多層パーセプトロンによるPresentaion MathML式の分類

長尾　悠真†　　鈴木　伸崇††

† 筑波大学大学院図書館情報メディア研究科　〒305–8550 茨城県つくば市春日
†† 筑波大学図書館情報メディア系　〒305–8550 茨城県つくば市春日
E-mail: †ynagao@klis.tsukuba.ac.jp, ††nsuzuki@slis.tsukuba.ac.jp

あらまし　MathMLは数式を記述するためのマークアップ言語であり，数式の表記を表すPresentation Markup形式と，数式の意味を表すContent Markup形式が存在する．Webページ等に数式を記述する際に用いられているのはPresentation Markupであるが，数式を計算するにはContent Markupが適している．数式中のトークンは複数の意味を持ち得るため，Presentation MarkupからContent Markupへの一意な変換は困難であるが，数式が属する分野を特定できればより適切な変換が行えると考えられる．そこで本研究では，多層パーセプトロンを用いてPresentation Markup形式のMathML式を分類する手法を提案する．評価実験の結果，本手法を用いてMathML式の分類を高精度に行えることが示せた．

キーワード　MathML，分類，多層パーセプトロン

1. はじめに

MathML (Mathematical Markup Language) は数式を記述するためのマークアップ言語で，2014年4月にW3Cによってバージョン3.0第2版が勧告された．また，MathMLはHTML5に組み込まれており，Webページ上で数式を表現するための標準的な規格といえる．

MathMLには数式の表記を表すPresentation Markup (Presentation MathML) と，意味を表すContent Markup (Content MathML) の2種類が存在する．Webページ等に数式を記述する際に用いられ，普及しているのはPresentation MathMLであるが，数式を計算する際にはContent MathMLが用いられる．そのため，Presentation MathMLからContent MathMLへの変換が可能になればMathML式の柔軟な利用が可能になる．しかし，Presentation MathMLからContent MathMLへの一意な変換は困難である．これは，数式の記述においてトークンは複数の意味を持ち得るためである．例として2つの数式を挙げる．

$$C_n = \frac{1}{n+1}\binom{2n}{n} \tag{1}$$

$$C(x) = \int_0^x \cos(t^2)\,dt \tag{2}$$

(1) は n 番目のカタラン数を表している．C はPresentation MathMLでは`<mi>C</mi>`と，Content MathMLでは`<ci>Catalan</ci>`と表記される．(2) はフレネル積分を表している．(1) と同様に C はPresentation MathMLでは`<mi>C</mi>`と表記されるが，Content MathMLでは`<ci>FresnelC</ci>`と表記される．このような異なる2つの意味が異なるクラスに属することが分かれば，Presentation MathMLからContent MathMLへの変換に有用だと考えられる．The Wolfram Functions Site [1] のカテゴリでは，(1) はConstantsに，(2) はGamma, Beta, Erfに属し得る．数式 e に C という文字が含まれていた場合に，e がConstantsクラスに属することが分かれば，C はカタラン数を表していると考えられる．

このように，与えられた数式に対して，その数式が属するクラスを特定できれば有用であると考えられる．そこで本研究では，Presentation MathML式の分類を行う手法を提案する．本手法では，深層学習の一種である多層パーセプトロンを用いてMathML式の分類を実現する．MathML式は可変サイズの木であるのに対して，多層パーセプトロンは固定長の入力を必要とする．したがって，MathML式をそのまま多層パーセプトロンに入力することは困難である．そこで本手法では，MathML式をBinary Branch Vector [2] を用いて固定長のベクトルに変換し，多層パーセプトロンによる分類を実現している．

関連研究

Presentation MathML式分類の先行研究としてKimら [3] のものが挙げられる．MathML式から特徴抽出を行い，SVMによる分類で高い精度の数式分類を実現している．しかし，この手法では葉ノードに近い部分しか木構造を考慮していない．それに対し，本研究ではMathML式全体の木構造を考慮した分類を行う．また，SVMなどによって分類する際には分類対象のデータに応じて適切な特徴を人手で抽出・選択する必要があるが，多層パーセプトロンのような深層学習ではその必要がない．MathML式の形式変換に関する研究としては，石山ら [4] や大瀬戸ら [5] によるものがある．Nghiemら [6] はPresentation MathML式とその周辺のテキストを用いてPresentation MathMLからContent MathMLへの変換を行っている．一方，本研究では周辺のテキストを必要とせず，MathML式のみを学習する．また，[6] では変換の際の入力の一部として数式のクラス情報を既知のものとして入力しているため，本研究によってクラスを推定できれば有用であると考えられる．MathPlayer [7] は数式の読み上げソフトであり，対象とする分野をGeometry, Probability, Statistics等から手動で指定することができる．し

かし，分野の自動判別は行われていない．

2. 諸 定 義

本章では，MathML の概要と Binary Branch Vector，多層パーセプトロンについて述べる．

2.1 MathML の概要

MathML は数式を記述するための XML ベースのマークアップ言語である．MathML には 2 種類の記法が存在する．1 つ目は Presentation MathML である．これは，数式の表記を記述するために用いられる．2 つ目は Content MathML で，数式の意味構造を記述するために用いられる．図 1 と図 2 に Presentation MathML と Content MathML で x^2+1 を表したものを示す．Presentation MathML では `mn` 要素は数字を表すために，`mi` は識別子を表すために，`mo` は演算子を表すために用いられる．これらの要素は子としてテキストノードを持つ．例えば，“2” という数字を表すには`<mn>2</mn>`と記述する．

```
<math>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
  <mo>+</mo>
  <mn>1</mn>
</math>
```

図 1 Presentation MathML 式

```
<math>
  <apply>
    <plus/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <cn>1</cn>
  </apply>
</math>
```

図 2 Content MathML 式

XML 文書はランクなし順序木 (unranked ordered tree) として表され，MathML 式も同様にランクなし順序木として表される．ランクなし木はノードが任意の数の子を持ち，順序木は兄弟ノードの順序を区別する．ラベル付き順序木 T をノードの集合 N，エッジの集合 E，T のルートノード $Root(T)$ を用いて $T=(N,E,Root(T))$ とする．ノード u からその子ノード v へのエッジを $(u,v)\in E$ と表す．また，二分木 $B(T)$ を $B(T)=(N,E_l,E_r,Root(T))$ とする．ここで，E_l は親ノードから左の子ノードへのエッジの集合，E_r は親ノードから右の子ノードへのエッジの集合を表す．

2.2 Binary Branch Vector

Binary Branch Vector [2] はラベル付き順序木のベクトル表現の一種である．本来は 2 つの木の間の距離の計算コストの削減のために提案されたものであるが，我々は多層パーセプトロンの入力として用いる．

木 T の完全二分木 $B(T)$ を左 (右) のエッジの集合 E_l (E_r) を用いて $B(T)=(N,E_l,E_r,Root(T))$ とする．v が u の最初の子ノードならば $\langle u,v\rangle_l\in E_l$，2 番目の子ノードならば $\langle u,v\rangle_r\in E_r$ とする．

Binary Branch [2] はルートノードと左右の子ノードから成る，二分木の深さ 1 の部分木である．二分木 $B(T)$ 中のノード u に対する Binary Branch $BiB(u)=(N_u,E_{u_l},E_{u_r},Root(T_u))$ を以下の条件を満たす二分木と定義する．

- $N_u=\{u,u_1,u_2\}(u\in N;u_i\in N\bigcup\{\epsilon\},i=1,2)$,
- $E_{u_l}=\{\langle u,u_1\rangle_l\}$,
- $E_{u_r}=\{\langle u,u_2\rangle_r\}$,
- $Root(T_u)=u$.

木 T の Binary Branch Vector $BRV(T)$ はデータセット中の Binary Branch の順序集合 Γ，Γ のサイズ $|\Gamma|$，$B(T)$ における i 番目の Binary Branch の出現回数 b_i を用いて，$BRV(T)=(b_1,b_2,...,b_{|\Gamma|})$ と表される．

2.3 多層パーセプトロン

多層パーセプトロン (Multilayer Perceptron) はディープラーニングのモデルの一種である．多くの分類問題を解くために SVM が用いられてきたが，近年では CNN (Convolutional Neural Network) を始めとしたディープラーニングのモデルが SVM よりも成果を挙げている．本研究ではディープラーニングのモデルで MathML 式を分類する最初のステップとして，最もシンプルなディープラーニングのモデルである多層パーセプトロンを分類器として用いる．

多層パーセプトロンは入力層，複数の中間層，出力層で構成される．それぞれの層は複数の入力を受け取り 1 つの値を出力するユニットで構成される．$l+1$ 層の j 番目のユニットの総入力 $u_j^{(l+1)}$ は重み $w_{ji}^{(l)}$，バイアス $b_j^{(l+1)}$，$u_i^{(l)}$ の出力 $z_i^{(l)}$ を用いて，

$$u_j^{(l+1)}=\sum_i w_{ji}^{(l)}z_i^{(l)}+b_j^{(l+1)} \tag{3}$$

と表す．ここで，活性化関数 f を用いて u_i の出力 z_i を $z_i=f(u_i)$ と表す．活性化関数の代表的なものとして以下の式で定義される ReLU がある．

$$f(u)=\max(0,u). \tag{4}$$

多クラス分類においては，一般的にソフトマックス関数

$$z_k=\frac{\exp(u_k)}{\sum_i\exp(u_i)} \tag{5}$$

が出力層の活性化関数として用いられる．ソフトマックスの出力 z_k は入力がクラス C_k に属する確立を表す．重みとバイアスは出力と正解の間の誤差を最小化するように更新される．多クラス分類の場合，交差エントロピー誤差

$$E=-\sum_k t_k\log z_k \tag{6}$$

が一般的に用いられている．ここで，z_k は出力，t_k は正解の k 番目の成分の値である．誤差を最小化するために用いられる最適化関数としては SGD や Adam が典型的なものとして知られている．

3. MathML 式の分類手法

MathML 式はサイズ可変の木として表されるのに対して，多層パーセプトロンの入力としては固定次元のベクトルが必要である．そのため，MathML 式を固定次元のベクトルに変換する必要がある．そこで，順序木の固定次元ベクトル表現である Binary Branch Vector を用いて MathML 式をベクトル形式に変換する．図 3 に提案手法の流れを示す．

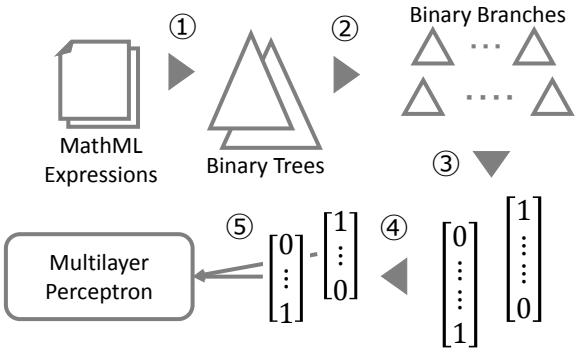

図 3　提案手法の流れ

提案手法は大きく分けて次の 5 ステップで構成される．

（1） MathML 式を二分木に変換．

（2） 二分木を Binary Branch に分割．

（3） Binary Branch を Binary Branch Vector に変換．

（4） Binary Branch Vector の次元を削減．

（5） (4) のベクトルを多層パーセプトロンに入力して分類．

ステップ 1 から 3 を 3.1 節で，ステップ 4 を 3.2 章で，ステップ 5 を 3.3 章で説明する．

3.1 ベクトルの構築

MathML 式は可変長の木として表される一方，多層パーセプトロンや SVM などの分類器は入力として固定長のベクトルを必要とする．そのため，MathML 式の分類のためには固定長のベクトルに変換することが必要である．本節では MathML 式を Binary Branch Vector に変換する手順を説明する．

T をランクなし順序木とする．Binary Branch Vector $BRV(T)$ は二分木 $B(T)$ から得られる．一般に，二分木 $B(T)$ は T から以下のような手順で得られる．初期状態として，$B(T)$ は T と同じノード集合を持ち，エッジを持たない．

（1） T におけるすべてのノード u に対して以下の操作を行う．

（a） $v_1, v_2, ..., v_n$ を u の子ノードとする．$B(T)$ に $2 \leqq i \leqq n$ でエッジ (v_{i-1}, v_i) を追加する．

（b） エッジ (u, v_1) を $B(T)$ に追加する．

（2） $B(T)$ のすべての内部ノードが 2 つの子ノードを持つように空ノード ϵ を挿入する．

Algorithm 1 にベクトル構築アルゴリズムを示す．このアルゴリズムでは MathML 式の探索と Binary Branch Vector の構築を同時に行う．[2] で提案されたオリジナルの Binary Branch Vector の構築アルゴリズムでは多層パーセプトロンの入力とする際には必要のない木の距離の計算のための処理を含んでいる．そのため，提案手法ではこの処理を省略している．与えられた MathML 式のデータセット D において，全ての MathML 式を探索し，*VectorConstruction* 関数によって連想配列 H に Binary Branch の出現回数を記録する．H のキーは Binary Branch のタプルであり，それぞれの要素は文字列として (“親ノードのラベル”，“左の子ノードのラベル”，“右の子ノードのラベル”) のように格納する．H の値はデータセット中の MathML 式の数のサイズの配列である．それぞれの要素は Binary Branch の出現回数である．*Traverse* 関数は MathML 式を再帰的に探索し，Binary Branch の出現回数を H に格納する．この関数では *first* と *next* はそれぞれカレントノードの最初の子ノードと右の弟ノードである．もし，カレントノードが子要素または弟ノードを持たなければ，$first$ または $next$ に空ノード ϵ を代入する．ここで，ϵ のラベルもまた ϵ とする．そして，i 番目の MathML 式における Binary Branch の出現回数を数える．H を構築後，H の値を $BRVs$ に挿入し，行ベクトルを得るために転置したものを返す．

Algorithm 1 Constructing binary branch vectors

Input:
A set D of MathML expressions
Output:
Binary branch vectors $BRVs$

```
1: function VectorConstruction(D)
2:     Initialize an associative array H to be empty
3:     i ⇐ 0
4:     for all MathML expression e in D do
5:         root ⇐ root node of e
6:         Traverse(i, root, H)
7:         i++
8:     end for
9:     for all value in H do
10:        Push value to BRVs
11:    end for
12:    return transposition of BRVs
13: end function

14: function traverse(i, node, H)
15:    first ⇐ first child of node
16:    next ⇐ next sibling of node
17:    H[(node, first, next)][i]++
18:    for all child in children of node do
19:        traverse(i, child, H)
20:    end for
21: end function
```

図 4 に例を挙げる．データセット中には MathML 式 T_1 と T_2 から変換された 2 つの二分木 $B(T_1)$ と $B(T_2)$ がある．図の下部にある表は各 MathML 式における Binary Branch の出現回数を表している．例えば，$(1, 1)$ 成分は Binary Branch (math, mn, ϵ) が $B(T_1)$ に 1 回出現することを示している．各行が各二分木に対する Binary Branch Vector を

表している．すなわち，$BRV(T_1) = (1,1,0,1,0,1,0,1,0)$，$BRV(T_2) = (1,0,1,0,1,0,1,0,1)$ である．

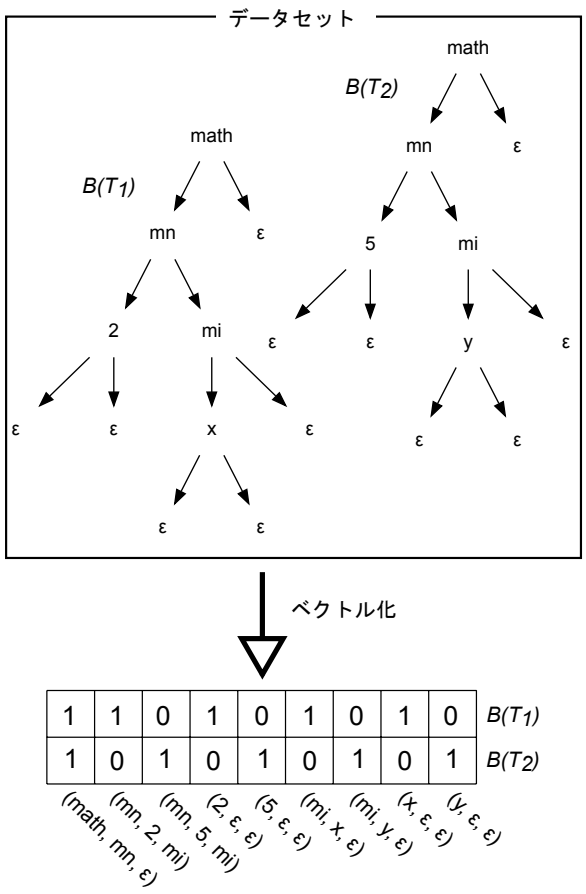

図 4　MathML 式の Binary Branch Vector への変換

3.2　ベクトルの次元削減

Binary Branch Vector のそれぞれの要素はデータセット中の Binary Branch の出現回数を表している．そのため，データセットが大きくなるにつれて，Γ のサイズが大きくなる．特に，MathML 式が構造は似ているが異なる数字を持っている Binary Branch の数が増えると，Γ のサイズが大きくなる傾向にある．このような構造の異なりは分類の正解率を下げる可能性がある．これは，類似した MathML 式は類似した構造を持っている傾向があるためである．したがって，提案手法では `mn` 要素のテキストを消去することによって，このような類似した構造を 1 つにまとめ，Binary Branch Vector の次元削減を行う．

Algorithm 2 に次元削減ベクトル構築アルゴリズムを示す．このアルゴリズムにおける *VectorConstruction* 関数は Algorithm 1 と同様のものであるため，Algorithm 2 では省略した．Algorithm 2 内の *traverse* 関数は Algorithm 1 のものと同様に e を探索し，Binary Branch の出現回数を H に格納する．主な違いは，`mn` 要素の処理である．カレントノード *node* のラベルが “mn” であれば，i 番目の MathML 式の Binary Branch $(node, \epsilon, next)$ と（“ mn ”, ϵ, ϵ）の出現回数を 1 増やす．それ以外の場合には，Algorithm 1 と同様の方法で *node* の子ノードを探索する．

図 5 に次元削減処理の例を挙げる．図 4 のデータセット中の

Algorithm 2 Constructing dimensionality reduced vector

Input:
A tree id i,
A node of the tree *node*,
An associative array H

```
1: function TRAVERSE(i, node, H)
2:     first ⇐ first child of node
3:     next ⇐ next sibling of node
4:     if label of node is "mn" then
5:         H[(node, ε, next)][i] + +
6:         H[("mn", "ϵ", "ϵ")][i] + +
7:     else
8:         H[(node, first, next)][i] + +
9:         for all child in children of node do
10:            TRAVERSE(i, child, H)
11:        end for
12:    end if
13: end function
```

Binary Branch を考える．`mn` 要素のテキスト (2 と 5) を ϵ に置き換える．その結果，親ノードが `mn` の Binary Branch が同じ Binary Branch に変換され，図 5 の 2 つの二分木 $B(T_1)'$ と $B(T_2)'$ が得られる．その結果，図 4 の Binary Branch Vector と比較すると，図 5 のものはサイズが小さくなっている．

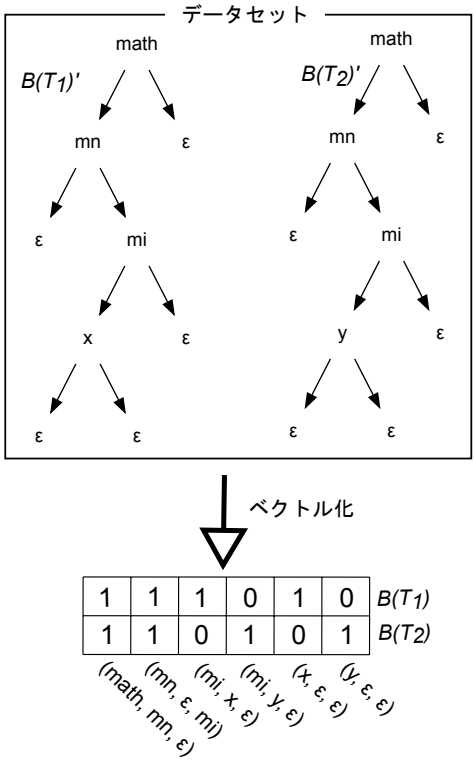

図 5　`mn` 要素のテキストを空ノード ϵ に置き換えることによる次元削減処理

3.3　学習モデル

提案手法では中間層 2 層の多層パーセプトロンを用いる．各中間層のユニット数は 512 とした．入力層，中間層の活性化関数として ReLU 関数を用い，出力層の活性化関数にはソフト

表 1　Kim ら [3] による MathML 式の特徴量

特徴	説明
Tag	要素名
Operator	mo 要素で表される演算子
Identifier	mi 要素で表される識別子
String	mtext 要素で表される文字列の 2-gram
Identifier & Operator (I&O)	識別子と演算子の 2-gram

マックス関数を用いる．学習時の最適化関数としては Adam を用いている．

4. 評 価 実 験

4.1 概　　要

評価実験を以下の (1) から (5) の手順で行う．

（1） データセット中の MathML 式を全て Binary Branch Vector に変換する

（2） (1) のベクトルと元の MathML 式が属するクラスの組を教師データ:評価データ=7:3 に分割する

（3） 教師データを分類器に学習させる

（4） 評価データを分類器にかけ，出力値が最も大きいクラスを割り当て，実際のクラスと一致するかを求める

（5） (4) を全ての評価データで行い，正解率を計算する

また，Kim ら [3] の SVM による MathML 式分類手法との正解率の比較を行う．Kim らは MathML 式の特徴として 1 のような特徴の組合せを用い，TF/IDF で重み付けをしている．彼らの評価実験の結果，Tags + Operators + Strings + I&Os が最も正解率が高く，次に Tags + Operators + Identifiers + I&Os の組合せが高いとされている．しかし，本評価実験で用いる The Wolfram Function Site (後述) では，mtext 要素が数式の見た目を調整するための空白にのみ用いられているため，今回は Tags + Operators + Identifiers + I&Os の組合せを比較対象とする．データセットから特徴ベクトルを構築し，提案手法と同様の手順で正解率を求める．また，提案手法の評価においても同様の理由で MathML 式から mtext 要素を消去し，Binary Branch Vector には mtext 要素とそのテキストを含めない. [3] にはペナルティ C の値が明記されていないため，本評価実験では $C = 2^{-10}, 2^{-9}, ..., 2^{10}$ で実験し比較した．

4.2 データセット

評価実験では，データセットとして The Wolfram Function Site [1] と MREC [8] を用いる．

4.2.1 The Wolfram Functions Site

The Wolfram Functions Site から網羅的に HTML ファイルを収集し，MathML 式を抽出する．抽出した MathML 式は Presentation Markup 形式であるが，semantics 要素や annotation-xml 要素によって Content Markup が埋め込まれている．そのため，annotation-xml 要素を消去し，semantics 要素の消去と子孫要素のネストの解除を行う．表 2 に MathML 式の属するクラスとその件数を示す．

4.2.2 MREC

MREC は arxiv.org の論文の一部を XML に変換したデータセットであり，数式は Presentation MathML で記述されている．MREC には MREC2011.3 と MREC2011.4 の 2 種類がある．後者には Content MathML が付与されているが，前者には付与されていない．提案手法では Content MathML によって明示された意味を用いないため，評価実験には前者を用いる．

MREC は変数や数字のみからなる短い MathML 式を含む．そのため，に mi 要素を持たないまたは math，mrow 以外の要素を 2 つ以下しか持たない MathML 式を除外した．MREC は 34 クラスから成るが，100,000 件以上の数式をもつ 20 クラスを抽出した．また，このクラスの間には親子関係や類似関係をもつものがある．例えば，Mathematics (math) は Algebraic Geometry (alg-geom), Differential Geometry (dg-ga), Quantum Algebra (q-alg) の親クラスである．そのようなクラスをまとめた結果，以下の 12 クラスになった．

（1） Astrophysics (astro-ph)
（2） Computer Science (cs)
（3） Condensed Matter (cond-mat)
（4） General Relativity and Quantum Cosmology (gr-qc)
（5） High Energy Physics - Experiment (hep-ex),
High Energy Physics - Lattice (hep-lat),
High Energy Physics - Theory (hep-th)
（6） Algebraic Geometry (alg-geom),
Differential Geometry (dg-ga),
Mathematics (math),
Quantum Algebra (q-alg)
（7） Mathematical Physics (math-ph)
（8） Chaotic Dynamics (chao-dyn),
Nonlinear Sciences (nlin),
Exactly Solvable and Integrable Systems (solv-int)
（9） Nuclear Experiment (nucl-ex),
Nuclear Theory (nucl-th)
（10） Physics (physics)
（11） Quantitative Biology (q-bio)
（12） Quantum Physics (quant-ph)

評価実験ではこの 12 クラスからそれぞれ 100,000 件ずつ，合計 1,200,000 件を用いる．

4.3 実 験 結 果

The Wolfram Functions Site における結果を表 3 に示す．提案手法は SVM($C = 2^8$) のものと比べ，同等以上の分類性能を示した．また，次元削減処理を施したものとそうでないものを比較すると，次元削減処理を施したものの方が正解率が高い．次元削減前のベクトルの次元数が 558,777 なのに対し，削減後

表 2　クラスと件数

クラス ID	クラス名	件数
0	Bessel-Type Functions	5,583
1	Complex Components	689
2	Constants	735
3	Elementary Functions	61,454
4	Elliptic Functions	7,248
5	Elliptic Integrals	1,236
6	Gamma, Beta, Erf	5,009
7	Generalized Functions	280
8	Hypergeometric Functions	218,241
9	Integer Functions	1,966
10	Mathieu & Spheroidal Functions	388
11	Number Theory Functions	904
12	Polynomials	2,372
13	Zeta Functions & Polylogarithms	1,571
	合計	307,676

のものは 2,184 であり大きく情報量を落としているが，次元削減処理による正解率の低下は見られなかった．

The Wolfram Functions Site はクラスごとの数式の数に偏りがあり，Elementary Functions (ID 3) と Hypergeometric Functions (ID 8) が 9 割を占める．そこで，詳細な結果を表 4 に示す．提案手法は従来の SVM を用いた手法に比べ F 値において 14 クラス中 13 クラスで同等以上であった．

表 3　The Wolfram Functions Site における各手法の正解率

手法	正解率
提案手法 (次元削減後)	**99.35**
提案手法 (次元削減前)	99.24
SVM [3]	99.03

MREC における結果を表 5 に示す．提案手法は次元削減処理の有無に関わらず従来手法 ($C = 2^0$) よりも高い正解率であった．しかしながら，The Wolfram Functions Site と比較するとどの手法も低い正解率である．これは，MREC の数式は複数著者によって記述されたものであるため，変数の文字や変数の順番に統一性がなく，整っていないことが一因として考えられる．この結果から提案手法は従来手法よりも整っていない数式に対してロバストであると言える．

表 4　提案手法と SVM ベースの手法 [3] における F 値

	F 値	
クラス ID	提案手法 (次元削減後)	SVM
0	0.97	0.97
1	**0.85**	0.80
2	**0.87**	0.85
3	0.99	0.99
4	0.99	0.99
5	**0.97**	0.93
6	**0.95**	0.94
7	**0.94**	0.90
8	0.99	0.99
9	**0.94**	0.93
10	0.95	0.95
11	**0.96**	0.92
12	0.85	**0.86**
13	**0.95**	0.94

表 5　MREC における各手法の正解率

手法	正解率
提案手法 (次元削減後)	**83.54**
提案手法 (次元削減前)	33.75
SVM [3]	28.16

5. むすび

本研究では，多層パーセプトロンを用いて Presentation Markup 形式の MathML 式を分類する手法を提案した．評価実験により提案手法は高精度に MathML 式を分類できることや正解率を落とさずに次元削減を行えることを示した．

今後の課題として，MREC において次元削減前後で正解率が大きく異なっている原因の解明が挙げられる．

文　献

[1] Wolfram Research, “The wolfram functions site”. http://functions.wolfram.com/

[2] R. Yang, P. Kalnis, and A.K.H. Tung, “Similarity evaluation on tree-structured data,” Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp.754–765, 2005.

[3] S. KIM, S. YANG, and Y. KO, “Classifying mathematical expressions written in mathml,” IEICE Transactions on Information and Systems, vol.E95.D, no.10, pp.2560–2563, 2012.

[4] 石山寿子，高野文子，佐藤浩史，原　俊介，大武信之，“Xml における数式の表示形式から意味形式への変換,” 電子情報通信学会技術研究報告. ET, 教育工学，vol.101，no.506，pp.23–30，2001.

[5] 大瀬戸良輔，甲斐　博，“数式データベースを用いた曖昧な数式の発見,” 第 73 回全国大会講演論文集，vol.2011，no.1，pp.723–724，2011.

[6] M.-Q. Nghiem, G.Y. Kristianto, G. Topić, and A. Aizawa, “A hybrid approach for semantic enrichment of mathml mathematical expressions,” Proceedings of the 2013 International Conference on Intelligent Computer Mathematics, pp.278–287, 2013.

[7] Design Science, “Mathplayer”. https://www.dessci.com/en/products/mathplayer/

[8] M. Líška, P. Sojka, M. Růžička, and P. Mravec, “Web interface and collection for mathematical retrieval: Webmias and mrec,” Towards a Digital Mathematics Library., eds. by P. Sojka and T. Bouche, pp.77–84, Masaryk University, Bertinoro, Italy, July 2011.