Vol.2016-DBS-163 No.7
Vol.2016-IFAT-123 No.7
2016/9/13

多層パーセプトロンによるMathML式の分類

長尾 悠真[1,a)]　鈴木 伸崇[2]

概要：MathML (Mathematical Markup Language) には数式の表記を表す 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，普及しているのは Presentation Markup であるが，数式を計算するには Content Markup が適している．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．また，数式の読み上げを行う際にも，その数式が属する分野が分かればより正確な読み上げが可能となる．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．

Classification of MathML Formulas Using Multilayer Perceptron

Yuma Nagao[1,a)]　Nobutaka Suzuki[2]

Abstract: MathML (Mathematical Markup Language) consists of two sets of elements: Presentation Markup and Content Markup. The former is more widely used to display math formulas in Web pages, while the latter is more suited to the calculation of math formulas. In general, a math formula in Presentation Markup cannot be uniquely converted into the corresponding formula in Content Markup. For a given math formula, if the class that the formula belongs to can be identified automatically, such conversions can be done more appropriately. Moreover, identifying the class of a given math formula is useful for text-to-speech of math formula. In this paper, we propose a method for classifying math formulas in Presentation Markup by using multilayer perceptron. Experimental results show that our method classifies math formulas with high accuracy.

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 と，意味を表す Content Markup 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 Markup であるが，数式を計算する際には Content Markup が用いられる．そのため，Presentation Markup から Content Markup 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 Markup から Content Markup への一意な変換は困難である．例えば，${}_nC_r = \frac{n!}{(n-r)!}$ の C という文字が積分定数であるのかコンビネーションを表すのかは数式自体に意味が付与されている訳ではないため，数式を見てどちらの意味かを推測する必要がある．ここで，この数式が場合の数と確率の式ということが分かれば C はコンビネーションと推測でき，適切な変換が行えると考えられる．また，数式の読み上げを行う際にも，その数式が属するクラス (分野) が分かればより正確な読み上げが可能になると考えられる．

このように，与えられた数式に対して，その数式が属す

[1] 筑波大学大学院図書館情報メディア研究科
Graduate School of Library, Information and Media Studies, University of Tsukuba
[2] 筑波大学図書館情報メディア系
Faculty of Library, Information and Media Science, University of Tsukuba
[a)] ynagao@klis.tsukuba.ac.jp

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 Markup 形式の 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[1] 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

MathML 式分類の先行研究として Kim ら [2] 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 [3] や大瀬戸ら [4] によるものがある．Nghiem ら [5] は Presentation Markup 形式の MathML 式とその周辺のテキストを用いて Presentation Markup から Content Markup 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．MathPlayer[6] 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. MathML 式のベクトル表現

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．以下に MathML 式を Binary Branch Vector に変換する手順を述べる．

2.1 ランク無し順序木の二分木化

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) 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．

```
<math>
  <mfrac>
    <mn>1</mn>
    <mi>x</mi>
  <mo>+</mo>
  <mn>3</mn>
</math>
```

図 1　変換前の MathML 式
Fig. 1　MathML formula before conversion

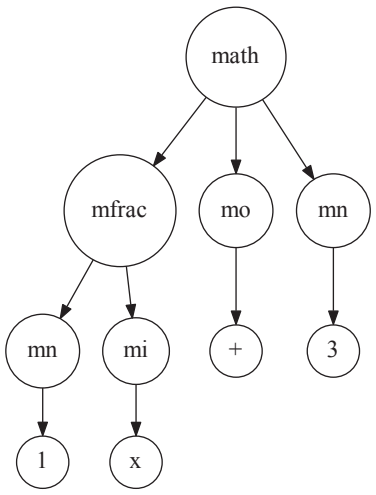

図 2　変換前の MathML 式の木構造
Fig. 2　Tree structure of MathML formula before conversion

MathML 式は，子要素数に制限のないランク無し順序木 (unranked ordered tree) として表される．これを完全二分木に変換する．完全二分木とは全てのノードが 0 または 2 つの子ノードを持つ二分木である．ランク無し順序木に対して，以下の (1) から (3) の手順で変換を行う．

(1) ノード v_n が親ノード u の $n(>1)$ 番目の子ノードであるとき，v_{n-1} から v_n にエッジ (v_{n-1}, v_n) を張る．すなわち，木の中の全ての隣り合う兄弟ノードに対して，兄から弟へのエッジで繋ぐ
(2) エッジ (u, v_1) と (1) で張ったものを除いた全てのエッジ消去
(3) 全てのノードが 0 または 2 つの子ノードを持つように子ノード数が 1 以下のノードに対して，子ノードが欠落している部分に ϵ を挿入

例として $\frac{1}{x} + 3$ の MathML 式を**図 1** に，それを木構造で表したものを**図 2** に，これを完全二分木に変換したものを**図 3** に示す．

Vol.2016-DBS-163 No.7
Vol.2016-IFAT-123 No.7
2016/9/13

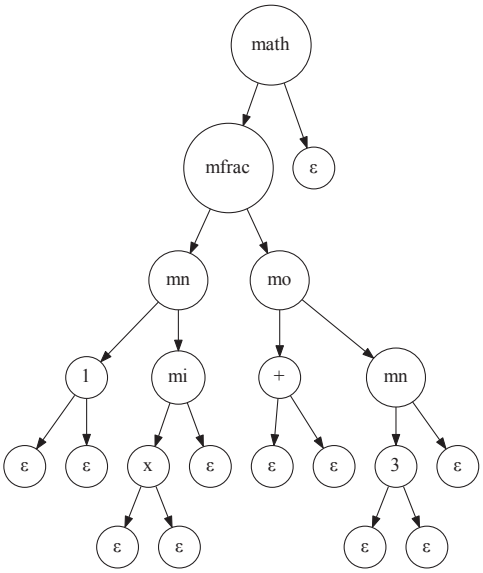

図 3 二分木変換後の MathML 式
Fig. 3 Binary tree of MathML formula

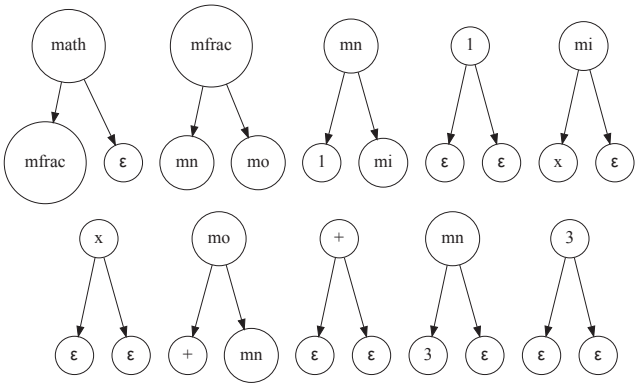

図 4 図 3 の二分木に対する Binary Branch
Fig. 4 Binary Branch of Figure 3

2.2 Binary Branch

任意のの $u, v_1, v_2 (\in N)$ において，v_1 が u の左の子であれば $\langle u, v_1 \rangle_l \in E_l$，$v_2$ が u の右の子であれば $\langle u, v_2 \rangle_l \in E_r$ とする．二分木 $B(T)$ 中のノード u に対する Binary Branch $BiB(u) = (N_u, E_{ul}, E_{ur}, Root(T_u))$ を以下の条件を満たす二分木と定義する．

- $N_u = \{u, u_1, u_2\}$
- $E_{ul} = \{\langle u, u_1 \rangle_l\}$
- $E_{ur} = \{\langle u, u_2 \rangle_r\}$
- $Root(T_u) = u$

図 4 に図 3 の二分木の各ノードに対する Binary Branch を示す．

表 1 図 4 に対する Binary Branch Vector
Table 1 Binary Branch Vector of Figure 4

Binary Branch			
親ノード	左の子	右の子	出現回数
math	mfrac	ϵ	1
mfrac	mn	mo	1
mn	1	mi	1
1	ϵ	ϵ	1
mi	x	ϵ	1
x	ϵ	ϵ	1
mo	+	mn	1
+	ϵ	ϵ	1
mn	3	ϵ	1
3	ϵ	ϵ	1

表 2 図 5 に対する Binary Branch Vector
Table 2 Binary Branch Vector of Figure 5

Binary Branch			
親ノード	左の子	右の子	出現回数
math	mfrac	ϵ	1
mfrac	mn	mo	1
mn	ϵ	mi	1
mi	x	ϵ	1
x	ϵ	ϵ	1
mo	+	mn	1
+	ϵ	ϵ	1
mn	ϵ	ϵ	1

2.3 Binary Branch Vector

木 T に対する Binary Branch Vector $BRV(T)$ を i 番目の Binary Branch の出現回数 b_i，データセット中のユニークな Binary Branch の数 $|\Gamma|$ を用いて $BRV(T) = (b_1, b_2, ..., b_{|\Gamma|})$ と表す．**表 1** に図 4 の Binary Branch とその出現回数を示す．この出現回数の列が Binary Branch Vector となる．

2.4 Binary Branch Vector の次元削減

Binary Branch Vector の各成分はデータセット中の全ての木に出現する Binary Branch の出現回数であるため，データセットが大きくなるとユニークな Binary Branch の数が増え，ベクトルの次元が増えることが想定される．特に同じような形の式だが数値が異なる場合に，ユニークな Binary Branch の数が増加すると考えられる．そこで本手法では MathML 式を二分木に変換する際に，mn 要素のテキストを無視し，ベクトルの次元削減を行う．例として図 2 の木を mn 要素のテキストを無視して完全二分木に変換したものを**図 5** に，**表 2** に図 5 の Binary Branch Vector を示す．図 1 のものと比べ，親ノードが数値である Binary Branch が消え，ベクトルの次元が削減されていることが分かる．

Vol.2016-DBS-163 No.7
Vol.2016-IFAT-123 No.7
2016/9/13

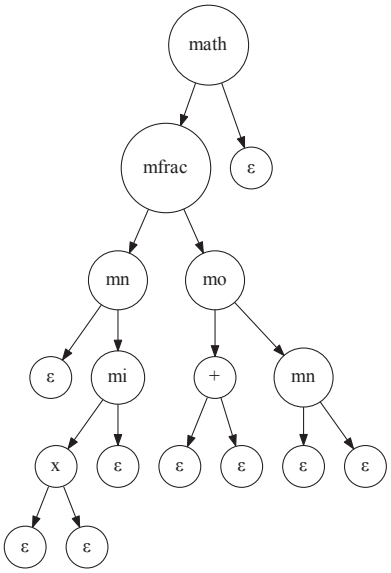

図 5　次元削減処理を施した完全二分木
Fig. 5　Complete binary tree of MathML formula after deletion of numerical value

3. MathML 式の分類

MathML 式の分類には多層パーセプトロン (Multilayer Perceptron) を用いる．

3.1 多層パーセプトロン

多層パーセプトロンは図 **6** のように入力層，任意の数の中間層，出力層から構成される．第 n 層のユニットを $i = 1, 2, .., I$，第 $n+1$ 層のユニットを $j = 1, 2, ..., J$ とする．第 $n+1$ 層のユニット j の入力値 u_j は，第 n 層の入力値 x_i，重み w_{ji}，バイアス b_j を用いて以下のように表される．

$$u_j = \sum_{i=1}^{I}(w_{ji}x_i + b_j) \tag{1}$$

本手法では，入力層のユニット数はデータセット中のユニークな Binary Branch の数である．また，出力値 z_j は活性化関数 f を用いて

$$z_j = f(u_j) \tag{2}$$

と表される．一般に，f にはシグモイド関数や ReLU 関数などがよく用いられる．ReLU 関数は

$$f(u_j) = \max(0, u_j) \tag{3}$$

と表される．出力層では，一般に線形出力関数やソフトマックス関数が用いられれる．ソフトマックス関数の場合，出力層はクラス数と同数のユニットを持ち，

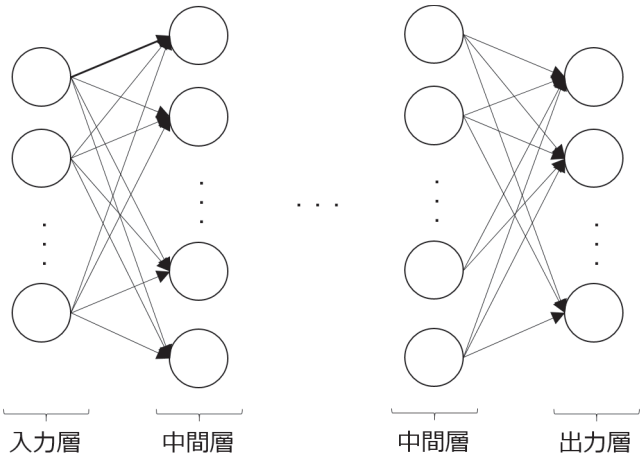

図 6　多層パーセプトロン
Fig. 6　Mulutilayer perceptron

$$f(u_j) = \frac{e^{u_j}}{\sum_{i=1}^{J} e^{u_i}} \tag{4}$$

で表される．J 個のユニットの出力値の総和は 1 であるため，各ユニットの出力値は，入力データがそのクラスに属する確率と解釈できる．

多層パーセプトロンでは，訓練データの目標出力値と実際の出力値との誤差 $E(w)$ が最小となるように w を更新することで学習を進める．この w を更新するため手法として，確率的勾配降下法や Adam などが用いられる．

また，多層パーセプトロンのような多層ネットワークでの学習の際に過学習を防ぐ仕組みとしてドロップアウトがある．これは，学習の際に入力層や中間層のユニットを確率 p で無視するというものである．これにより，学習時のネットワークを小さくし，過学習を防ぐことができる．

3.2 学習モデル

提案手法では中間層 5 層の多層パーセプトロンを用いる．入力層に近い層から $1, 2, ..., 5$ 層とすると，中間層のユニット数は 1 層を 2,048, それ以外を 1,024 とした．また，ドロップアウト率を 1,2,3 層では 50%，4,5 層では 20%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

提案手法を深層学習ライブラリの Keras を用いて Python で実装し，評価実験を行った．評価実験の環境は以下の通りである．

OS　Ubuntu 14.04 (64bit)
CPU　Intel Xeon CPU E5-1620 3.50GHz
メモリ　32GB
GPU　NVIDIA Tesla K40m

以下，評価実験の方法と得られた結果について述べる．

表 3　Kim ら [2] による MathML 式の特徴量
Table 3　Features of MathML formula

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

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:訓練データ=7:3 に分割する
(3) 教師データを分類器に学習させる
(4) テスト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) を全てのテストデータで行い，正解率を計算する

また，Kim ら [2] 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**表 3** 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．

4.2　データセット

本研究では，データセットとして The Wolfram Function Site[7] の数式を用いる．このサイトから網羅的に 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，**図 7** のように semantics 要素や annotation-xml 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 4** に MathML 式の属するクラスとその件数を示す．

4.3　実験結果

評価実験の結果を**表 5** に示す．SVM については，2 種のカーネル (Linear, RBF) と 2 種の多クラス分類方法を試したが，いずれの場合も提案手法が SVM による分類を上回っている．また，入力ベクトルの次元は Kim ら [2] の特徴ベクトルが 8,113 次元であったのに対し，Binary Branch Vector は 2,184 次元であった．よって，提案手法は Kim ら [2] のものより小さい次元数で，より MathML 式の特徴を反映した分類が可能であると考えられる．

```
<math xmlns='http://www.w3.org/1998/Math/MathML' ...>
  <semantics>
    <mrow>
      <msqrt>
        <mi> z </mi>
      </msqrt>
       ...
    </mrow>
    <annotation-xml encoding='MathML-Content'>
      <apply>...</apply>
    </annotation-xml>
  </semantics>
</math>
```

図 7　Content Markup の埋め込み
Fig. 7　Embedded Content Markup

表 4　クラス名と件数
Table 4　Class name and number of MathML formula

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

表 5　正解率
Table 5　Accuracy of each method

手法	正解率
提案手法	99.2%
SVM(Linear, One-against-one)	94.4%
SVM(Linear, One-against-rest)	96.0%
SVM(RBF, One-against-one)	70.9%
SVM(RBF, One-against-rest)	71.0%

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 The Wolfram Function Site の数式では正解率が 99.2%と高精度な分類を行えることが分かった．

今後の課題として，本手法で分類したクラス情報を用いて Presentation Markup から Content Markup への変換を行うことがある．また，[8] や [9] のような木構造データに対するカーネル用いて MathML 式の分類を行い，提案手法と正解率を比較することを検討している．

参考文献

[1] Yang, R., Kalnis, P. and Tung, A. K. H.: Similarity Evaluation on Tree-structured Data, *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pp. 754–765 (2005).
[2] KIM, S., YANG, S. and KO, Y.: Classifying Mathematical Expressions Written in MathML, *IEICE Transactions on Information and Systems*, Vol. E95.D, No. 10, pp. 2560–2563 (2012).
[3] 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).
[4] 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).
[5] Nghiem, M.-Q., Kristianto, G. Y., Topić, G. and Aizawa, A.: A Hybrid Approach for Semantic Enrichment of MathML Mathematical Expressions, *Proceedings of the 2013 International Conference on Intelligent Computer Mathematics*, pp. 278–287 (2013).
[6] Design Science: MathPlayer, Design Science (online), available from ⟨https://www.dessci.com/en/products/mathplayer/⟩ (accessed 2016-08-19).
[7] Wolfram Research: The Wolfram Function Site, Wolfram Research (online), available from ⟨http://functions.wolfram.com/⟩ (accessed 2016-08-03).
[8] 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, pp. 113–121 (2006).
[9] 木村大翼，久保山哲二，渋谷哲朗，鹿島久嗣：部分パスに基づいた木カーネル，人工知能学会論文誌，Vol. 26, No. 3, pp. 473–482 (2011).