r)|, |F_i(G, r')|\} \end{aligned}$$

これにより，リソース r, r' の長さ i の PRO の一致数が多いほど，それらの類似度は高くなる．このとき，r, r' の長さ i の PRO 集合の最大要素数 C_1 で正規化する．

特徴として使用する PRO の最大長を m に制限したリソースカーネルを設計する．RDF グラフ G 上の 2 つのリソース r, r' の長さ m 以下の全ての PRO を特徴としたリソースカーネルを以下に定める．

$$\begin{aligned} K^{\leq m}_{res}(r, r') &= \frac{1}{C_2} \sum_{0 \leq i \leq m} \frac{1}{i+1} K^i_{res}(r, r') \\ C_2 &= \sum_{0 \leq j \leq m} \frac{1}{j+1} \end{aligned}$$

これにより，リソース r, r' の長さ m 以下の PRO の一致数による類似度を返す．但し，特徴となる PRO が長いほど始点と終点のリソースの関係性が薄れ，類似度への影響は少なくなると考えられる．そのため，右辺の各リソースカーネルに重みとして $\frac{1}{i+1}$ を与える．また，右辺の各重みつきリソースカーネルの和の最大値 C_2 で正規化する．

3.4 情報利得率による PRO の削減

リソースカーネルは，(i) RDF パスを抽象化して PRO を抽出し，(ii) PRO の長さを制限する，2 つの手順で求められる．本節では情報利得率 [6] を用いて，リソースから生成される PRO 集合から冗長な特徴を削除する．これにより学習の計算コストや過学習を抑えることが期待できる．

ある RDF グラフ G を与えたとき，その中で学習対象となる全てのリソースの集合を $\boldsymbol{R}_G$ とする．$\boldsymbol{R}_G$ に出現する正例と負例のリソースの集合をそれぞれ $\boldsymbol{R}^+_G$ と $\boldsymbol{R}^-_G$ で表す．すなわち $\boldsymbol{R}_G = \boldsymbol{R}^+_G \cup \boldsymbol{R}^-_G$ となる．また，$\boldsymbol{R}_G$ に出現する，G から PRO f を生成できるリソースの集合を $\boldsymbol{R}^f_G = \{r \in \boldsymbol{R}_G \mid f \in F(G, r)\}$，生成できないリソースの集合を $\boldsymbol{R}^{-f}_G = \boldsymbol{R}_G \setminus \boldsymbol{R}^f_G$ とする．すなわち $\boldsymbol{R}_G = \boldsymbol{R}^f_G \cup \boldsymbol{R}^{-f}_G$ となる．

リソース集合 $\boldsymbol{R}_G$ に対する $\boldsymbol{R}^+_G$ と $\boldsymbol{R}^-_G$ の要素数の偏り表す情報量 $\mathit{Info}(\boldsymbol{R}_G)$，$\boldsymbol{R}^f_G$，$\boldsymbol{R}^{-f}_G$ の情報量の平均 $\mathit{Info}_f(\boldsymbol{R}_G)$，$\boldsymbol{R}_G$ に対する $\boldsymbol{R}^f_G$ と $\boldsymbol{R}^{-f}_G$ の要素数の偏りを表す分割情報量 $\mathit{SplitInfo}_f(\boldsymbol{R}_G)$ を以下のように定める．

$$\begin{aligned} \mathit{Info}(\boldsymbol{R}_G) &= -\sum_{c \in \{+,-\}} \frac{|\boldsymbol{R}^c_G|}{|\boldsymbol{R}_G|} \log_2 \frac{|\boldsymbol{R}^c_G|}{|\boldsymbol{R}_G|} \\ \mathit{Info}_f(\boldsymbol{R}_G) &= \sum_{x \in \{f,-f\}} \frac{|\boldsymbol{R}^x_G|}{|\boldsymbol{R}_G|} \mathit{Info}(\boldsymbol{R}^x_G) \\ \mathit{SplitInfo}_f(\boldsymbol{R}_G) &= -\sum_{x \in \{f,-f\}} \frac{|\boldsymbol{R}^x_G|}{|\boldsymbol{R}_G|} \log_2 \frac{|\boldsymbol{R}^x_G|}{|\boldsymbol{R}_G|} \end{aligned}$$

また，$\boldsymbol{R}_G$ を $\boldsymbol{R}^f_G$ と $\boldsymbol{R}^{-f}_G$ に分割した場合の情報量の減少量を表す情報利得 $\mathit{Gain}_{\boldsymbol{R}_G}(f)$，情報利得を正規化した情報利得率 $\mathit{GainRate}_{\boldsymbol{R}_G}(f)$ を以下のように定める．

$$\begin{aligned} \mathit{Gain}_{\boldsymbol{R}_G}(f) &= \mathit{Info}(\boldsymbol{R}_G) - \mathit{Info}_f(\boldsymbol{R}_G) \\ \mathit{GainRate}_{\boldsymbol{R}_G}(f) &= \frac{\mathit{Gain}_{\boldsymbol{R}_G}(f)}{\mathit{SplitInfo}_f(\boldsymbol{R}_G)} \end{aligned}$$

PRO f の情報利得率 $\mathit{GainRate}_{\boldsymbol{R}_G}(f)$ が高い場合，$\boldsymbol{R}_G$ より $\boldsymbol{R}^f_G$，$\boldsymbol{R}^{-f}_G$ の方がクラスラベルが大きく偏っているので，f は分類を判定する特徴に有用である．

有益な PRO をフィルタリングするため，情報利得率のボーダーを ϵ とする．$F_n(G, r)$ から重要度の低い特徴を削減した $F_n(G, r)^-$ を以下に定義する．

$$F_n(G, r)^- = \{f \in F_n(G, r) \mid \mathit{GainRate}_{\boldsymbol{R}_G}(f) \geq \epsilon\}$$

これをリソースカーネルの計算に用いるには，$K^i_{res}(r, r')$ 内の $K_{pro}(f, f')$ の和に現れる $F_i(G, r)$ と $F_i(G, r')$ を $F_i(G, r)^-$ と $F_i(G, r')^-$ に書き換える．

4 実験

DBpediaJapanese 上の wikiPageWikiLink をプロパティとする RDF データ（約 7GB）を用いて分類実験を行う．ここでは日本の国立大学を正例，私立大学を負例としてそれぞれ 50 リソースずつランダムに選択して分類し，10 分割交差検証により評価する．実験環境は，OS:Windows 8.1 Enterprise 64bit，CPU:Intel(R) Core(TM) i7-5820K @3.30GHz 3.30GHz，実装メモリ:64.0GB である．SVM の最適化問題を解くために

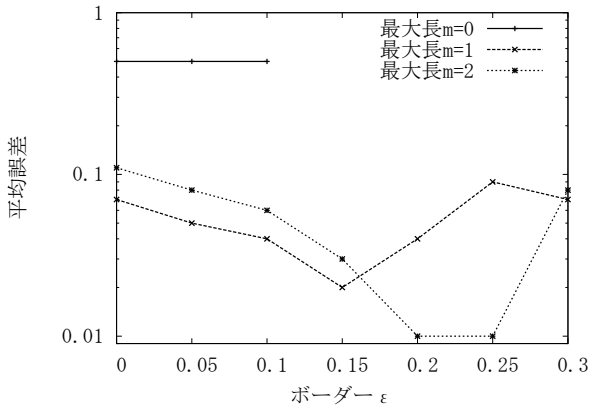

図 3: 各最大長のボーダーに対する平均誤差

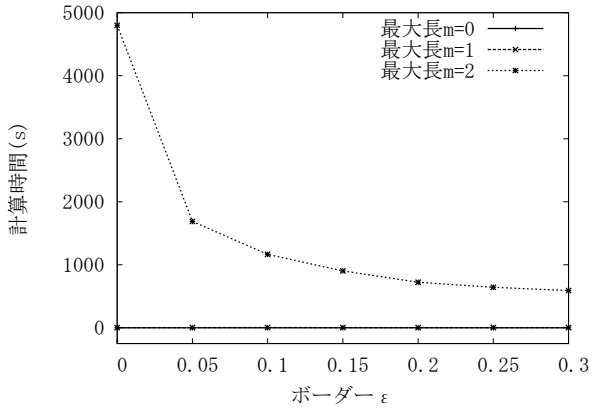

図 4: 各最大長のボーダーに対する計算時間

SMO（逐次最小最適化）アルゴリズムがある [7]．本研究ではこのアルゴリズムを採用し，Python によるプログラム [8] を参考にして，Java で実装する．

図 3，4 は，リソースカーネルの最大長と PRO の情報利得率のボーダーに対する平均誤差とカーネル行列の計算時間の結果である．

4.1 交差検証の平均誤差

最大長を固定したとき，各ボーダーに対する平均誤差は下に凸の関係である．最大長 0（リソースの一致のみが特徴）のとき，ボーダー 0.00 から 0.10 までの平均誤差は全て 0.50 である．これは完全にランダムに分類する場合と変わらず，最悪の精度である．また，ボーダー 0.15 から 0.30 に関しては，カーネル行列の全要素が 0 となり分類不可能であった．よって，最大長 0 のリソースカーネルは分類に不適切である．最大長が 1 のとき，ボーダー 0.15 で，平均誤差は最小の 0.02 となる．最大長が 2 のとき，ボーダー 0.20，0.25 で，平均誤差は最小の 0.01 となる．低いボーダーのとき，最大長 2 の方が最大長 1 よりも精度が低い．これは，最大長 2 の特徴が既に冗長な長さ 1 の PRO を含み，加えて冗長な長さ 2 の PRO も含むからだと考えられる．逆に高いボーダーのとき，最大長 2 の方が最大長 1 よりも精度が高い．これは，特徴から削減しすぎてしまった長さ 1 の PRO を長さ 2 の PRO で補っているためだと考えられる．本実験の結果により，最大長が大きく，その長さに見合った適度なボーダーが分類の精度を向上させることを示す．

4.2 カーネル行列の計算時間

最大長 0 の計算時間は約 60 ミリ秒，最大長 1 の計算時間は約 3.5 秒であり，ボーダーに対して一定である．これは，情報利得率による削減で減少する計算が，全体の計算よりもはるかに少ないためである．最大長 2 のとき，ボーダー 0.00 の計算時間は約 1 時間 20 分であり，ボーダー 0.05 で大幅に減少して約 28 分となり，それ以降も緩やかに減少している．これは，非常に小さいボーダーでも膨大な特徴を削減できることを表し，特徴削減の重要性を示している．またボーダーに対して計算時間が単調に減少していることから，精度を保つ限りはボーダーを上げるべきである．

5 まとめ

本研究では，RDF データの冗長な特徴の削減を行うために，RDF パスの PRO による抽象化，PRO の最大長の制限，情報利得率による PRO の削減を組み込んだリソースカーネルを提案した．分類実験の結果から，提案したリソースカーネルにより，分類精度の向上と計算時間の短縮が両立できることを示した．

今後の課題として，どのように適切な PRO の最大長と情報利得率のボーダーを取得するかという問題がある．現状，交差検証によって良い最大長とボーダーを見つけているが，離散的かつ多大な実行時間がかかるため，これを高速に解ける解析的または近似的手法が必要である．また多様なデータに対応するために，本研究で採用した PRO 以外にも柔軟に特徴を取り入れる等，カーネル関数の頑健性の向上が必要である．加えて，より長い PRO や大規模データに対応するために，カーネル値を高速に求めるアルゴリズムも検討中である．

参考文献

[1] P. Exner and P. Nugues. Entity extraction: From unstructured text to DBpedia RDF triples. In *The Web of Linked Entities Workshop (WoLE 2012)*, 2012.

[2] V. Bicer, T. Tran, and A. Gossen. Relational kernel machines for learning from graph-structured RDF data. In *The Semantic Web: Research and Applications*, pp. 47–62. Springer, 2011.

[3] N. Fanizzi, C. d'Amato, F. Esposito. Induction of robust classifiers for web ontologies through kernel machines. *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 11, pp. 1–13, 2012.

[4] 鹿島久嗣, 坂本比呂志, 小柳光生. 木構造データに対するカーネル関数の設計と解析. 人工知能学会論文誌, Vol. 21, No. 1, 2006.

[5] 高村大地. 言語処理のための機械学習入門, pp. 117–131. コロナ社, 2010.

[6] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[7] C. Platt, et al. Using analytic QP and sparseness to speed training of support vector machines. *Advances in neural information processing systems*, pp. 557–563, 1999.

[8] P. Harrington. *Machine learning in action*, pp. 101–128. Manning, 1st edition, 2012.