Paragraph Vectorのための二分法によるパラメータサーチの効率化の一手法

久保田大貴† 新妻 弘崇†† 太田 学††

† 岡山大学工学部情報系学科 〒 700–8530 岡山県岡山市北区津島中三丁目 1 番 1 号
†† 岡山大学大学院自然科学研究科 〒 700–8530 岡山県岡山市北区津島中三丁目 1 番 1 号
E-mail: †ptc13pgv@s.okayama-u.ac.jp, ††{niitsuma，ohta}@de.cs.okayama-u.ac.jp

あらまし Paragraph Vector を用いると文章を特徴ベクトルで表すことができ，この特徴ベクトルにより文書分類を高精度で行うことが可能であるとされている．しかし，高精度な文書分類を行うためには，Paragraph Vector の多数のパラメータをそれぞれ適切な値に設定する必要がある．ところが，パラメータの数とその範囲が大きいため，パラメータの探索に膨大な時間がかかる．そこで，我々は効率的にパラメータを探索する手法として，多次元二分法によるパラメータサーチ手法を提案する．この手法はパラメータ探索を効率化し，低コストで高精度な文書分類を実現する．本稿では Stanford Sentiment Treebank Dataset の映画のレビュー文の分類実験により提案手法の有効性を評価した．

キーワード word2vec，Paragraph Vector，ニューラルネットワーク，パラメータサーチ

1. はじめに

近年 Mikolov らによって提案された word2vec [1], [2] が様々な用途で利用されている．word2vec は，単語の特徴を低次元の数値ベクトルとして表現する手法である．その際，意味の近い単語同士は類似したベクトルになる．word2vec は，文章中のある単語をその単語の前後関係から予測するというタスクをニューラルネットワークに学習させた時に，そのニューラルネットワークの中間層で使われている重みの値を特徴ベクトルとして抽出するものである．この単語の前後関係を予測する方法としては，Skip-gram モデルと呼ばれる中心の単語から周辺の単語を予測するものと，Countinuous Bag-of-Words (CBOW) モデルと呼ばれる周辺の単語から中心の単語を予測するものがある．

文章の特徴を表現する手法には，TF-IDF や Bag-of-Words (BOW) のような単語の出現頻度を用いる手法が存在する．しかし，これらの方法では語順を考慮することができない．また同義語に対して類似したベクトルにならないことがあるという問題がある．word2vec はこれらの問題を解決することはできるものの，文章の特徴ベクトルを生成することはできない．そこで，word2vec と同じ原理で文章の特徴ベクトルを生成できるように word2vec を拡張したのが Paragraph Vector [3] である．Paragraph Vector を使うと映画のレビュー文章からレビュアがつけた映画の評価値を推定する問題 [4] において，Recurrent Neural Network を用いた手法よりも高精度な推定を実現できたことが報告されている [3].

原らは，Paragraph Vector のための二分法とランダムサーチを組み合わせた効率的なパラメータサーチ手法を提案した [5]. この手法は 1 つのパラメータを二分法で探索しつつ，他のパラメータをランダムサーチで探索する手法である．

本稿では原らの手法を拡張し，パラメータサーチをより効率的に行う手法を提案する．具体的には，二分法を適用するパラメータの数を増やし，多次元の二分法で探索することで，パラメータサーチにかかる時間を短縮する．

評価実験では，gensim ライブラリ [6] の doc2vec class を用いて Paragraph Vector を利用したロジスティック回帰による文書分類機能を実装し，これを用いて Stanford Sentiment Treebank Dataset [8] の映画レビュー文の分類を行う．

2. 関連研究

本研究で使用する分散表現である Paragraph Vector とその元になった word2vec，またパラメータサーチ手法について説明する．

2.1 分散表現

Paragraph Vector は文章を特徴ベクトルで表し，word2vec は単語を特徴ベクトルとして表す手法である．Mikolov らは word2vec で生成したある 2 つの単語ベクトルの差は，それぞれの単語ベクトルの関係を表現していると主張した [1], [2]．以下の式を例にこの単語ベクトルについて説明する．

$$\text{king} - \text{man} + \text{woman} \tag{1}$$

式 (1) は king，man，woman の単語ベクトルを用いた演算を表している．この式の演算結果は king と man の関係を woman に適用したものである．Mikolov らは word2vec で生成した単語ベクトルでこの演算をした場合，queen の単語ベクトルが生成されると主張した．

word2vec は特徴ベクトルの生成に学習コーパスが必要である．コーパス中の単語の前後関係を用いてニューラルネットワークにコーパスの単語を学習させ，特徴ベクトルを生成する．Mikolov らはここで用いられるコーパスの単語数が多い場合，単語ベクトルの次元数も多くすることで式 (1) のような演算の

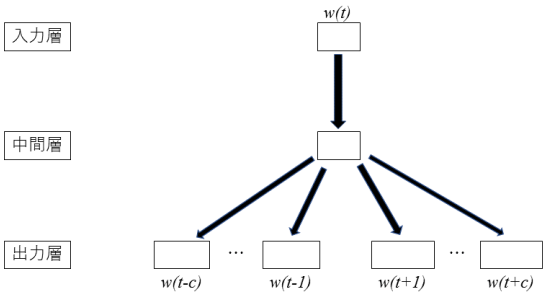

図 1　Skip-gram モデル

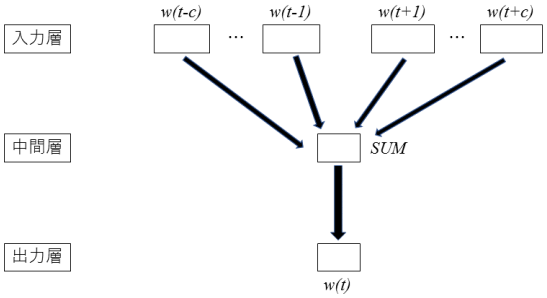

図 2　CBOW モデル

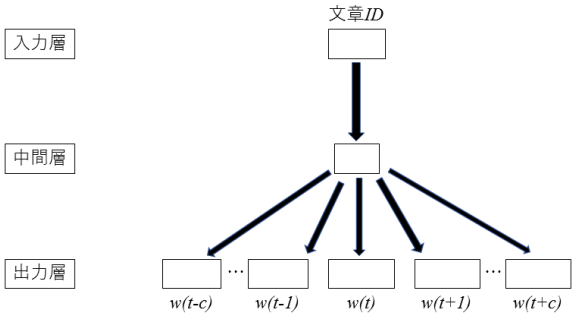

図 3　PV-DBOW モデル

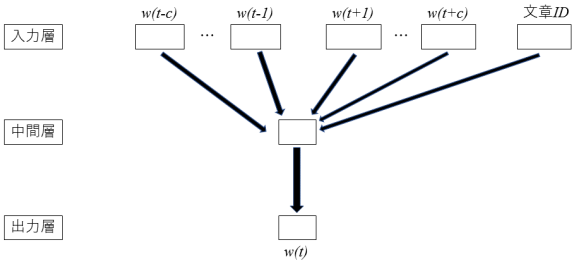

図 4　PV-DM モデル

精度が上がることを示した [2]．そのため高い精度で文書分類を行うためには，単語ベクトルの次元数を適切に調整する必要がある．また，単語ベクトルの次元数以外にも，学習の重みを決める学習率や，反復学習の回数，考慮しない単語を決める出現頻度の閾値など調整が必要なパラメータが存在する．これは Paragraph Vector でも同じことがいえる．

2.1.1　word2vec

word2vec では，Skip-gram モデルと CBOW モデルの 2 つのニューラルネットワークが提案されている [1], [2]．以下でこれら 2 つのモデルについて説明する．

Skip-gram モデル:　図 1 に Skip-gram モデルのニューラルネットワークを示す．このモデルは入力層，中間層，出力層の 3 層からなり，文章中のある単語 $w(t)$ を入力し，その前後の単語 $w(t-c), ..., w(t-1), w(t+1), ..., w(t+c)$ を出力とするニューラルネットワークである．c は同じ文脈として考慮する単語数であり，後述の window で定めるパラメータである．

CBOW モデル:　図 2 に CBOW モデルのニューラルネットワークを示す．このモデルは Skip-gram モデルと同じく，入力層，中間層，出力層の 3 層で構成されているが，入力と出力が Skip-gram モデルと逆になっている．前後の単語 $w(t-c), ..., w(t-1), w(t+1), ..., w(t+c)$ を入力することで中心の単語 $w(t)$ が出力されるニューラルネットワークである．Skip-gram モデルとは反対に，周辺の単語から中心にある単語を推定する問題を学習させている．

2.1.2　Paragraph Vector

Paragraph Vector は word2vec を拡張し，単語ではなく文章の特徴ベクトルを計算する手法である [3]．Paragraph Vector にも word2vec と同様に 2 つモデルがあり，それぞれ Skip-gram モデルを拡張した，Paragraph Vector with Distributed Bag of Words (PV-DBOW) モデルと，CBOW モデルを拡張した，Paragraph Vector with Distributed Memory (PV-DM) モデルである．以下でこれら 2 つのモデルについて説明する．

PV-DBOW モデル:　図 3 に PV-DBOW モデルのニューラルネットワークを示す．このモデルは word2vec の Skip-gram モデルを拡張したものである．まず最初に word2vec の Skip-gram モデルで学習したニューラルネットワークを作り，そのニューラルネットワークの入力層を文章の ID を入力するネットワークと入れ替える．文章の ID のみを word2vec と同じ手法で目的の文章に対して学習することで，文章の特徴ベクトルを得ることができる．

PV-DM モデル:　図 4 に PV-DM モデルのニューラルネットワークを示す．このモデルは word2vec の CBOW モデルを拡張したものである．まず最初に word2vec の CBOW モデルで学習したニューラルネットワークを作り，そのニューラルネットワークの入力層に文章の ID を入力するネットワークを追加する．追加されたネットワーク部分のみを word2vec と同じ手法で目的の文章に対して学習することで，文章の特徴ベクトルを得ることができる．

2.2　パラメータサーチ手法

高精度な文書分類を実現するためには，Paragraph Vector

のパラメータの調整をするためのパラメータサーチが必要である．主なパラメータサーチ手法としては，オープンソースの機械学習ライブラリである scikit-learn [9] が提供しているグリッドサーチとランダムサーチが存在する．以下ではその 2 つについて説明する．

グリッドサーチ: グリッドサーチは各パラメータの探索範囲を定め，全ての値の組み合わせで学習を行い，最も良いスコアを算出する組み合わせのパラメータの値を出力する．全ての組み合わせを試すため，探索範囲の中で最もその学習データに適したパラメータの値を求めることができる．しかし，そのため膨大な時間がかかる．

ランダムサーチ: ランダムサーチは各パラメータの探索範囲を定め，ランダムな組み合わせで学習を行い，最も良いスコアを算出する組み合わせのパラメータの値を出力する．ランダムサーチはユーザが回数を指定することができるため，回数を少なくすれば探索時間を短くすることができる．しかし，パラメータの探索範囲が広い場合や，ランダムサーチを行う回数が少ない場合は十分に探索できず，得られたパラメータの値が必ずしも適切とはいえない．

原ら [5] は Paragraph Vector のパラメータを効率的に探索する手法として，二分法とランダムサーチを組み合わせた手法を提案した．彼らはまた，Brown Corpus [7] の英文の分類実験において，ランダムサーチと比べて二分法が探索コストにおいて優れていることを示した．

3. 二分法によるパラメータサーチ

ここでは二分法によるパラメータサーチの効率化について説明する．まず 3.1 節で原らの手法について説明し，それから 3.2 節で提案手法を説明する．

3.1 原らの手法

Paragraph Vector の実装に用いられる doc2vec には調整が必要なパラメータが多数存在する．精度向上の観点から，それらのパラメータを適切に調整することは重要である．しかし，パラメータ調整は経験則で行われており，学習データが変わるとパラメータの再調整が必要である．そこで原らは二分法とランダムサーチを組み合わせた効率的なパラメータサーチを提案した [5]．原らは doc2vec の以下の 4 種類のパラメータに注目した．

- alpha：学習率
- iter：反復学習の回数
- min_count：学習対象に含む単語の最小出現回数
- window：文脈を考慮するために関連付ける前後の単語数 (図 3 や図 4 の c)

これら 4 種類のパラメータは doc2vec のパラメータの中で探索する範囲が広く，また生成されるベクトルに大きく影響を与える．一方，doc2vec には他にもいくつかパラメータが存在するが，それらのパラメータの値はデフォルト値とした．

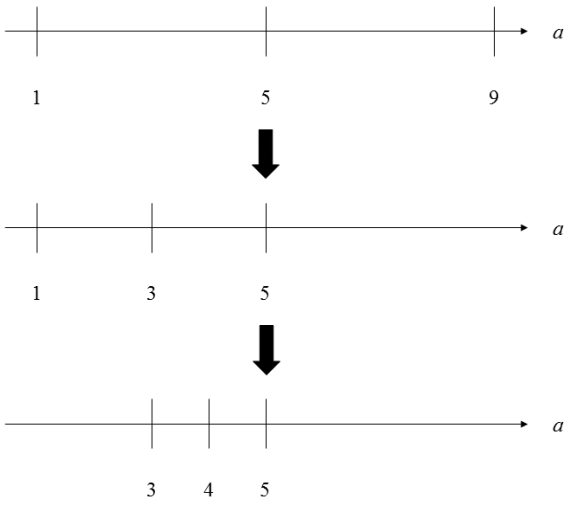

図 5 区間上限と区間下限とその中間点の推移例

原らの提案手法では，まずはじめに上記の 4 種類のパラメータの中から二分法で探索したいパラメータを 1 つ選択する．選択しなかった残り 3 種類のパラメータはランダムサーチで探索する．この際，学習に使うデータセットで 5 分割交差検定を行い，その検証データの正解率の平均を求め，それに基づいてパラメータの値を決める．この時，4 種類全てのパラメータの探索範囲となる区間上限と区間下限を決める．二分法で探索するパラメータは区間上限と区間下限の値で固定し，残りのパラメータは探索区間内の値でランダムサーチを複数回行う．この時のランダムサーチの回数はユーザが決める．そのランダムサーチの結果を用い，区間上限を用いた結果と区間下限を用いた結果を比べ，良い方のパラメータの値と，区間上限と区間下限の中間点の値を二分探索するパラメータの新たな区間上限または区間下限とする．この探索を複数回繰り返す．

図 5 は二分法で探索するパラメータを a とし，区間上限を 9，区間下限を 1 としたときの区間上限と区間下限とその中間点の推移の例である．この例ではパラメータ $a = 9$ として他のパラメータに対してランダムサーチを指定回数行う．例えば，ランダムサーチの回数を 4 回とし，以下の結果が得られたとする．

- $a = 9$，$b = 7$，$c = 2$，$d = 8$，正解率 $= 0.5$
- $a = 9$，$b = 3$，$c = 9$，$d = 2$，正解率 $= 0.4$
- $a = 9$，$b = 1$，$c = 6$，$d = 3$，正解率 $= 0.7$
- $a = 9$，$b = 9$，$c = 4$，$d = 5$，正解率 $= 0.6$

この場合，正解率が最も高い 0.7 がパラメータ a の値が 9 であるときの正解率となる．パラメータ a の値が 1 であるときの正解率も同様にして求め，それが 0.8 であったとする．$a = 9$ の正解率より $a = 1$ の正解率が高いので，正解率の低い区間上限を中間点の値で置き換える．すなわち図 5 にあるようにパラメータ a の探索範囲が 1 から 5 に更新される．以降，同様に新たな探索範囲の区間上限と区間下限の正解率を算出し探索範囲が狭くなるように更新していく．この繰り返し処理も初めにユーザが指定した回数行われる．

3.2 提案手法

本研究では原らの手法を拡張する．具体的には，二分法探索を多次元化する．さらに原らの研究では探索していなかった特徴ベクトルの次元を表す size も本研究では加える．

多次元化した提案手法について説明する．初めに二分法で探索するパラメータを任意の数選択する．このとき選んだパラメータが 1 種類の場合，原らと同じ二分探索となる．本研究では探索するパラメータの種類を 5 種類としたため，最大 5 次元の二分法で探索する．5 種類のパラメータの探索範囲である区間上限と区間下限をそれぞれ決める．二分法を適用するパラメータのそれぞれの区間上限と区間下限の組み合わせで 5 分割交差検定を行い正解率を算出する．原らの手法と同様に二分法を適用しないパラメータはランダムサーチを行う．二分法で探索するパラメータの組み合わせの総数は n 次元二分法で 2^n 通りである．それらの内最も正解率の高かったパラメータの組み合わせと，それぞれのパラメータの中間点で新たな区間上限と区間下限を定める．このような探索を探索範囲を狭めながら複数回繰り返す．

図 6 はパラメータ 2 つを 2 次元二分法で探索する例である．この 2 つのパラメータを以下では x，y とする．x，y のどちらも区間上限は 9，区間下限は 1 である．パラメータ x を 1，パラメータ y を 1 に固定し，他のパラメータをランダムに変えてランダムサーチを複数回行い正解率を算出する．最も正解率が高かったパラメータの組み合わせを $x = y = 1$ のときの正解率とする．またこれを以下の例のようにパラメータ x とパラメータ y の区間上限と区間下限でできる全ての組み合わせで行う．

- $x = 1$，$y = 1$，$z = 3$，$w = 4$，$v = 6$，正解率 $= 0.6$
- $x = 1$，$y = 9$，$z = 8$，$w = 9$，$v = 1$，正解率 $= 0.7$
- $x = 9$，$y = 1$，$z = 2$，$w = 7$，$v = 8$，正解率 $= 0.5$
- $x = 9$，$y = 9$，$z = 5$，$w = 6$，$v = 3$，正解率 $= 0.8$

この場合，正解率が最も高いのは $x = 9$，$y = 9$ のときである．最も良い正解率のパラメータの組み合わせとそれぞれの中間点 x_c，y_c で探索範囲を更新する．1 と 9 の中間の値は 5 なので，x と y のどちらも区間上限は 9，区間下限は 5 に更新される．以降も同様に新たな組み合わせで正解率を算出し，最も良かった組み合わせと x，y それぞれの中間点で探索範囲を更新する．この繰り返しをユーザの指定回数行い，最後に最も良かった正解率のパラメータの組み合わせを最適なパラメータの値として出力する．図 6 では，1 回目の探索が青，2 回目の探索をが橙，探索するパラメータの組み合わせを丸，中間点を四角で表している．

4. 評価実験

Stanford Sentiment Treebank Dataset [8] に含まれるレビュー文を Positive と Negative の 2 値に分類する評価実験を行う．

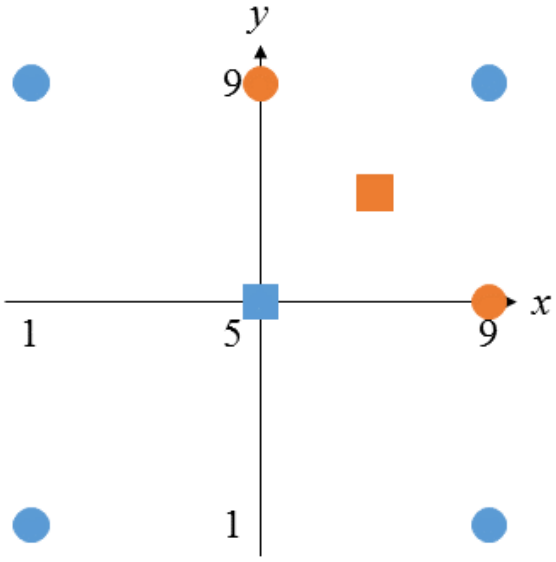

図 6　2 次元二分法の区間上限と区間下限とその中間点の推移例

表 1　Stanford Sentiment Treebank Dataset の評価ラベル数

評価ラベル	訓練データ
Very Negative	1,092
Negative	2,218
Neutral	1,624
Positive	2,322
Very Positive	1,288
all	8,544

4.1 データセット

このデータセットは，Amazon Mechanical Turk(注1) を利用して，映画レビュー文に評価ラベルを付与した構文木コーパスである．また Amazon Mechanical Turk とは，ソフトウェアが実行するよりも，人間が行う方が効率的であると思われる作業を Web を通して代行を依頼するサービスである．Stanford Sentiment Treebank Dataset は，あらかじめ，訓練データ 8,544 件とテストデータ 2,210 件に分割されており，各レビュー文に Very Negative，Negative，Neutral，Positive，Very Positive の 5 種類の評価ラベルが付与されている．表 1 に実験に用いる訓練データのラベル毎の内訳を示す．

以下は，Stanford Sentiment Treebank Dataset の 1 文である．Very Negative から Very Positive に対応する 0 から 4 までの数字で評価を表しており，単語や熟語毎に評価ラベルが付けられている．この例文は，先頭の 3 から Positive のラベルの付いた文であることが分かる．

```
(3 (2 Yet) (3 (2 (2 the) (2 act)) (3 (4 (3 (2 is) (3
(2 still) (4 charming))) (2 here)) (2 .))))
```

また学習の際，上記の 1 文を 2 単語以上のラベル毎に区切った断片を利用している．以下に先ほどの例文を断片にしたものを示す．

```
[['Yet', 'the', 'act', 'is', 'still', 'charming', 'h
```

(注1)：https://www.mturk.com/mturk/welcome

表 2　探索するパラメーター一覧

パラメータ	探索するパラメータの値
size	300，325，350，375，400，425，450，475，500
alpha	0.005，0.066875，0.12875，0.190625，0.2525，0.314375，0.37625，0.438125，0.5
iter	1，2，3，4，5，6，7，8，9
min_count	1，2，3，4，5，6，7，8，9
window	1，2，3，4，5，6，7，8，9

```
ere', '.'], ['the', 'act', 'is', 'still', 'charming', 'here', '.'], ['the', 'act'], ['is', 'still', 'charming', 'here', '.'], ['is', 'still', 'charming', 'here'], ['is', 'still', 'charming'], ['still', 'charming']]
```

この例では 1 文が 7 つの断片になっている．訓練データの全件である 8,544 件を断片にした場合，断片数は 126,632 件である．

Stanford Sentiment Treebank Dataset の Very Negative，Negative，Neutral，Positive，Very Positive の 5 種類の評価ラベルで表された問題は以下の手順で Positive と Negative の 2 値分類問題に変換する．まず Neutral となっている断片は除外する．Very Negative と Negative を Negative にまとめ，Positive と Very Positive を Positive にまとめる．こうすることで 5 値分類の問題を Positive と Negative の 2 値分類問題に変換する．

4.2　レビュー文の分類実験

Stanford Sentiment Treebank Dataset を Paragraph Vector を用いて計算した特徴ベクトルをロジスティック回帰で分類し，その正解率を算出する．使用した Paragraph Vector の実装には gensim [6] ライブラリの doc2vec，グリッドサーチ，ランダムサーチ及び，ロジスティック回帰の実装には scikit-learn [9] ライブラリを使用する．なお，ロジスティック回帰のパラメータはデフォルト値とする．

実験は提案手法，原らの二分法探索手法 [5]，ランダムサーチ，グリッドサーチで行う．使用するデータとしては，前述のレビュー文の断片データの先頭 4,000 件とデータセット全件を利用する．提案手法の二分法の繰り返し回数は 3 回とする．5 次元探索のため，探索するパラメータの組み合わせは 32 通りを 3 回と最後の中間点を含めた 97 から，探索範囲の更新の際に，前の範囲の最も良い組み合わせが残ることから 2 を引いた 95 通りである．4 次元探索の場合，探索するパラメータの組み合わせは 16 通りなので，この時のランダムサーチの回数を 2 回にすることで，同様に 32 通りを 3 回から 2 を引いた 95 回とする．3 次元探索も同様に 8 通りの組み合わせのランダムサーチを 4 回行うことで，32 通りを 3 回から 2 を引いた 95 回とした．ランダムサーチのみの探索回数は提案手法の探索回数と同等にするため 95 回とする．原らの二分法探索の各ランダムサーチ回数は 10 回，二分法を行う回数は 3 回とする．すなわち二分探索するパラメータの上限，下限，中間点で各 10 回，計 30 回のランダムサーチを 3 回繰り返すため，90 通りの組み合わせを探索する．原らの手法では二分法で求めるパラメータは先頭 4,000 件の場合は 5 種類から 1 種類，全件の場合は size，window のいずれか 1 種類とする．残りのパラメータ 4 種類はランダムサーチを行う．探索するパラメータは表 2 である．二分探索するパラメータは最大値と最小値を上限と下限とし，ランダムサーチの場合は，この表からランダムな値を使用する．表 3 及び，表 4 に先頭 4,000 件と全件での正解率と実行時間をそれぞれ示す．

表 3　先頭断片 4,000 件での正解率と実行時間

手法	正解率	実行時間 [min]
5 次元二分法	0.623	13.1
1 次元二分法 (size)	0.612	14.0
1 次元二分法 (alpha)	0.614	13.3
1 次元二分法 (iter)	0.623	13.5
1 次元二分法 (min_count)	0.619	13.9
1 次元二分法 (window)	0.622	14.2
ランダムサーチ	0.619	15.3
グリッドサーチ	0.644	13,102.5

表 4　全件断片 126,632 件での正解率と実行時間

手法	正解率	実行時間 [min]
5 次元二分法	0.718	407.7
1 次元二分法 (size)	0.732	408.2
1 次元二分法 (alpha)	0.727	462.9
1 次元二分法 (iter)	0.733	389.2
1 次元二分法 (min_count)	0.724	390.5
1 次元二分法 (window)	0.723	416.6
ランダムサーチ	0.737	439.8

表 5　先頭断片 4,000 件での各 4 次元二分法の正解率と実行時間

手法	正解率	実行時間 [min]
4 次元二分法 (size 以外)	0.623	15.8
4 次元二分法 (alpha 以外)	0.625	15.7
4 次元二分法 (iter 以外)	0.621	16.2
4 次元二分法 (min_count 以外)	0.623	15.3
4 次元二分法 (window 以外)	0.623	15.1

表 3 より，先頭 4,000 件での提案手法の正解率は，ランダムサーチ，原らの手法より低くはなかった．また，実行時間は提案手法が最も短かったが，どちらも大きな差ではなかった．表 4 の全件データを利用した実験では，提案手法の正解率が最も低く，また実行時間は原らの手法と変わらない結果となった．5 次元二分法はこれらの実験から先頭 4,000 件では原らの手法と同程度，全件では正解率が劣ることが分かった．

4.3　二分法が有効なパラメータ

1 つのパラメータについてランダムサーチを行い，残りのパラメータについて 4 次元二分探索をする実験も行った．4 次元二分法の探索回数も，先述の通り 5 次元二分探索に合わせた 95 回とする．その正解率と実行時間を表 5 に示す．

表 5 では，alpha をランダムサーチにした場合，正解率が他と比べて少し高かったものの，大きな差はない．表 3 の 1 次元二分法の正解率を比べると size と alpha が低く，iter と 1 ポイ

表 6　先頭断片 4,000 件での 3 次元二分法の正解率と実行時間

手法	正解率	実行時間 [min]
3 次元二分法 (size，alpha 以外)	0.627	15.4
3 次元二分法 (size，iter 以外)	0.624	15.7
3 次元二分法 (size，min_count 以外)	0.623	16.2
3 次元二分法 (size，window 以外)	0.620	15.3
3 次元二分法 (alpha，iter 以外)	0.619	16.3
3 次元二分法 (alpha，min_count 以外)	0.621	14.9
3 次元二分法 (alpha，window 以外)	0.620	16.0
3 次元二分法 (iter，min_count 以外)	0.621	15.2
3 次元二分法 (iter，window 以外)	0.623	13.5
3 次元二分法 (min_count，window 以外)	0.623	14.3

ントの差があった．そこで，2 つのパラメータについてランダムサーチ，残りのパラメータを二分探索にした 3 次元二分法でパラメータ探索を行った．3 次元二分法の探索回数も 5 次元二分探索と同様に 95 回とした．その結果を表 6 に示す．表 6 から良くなる傾向がみられるが，さらなる実験が必要である．

5. ま と め

本研究では，Paragraph Vector のためのパラメータサーチの効率化手法として，原らの 1 次元二分法を拡張した多次元二分法を提案した．多次元二分法は複数のパラメータについて，二分法により探索する手法である．

評価実験として Stanford Sentiment Treebank Dataset の分類実験を行い，ランダムサーチと原らの提案した 1 次元二分法と比較した．訓練データのレビュー文の断片 4,000 件を使った 5 分割交差検定の実験では，正解率と実行時間ともに他の手法よりよかったが，大きな差ではなかった．訓練データの断片全件 (126,632 件) の実験では，実行時間は同程度，正解率は劣る結果となった．また，3 次元二分法や 4 次元二分法では，1 次元二分法と同程度の有効性を確認した．

今後の課題としては，この訓練データ全件の二分法を用いた実験による提案手法の評価，パラメータの探索範囲を広げた評価実験，Paragraph Vector 以外のニューラルネットワークでの評価実験を行うことなどがあげられる．

文　献

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J.: Distributed representations of words and phrases and their compositionality, *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[2] Mikolov, T., Chen, K., Corrado, G., and Dean, J.: Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781*, pp. 1–12, 2013.

[3] Le, Q. and Mikolov, T.: Distributed Representations of Sentences and Documents, *CoRR*, abs/1405.4053, pp. 1–9, 2014.

[4] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manningand, C. D., Ng, A. Y., and Potts, C.: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, Association for Computational Linguistics, pp. 1631–1642, 2013.

[5] 原裕貴，新妻弘崇，太田学: Paragraph Vector のための効率的なパラメータサーチの検討，DEIM Forum 2017，B6-1，2017.

[6] Řehůřek, R. and Sojka, P.: Software Framework for Topic Modelling with Large Corpora, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50, 2010.

[7] Francis, W. N., and Kucera, H.: Brown Corpus Manual, Brown University, 1979.

[8] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C.: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, *Conference on Empirical Methods in Natural Language Processing*, pp. 1–12, 2013.

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É.: Scikit-learn: Machine Learning in Python, *The Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.