半環に基づく前向き後ろ向きアルゴリズムの一般化

Semiring-based Generalization of the Forward-Backward Algorithm

東 藍[1*]　新保 仁[1]　松本 裕治[1]
Ai Azuma[1]　Masashi Shimbo[1]　Yuji Matsumoto[1]

[1] 奈良先端科学技術大学院大学
[1] Nara Institute of Science and Technology

Abstract: When we apply machine learning or data mining technique to sequential data, it is often required to take a summation over all the possible sequences. We cannot calculate such a summation directly from its definition in practice. Although the ordinary forward-backward algorithm provides an efficient way to do it, it is applicable to quite limited types of summations. In this paper, we propose general algebraic frameworks for generalization of the forward-backward algorithm. We show some examples falling within this framework and their importance.

1 はじめに

近年，機械学習やデータマイニングの適用範囲はますます広がっており，その対象は内部に複雑な構造を有するデータにも広がりつつある．本論文では，そのような構造や依存関係を有するデータのうち最も単純な部類である**系列データ**を対象に議論する．

系列データに対する機械学習・データマイニングを行う上で，可能な全ての系列にわたって何らかの総和を計算する必要に迫られる局面は多数見受けられる．この可能な全ての系列の集合が，有向非循環グラフ上の有向パスとしてコンパクトに表現されている場合，単純な形式の総和に対しては動的計画法に基づく効率的な計算方法 (例：前向き後ろ向きアルゴリズム) が広く知られている．しかしながら，より一般的・複雑な形式の和については動的計画法による効率的な計算方法は知られていないことが多い．こういった複雑な総和計算を必要とする場合，既存の研究においては独立かつ個々に計算アルゴリズムを設計している例も散見される [2][8][4] .

本論文では，より幅広い形式の和を有向非循環グラフ上で計算するためのより統一的な枠組みとして，代数的な抽象化に基づいた前向き後ろ向きアルゴリズムの一般化を提案する．また，この一般化の枠組みに当てはまる具体的な代数のうち，系列データに対する機械学習・データマイニングにおいて重要な役割を果たすと思われるものをいくつか取り上げる．そして，この一般的なアルゴリズムが系列データに対する具体的な機械学習・データマイニングの文脈において，どのように具体化されることで有用となるのかを示す．

*連絡先：奈良先端科学技術大学院大学
情報科学研究科 自然言語処理学講座
〒 630-0192 奈良県生駒市高山町 8916-5
E-mail: ai-a@is.naist.jp

2 研究背景

まず最初に，本論文において何を一般化の対象とするかについて述べておく．有向非循環グラフ $G = (V, E)$ を考える．入次数 0 の頂点がただ 1 つしかない場合を考え，これを src と表記する．同じく，出次数 0 の頂点がただ 1 つしかない場合を考え，これを snk と表記する．入次数 0 の頂点，出次数 0 の頂点がそれぞれ複数ある場合に対しても，以下の議論は容易に拡張できるため，その詳細は割愛する．src を始点とし snk を終点とする G 上の有向パスの集合を $\Pi(\mathrm{src}, \mathrm{snk})$ とおく．また，V 上に実数値関数 ϕ, f が定義されているとする．このように定めた表記の下で，本論文で一般化の対象とするのは，我々が系列データに対する機械学習・データマイニングにおいて極めて重要な役割を果たすと考える以下のように定式化される計算である．

$$\sum_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \prod_{v \in \pi} \phi(v) \tag{1}$$

$$\sum_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \left(\sum_{v \in \pi} f(v) \right) \left(\prod_{v \in \pi} \phi(v) \right) \tag{2}$$

ここで有向パスを表す記号 π を，π 上に現れる頂点の集合を表す記号としても扱っている．この表記は特に混乱を来たさないと思われるので，以下でも特に断りなく用いることにする．

以下，(1) および (2) の形式，およびその一般化が系列データを対象とする機械学習・データマイニング

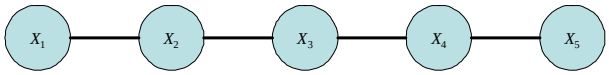

図 1: 直鎖状グラフィカルモデルの例

において重要であると我々が考える理由と背景について述べる．

(1) は，考慮している系列の集合が $\Pi(\mathrm{src}, \mathrm{snk})$ として表現されており，各系列に対する確率ポテンシャルがその系列上の頂点のポテンシャル関数の積として表現されているときに，集合 $\Pi(\mathrm{src}, \mathrm{snk})$ に対する**周辺確率**あるいは**正規化定数**を計算することに対応する．たとえば，系列データの解析に頻繁に用いられる図 1 のような直鎖状のグラフィカルモデル $\mathcal{G}$ を考え，$\mathcal{G}$ の周辺確率あるいは正規化定数を計算することを考えてみる．今，仮に $\mathcal{G}$ の各頂点に A, B, C の 3 つの状態が割り当てられるものとする．その可能な状態の割り当てを全て展開したものを有向非循環グラフとして表現したものを図 2 に示す．図 2 で表現されるような図，あるいはデータ構造はしばしば**ラティス**あるいは**トレリス**と呼ばれる．ここでトレリス中の辺の向きは単に系列の前後を区別するために便宜的に与えるものであり，グラフィカルモデルにおける有向・無向の概念とは無関係である．$\mathcal{G}$ のような直鎖状のグラフィカルモデルにおいては，全頂点に対する状態の割り当てに対する結合確率は頂点と辺の上に定義された非負実数値のポテンシャル関数の積に比例する．一方，トレリス G 上に定義される有向パスの集合 $\Pi(\mathrm{src}, \mathrm{snk})$ は $\mathcal{G}$ に対する可能な状態の割り当ての集合に 1 対 1 で対応するのは明らかである．図 2 に示されるトレリス $G = (V, E)$ に対して，全ての辺を新たに頂点に置き換えるように変形した結果得られるトレリスを $G' = (V', E')$ と置く．つまり，$V' \stackrel{\mathrm{def}}{=} V \cup E$ とし，E' の要素 (x, y) は $x = v \wedge y = (v, v')$，$(v, v') \in E$ または $x = (v, v') \wedge y = v'$，$(v, v') \in E$ のときかつこのときのみに存在するとする．この変形は，$\mathcal{G}$ をファクターグラフに変形することに対応する．[1]このような定義の下で，G' に対する (1) はまさにグラフィカルモデル $\mathcal{G}$ に対する周辺確率あるいは正規化定数を与える．この文脈では，(1) における ϕ は $\mathcal{G}$ のクリーク上に定義されたポテンシャル関数に対応する．

より一般に，ある系列において特定の状態に滞留する時間が単一時間長さとは限らないようなモデル，すなわち準マルコフモデルを考えることもできる．このようなモデルもやはり系列データに対する機械学習・データマイニングで実用上多用されている [7][3]．(1) の形式はこのような準マルコフモデルも包含している

[1]この変形によって，頂点および辺の上で定義された関数を頂点の上だけで定義された関数として扱える．以下，表記を簡潔にするためにこのような頂点上にのみ関数が定義された形式のみで論じるものとする．読者におかれては，文脈の必要に応じてこのような変形が暗黙になされているものとして読み替えていただきたい．

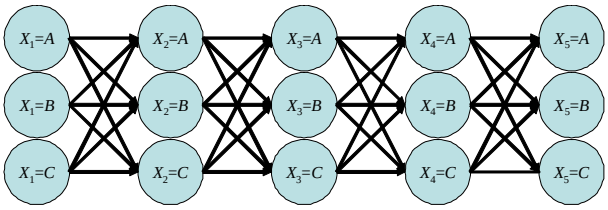

図 2: トレリスの例

形式であることに注意されたい．

(1) は，通常定義どおりに計算することが困難であるが，これを動的計画法で効率的に計算できることが知られている．つまり，$\Pi(\mathrm{src}, \mathrm{snk})$ は巨大な集合になりうるため，$\Pi(\mathrm{src}, \mathrm{snk})$ の全要素をわたる和を直接計算することは難しい．しかしながら，(1) の値は G の topological order に沿った動的計画法を実行することで，高々 $|V| + |E|$ 程度の計算量で計算することができることが広く知られている．このアルゴリズムはちょうど，前向き後ろ向きアルゴリズムにおける前向きの再帰，あるいは前向きの message passing に対応するものである．

さて，このようなトレリスにエンコードされた系列集合を対象とした計算のうち，実用上重要な他の例として最尤パスの尤度計算が挙げられる．対数関数が単調増加関数であることから，最尤パスの対数尤度は以下のように定式化される．

$$\max_{\pi \in \Pi(\mathrm{src}, \mathrm{snk})} \sum_{v \in \pi} \log \phi(v) \tag{3}$$

本論文における一般化の出発点として，(1) と (3) の代数的な共通性を論じることにする．すなわち，ある集合 $\mathbb{K}$ 上に加算および乗算に相当する 2 項演算が定義されて，それらが $\oplus, \otimes$ という記号で表記することにする．このとき，(1) と (3) はともに以下のように代数的に抽象化した形式で統一的に記述できる．

$$\bigoplus_{\pi \in \Pi(\mathrm{src}, \mathrm{snk})} \bigotimes_{v \in \pi} \phi(v) \tag{4}$$

ここで，$\bigoplus$ は 2 項演算 $\oplus$ の列 $\cdot \oplus \cdot \oplus \cdots$ を表し，$\bigotimes$ は 2 項演算 $\otimes$ の列 $\cdot \otimes \cdot \otimes \cdots$ を表す．また，$\phi : V \to \mathbb{K}$ である．(1) は $\mathbb{K}$ が $\mathbb{R}$ $(\mathbb{R}_+)$，$\oplus$ が実数の通常の加算，そして $\otimes$ が実数の通常の乗算の場合に対応する．また，(3) は $\mathbb{K}$ が $\mathbb{R} \cup \{-\infty\}$，$\oplus$ が max，そして $\otimes$ が通常の加算の場合である．なお，(4) が明確に定義でき，なおかつこれを効率的に計算する動的計画法を導出するためにはこれらの 2 項演算が満たすべき一定の規則が必要となるが，これは**半環**と呼ばれる代数的枠組みとして形式化される．この半環の厳密な定義については 2 節で触れる．

このような代数的抽象化に基づいた考察を本論文で展開する理由は，単に計算の対象となるものを統一的

かつ可能な限り広範に形式化できるばかりでなく，それらを計算するアルゴリズムまでをも統一的かつ広範な適用範囲を持つものとして記述できる点にある．本論文における代数的抽象化に基づくアルゴリズムの記述や計算量，アルゴリズムの記述における代数的抽象化の意義については 3 節で触れる．

半環に基づいて (4) の形式を含む範囲の対象について代数的に抽象化し，これを計算するアルゴリズムを定式化した既存の研究としていくつか挙げることができる．たとえば [5] の枠組みでは，任意のグラフに対して半環によって一般化した最短距離問題を定式化している．(4) の形式は，彼の枠組みにおいて有向非循環グラフに限定し，queue decipline として topological order による優先順位付キューを採用することに対応する．しかしながら [5] の主眼は最短距離問題であり，$k-$最短距離問題などを代数的抽象化の具体例としてインスタンス化している程度である．無論，$k-$最短距離問題 も系列データに対する機械学習・データマイニングの文脈においてはいわゆるビームサーチの形式化として咀嚼できるものであり，有意義である．とはいえ，このような代数的抽象化に当てはまるものの従来顕著には指摘されてこなかった例もある．そして，それらの中には系列データに対する機械学習・データマイニングの文脈において相当の貢献が見込まれるものもある．本論文ではそのような具体例を指摘し，なおかつ系列データに対する解析においてそれがどのようにその意義を発揮できるのかを示す．

また，[5] の枠組みは系列データに対する機械学習・データマイニングの文脈に当てはめて考えると，前向き後ろ向きアルゴリズムのうちの前向きの再帰のみ，別の言葉で言えば前向きの message passing に対しての代数的抽象化に言及しているに過ぎないと我々は考える．一方で，前向き後ろ向きアルゴリズムという言葉に表れているように，前向きの再帰・message と後ろ向きの再帰・message と組み合わせる形式のアルゴリズムも良く知られている．しかしながら，後者に対する代数的抽象化については我々の知る限り既知ではないと思われる．

ここで，(2) の形式の意義と重要性について触れてこなかったので簡単に述べておく．(2) の形式は，まさに従来の前向き後ろ向きアルゴリズムが対象としている計算である．このことを系列データに対する 2 つの具体的な計算で見てみる．

系列データに対する機械学習・データマイニングにおいては，たとえばグラフィカルモデルのある頂点に特定の状態が割り当てられる確率，あるいはある 2 つの頂点状態間の遷移確率がどの程度かを計算する必要に迫られる局面が多い．いわゆる周辺化・周辺確率の計算である．これはトレリスによる表現で言い換えれば，トレリス上のある頂点を通る有向パスの確率ポテンシャルのみに関して和を取る計算となる．たとえば，図 2 の $X_3 = A$ なる頂点 (今，仮にこの頂点を v と置く) を通る有向パスの確率ポテンシャルのみを総和する計算は

$$\sum_{\pi\in\Pi(\mathrm{src},\mathrm{snk})} \delta_{v\in\pi} \prod_{v'\in\pi} \phi(v') = \sum_{\pi\in\Pi(\mathrm{src},\mathrm{snk})} \left(\sum_{v'\in\pi} \delta_{v'=v}\right)\left(\prod_{v'\in\pi} \phi(v')\right) \tag{5}$$

で与えられる．ここで $\delta_{\mathcal{P}}$ は叙述 $\mathcal{P}$ に対する指示関数である．

また，ある状態 (ここでは一般性を失うことなく A とする) が割り当てられる期待値——これも 1 種の周辺化であるが——を計算する必要が生じる場面もある．これは，

$$\sum_{\pi\in\Pi(\mathrm{src},\mathrm{snk})} \left(\sum_{v\in\pi} \delta_{\mathrm{state}(v)=A}\right)\left(\prod_{v'\in\pi} \phi(v')\right) \tag{6}$$

で計算できる． $\mathrm{state}(v) = A$ は， v が状態 A の割り当てに対応する頂点であることを意味するものとする．$\sum_{v\in\pi} \delta_{\mathrm{state}(v)=A}$ が，状態 A を割り当てられた頂点が有向パス π 中に現れる個数を表すことに注意されたい．これらの計算が (2) の形式に当てはまっていることは容易に確認できる．我々が (2) もまた，系列データに対する機械学習・データマイニングの文脈で重要であると認識する理由がここにある．そしてこれらの計算が，前向きの再帰における中途の状態を記録した変数 (前向きの message) と後ろ向きの再帰における中途の状態を記録した変数 (後ろ向きの message) とを組み合わせる 1 種の動的計画法で効率的に計算できることは広く知られている．従って， (2) の形式，およびこれを効率的に算出する動的計画法の代数的抽象化を導出することは極めて自然な進展の方向であるように思われる．

そして最後に，系列データに機械学習・データマイニングを適用する上で，トレリス中の各系列に対する結合確率がその頂点の上に定義されたポテンシャル関数の積として因数化できない場合についても考察や実験の俎上に載せる意義についても言及しておく．トレリス中の各系列に対する結合確率がその頂点の上に定義されたポテンシャル関数の積として因数化できる場合，これらは学習や推論に必要とされる主要な計算としては (1) (2) の形式に対応し，同時にこれはたとえば直鎖グラフィカルモデルの文脈では確率変数間の依存関係の直感的な理解のしやすさに直結している．これらは言うまでもなく系列データの解析において重要なモデルのクラスを形成している．一方で，もはやこのような範疇に収まらない結合確率の形式を持つモデルについて考察や理解を深めることも重要な意義を持

つと考える．具体化や適用方法については今後検討の余地は大いにあると思われるが，たとえば Gaussian, logistic regression, kernel-induced feature space の利用などが想定される．特に，これら様々なモデルを具体的・実用的なデータに対してその性能を比較検討するあり方が確立されることには十二分の意義があると考える．同時に，系列データの解析におけるそれらのモデルに対する最適化の基準や最適化アルゴリズムを決定する際に，より幅広い選択肢を提供できることもその意義も大きいだろう．これらもまた，それらを実データに基づいて比較できることが可能となればその意義は大きいと考える．

以上を踏まえて，本論文では

1. 前向きの再帰を一般化した形式だけではなく，前向きの再帰と後ろ向きの再帰を組み合わせることで周辺確率の計算を行う形式も一般化する．
2. このような代数的抽象化の形式に当てはまる具体例のうち，系列データに対する機械学習・データマイニングにおいて有効であると思われるものの従来顕著に指摘されてこなかったものをいくつか挙げ，これらにより，より幅広いモデル・より幅広い最適化戦略を実データに基づいて比較できる可能性を指摘する．
3. 系列データに対する機械学習・データマイニングの具体的な文脈において，これらがどのように活用されるのかを実演する．

3 定義

前向き後ろ向きアルゴリズムを一般化するにあたって，本論文で用いるいくつかの定義について本節で述べておく．

3.1 半環および群上の半環

定義 1 (半群, monoid) 半群 $(\mathbb{K}, \oplus, \bar{0})$ とは，閉じた *2* 項演算 $\oplus$ の定義された集合 $\mathbb{K}$ であり，以下を満たす．

1. $^\forall a, b, c \in \mathbb{K}$ に対して $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ が成立 *(結合法則)*
2. $\bar{0} \in \mathbb{K}$ が存在し $^\forall a \in \mathbb{K}$ に対して $\bar{0} \oplus a = a \oplus \bar{0} = a$ が成立 *(単位元の存在)*

定義 2 (可換半群, commutative monoid) 可換半群とは，$^\forall a, b \in \mathbb{K}$ に対して $a \oplus b = b \oplus a$ *(交換法則)* が成り立つ半群 $(\mathbb{K}, \oplus, \bar{0})$ である．

定義 3 (半環, semiring) $\mathbb{K}$ を集合とする．$\oplus, \otimes$ を $\mathbb{K}$ 上の *2* 項演算，$\bar{0}, \bar{1}$ を $\mathbb{K}$ の要素とする．このとき，システム $\mathcal{T} = (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ が**半環**であるとは以下が成り立つことである．

1. $(\mathbb{K}, \oplus, \bar{0})$ が可換半群である，
2. $(\mathbb{K}, \otimes, \bar{1})$ が半群である，
3. $\otimes$ が $\oplus$ に対して分配する．すなわち，$^\forall a, b, c \in \mathbb{K}$ に対して以下が成り立つ
   (a) $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
   (b) $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$
4. $\bar{0}$ が $\otimes$ に対するゼロイデアルとなる．つまり $^\forall a \in \mathbb{K}, a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$

ここで，実数とそれに対する通常の加算･乗算 $(\mathbb{R}, +, \times, 0, 1)$ は半環であることに注意されたい．本論文で (1) (2) を一般化するというのは，つまりこの実数の半環以外について考えていくということである．

定義 4 (群上の半環) 半環 $\mathcal{T}' = (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ が群 $(\mathbb{G}, +, 0)$ 上の半環であるとは，$\cdot : \mathbb{G} \times \mathbb{K} \to \mathbb{K}$ が存在し以下を満たすことである．

1. $^\forall c \in \mathbb{G}$, $^\forall x, y \in \mathbb{K}$ に対して $c \cdot (x \oplus y) = c \cdot x \oplus c \cdot y$
2. $^\forall c, d \in \mathbb{G}$, $^\forall x \in \mathbb{K}$ に対して $(c + d) \cdot x = c \cdot x \oplus d \cdot x$

3.2 最短距離および周辺化最短距離

トレリス $G = (V, E)$, 半環 $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$, V から $\mathbb{K}$ への関数 w が与えられたとする．v を引数としたときの w の値を $w[v]$ と書き，頂点 v の**重み**と呼ぶことにする．また，この定義を拡張し G 上の有向パス $\pi = (v_1, \ldots, v_{|\pi|})$ の重みを $w[\pi] \stackrel{\text{def}}{=} w[v_1] \otimes \cdots \otimes w[v_{|\pi|}]$ と定義する．このとき，(1) において “+” を “$\oplus$” に，“$\times$” を “$\otimes$” に置き換えて一般化した次のような和の形式を定義する．

定義 5 (最短距離) $u, v \in V$, 半環 $\mathcal{T} = (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ および頂点の重み w が与えられたとき，u, v 間の**最短距離** $\mathcal{D}_{\mathcal{T},w}(u, v)$ を以下で定義する．

$$\mathcal{D}_{\mathcal{T},w}(u, v) \stackrel{def}{=} \bigoplus_{\pi \in \Pi(u,v)} w[\pi] \tag{7}$$

同様に，(2) を一般化した次のような和の形式を定義する．

Data: A trellis $G = (V, E)$,
vertex weight $w : V \to \mathbb{K}$
Result: $\alpha[v] = \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, v)$

$\alpha[\mathrm{src}] \leftarrow w[\mathrm{src}]$
for $v \in V \setminus \mathrm{src}$ *with a topological order* **do**
  $\alpha[v] \leftarrow \bigoplus_{x \in \mathrm{prev}(v)} \alpha[x] \otimes w[v]$
end

Algorithm 1: 一般化した前向き再帰

Data: A trellis $G = (V, E)$,
vertex weight $w : V \to \mathbb{K}$
vertex function $f : V \to \mathbb{G}$
Result: $E = \mathcal{E}_{\mathbb{K},w}[f]$

$\beta[\mathrm{snk}] \leftarrow \bar{1}$
for $v \in V \setminus \mathrm{snk}(G)$ *with a topological order* **do**
  $\beta[v] \leftarrow \bigoplus_{x \in \mathrm{next}(v)} w[x] \otimes \beta[x]$
end
$E \leftarrow \bar{0}$
for $v \in V$ **do**
  $E \leftarrow E \oplus f(v) \cdot \alpha[v] \otimes \beta[v]$
end

Algorithm 2: 一般化した前向き後ろ向きアルゴリズム

定義 6 (周辺化最短距離) 群 $(\mathbb{G}, +, 0)$ 上の半環 $\mathcal{T} = (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$, 重み $w : V \to \mathbb{K}$, が与えられたとき, $f : V \to \mathbb{G}$ によって**周辺化した最短距離**を以下のように定義する.

$$\mathcal{E}_{\mathcal{T},w}[f] \overset{def}{\equiv} \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \left(\sum_{v \in \pi} f(v) \right) w[\pi] \tag{8}$$

4 前向き再帰および前向き後ろ向きアルゴリズムの一般形

3 節における定義に基づいて, 一般化した前向きの再帰を Algorithm 1 に, 一般化した前向き後ろ向きアルゴリズムを Algorithm 2 に掲げる. Algorithm 1 において $\mathrm{prev}(v)$ は v の直前のノードの集合 $\mathrm{prev}(v) \overset{\mathrm{def}}{\equiv} \{x \mid (x, v) \in E\}$, Algorithm 2 において $\mathrm{next}(v)$ は v の直後のノードの集合 $\mathrm{next}(v) \overset{\mathrm{def}}{\equiv} \{x \mid (v, x) \in E\}$ を各々表す.

まず, Algorithm 1 において

$$\alpha[v] = \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, v) \tag{9}$$

であることを簡単に説明しておく. まず, $\alpha[\mathrm{src}] = \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, \mathrm{src})$ は定義により自明である. 以下, ある $x \in V$ を考え ${}^{\forall}v \in \mathrm{prev}(x)$ に対して (9) が成り立っていると仮定すると,

$$\begin{aligned}
\alpha[x] &= \bigoplus_{v \in \mathrm{prev}(x)} \alpha[v] \otimes w[x] \\
&= \bigoplus_{v \in \mathrm{prev}(x)} \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, v) \otimes w[x] \\
&\qquad (\because \text{仮定 (9) から}) \\
&= \bigoplus_{v \in \mathrm{prev}(x)} \left(\bigoplus_{\pi \in \Pi(\mathrm{src},v)} w[\pi] \right) \otimes w[x] \\
&\qquad (\because \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, v) \text{ の定義から}) \qquad (10) \\
&= \bigoplus_{v \in \mathrm{prev}(x)} \bigoplus_{\pi \in \Pi(\mathrm{src},v)} (w[\pi] \otimes w[x]) \\
&\qquad (\because \text{分配法則}) \\
&= \bigoplus_{\pi \in \Pi(\mathrm{src},x)} w[\pi] \\
&(\because 2 \text{ つの } \textstyle\bigoplus \text{ と系列の重みをまとめた}) \\
&= \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, x)
\end{aligned}$$

となる. 従って, G の topological order に沿って α を再帰計算しているため, ${}^{\forall}v \in V$ に対して (9) は成立する.

次に, Algorithm 2 において $E = \mathcal{E}_{\mathcal{T},w}[f]$ であることを示す. まず, (9) と同様, $\beta[v]$ に対して

$$\beta[v] = \bigoplus_{x \in \mathrm{next}(v)} \mathcal{D}_{\mathcal{T},w}(x, \mathrm{src}) \tag{11}$$

が成り立つことが示せる. ここから,

$$\begin{aligned}
\mathcal{E}_{\mathcal{T},w} &= \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \left(\sum_{v \in \pi} f(v) \right) \cdot w[\pi] \\
&= \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \bigoplus_{v \in \pi} f(v) \cdot w[\pi] \\
&\qquad (\because \text{“}\cdot\text{”} \text{ の分配}) \\
&= \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \bigoplus_{v \in V} f(v) \cdot (\delta_{v \in \pi} \otimes w[\pi]) \\
&(\delta_{\mathcal{P}} \text{ は } \mathcal{P} \text{ が真のとき } \bar{1}, \text{ それ以外は } \bar{0} \text{ と定義}) \\
&= \bigoplus_{v \in V} \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} f(v) \cdot (\delta_{v \in \pi} \otimes w[\pi]) \\
&\qquad (\because \oplus \text{ の交換}) \\
&= \bigoplus_{v \in V} f(v) \cdot \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \delta_{v \in \pi} \otimes w[\pi] \\
&\qquad (\because \text{“}\cdot\text{”} \text{ の分配})
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{v\in V} f(v)\cdot \bigoplus_{\pi\in\Pi_v(\mathrm{src},\mathrm{snk})} w[\pi] \\
&\quad (\Pi_v(\mathrm{src},\mathrm{snk}) \stackrel{\mathrm{def}}{\equiv} \{\pi\in\Pi(\mathrm{src},\mathrm{snk}) \mid v\in\pi\}) \\
&= \sum_{v\in V} f(v)\cdot \left(\bigoplus_{\pi\in\Pi(\mathrm{src},v)} w[\pi]\right) \\
&\qquad\qquad \otimes \left(\bigoplus_{x\in\mathrm{next}(v)} \bigoplus_{\pi\in\Pi(x,\mathrm{snk})} w[\pi]\right) \\
&= \sum_{v\in V} f(v)\cdot \alpha[v]\otimes\beta[v]
\end{aligned}
\tag{12}
$$

となり，これは E の計算に一致する．

計算量について言及しておく． Algorithm 1 においては $\mathcal{O}(|E|\,\mathcal{T}_\oplus + |V|\,\mathcal{T}_\otimes)$ の時間計算量が必要となる．ここで $\mathcal{T}_\oplus, \mathcal{T}_\otimes$ は，各々 $\oplus, \otimes$ を実行するために必要な時間計算量である．また，Algorithm 2 においては $\mathcal{O}(|E|\,\mathcal{T}_\oplus + |V|\,\mathcal{T}_\otimes + \mathcal{T}_\cdot)$ の時間計算量が必要となる．ここで $\mathcal{T}_\cdot$ は $\cdot$ を実行するために必要な時間計算量である．さらに， Algorithm 2 においては各頂点に $\mathbb{K}$ の要素を記録する必要があるため $\mathcal{O}(|V|\,\mathcal{S}_\mathbb{K})$ の空間計算量が必要となる．ここで $\mathcal{S}_\mathbb{K}$ は $\mathbb{K}$ の要素 1 つを記録するために必要な空間計算量である．

5 具体例

本節では，上に挙げた枠組みに当てはまる具体的な半環および群上の半環の具体例をいくつか挙げる．これらは，6 節で解説するように，系列データに対する機械学習・データマイニングにおいて重要な役割を担うと考えられる．

5.1 2 項畳み込み

n 次元実数ベクトル上における代数系 $\{\mathbb{R}^n, +, \diamond, \bar{0}, \bar{1}\}$ を以下のように定義する．ただし，$\vec{x} = (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$， $\vec{y} = (y_1, \ldots, y_n)^{\mathrm{T}} \in \mathbb{R}^n$ とする．

$$
\begin{aligned}
&\vec{x} + \vec{y} = (x_1 + y_1, \cdots, x_n + y_n)^{\mathrm{T}} \\
&\vec{x} \diamond \vec{y} = (\binom{0}{0} x_1 y_1, \binom{1}{0} x_1 y_2 + \binom{1}{1} x_2 y_1, \cdots, \\
&\qquad\qquad \sum_{m=1}^{k} \binom{k-1}{m-1} x_m y_{k-m+1}, \cdots)^{\mathrm{T}} \\
&\bar{0} = (0, \cdots, 0)^{\mathrm{T}} \\
&\bar{1} = (1, 0, \cdots, 0)^{\mathrm{T}}
\end{aligned}
\tag{13}
$$

$+$ は n 次元実数ベクトルに対する通常の加算であり，$\diamond$ は **2 項畳み込み** (binomial convolution) などと呼ばれる演算である．これらの定義によるシステム $(\mathbb{R}^n, +, \diamond, \bar{0}, \bar{1})$ が半環を成すことは容易に確かめられる．

$\phi : V \to \mathbb{R}$ とする．この半環において特記すべきこととして，$\vec{\phi} : V \to \mathbb{R}^n$， $\vec{\phi}(v) = (1, \phi(v), \phi^2(v), \ldots, \phi^{n-1}(v))$ とすると

$$
\bigotimes_{v\in\pi} \vec{\phi}(v) = \left(1, \phi(\pi), \phi^2(\pi), \ldots, \phi^{n-1}(\pi)\right)^{\mathrm{T}} \tag{14}
$$

である．ただし， G 上の有向パス π に対して $\phi^n(\pi) \stackrel{\mathrm{def}}{\equiv} \left(\sum_{v\in\pi} \phi(v)\right)^n$ とする．ここから，ある系列上においてその頂点上に定義された関数の和の任意のべき乗をすべての系列について総和することが可能となることが分かる．

5.2 複素数

実数と同様，複素数 $\mathbb{C} \stackrel{\mathrm{def}}{\equiv} \{x+iy \mid x, y \in \mathbb{R}, i \text{は虚数単位}\}$ もまたその通常の加算・通常の乗算によって半群を成す．すなわち， $(\mathbb{C}, +, \times, 0, 1)$ は半群となり，頂点上に複素数の重みを定義した場合でも最短距離や周辺化最短距離は定義でき，それらは Algorithm 1, Algorithm 2 で計算可能である．ここで特記すべきことは $\phi : V \to \mathbb{R}$ に対して，頂点の重み w を $w[v] = \cos(\phi(v)) + i \sin(\phi(v))$ と定義することで以下が成立することである．

$$
w[\pi] = \cos\left(\sum_{v\in\pi} \phi(v)\right) + i \sin\left(\sum_{v\in\pi} \phi(v)\right) \tag{15}
$$

すなわち，ある系列上においてその頂点上に定義された関数の和の値に関する余弦および正弦をすべての系列について総和することが可能となることが分かる．

この事実とフーリエ級数展開を用いて，非常に広範な形式の和が近似計算できる可能性があることを示しておく．

今，$^\forall\pi \in \Pi(\mathrm{src},\mathrm{snk})$ に対して，π 上の頂点に定義された関数 ϕ の和が，何らかの関数 h の引数となっている場合 $h\left(\sum_{v\in\pi} \phi(v)\right)$ を考える．h が区間 $[\min_{\pi\in P(\mathrm{src},\mathrm{snk})} \sum_{v\in\pi} \phi(v), \max_{\pi\in P(\mathrm{src},\mathrm{snk})} \sum_{v\in\pi} \phi(v)]$ を含む適切な範囲 $[-L, L]$ で周期関数であり，フーリエ級数展開可能だとすると，

$$
\begin{aligned}
&\sum_{\pi\in\Pi(\mathrm{src},\mathrm{snk})} h\left(\sum_{v\in\pi} \phi(v)\right) \\
&= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_k \sum_{\pi\in\Pi(\mathrm{src},\mathrm{snk})} \cos\left(\frac{n\pi \sum_{v\in\pi} \phi(v)}{L}\right)\right) \\
&\qquad + \sum_{n=1}^{\infty} \left(b_k \sum_{\pi\in\Pi(\mathrm{src},\mathrm{snk})} \sin\left(\frac{n\pi \sum_{v\in\pi} \phi(v)}{L}\right)\right)
\end{aligned}
\tag{16}
$$

ここで $a_k = \frac{1}{L}\int_{-L}^{L} h(x)\cos\left(\frac{n\pi s}{L}\right)dx$ $b_k = \frac{1}{L}\int_{-L}^{L} h(x)\sin\left(\frac{n\pi s}{L}x\right)dx$ である．上式において a_k, b_k はトレリスの構造と無関係に h だけから決まる値であり，また，和 $\sum_\pi \cdot, \sum_\pi \cdot$ は先に述べたとおり $(\mathbb{C}, +, \times, 0, 1)$ に対する前向きの再帰によって求まる．このことから，この形式の和はフーリエ級数展開したものを有限項数で打ち切って計算することで近似計算できることが示唆される．

5.3 双対数

複素数の変種である，**双対数**について触れておく．$\mathbb{D}$ を $\left\{x + y\varepsilon \mid x \in \mathbb{R}, y \in \mathbb{R}, \varepsilon^2 = 0\right\}$ なる集合とする．ε は虚数単位の一種で，通常の虚数単位が 2 乗すれば -1 となるのに対して，ε は 2 乗すれば 0 となる虚数単位である．この集合上で複素数と同様，可算 $(x_1 + y_1\varepsilon) + (x_2 + y_2\varepsilon) = (x_1 + x_2) + (y_1 + y_2)\varepsilon$ および乗算 $(x_1 + y_1\varepsilon)\cdot(x_2 + y_2\varepsilon) = x_1x_2 + (x_1y_2 + x_2y_1)\varepsilon$ が定義できる．そして，複素数同様やはりこの双対数も $(\mathbb{D}, +, \times, 0, 1)$ が半群をなす．

このような数を持ち出す利点は，これが導関数の数値評価を行うための便宜上の代数として扱えることにある．たとえば，変数 $x \in \mathbb{R}$ によってパラメタ化された実数上の半環 $\mathcal{T} = (\mathbb{R}, \oplus, \otimes, \bar{0}, \bar{1})$ による以下のような和を考える．

$$\mathcal{D}_{\mathcal{T},w}(\mathrm{src}, \mathrm{snk}; x) = \bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \bigotimes_{v \in \pi} w[v; x] \tag{17}$$

ここで，$\oplus, \otimes$ は四則演算および初等関数を有限個組み合わせた計算からなるものとする．このとき，$w[v; x]$ を $w[v; x] + \varepsilon\frac{\partial}{\partial x}w[v; x]$ と置き換えることによって得られる双対数上の半環 $(\mathbb{D}, \oplus, \otimes, \bar{0}, \bar{1})$ による和について以下が成り立つ．

$$\begin{aligned}&\bigoplus_{\pi \in \Pi(\mathrm{src},\mathrm{snk})} \bigotimes_{v \in \pi} (w[v; x] + \varepsilon\frac{\partial}{\partial x}w[v; x]) \\ &= \mathcal{D}_{\mathcal{T},w}(\mathrm{src}, \mathrm{snk}; x) + \varepsilon\frac{\partial}{\partial x}\mathcal{D}_{\mathcal{T},w}(\mathrm{src}, \mathrm{snk}; x)\end{aligned} \tag{18}$$

すなわち，(17) における計算をすべて双対数のものに取り替えることで，計算の結果の虚部に (17) の導関数の数値評価が現れる．ただし，$\oplus, \otimes$ における実数の四則演算は $\mathbb{D}$ のものと置き換え，また任意の実数上の初等関数 $f(x)$ は双対数に対してその定義を $f(x + y\varepsilon) = f(x) + \varepsilon y\frac{\partial}{\partial x}f(x)$ と拡張する．

この操作は**自動微分**と呼ばれる数値微分法を双対数という代数的な表現で定式化したものに相当する [6][1]．これによって，最適化において重要な計算となる微分勾配の計算をもこのような代数的な抽象化で統一的に記述できることが分かる．なお，たとえば (17) の和を計算する場合，トレリスはそれ自身が (17) に対する簡素化された計算グラフをとして捉えることができる．従って，前向きの再帰による動的計画法による計算過程を双対数に置き換えたものは forward accumulation と呼ばれる形式の自動微分に対応する．一方で，前向き後ろ向きアルゴリズムにおける計算過程を双対数に置き換えたものは backward accumulation と呼ばれる形式に対応する．$\mathbb{R}^n \to \mathbb{R}$ $(n >> 1)$ なる関数の場合，特性上 backward accumulation が有利である．系列データに対する機械学習・データマイニングの文脈では，ポテンシャル関数などが高次元ベクトル空間上で定義されている場合が多いため，用いられるとすれば backward accumulation が多用されるであろう．

6 応用

この節では，本論文で示した代数的抽象化や，それに当てはまる具体的な代数の例がどのような実際的な応用を持ちうるのかを説明する．

まず，代数的抽象化それ自身の応用と言える実装上の恩恵について説明する．本論文に示したような代数的な枠組みに基づいてアルゴリズムを記述することは，そのまま**汎用アルゴリズム**を構築することへとつながる．この意味は，本論文で定義した代数的な性質を満たすものに対してならば何にでも機能する汎用なアルゴリズムの鋳型が提供できるのである．これにより，本論文が示す枠組みに沿うことさえ確認すれば，再び同じ目的のアルゴリズムを再発明・再実装する必要を可能な限り避けることができるようになる．本論文における代数的抽象化に基づけば，たとえば C++ のような総称プログラミングによる設計と実装が可能なプログラミング言語であれば，テンプレートのようなプログラミング言語機能を用いることにより，汎用なアルゴリズムの雛形を用意しておくことができるだろう．

系列データに対する機械学習・データマイニングにおいては，系列上の頂点や辺に関する特徴を捉える素性関数の和をその系列の特徴を捉える量とすることが極めて多い．したがって，系列上の頂点に定義された関数の和を基に，それを変形・変換した形式の総和が取れる意義は大きい．実際，5.1 に挙げた例から，このような系列上での和をべき乗しつつ適当なポテンシャルと組み合わせることで，そのようなポテンシャルが形成する分布上でこれら素性関数の和の，高次も含めたモーメントを算出することができる．また，複数の素性関数の多元モーメントを計算できるよう代数を設計することも可能である．特に各系列が対数線型モデルをポテンシャルに持つ場合，正規化定数自身が素性関数のモーメント母関数に，正規化定数の対数がキュムラント母関数となるため，素性関数のモーメントを

計算できることはただちにこのようなモデルに対する最適化の選択肢も広がることに直結する．

学習・最適化における微分演算の重要性は明らかである．この重要な演算のアルゴリズムが，通常なされるような記号操作だけで導出されるばかりでなく，代数的な置き換えによって比較的簡便にアルゴリズムが導出できる意義は大きい．特に汎用アルゴリズムや総称プログラミングの文脈では，総和計算のアルゴリズムがほぼ機械的に総和計算の導関数の値を計算するアルゴリズムとして再利用される．系列データに対する最適化の文脈での具体的な使用例は存在するが，代数的抽象化を通した統一的な記述や汎用アルゴリズム・総称プログラミングの文脈との親和性の高さについては，それほど強く主張されてこなかったように思われる．

7　今後の課題とまとめ

本論文では，前向き後ろ向きアルゴリズムに対して代数的な一般化を提案した．また，この代数の枠組みに収まる具体例のうち，系列データに対する機械学習・データマイニングに大きな意義のあるものをいくつか挙げた．今後は本論文で提案した枠組みを元に，より高度で柔軟な機械学習・データマイニング・各種最適化に関して，それが系列データの解析などにおいても選択肢が出来，柔軟に対処することが重要だと思います．

参考文献

[1] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics (SIAM), 2008.

[2] S. Kakade, Y. W. Teh, and S. Roweis. An alternate objective function for Markovian fields. In *Proc. of the ICML 2002*, Vol. 19, 2002.

[3] T. Kudo, K. Yamamoto, and Y. Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP*, Vol. 2004, 2004.

[4] Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *HLT-NAACL 2007; Companion Volume, Short Papers*, pp. 109–112, April 2007.

[5] M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, Vol. 7, No. 3, pp. 321–350, 2002.

[6] D. Piponi and ESC Entertainment. Automatic differentiation, c++ templates, and photogrammetry. *The Journal of Graphics Tools*, Vol. 9, p. 61, 2004.

[7] S. Sarawagi and W.W. Cohen. Semi-markov conditional random fields for information extraction. *Advances in Neural Information Processing Systems*, Vol. 17, pp. 1185–1192, 2005.

[8] S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. of ICML 2006*, pp. 969–976, 2006.