Vol.2015-NL-221 No.16
Vol.2015-SLP-106 No.16
2015/5/26

逐次最適解更新による頑健な単語分散表現の学習方式

鈴木　潤[1,a)]　永田昌明[1,b)]

概要：SkipGram, GloVe といった対数双線形言語モデルに属する単語分散表現のモデルは，これまで確率的勾配法 (SGD) やその拡張である AdaGrad といった勾配に基づくオンライン学習アルゴリズムを用いてパラメタ推定を行ってきた．しかし，対数双線形言語モデルと勾配に基づくパラメタ推定法の組み合わせは，解の収束性や再現性といった観点で，必ずしも適切な選択とは言えない．本稿では，より信頼性の高い単語分散表現を獲得する枠組みを構築することを目的として，対数双線形言語モデルが持つ性質に対応したパラメタ推定法を提案する．

1. はじめに

単語間の意味的な類似度，或いは，関連度といった情報をベクトル空間上に埋め込む単語分散表現の研究は，自然言語処理の研究分野で古くから扱われてきた研究トピックである．SkipGram，CBoW[1] に代表される対数双線形言語モデル (log-bilinear language model)[2] に属する単語分散表現のモデルは，近年多くの注目を集め様々な派生研究が盛んに行われている．また，ここ 2〜3 年で急速に研究が発展した比較的新しいモデルであるにもかかわらず，多くの自然言語処理タスクの基本ツールとして既に様々な場面で利用されている．このように，対数双線形言語モデルによる単語分散表現が自然言語処理分野で注目されている理由には，大きく二つの要因が考えられる．

一つ目は，単語間の関係をベクトル空間内の単語ベクトル間の加減算の関係として埋め込む新しい単語埋め込みタスクである単語アナロジータスクが提案されたことである．対数双線形言語モデルに属するモデルは，特にこのアナロジータスクにおいて，他のモデルと比較して非常に高い性能が出せることが知られている．

もう一つの要因は，モデルが非常に単純であるため，学習データ，および，語彙数が大規模化しても現実的な時間でパラメタ推定が可能という点である．現状，パラメタ推定アルゴリズムの更なる高速化や処理の簡略化といった様々な改良を経て，高品質な単語分散表現を誰でも比較的手軽に獲得できるようになった．特に，分散並列処理のような大規模な計算環境や特殊な計算機がなくても，通常のデスクトップ PC やラップトップ PC でも，無理なく利用できる．また，フリーのプログラムが公開されているため，一般ユーザが使いやすい状況が確立している．これらのことが爆発的な普及の大きな要因になっていると考えられる．

現在，単語分散表現としての機能拡張や単語間の新しい関連性を埋め込むことを可能とする拡張といった研究トピックが，対数双線形言語モデルの主な研究トピックであると考えられる．それに対し，本稿では，対数双線形言語モデルの学習法（パラメタ推定法）の改良に焦点を当てる．対数双線形言語モデルに関する既存研究では，基本的に勾配法やその拡張である AdaGrad[3] など勾配に基づく最適化アルゴリズムを用いてパラメタ推定が行われており，パラメタ推定法そのものの改良に類する研究はほとんど見られない．しかし，勾配法は，一般の凸最適化問題では非常に良好なアルゴリズムの一つして知られているが，非凸最適化問題である対数双線形言語モデルに対しても同様に適したアルゴリズムであるかは定かでは無い．本稿で議論するが，実際，対数双線形言語モデルにおいてはこれら勾配に基づく学習アルゴリズムでは幾つかの潜在的な観点で考慮すべき課題が存在する．

そこで本稿では，既存のパラメタ推定時の課題を挙げ，それらの課題に対応した新しいパラメタ推定法を提案する．より具体的には，パラメタ推定に用いる既存の目的関数を凸最適化問題，かつ，解析的に解が求まる部分最適化問題に分割し，その部分最適化問題を順次解いていくことで，元の最適化問題の解を見つけるといった，従来の勾配法とは全く異なるアプローチにより構築されたパラメタ推定アルゴリズムである．実際に提案法を用いることで，従来法が潜在的に持つ課題の幾つかを克服したり軽減したり

[1] 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation
[a)] suzuki.jun@lab.ntt.co.jp
[b)] masaaki.nagata@lab.ntt.co.jp

できることを示す．

実験では単語分散表現の研究で広く用いられるベンチマークデータを用いて，提案法のパラメタ推定時の挙動と，パラメタ推定後に得られた単語分散表現の品質を比較することで，提案法の利点，欠点を調査する．最終的に提案法は，従来用いられてきた SDG や AdaGrad の代替法として有効であることを示す．

2. 対数双線形言語モデル（LBL モデル）

これまでに単語分散表現を獲得する多種多様な方法が提案されてきた．本稿では，対数双線形言語モデルとみなすことができる方法を取りあげて議論を行う．以降，対数双線形言語モデルを ‘**LBL モデル** (Log-bilinear language model)’ と略記する．

LBL モデルは，ニューラルネット言語モデルの一種に分類される．ニューラルネット言語モデルでは，一般的に得られる単語分散表現の品質は良好である反面，大規模なデータや語彙を用いた学習が計算量的に高コストであるという欠点を持つ．この問題意識から，ニューラルネット言語モデルを簡略化した派生形として考案されたのが LBL モデルである．事実，入力層と出力層のみで中間層が存在しない，最も単純なニューラルネット言語モデルである．

2.1 LBL モデルの定義

$\mathcal{V}$ を語彙，或いは単語の集合とする．本稿では，集合の絶対値の形式で集合に含まれる要素数を表す．よって，$|\mathcal{V}|$ は語彙数，或いは，集合 $\mathcal{V}$ に含まれる単語数である．

本稿で取り扱う LBL モデルでは，二種類のベクトルを語彙中の各単語にそれぞれ割り当てることを仮定する[*1]．また，それらのベクトルは全て D 次元の固定長と仮定する．ここでは，ニューラルネット言語モデルの形式に則って，議論の際に二種類のベクトルを区別するために $\mathbf{o}$ を「出力ベクトル」，$\mathbf{e}$ を「入力ベクトル」と便宜上呼ぶこととする．

次に，$\mathbf{o}_j$ を j 番目の単語に割り当てられた出力ベクトル，$\mathbf{e}_i$ を i 番目の単語に割り当てられた入力ベクトルとする．本稿では，曖昧さを排除する目的で，j を出力ベクトル $\mathbf{o}$ の添字，i を入力ベクトル $\mathbf{e}$ の添字として必ず用いることとする．$1 \leq i \leq |\mathcal{V}|$，$1 \leq j \leq |\mathcal{V}|$ の関係が必ず成り立ち，$i = j$ の場合は同じ単語に割り当てられた入力および出力ベクトルを意味する．

このとき，LBL モデルの一般形として．ある単語（列）が出現した条件下で，j 番目の単語が出現する確率を，単語列を入力ベクトル添字のリスト $\mathcal{H}$ を使って以下の式で表すことができる．[*2]

$$p(j|\mathcal{H}) \propto s_{\mathcal{H},j} = \sum_{i \in \mathcal{H}} \mathbf{e}_i \cdot \mathbf{o}_j \tag{1}$$

LBL モデルでは，$s_{\mathcal{H},j}$ を用いて様々なモデルを定義できる．実際，SkipGram，CBoW[1]，GloVe[4] といった有名なモデルは，全て LBL モデルの特別な形と見做すことができる．

また，SkipGram や GloVe では，入力と出力の一対一の関係でモデル化する．よって入力は必ず一つなので，LBL モデルの特殊ケースとして以下のように入力と出力ベクトルの内積で定義される．

$$p(j|i) \propto s_{i,j} = \mathbf{e}_i \cdot \mathbf{o}_j, \tag{2}$$

本稿では，以降の議論を単純化するために，式 2 のように入力と出力のペアによる LBL モデルの形式を用いて議論を行う．

2.2 パラメタ推定方式

学習データを $\mathcal{D}$ とする．SkipGram や GloVe の学習データは，語彙の添字を用いて $\mathcal{D} = \{(i_n, j_n)\}_{n=1}^{N}$ と書ける．

$\mathcal{O}$ と $\mathcal{E}$ をそれぞれ，出力ベクトルまたは入力ベクトルを添字の順番で並べた長さ $|\mathcal{V}|$ のリストとする．つまり，$\mathcal{O} = (\mathbf{o}_1, \ldots, \mathbf{o}_{|\mathcal{V}|})$，$\mathcal{E} = (\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{V}|})$ である．このとき，LBL モデルのパラメタ推定は，目的関数 J を用いて以下の最適化問題の形式で定義できる．

$$(\hat{\mathcal{E}}, \hat{\mathcal{O}}) = \arg\min_{\mathcal{E},\mathcal{O}} \{J\} \tag{3}$$

2.2.1 GloVe の目的関数

本稿では，後述する提案法との兼ね合いで特に GloVe の目的関数に着目する．まず GloVe では，学習データの共起頻度 $F_{i,j}$ は事前に数え上げられていることを前提とする．次に，（重み付き）共起頻度が 0 より大きい値を獲得した単語ペアに対して，（重み付き）共起頻度の対数 $\log(F_{i,j})$ が i, j 成分となる行列を考える．最後に GloVe の目的関数は，その行列成分をに対し，（重み付き）二乗誤差が最小になるように入力ベクトル $\mathcal{O}$ および出力ベクトル $\mathcal{E}$ のパラメタを推定する問題として以下のように定式化される．

$$J = \sum_i \sum_j \beta_{i,j} (s_{i,j} - \log(F_{i,j}))^2 \tag{4}$$

ただし，$\beta i, j$ は各共起頻度の近似の優先度に関わる係数であり，$F_{i,j} = 0$ のとき $\beta_{i,j} = 0$ が成り立つように定義する．これは，共起しなかった入力と出力はパラメタ推定時に考慮しないための措置である．

2.3 最適化アルゴリズム

[*1] LBL モデルの初期は行列を割り当てているが，近年ではベクトルを用いるのが一般的なので，こちらを用いる．

[*2] $Q_{\mathcal{H},j} = \mathbf{q}_{\mathcal{H}} \cdot \mathbf{o}_j + b_j$ のように，バイアス項 b_j を明示的に入れて定式化する場合もあるが，例えば各ベクトルを $D+1$ 次元にし，すべての i に対して $c_{i,D+1} = 1$ と固定することで $b_j = w_{j,D+1}$ で代用して等価な式が得られる．よってここでは以降の議論に影響を与えないので簡単のためバイアス項は明示的には扱わない．

次に，最適化アルゴリズムについて議論を行う．主に機械学習の研究分野で，様々な特徴を持った多種多様な最適化アルゴリズムが考案されている．一般論として，一つの同じ目的関数に対して，選択可能な最適化アルゴリズムは幅広い選択肢がありえる．ただし，適用可能な最適化アルゴリズムを全て網羅して議論を行うのは困難である．そこでここでは，現在広く用いられているフリーソフトウェアに実装されている最適化アルゴリズムに着目して議論を行う．例えば，以下のソフトウェアを考える．

- 文献 [1], [5] のオリジナル実装であり，SkipGram, CBoW の実装を含む `word2vec`[*3].
- 文献 [6] の実装であり，`word2vec` の再実装および機能拡張に相当する `word2vecf`[*4].
- 文献 [4] のオリジナル実装である `glove`[*5].

これらの実装では，全てオンライン学習での確率的勾配法 (SGD)，または，その拡張にあたる AdaGrad[3] を用いてパラメタ推定が行われている．

例えば，式 4 の勾配は以下のように書ける．

$$\nabla_{\mathbf{e}_i} J = \sum_j 2\beta_{i,j}(\mathbf{e}_i \cdot \mathbf{o}_j - \log(F_{i,j})) \tag{5}$$

$$\nabla_{\mathbf{o}_j} J = \sum_i 2\beta_{i,j}(\mathbf{e}_i \cdot \mathbf{o}_j - \log(F_{i,j})) \tag{6}$$

勾配法に基づく方法では，基本的にこの勾配に学習率などの係数を乗算したものをパラメタに加算することでパラメタを更新していく．

3. パラメタ推定アルゴリズムの課題

$\mathcal{O}$ と $\mathcal{E}$ をそれぞれ，出力ベクトルまたは入力ベクトルを添字の順番で並べた長さ $|\mathcal{V}|$ のリストとする．つまり，$\mathcal{O} = (\mathbf{o}_1, \ldots, \mathbf{o}_{|\mathcal{V}|})$，$\mathcal{E} = (\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{V}|})$ である．このとき，例えば式 4 は，$\mathcal{E}$ に属するパラメタを全て任意の実数値で固定した場合 $\mathcal{O}$ に属するパラメタに関しては凸最適化問題になる．同様に，$\mathcal{E}$ に属するパラメタを全て任意の実数値で固定した場合 $\mathcal{O}$ に属するパラメタに関しても凸最適化問題である．こういった性質を持つ最適化問題を「**双凸 (biconvex) 最適化問題**」[*6]という [7]．双凸最適化問題となるのは，LBL モデルが内積の形式でモデルが定義されていることが大きな要因となっているためであり，ここでは明確に示さないが，SkipGram や CBoW の目的関数も全て双凸最適化問題のクラスに属する．

一般論として，凸最適化問題に属するパラメタ推定問題は，主に機械学習の研究分野でこれまで多くの研究成果があり，かつ非常に優れた最適化アルゴリズムが各種存在する．また，大域的最適解を求めることができ，得られる結果も毎回（計算誤差などを除いて）同じものを得ることも可能である．

一方，非凸最適化問題は，凸最適化問題と比べて最適解を得るのが計算量的に容易でない場合が多い．一般論としては，局所最適解を見つけてそれを解とみなしたり，初期値を複数回変えて最もよい結果を選択して使うなど，本質的に凸最適化より多くの考慮すべき点が存在する．また，大域的最適解を得難いという観点で，設定によって毎回最適化の結果が違うということが往々にしてあり得る．

本稿の議論の対象となっている LBL モデルのパラメタ推定問題は，前述の通り双凸最適化問題に属するが，これは，非凸最適化問題の一種である．よって，パラメタ推定時に考慮すべき点がいくつか存在する．具体的に，LBL モデルのパラメタ推定問題に関して，本稿では以下の 5 点の課題を取り上げる．

(1) パラメタ（ベクトルの要素）の初期値
(2) 学習率
(3) 並列実行性
(4) 分散表現の各要素の収束性が理論的に保証できない
(5) 分散表現の再現性

以下各項目の詳細について述べる．

3.1 パラメタ（ベクトルの要素）の初期値

LBL モデルのパラメタ推定は，非凸最適化なので，勾配法に基づく局所最適解を求めるパラメタ推定アルゴリズムでは，最終的に得られる解は初期値に大きく依存する．

また，LBL モデル特有の問題として，一般的な勾配法に基づくアルゴリズムを用いている場合，例えば全てのベクトルを同じ値，例えば 1，で初期化すると問題が生じる．それは，最終的に得られるベクトルが，ベクトル毎に全ての要素の値が同じになるといった現象である．具体的には，$\mathbf{o} = (0.5, 0.5, \ldots, 0.5)$，$\mathbf{e} = (0.1, 0.1, \ldots, 0.1)$ のような状態となる．

なぜそのようなことが起きるかと言えば非常に単純なことで，勾配法によるパラメタ更新では，各ベクトル単位で要素の値が全て同じだと，勾配の値も同じになるので，結局更新幅も全ておなじとなり，いつまでたっても同じ値を保ったままパラメタ推定の処理が続くことになる．端的な例として，$\beta_{1,1} = 0.5$，$\log(F_{1,1}) = 0.05$ のとき，$\mathbf{o}_1 = (0.5, 0.5, 0.5)$，$\mathbf{e}_1 = (0.1, 0.1, 0.1)$ の場合，式 5 や 6 に従って勾配を計算すると，$\nabla_{\mathbf{o}_1} J = (0.01, 0.01, 0.01)$，$\nabla_{\mathbf{e}_1} J = (0.05, 0.05, 0.05)$ のようになって，更新後も同じベクトル内の要素の値は同じ状態が続いてしまう．

このような現象を回避するには，単純に乱数を用いて全てのパラメタを初期化すれば十分であり，実際の実装でもそのように組み込まれている．ただし，起きる可能性は低いにせよ，潜在的にはベクトルの一部でこのような現象が

[*3] https://code.google.com/p/word2vec/
[*4] http://www.bitbucket.org/yoavgo/word2vecf
[*5] http://nlp.stanford.edu/projects/glove/
[*6] 「biconvex」に相当する日本語が不明なので，ここでは前述の「bilinear」が「双線形」と訳されたのに合わせて「双凸」と記述することにする．

起こる可能性を完全に排除することは，既存手法の範囲内では難しい．

3.2 学習率

詳細は実験結果にて述べるが，AdaGrad を用いる場合でも，初期学習率の設定により，得られる単語分散表現の質はしばしば大きく異なる．よって，例えばもし真面目により良い単語分散表現を得たいと思ったら，複数回実行し，それらを全て評価して最良の分散表現を選択する必要が生じる．

3.3 並列実行性

`word2vec` や `glove` の実装では非同期型の並列処理が用いられている．これによりパラメタ推定が高速化される反面，パラメタはお互いに依存関係を持つので，パラメタ更新時のタイミングに依存してパラメタ更新の一意性は損なわれる．

ただし，これは並列実行をやめるか，同期処理を導入するといった容易な解決策が存在する．これらの解決策は，計算速度と解の一意性担保のトレードオフと考えることもできる．特に学習データや扱う語彙数が大規模になる場合は，非同期並列実行を利用しないと現実的な時間でパラメタ推定を終えるのは難しい場合が多い．

3.4 分散表現の各要素の収束性が理論的に保証できない

双凸最適化問題では，目的関数の値としては局所最適解に収束していっても，その際のパラメタは発散していっているといった現象が起こり得る [7]．これも端的な例で示すことができる．例えば，$\beta_{1,1} = 1$，$\log(F_{1,1}) = 2$ のとき，$\mathbf{o}_1 = (0.5, 2.0), \mathbf{e}_1 = (2.2, 0.1, 0.5)$ の場合，$J = 0.01$ である．また，$\mathbf{o}_1 = (440000.0021, 20000.0)$，$\mathbf{e}_1 = (10000, -22000)$ の場合も，$J = 0.01$ である．このように，乗算の総和を計算することから，プラスとマイナスで値を打ち消すことにより，個々の要素の値の絶対値はいくらでも大きくすることができる．特に勾配法に基づく既存手法だと，初期値，学習率の設定などを間違えると容易にパラメタが発散する現象がおきる．この課題に関しても実験にて現象を示す．

3.5 分散表現の再現性

上記課題 1 から 4 全てに関わることであるが，勾配法に基づくパラメタ推定では設定を少し変更しただけで，最終的に得られる単語分散表現の各要素の値は劇的に違う場合がしばしば起こり得る．場合によっては，設定を変更しなくても，得られるベクトルが同一にならない場合もある（課題 3）．研究的な側面では，例えば単語分散表現のベンチマークデータによる評価で大きな差異が出ないなら，単語分散表現の中身の値自体は考慮しなくても良いと言えるかもしれない．しかし実用面では，単語分散表現の各要素の値を利用して適用先のパラメタ調整などをするため，別の分散表現を同じパラメタで利用することは困難になるといったデメリットが生じる．

このことから，理想的には，凸最適化問題と同じように，同じあるいは似た設定で得られた単語分散表現は極力同じか類似しているといった性質があることが望ましい．

4. 逐次最適解更新アルゴリズム

本節では，第 3 節で議論した LBL モデルのパラメタ推定時の課題を克服または軽減することを目的とした新たなパラメタ推定法を提案する．

まず提案法を説明する前に，提案法の基本となる最適化アルゴリズムを説明する．その後，提案法の詳細を述べる．

4.1 準備

4.1.1 Alternating Convex Optimization (ACO)

双凸最適化問題に特化した最適化アルゴリズムの一つに **alternating convex optimization (ACO)** と呼ばれる方法がある．双凸最適化問題は 2 種類のパラメタ集合の一方を固定すると凸最適化問題と見なせるという性質を利用し，ACO では，2 種類のパラメタ集合のうち一方を固定した上でもう一方のパラメタのみを使って凸最適化問題を解く，という手続きを，固定するパラメタ集合を交互に替えながら繰り返し最適化を行う．例えば，式 3 に ACO を適用した場合，事前に定めた収束判定条件を満たすまで，ACO は以下の二つの最適化問題を繰り返し交互に解く．

$$\begin{aligned} \hat{\mathcal{E}} &= \arg\min_{\mathcal{E}} \{J(\mathcal{E}|\mathcal{O})\} \\ \hat{\mathcal{O}} &= \arg\min_{\mathcal{O}} \{J(\mathcal{O}|\mathcal{E})\} \end{aligned} \tag{7}$$

ACO の特徴として，特殊な場合を除いて双凸最適化問題で必ず局所最適解が得られることが理論的に保証できる [7]．また，双凸最適化問題においては，汎用的な非凸最適化アルゴリズムを適用するより ACO は良好な結果が得られると経験的に知られている．例えば，ACO またはその亜種は非負行列分解の解法として広く用いられている [8]．

4.1.2 Cyclic Ccoordinatewise Optimization

凸最適化法の一つとして，**cyclic coordinatewise optimization (CCO)** と呼ばれる方法がある [9]．CCO の基本概念の一つとして，個々の次元毎に最適化する計算コストは非常に小さいと見積もれる必要がある． 特に最適解が解析的に求まる場合は，最適化時の反復計算が不要になるため，一般的に計算コストを非常に低くすることができる．

4.2 定式化

大まかに言うと提案法は，前述の ACO と CCO の要素を組み合わせた最適化法である．$\mathcal{O}_d$ を，各出力ベクトルの d 番目の要素のみを抽出したリストとする．同様に $\mathcal{E}_d$

Algorithm 1: SPCU アルゴリズム

Input: $\mathcal{F}$: 共起頻度行列, D: ベクトルの次元, ϵ: 収束判定用定数

1:全出力ベクトル $\mathcal{O}$ と，全入力ベクトル $\mathcal{E}$ を準備
2:$t = 1$
3:**repeat** // outer loop
4: **for** $d \leftarrow 1, 2, \ldots, D$
5: **if** $t = 1$ // 初期値を設定するために初回のみ処理
6: $o_{j,d} \leftarrow 1\ \forall j$
7: $e_{i,d} \leftarrow$ 式 12 に基づいて値をセット $\forall i$
8: $(o_{j,d}, e_{i,d}) \leftarrow$ `CalcInitlVal`$(o_{j,d}, e_{i,d})$
9: **end_if**
10: **repeat** // ASO part (inner loop)
11: $o_{j,d} \leftarrow$ 式 11 に基づいて更新 $\forall j$ // CCO part
12: $e_{i,d} \leftarrow$ 式 12 に基づいて更新 $\forall i$ // CCO part
13: **until** `InnerLoopConvergenceCheck(`ϵ`)`=true
14: **end_for**
15: $t \leftarrow t + 1$
16:**until** `OuterLoopConvergenceCheck(`ϵ, t`)`=true

Output: $(\mathcal{O}, \mathcal{E})$

図 1 GloVe 目的関数に対する逐次最適解更新 (SPCU) アルゴリズムによる単語分散表現のパラメタ推定

を，各入力ベクトルの d 番目の要素のみを抽出したリストとする．$o_{j,d}$ を j 番目の出力ベクトル内の d 番目の要素を表すとすると，$\mathcal{O}_d = (o_{1,d}, \cdots, o_{|\mathcal{V}|,d})$ の関係が成り立つ．

更に, $\mathcal{O}_{-d}$ と $\mathcal{E}_{-d}$ を入力または出力ベクトルの d 番目の要素 '以外' の全ての要素で構成されるリストとする．このとき，各 $\mathcal{O}_d$ 或いは $\mathcal{E}_d$ を求める部分最適化問題は以下のように書ける．

$$\begin{aligned}\hat{\mathcal{O}}_d &= \arg\min_{\mathcal{O}_d} \{J(\mathcal{O}_d|\mathcal{O}_{-d}, \mathcal{E})\} \\ \hat{\mathcal{E}}_d &= \arg\min_{\mathcal{E}_d} \{J(\mathcal{E}_d|\mathcal{O}, \mathcal{E}_{-d})\}.\end{aligned} \tag{8}$$

ここでの提案法では，式 8 に示した部分最適問題を一定の順番で解いていくことで，元の最適化問題の解を得る方法論になる．この時，ポイントとなるのが，式 8 に示した個々の部分最適化問題は凸最適問題である上に，大域的最適解が，解析的に求めることができるという性質があることである．

まず，個々の最適化問題の最適解 $(\hat{o}_{j,d})_{j=1}^{|\mathcal{V}|} = \hat{\mathcal{O}}_d$ と $(\hat{e}_{i,d})_{i=1}^{|\mathcal{V}|} = \hat{\mathcal{E}}_d$ は，各部分最適化問題が対象とする固定されていないパラメタの偏微分が全て 0 になる点で得られる．例えば，目的関数 $J(\mathcal{O}_d|\mathcal{O}_{-d}, \mathcal{E})$ の $o_{j,d}$ に関する偏微分は以下のように書ける．

$$\partial_{o_{j,d}} J(\mathcal{O}_d|\mathcal{O}_{-d}, \mathcal{E}) = \sum_i \beta_{i,j} \left(e_{i,d} o_{j,d} - \tilde{z}_{i,j,d}\right) e_{i,d} \tag{9}$$

ただし,

$$\tilde{z}_{i,j,d} = \log(F_{i,j}) - \sum_{d':d \neq d'} e_{i,d'} o_{j,d'} \tag{10}$$

最適解 $\hat{o}_{j,d}$ は，$\partial_{o_{j,d}} J(\mathcal{O}_d|\mathcal{O}_{-d}, \mathcal{E}) = 0$ の時のパラメタなので，式 9 を変形して以下の関係が得られる．

$$\hat{o}_{j,d} = \frac{\sum_i \beta_{i,j} e_{i,d} (\tilde{z}_{i,j,d})}{\sum_i \beta_{i,j} (e_{i,d})^2} \tag{11}$$

同様にして，最適解 $\hat{e}_{i,d}$ も以下の関係が得られる．

$$\hat{e}_{i,d} = \frac{\sum_j \beta_{i,j} o_{j,d} (\tilde{z}_{i,j,d})}{\sum_j \beta_{i,j} (o_{j,d})^2} \tag{12}$$

つまり，個々の最適化問題は反復計算なしに最適解が容易に求まることがわかる．

この提案法の概要としては，大域的最適解が解析的に得られるまで問題を小さく分割し，最適解を逐次得ながら最適化していく方法とみなすことができる．よって，提案法を「**逐次最適解更新 sequential piece-wise closed form update (SPCU)** アルゴリズム」と呼ぶこととする．

図 1 に，SPCU の最適化アルゴリズムを示す．10 から 13 行目がメインのパラメタ更新の手続きである．注意点として，各次元のループはこの内側のパラメタ更新手続きの外側にある．これの意味することは，各次元ごとに最適値を決定していく手順になっていることである．

この部分は SPCU の大きな特徴にもなっている．つまり，SGD や AdaGrad を用いた従来の最適化では，全ての次元に対して一括で少しずつパラメタを更新する方法と言えるが，SPCU は，各次元毎に限定したパラメタ内の範囲で一気に最適解へパラメタを更新しながら全体として良いパラメタを得るという大きな違いがある．

4.3 課題に対する効果

第 3 節で議論した課題について，提案法がどのような位置づけにあるかを議論する．以下，各課題に対する SPCU アルゴリズムの位置づけの概略である，

(1) パラメタ（ベクトルの要素）の初期値
乱数不要.
(2) 学習率
学習率そのものが不必要.
(3) 並列実行性
依存関係が無いパラメタで並列化が可能．更新タイミングは結果に影響を与えない
(4) 分散表現の各要素の収束性が理論的に保証できない
各次元毎に最適化するので，問題は発生しない.
(5) 分散表現の再現性
収束判定を固定すれば，毎回必ず同じベクトルが獲得できる.

以下各項目の詳細について述べる.

4.3.1 パラメタ（ベクトルの要素）の初期値

SPCU ではベクトルの全ての要素の初期値として 0 より大きい一定の実数，例えば 1，を初期値に設定したとしても，理論的にも実験的にも全く問題なくパラメタ推定が行える．しかし，GloVe の目的関数である式 4 からもわか

るように，入力ベクトルと出力ベクトルの内積を用いて二乗誤差を最小化するので，入力ベクトルか出力ベクトルの一方が極端に大きい値をとって，もう一方が小さい値をとるといった，不均一なパラメタを得る可能性がある．例えば，$\log F_{i,j} = 1$ の時，$e_{1,d} = 100$ と $o_{1,d} = 0.01$ でも $e_{1,d}o_{1,d} - \log F_{i,j} = 0$ となるし，$e_{1,d} = 1$ と $o_{1,d} = 1$ でも $e_{1,d}o_{1,d} - \log F_{i,j} = 0$ である．一般論として，後者の $e_{1,d} = 1$ および $o_{1,d} = 1$ を得る方がよりよいと考え，以下の式を用いて図 1 中の初期値を決定する処理 `CalcInitVal` を以下のように定義する．

$$\begin{aligned} o_{j,d} &= \mathrm{sgn}(o_{j,d})\sqrt{|o_{j,d}e_{j,d}|} = \sqrt{|e_{j,d}|} \\ e_{i,d} &= \mathrm{sgn}(e_{i,d})\sqrt{|o_{i,d}e_{i,d}|} = \mathrm{sgn}(e_{i,d})\sqrt{|e_{i,d}|}, \end{aligned} \tag{13}$$

ただし，$\mathrm{sgn}(a)$ を，$a < 0$ のとき $\mathrm{sgn}(a) = -1$，それ以外の時に $\mathrm{sgn}(a) = 1$ とする．図 1，6 行目からわかるように必ず $w_{j,d} = 1$ なので，ここでの初期値は，一度計算した入力ベクトルの最適解の平方根を初期値として用いることを意味する．

4.3.2 学習率

SPCU では必ず各部分問題の最適解を計算して更新する処理となるので，学習率に相当するパラメタは不要であり考慮する必要がない．このことから，人手により設定する必要があるパラメタが一つ減ったことも，付随する SPCU の利点の一つと言える．

実際に，各部分最適化問題の最適解を逐次更新するパラメタ推定は，各次元毎にラインサーチをしながらパラメタ更新を行っていると見做すこともできる．つまり，各部分最適化問題の最適解の計算に適合的に学習率を決めている部分が陰に含まれているとみなせる．

4.3.3 並列実行性

図 1 中の 11 行目と 12 行目の処理は各 i または j に対して完全に依存関係が存在しない処理である．よって，全ての i または全ての j の処理は，並列実行可能である．この並列実行は，`word2vec` の実装などで用いられている非同期型の並列処理とは違い，個別の処理結果が他の処理に全く影響を与えないので，並列実行しても得られる解に影響を与えないという性質をもつ．この理由により，既存手法と比較してパラメタ更新時のタイミングに依存して一意性が損なわれる現象を排除することができる．この点も，SPCU に付随する利点の一つに数えられる．

4.3.4 分散表現の各要素の収束性が理論的に保証できない

要素の値が発散するそもそもの原因は，複数のパラメタを同時に更新することにより，結果として正負の値で打ち消しあって要素の値の絶対値を大き方向へ向かわせることが容易に可能なことに由来する．

一方，SPCU では，各部分最適化問題に対して，各ベクトル毎に一つの自由パラメタしか持たない．よって，SPCU の枠組みに則ってパラメタ更新を行っている限り，発散することはない．

4.3.5 ベクトルの各要素の値に対する再現性

課題 1 から 4 にも関連する事項であるが，これらが解決したことにより，パラメタ推定後に得られる解の頑健性が高くなる．具体的には例えば，収束判定のパラメタを固定することで，毎回必ず同じ単語分散表現を獲得できるようになる．

これらの理由から SPCU は，特に意識しなくても同じ学習環境を用いている限りは，必ず毎回同じ結果が得られるという利点が挙げられる．

5. 実験

本稿では，単語分散表現の性質の良さを評価する際によく用いられるベンチマークデータを利用して，パラメタ推定時の挙動と，パラメタ推定後に得られた単語分散表現に関して比較実験を行う．

5.1 比較手法

LBL モデル，特に GloVe の目的関数に関してのパラメタ推定法が本稿の主たる議論の対象なので，比較対象はパラメタ推定法である．ここでは提案法 (**SPCU**) と，`glove`[*7] に実装されている **AdaGrad**[3] の比較を行う．

パラメタ推定時の人手設定のハイパーパラメタは基本的に文献 [4] に記述されている通りの値を用いた．例えば，文脈単語の距離 $p = 10$，down scaling パラメタ $x_{\max} = 100$，および $\alpha = 3/4$ などである．

5.2 学習データ

学習データには，既存研究で最もよく用いられる英語 Wikipedia のデータを用いた．前処理として，英語 Wikipedia アーカイブから 2014 年 8 月のダンプをダウンロードし，xml 形式から本文部分を抽出，文区切り，標準的なトークン区切り，小文字化処理を行った．最終的に，18 億 (1.8B) トークンのデータとなった．また，文献 [4] の実験設定に合わせて，語彙数を頻度順に並べて 40 万単語に固定した（$|\mathcal{V}| = 400,000$）．

5.3 単語分散表現評価用ベンチマークデータ

本稿では，既存研究で用いられてきた数多くのベンチマークデータを利用して評価実験を行う．本稿で用いたベンチマークデータの概要ををを表 1 に示す．表中の語彙内データ数とは，前述の学習データで定義した語彙中にベンチマークデータの単語が含まれている数である．つまり，語彙に含まれていない場合は絶対に正解することができないので，語彙内データ数が正解の上限となる．

[*7] http://nlp.stanford.edu/projects/glove/

abbr.	データ数	語彙内データ数	参考文献
Similarity ベンチマーク (Similarity)			
MEM	3,000	3,000	Bruni [10]
MTK	287	285	Radinsky [11]
M&C	30	30	Miller [12]
RAR	2,034	1,721	Luong [13]
R&G	65	65	Rubenstein [14]
SCWS	2,003	1,979	Huang [15]
WSR	252	238	Agirre [16]
WSS	203	196	Agirre [16]
Analogy ベンチマーク (Analogy)			
GSE	8,869	8,869	Mikolov [1]
GSY	10,675	10,675	Mikolov [1]
MSY	8,000	8,000	Mikolov [17]

表 1 ベンチマークデータ

5.3.1 Similarity ベンチマーク

ベンチマークデータの一つ目のカテゴリは，単語間の意味的な類似性，或いは，関連性を評価するデータである．このカテゴリに属するベンチマークデータを総称して，本稿では '**Similarity**' ベンチマークと記述する．

Similarity ベンチマークに属するのデータは，基本構成要素として，任意の二つの単語に対して意味的に類似している，或いは，関連している度合いを，人間の主観評価によるレートが付与されていることを仮定する．よって，データ全体としては，（単語 1，単語 2，レート）という 3 つの情報を一つのエントリとしたリストで構成される．

単語分散表現のベンチマークデータとして利用する場合は，各エントリの二つの単語間の類似度を，例えば，各単語に割り当てられた単語分散表現間のコサイン類似度により計算する．その後，二単語間のコサイン類似度による順序と，正解の類似度の順序をスピアマンの順位相関係数を用いて比較する．最終的に，より順位相関係数が高い順序を与える単語分散表現がより品質の高い単語分散表現であると判定する．

本稿では，Similarity に属する全てのベンチマークデータのマイクロ平均を用いて最終的な評価を行った．

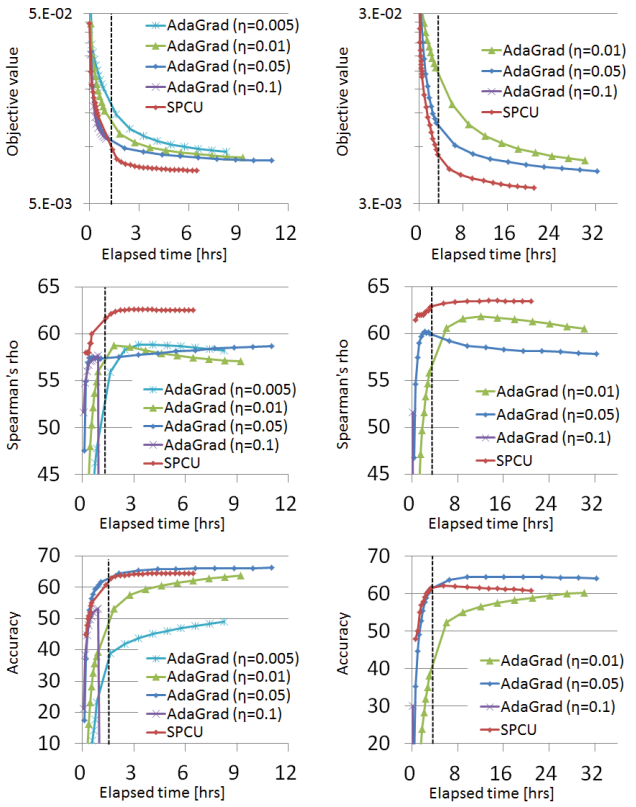
5.3.2 Analogy ベンチマーク

3 つの単語 a，b，c が与えられた時，Analogy ベンチマークは，「a is to b as c is to ?」という質問の答えとなる単語 d を予測する問題といえる．

Analogy ベンチマークでは，以下の式を用いて単語 d を予測する．

$$\hat{d} = \arg\max_{d \in \mathcal{V}} \{\cos(\mathbf{o}_d, \mathbf{o}_c - \mathbf{o}_a + \mathbf{o}_b)\} \qquad (14)$$

この単語 d の予測結果の正解率をが Analogy タスクの一般的な評価指標である．本湖では複数のデータを合わせて用いるので，全てのデータの解率のマイクロ平均で最終的に評価を行う．



(a) $D = 300$ (b) $D = 1000$

図 2 学習曲線: x 軸は実行時間 (hours) を表す．y 軸はサンプル単位の目的関数の値 (上段)，Similarity ベンチマークのスピアマンの順位相関のマイクロ平均 (中段)，Analogy ベンチマークの精度のマイクロ平均 (下段)，

6. 実験結果

図 2 に学習曲線とそれに合わせた評価結果を示す．まず，図からわかるように提案法である SPCU は，AdaGrad より，良好な目的関数の値を得ることができていることがわかる．また同時に，AdaGrad では，初期学習率によって大幅に学習の精度が違うことが容易に見て取れる．

結果を簡単にまとめると，AdaGrad では，なるべく大きな初期学習率から始めたいが，あまり大きくしすぎると（ここでは例えば $\eta = 0.1$）ベクトルが発散してしまい，学習が不可能となっている．逆に，初期学習率が小さすぎると（ここで例えば $\eta = 0.01$ 以下）学習があまりうまく進まう，目的関数の観点でもタスクの評価としても相対的に不十分な結果となっている．このように，ちょうど良い初期学習率 $\eta = 0.05$ を選択するのは，それなりにコストのかかることであり，かつ，選択に失敗した場合の悪影響が大きい．一方，SPCU はそもそも学習率のようなパラメタが存在しないので，特に注意深くパラメタ設定などしなくても頑健な結果を得られている．

次に，SPCU により得られた単語分散表現は，（最も良い）AdaGrad で得られた単語分散表現と比較して，Similarity

ベンチマークの精度が大幅に高く，Analogy ベンチマークの精度がやや低いという結果になった．実際には単語分散表現の中身を調査するなどして詳細に分析する必要があるが，一つの可能性として，SPCU アルゴリズムは次元毎に値を決定することで，様々な定性的なメリットを享受しているが，一方で，Analogy タスクでは他の最適化アルゴリズムより不向きな可能性が考えられる．この理由は，各次元毎に最適化を行うために D 次元空間内で軸方向に沿ってしか値を更新できないという制限が，Analogy タスクのように単語ベクトルの位置関係が非常に重要な問題では，悪影響を及ぼしている可能性が予想される．

7. おわりに

本稿では，LBL モデル，特に GloVe の目的関数に対して，元の最適化問題を凸最適化問題，かつ，解析的に解が求まる部分問題に分割し，その分割した最適化問題の最適解を逐次更新することで，元の目的関数の最適化を行う‘逐次最適解更新 (SPCU) アルゴリズム’を提案した．次に，本稿で取り上げた 5 つの観点で提案法はパラメタ推定時の課題を軽減もしくは解決していることを定性的に示した．また，実験では，これまで広く用いられてきた確率的勾配法 (SGD) の拡張である AdaGrad と比較を行い，最適化時の学習率に依存せずに頑健なパラメタ推定が可能であることを示した．今後は，SPCU アルゴリズムを LBL モデルの標準的なパラメタ推定アルゴリズムとなることを目指して，更なる利点と欠点の調査を行い，よりよいアルゴリズムに仕上げていく予定である．

参考文献

[1] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, *CoRR*, Vol. abs/1301.3781 (2013).

[2] Mnih, A. and Kavukcuoglu, K.: Learning word embeddings efficiently with noise-contrastive estimation, *Advances in Neural Information Processing Systems 26* (Burges, C., Bottou, L., Welling, M., Ghahramani, Z. and Weinberger, K., eds.), Curran Associates, Inc., pp. 2265–2273 (online), available from ⟨http://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive-estimation.pdf⟩ (2013).

[3] Duchi, J., Hazan, E. and Singer, Y.: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *J. Mach. Learn. Res.*, Vol. 12, pp. 2121–2159 (online), available from ⟨http://dl.acm.org/citation.cfm?id=1953048.2021068⟩ (2011).

[4] Pennington, J., Socher, R. and Manning, C.: Glove: Global Vectors for Word Representation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics, pp. 1532–1543 (online), available from ⟨http://www.aclweb.org/anthology/D14-1162⟩ (2014).

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J.: Distributed Representations of Words and Phrases and their Compositionality, *Advances in Neural Information Processing Systems 26* (Burges, C., Bottou, L., Welling, M., Ghahramani, Z. and Weinberger, K., eds.), Curran Associates, Inc., pp. 3111–3119 (online), available from ⟨http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf⟩ (2013).

[6] Levy, O. and Goldberg, Y.: Dependency-Based Word Embeddings, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland, Association for Computational Linguistics, pp. 302–308 (online), available from ⟨http://www.aclweb.org/anthology/P14-2050⟩ (2014).

[7] Gorski, J., Pfeuffer, F. and Klamroth, K.: Biconvex sets and optimization with biconvex functions: a survey and extensions., *Math. Meth. of OR*, Vol. 66, No. 3, pp. 373–407 (online), available from ⟨http://dblp.uni-trier.de/db/journals/mmor/mmor66.html#GorskiPK07⟩ (2007).

[8] Kim, J., He, Y. and Park, H.: Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework, *Journal of Global Optimization*, Vol. 58, No. 2, pp. 285–319 (online), available from ⟨http://EconPapers.repec.org/RePEc:spr:jglopt:v:58:y:2014:i:2:p:285-319⟩ (2014).

[9] Khare, K. and Rajaratnam, B.: Convergence of cyclic coordinatewise l1 minimization, *ArXiv e-prints* (2014).

[10] Bruni, E., Tran, N. K. and Baroni, M.: Multimodal Distributional Semantics, *J. Artif. Int. Res.*, Vol. 49, No. 1, pp. 1–47 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2655713.2655714⟩ (2014).

[11] Radinsky, K., Agichtein, E., Gabrilovich, E. and Markovitch, S.: A Word at a Time: Computing Word Relatedness Using Temporal Semantic Analysis, *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, New York, NY, USA, ACM, pp. 337–346 (online), DOI: 10.1145/1963405.1963455 (2011).

[12] Miller, G. A. and Charles, W. G.: Contextual Correlates of Semantic Similarity, *Language & Cognitive Processes*, Vol. 6, No. 1, pp. 1–28 (1991).

[13] Luong, T., Socher, R. and Manning, C.: Better Word Representations with Recursive Neural Networks for Morphology, *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, Sofia, Bulgaria, Association for Computational Linguistics, pp. 104–113 (online), available from ⟨http://www.aclweb.org/anthology/W13-3512⟩ (2013).

[14] Rubenstein, H. and Goodenough, J. B.: Contextual Correlates of Synonymy, *Commun. ACM*, Vol. 8, No. 10, pp. 627–633 (online), DOI: 10.1145/365628.365657 (1965).

[15] Huang, E. H., Socher, R., Manning, C. D. and Ng, A. Y.: Improving Word Representations via Global Context and Multiple Word Prototypes, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, Association for Computational Linguistics, pp. 873–882 (2012).

[16] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M. and Soroa, A.: A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches, *Proceedings of Human Language Technologies: The 2009 Annual Conference*

of the North American Chapter of the Association for Computational Linguistics, NAACL '09, Stroudsburg, PA, USA, Association for Computational Linguistics, pp. 19–27 (online), available from ⟨http://dl.acm.org/citation.cfm?id=1620754.1620758⟩ (2009).

[17] Mikolov, T., Yih, W.-t. and Zweig, G.: Linguistic Regularities in Continuous Space Word Representations, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, Association for Computational Linguistics, pp. 746–751 (online), available from ⟨http://www.aclweb.org/anthology/N13-1090⟩ (2013).