

卒業論文

計算問題の特徴分布に基づく類題選出による
自己学習支援

Self Learning Support by Automatic Selection of
Calculation Exercises based on Feature Distribution of
Exercises

成蹊大学理工学部情報科学科

S152114 宮地 雄也

要旨

本研究では自然言語処理の技術を人工言語の数式に適応し，分類ができるかどうかを目的とした．またその評価指標として累代選出も行い実際まちがえているのかも検討した．分散表現で文字の特徴量を抽出したのちその特徴量を用いて数式の分散表現化も行い，式のベクトルを得る．この結果から現在では間違った問題から人をグループ化していたシステムから問題をグループ化してその問題を間違えた人の最適化学習支援を行い，より繊細で効果的な学習が期待できる．この研究の先には大量の個人情報を扱うアダプティブラーニングにおいて問題から分類できるようになれば，学習データが少なくても十分に効果を発揮することが期待できる．

目次

第 1 章	序論	1
第 2 章	分散表現	3
2.1	Continuous Bag-of-Words Model	4
2.2	Skip Gram	5
第 3 章	LSTM(Long short-term memory)	6
3.1	forget ゲート	6
3.2	output ゲート	7
3.3	hidden ゲート	7
3.4	input ゲート	7
3.5	LSTM のまとめ	8
第 4 章	Exercises Vector System	9
4.1	システム全体の流れ	9
4.2	計算式の特徴量抽出	9
4.3	正誤判定機	13
第 5 章	結果とその検討	14
5.1	実験	14
5.2	実験結果	15
5.3	考察	16
第 6 章	関連研究	18
第 7 章	結論と今後の課題	19
	参考文献	21

第 1 章

序論

昨今、小・中学生の理系離れが問題視されている。平成 30 年度全国学力・学習状況調査（全国学力テスト）の結果では平均正答率は小学校では算数 B が 51.7%，中学校数学では 47.6% とどちらも最も低く、ついで国語，理科の順で正答率が低い。小中どちらとも理系教科の習熟度が低いことを示している。この要因の一つに，数学は一つの計算方法が様々な分野に横断していくことが一度，苦手を生んでしまったらそこからの分野の理解度が下がり，次の分野での応用がきかないために連鎖的に苦手が蓄積することが原因ではないかと考えた。各単元のちょっとした積み残しが，後々，尾を引いていることが全国学力テストの結果から見て取れる。この状況を打破するには子供一人一人の苦手と向き合い，苦手と感じる前に理解していくしかない。しかしながら，生徒と向き合うべき教師の労働時間は過酷を極めており，ベネッセ教育総合研究所の調査では小中高の教員の指導時間は増加の一途を辿っていることを明らかにした。表 1.1 は文献 [2] での調査の結果の抜粋である。

表 1.1 出勤時刻・退勤時刻・学校にいる時間（平均時間、経年比較（教員年齢別〔公立全体〕））

	調査年	25 歳以上	26～30 歳	31～40 歳	41～50 歳	51～60 歳
出勤時間	2010	7:44	7:43	7:44	7:42	7:42
	2016	7:44	7:43	7:44	7:42	7:42
退勤時間	2010	19:30	19:40	19:10	18:57	18:31
	2016	20:00	19:54	19:26	19:05	18:46
学校にいる時間	2010	11 時間 46 分	11 時間 57 分	11 時間 26 分	11 時間 15 分	10 時間 49 分
	2016	12 時間 26 分	12 時間 18 分	11 時間 46 分	11 時間 26 分	11 時間 06 分

表 1.1 によると，教員の労働時間は 2010 年に比べて 2016 年の方が各年次とも増加しており，教員のやることが増えている一方で，主であるはずの教材研究や教務準備に時間が避けていないことを示している。この状況では先生が生徒一人一人に時間をさき，指導することは難しい現状がみついている。

この打開策として，IT 技術駆使した個人別最適化学習に注目が集まっている，しかし，教育の情報は，生徒の情報と結びついている個人情報なためオープン化できず，現在でているサービスでは各サービス利用者の利用状況からデータを取得し，その運用に利用しているため一部の大手企業が情報を独占している。そこで個人の統計データではなく，解く数式の方に着目し，計算式自体の特徴を抽出し，間違えた問題と同様の特徴を持つ問題が復習する類題として最適なのではないかという仮定のもと，本論文では数式の特徴を掴むために自然言語処理の分野で使用される分散表現を

適用し，さらに再帰ニューラルネットワークを用いて数式ベクトルを作り出すことを目標とし，そのベクトルを用いて実際に復習問題生成を行った．

第 2 章

分散表現

自然言語処理ではコンピュータで演算するために各単語を判別するために単語一つ一つを onehot ベクトルというものに置き換える．onehot ベクトルとはある語彙数 V の文章の中の単語 w_i がある時，次元数 V のベクトルに対し i 番目の要素のみ 1 で残りが全て 0 になっているベクトルのことである (式 2.1)．

$$\mathbf{A} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad (i = 3 \text{ の時}) \quad (2.1)$$

これにより単語一つ一つを別々のベクトルとして区別して表記することができる．しかし onehot ベクトルには問題点があり，一つ目として，ある単語の語彙数 V が増加すると比例して onehot ベクトルの次元 V も大きくなり，処理に時間がかかる点がある．また二つ目として onehot ベクトルでは情報が 0 か 1 しかないので疎なベクトルができる．疎なベクトルとは次元数が大きくてもそのベクトルの持つ意味が薄いベクトルをさす，このような問題を解決しようと考えられたのが分散表現である．

分散表現とは疎な onehot ベクトルを密な密な実数値をの集合をベクトルとして扱い，その実数値で単語の意味を表そうとする手法である．これにより疎なベクトルであった単語のベクトルを密な表現ができ，また，密度が上がることでベクトルの次元数を削減することができる．以下，3 手法はその分散表現を得るにあたって機械学習を応用した推論をベースとして編み出された手法である．

2.1 Continuous Bag-of-Words Model

Continuous Bag-of-Words Model(以降, CBOW と略記) は [1] で提唱された手法で, ある単語数 v の単語列 $w_0, w_1, w_2, \dots, w_{v-2}, w_{v-1}, w_v$ がある時, 単語 w_t が文脈中で前後 m 単語をコンテキスト C とする時の前後 m 単語が共に共起する事前確率が増大するように学習するモデルである(図 2.2).

図 2.1 より w_t をターゲットにした時の周辺単語 m 個を対象としたときの事後確率は式 2.2 のように書くことができる. CBOW モデルでは式 2.2 の最大化問題と見ることができる. そこで CBOW の損失関数は, 単に式 2.2 の確率に対数を取りマイナスをつけ, 最小化問題として解く. 式 2.3 はコーパス全体に拡張した形を示す.

$$[tb] \quad P(w_t | w_{t-m}, \dots, w_{t+m}) \quad (2.2)$$

$$L = -\frac{1}{T} \sum_{t=0}^T \log P(w_t | w_{t-m}, \dots, w_{t+m}) \quad (2.3)$$

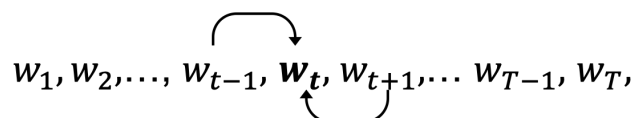


図 2.1 単語の列からターゲットとなる単語を推測する ($m = 1$)

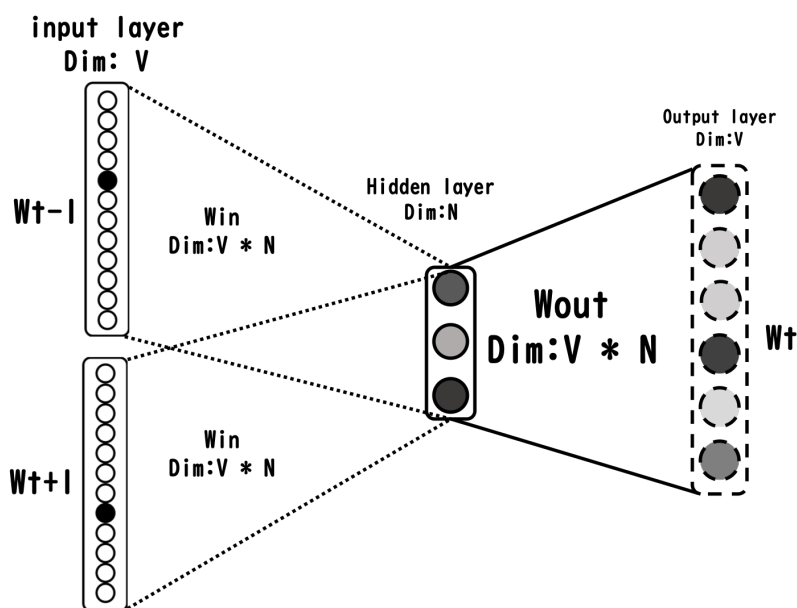


図 2.2 CBOW のネットワーク構成モデル模式図 ($m = 1$)

2.2 Skip Gram

Skip Gram は文献 [1] で紹介されている章 2.1 で述べた CBOW とは別の手法である。CBOW とは逆に中心の単語 w_t から m 個の周辺単語を推測するモデルである。ネットワーク構成は図 2.4 のようになる。

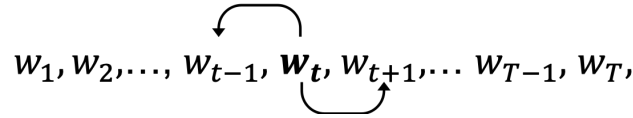


図 2.3 単語の列からターゲットとなる単語を推測する ($m = 1$)

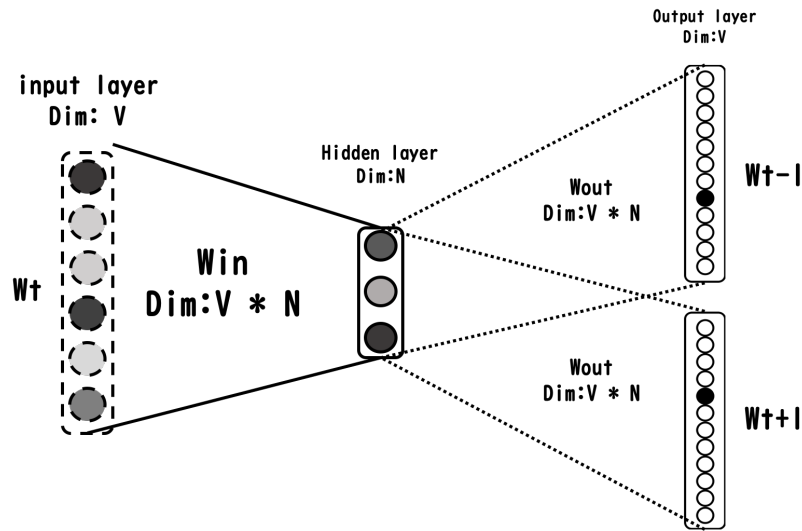


図 2.4 SkipGram のネットワーク構成モデル模式図 ($m = 1$)

ある単語 w_t が入力層に入力され、周辺単語の w_{t-m}, \dots, w_{t+m} を推測する時、その全てが同時に起こる確率は式 2.4 となる。

$$P(w_{t-m}, \dots, w_{t+m} | w_t) \quad (2.4)$$

ここで SkipGram モデルは w_{t-m}, \dots, w_{t+m} のそれぞれのあいだに関係性がないと仮定し、交差エントロピーを用いて損失関数 (式 2.5) を定義する。

$$\begin{aligned} \log P(w_{t-m}, \dots, w_{t+m} | w_t) &= -\log \prod_{k=0}^m P(w_{t-k} | w_t) \\ &= -\sum_{k=0}^m \log P(w_{t-k} | w_t) \end{aligned} \quad (2.5)$$

そして式 2.5 をコンテキスト C をコーパス全体に拡張すると式 2.6 となり、この式を学習によって最小化していく。

$$\begin{aligned} \sum_C \log P(w_{t-m}, \dots, w_{t+m} | w_t) &= \sum_C -\log \prod_{k=0}^m P(w_{t-k} | w_t) \\ &= -\sum_C \sum_{k=0}^m \log P(w_{t-k} | w_t) \end{aligned} \quad (2.6)$$

第 3 章

LSTM(Long short-term memory)

順伝播ニューラルネットワークでは前の情報はつかわないため文章や音声など、一つ前の情報に影響を受けるデータに対しては有用ではない。そこで、ある時刻 t の出力の際、過去の情報も扱う再起ニューラルネットワーク（以降 RNN と省略）というものが発案された。これは時刻 t の入力を x_t とする時、各ネットワークは過去の情報を記憶しているようにみなすことができる。しかしながら通常の RNN の場合、時間が経つほどに前の情報は薄れていく勾配消失が起こることがあった。これを解決しようとしたのが記憶情報ごとにメモリーの役割を果たすネットワークを分けた Long short-term memory（以降, LSTM と省略）というモデルである。LSTM では前の時刻 $t-1$ の C_{t-1} , h_{t-1} , x_t が入力ベクトルとして受け取る。 C_{t-1} は LSTM で導入された時刻 0 から $t-1$ までの情報を記録しているベクトル, h_{t-1} は一時刻前の出力ベクトル, x_t は時刻 t の入力ベクトルである。それぞれのセルを計算するために 4 つのゲートを導入する。それぞれのゲートは入力セルの情報をどれくらい加味するかを重み付けするゲートである。様々なバージョンがある中でもっとも基本的なネットワーク構成を模式図を図 3.1 に示す。以下各ゲートに対して説明する。

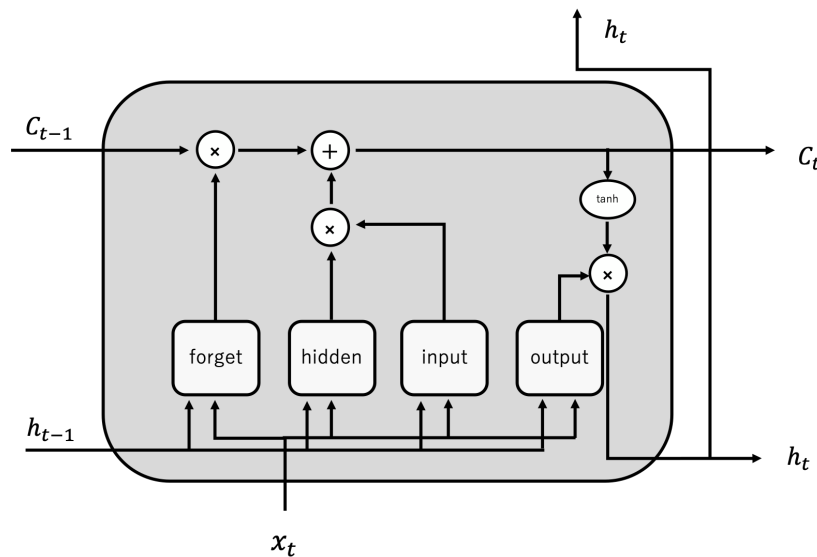


図 3.1 LSTM のネットワーク構成モデル模式図：図中の forget,hidden,input output はそれぞれゲートが扱う情報を示している

3.1 forget ゲート

forget ゲートは過去の情報が詰まっている C セルをどの程度次の今の時刻に使うかを定める重みである。その重みは入力と一時刻前の隠れ層の出力が使われており、その式を 3.1 に示す

$$f = \text{sigmoid}(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)}) \quad (3.1)$$

これにより時刻 t の C_t は 3.2 と求められる.

$$C_t = f \odot C_{t-1} \quad (3.2)$$

3.2 output ゲート

output ゲートでは次の時刻にどれくらい現在の情報を渡すかを定める重みである. その重みは入力と一時刻前の隠れ層の出力が使われており, その式を 3.3 に示す

$$o = \text{sigmoid}(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)}) \quad (3.3)$$

これにより時刻 t の C_t は 3.4 と求められる.

$$C_t = o \odot \tanh(C_t) \quad (3.4)$$

3.3 hidden ゲート

hidden ゲートでは forget ゲートで差し引かれた記憶セル C に新たな情報を加えるゲートである.

その情報は入力と一時刻前の隠れ層の出力を用いて算出する. (式 3.5)

$$\bar{h} = \tanh(x_t W_x^{(h)} + h_{t-1} W_h^{(h)} + b^{(h)}) \quad (3.5)$$

この \bar{h} が一時刻前の C_{t-1} に加算されることで新たな記憶を追加する.

3.4 input ゲート

input ゲートでは hidden ゲートで追加しようとした新たな記憶をどのくらい加算するかを決める重みを求めるゲートである. その重みは入力と一時刻前の隠れ層の出力が使われており, その式を 3.6 に示す

$$i = \text{sigmoid}(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)}) \quad (3.6)$$

これにより時刻 t の C_t は 3.7 と求められる.

$$C_t = \text{sigmoid}(C_t + i) \quad (3.7)$$

3.5 LSTM のまとめ

章 3.1, 章 3.2, 章 3.4, 章 3.5 で紹介したものをまとめると 3.8 のようになる.

$$\begin{aligned}
 f &= \text{sigmoid}(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)}) \\
 o &= \text{sigmoid}(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)}) \\
 \bar{h} &= \tanh(x_t W_x^{(h)} + h_{t-1} W_h^{(h)} + b^{(h)}) \\
 i &= \text{sigmoid}(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)})
 \end{aligned} \tag{3.8}$$

$$c_t = f \odot c_{t-1} + g \odot i \tag{3.9}$$

$$h_t = o \odot \tanh(c_t) \tag{3.10}$$

LSTM モデルを 3 時刻分展開したものを図 3.2 に示す.

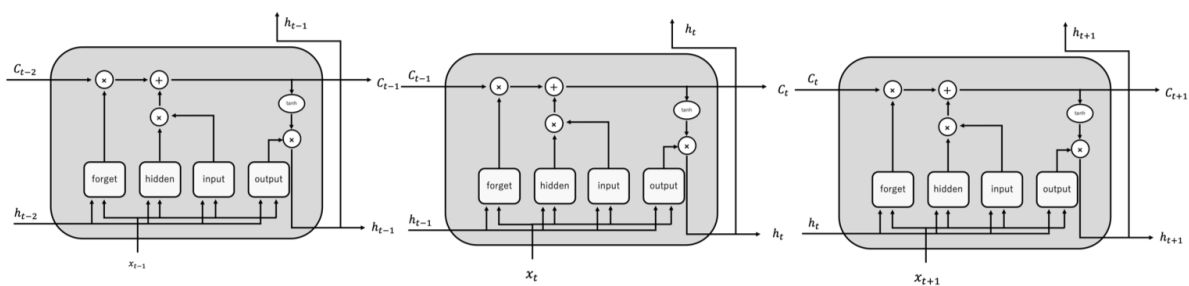


図 3.2 LSTM の 3 時刻展開:記憶を守る機構によりより高い濃度で次の時刻に情報を渡すことができる

第 4 章

Exercises Vector System

4.1 システム全体の流れ

数式の特徴を取り出し S 次元のベクトルに変換し、その分布から類題選出を行うシステムを作成し概要を図 4.1 に示す、以下、各システムの箇所ごとに説明する。

Exercises Vector System 概要

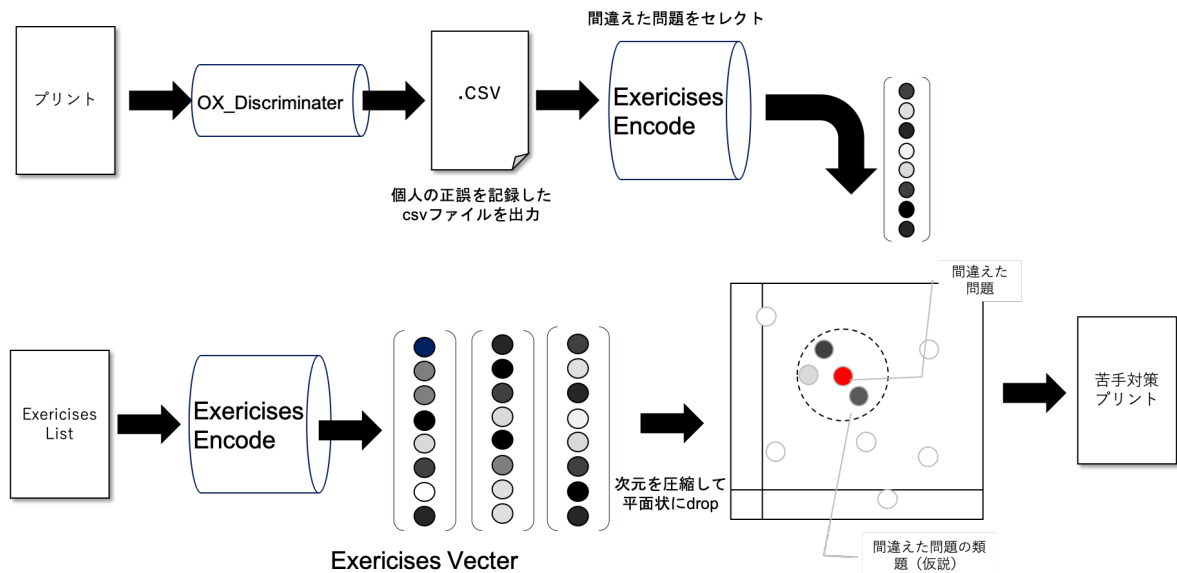


図 4.1 LSTM の 3 時刻展開:記憶を守る機構によりより高い濃度で次の時刻に情報を渡すことができる

4.2 計算式の特徴量抽出

4.2.1 概要

数式を分布化する際、そのベクトルの中に数式の特徴を入れ込んだベクトルを生成する手法が確立していない。そこで本論文では数式の各文字、記号を単語のようにみなし、onehot ベクトルを作成し、それを埋め込み層で特徴を踏まえた低次元ベクトルに変換したのち、系列変換モデルで読み込むことで低次元で数式の特徴を掴んだベクトルを生成できないかと考えた。

この考えを実現するために数式は我々が目にする $2x + 3 = 5$, $\frac{3x-1}{2} + 4 = \frac{2}{5}$ ではなく、

テキスト化かつその特徴を強く受けた形に変換する必要がある．そこで本論文では数式をある一定のルールの中でテキスト化されている \TeX 形式の数式を用いる．上記の計算式なら `2x+3=5,\frac{3x-1}{2}+4=\frac{2}{5}` とし，このテキストデータを用いて文字単位の埋め込んだベクトルを作成する．

実験を行った手法は以下の 2 種法で行い、それぞれ分布を python を用いて確認した.

- CBOW
- SkipGram

onehot ベクトルの置き方は以下の 2 手法を試し、それぞれ PCA と T-SNE で 2 次元に落とし plot し状況を確認する予備実験を行った。

- 出てきた数字, 数式の塊を onehot を置く方法 ($[0, 1, 2, \dots 3.6, 0.11, \dots = .+, -]$)
- 3 桁までの数字, 数式に現れる記号

4.2.2 文字分布の予備実験結果

予備実験として CBOW, SkipGram で文字の特徴が得られているかを確認した．以下その結果である．図 4.2 や図 4.3 を見ると小数は小数，整数は整数でまとまっているように見える．しかし，文字の分散表現と，その集合である式の分散表現の精度が一致することはまだわからない．また，出現文字を全て別の 1-of-k 表現した場合，一つ一つの情報が疎になってうまく学習が進まないことがわかった．

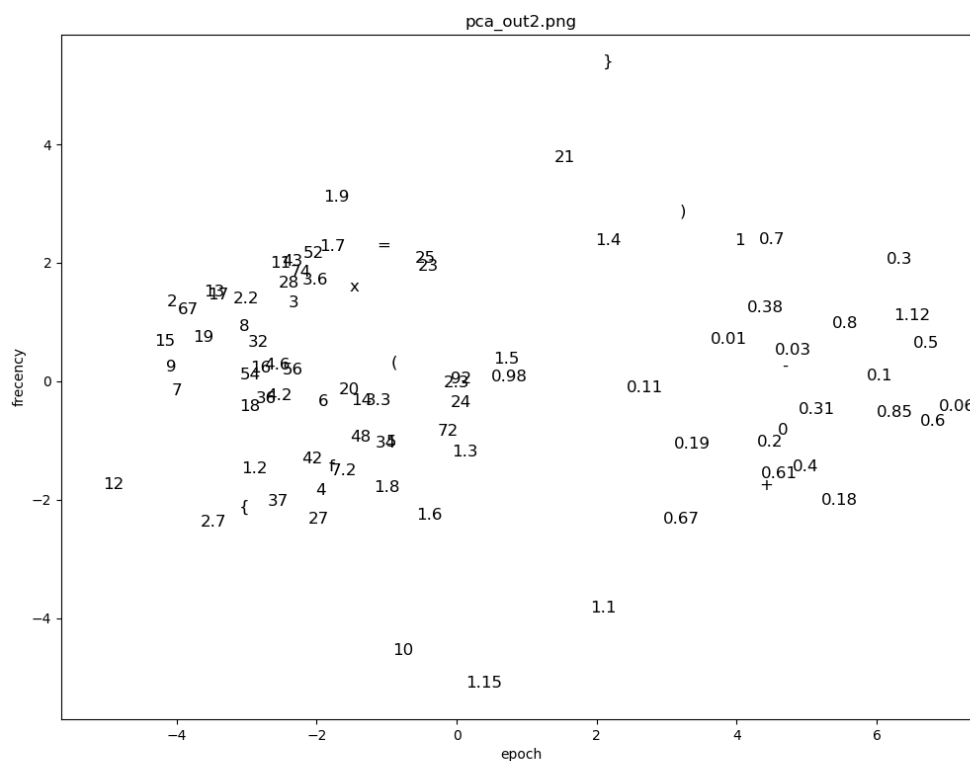


図 4.2 CBOW 89 次元を 2 次元に分散表現を圧縮したのち pca で再構成しグラフ化した結果

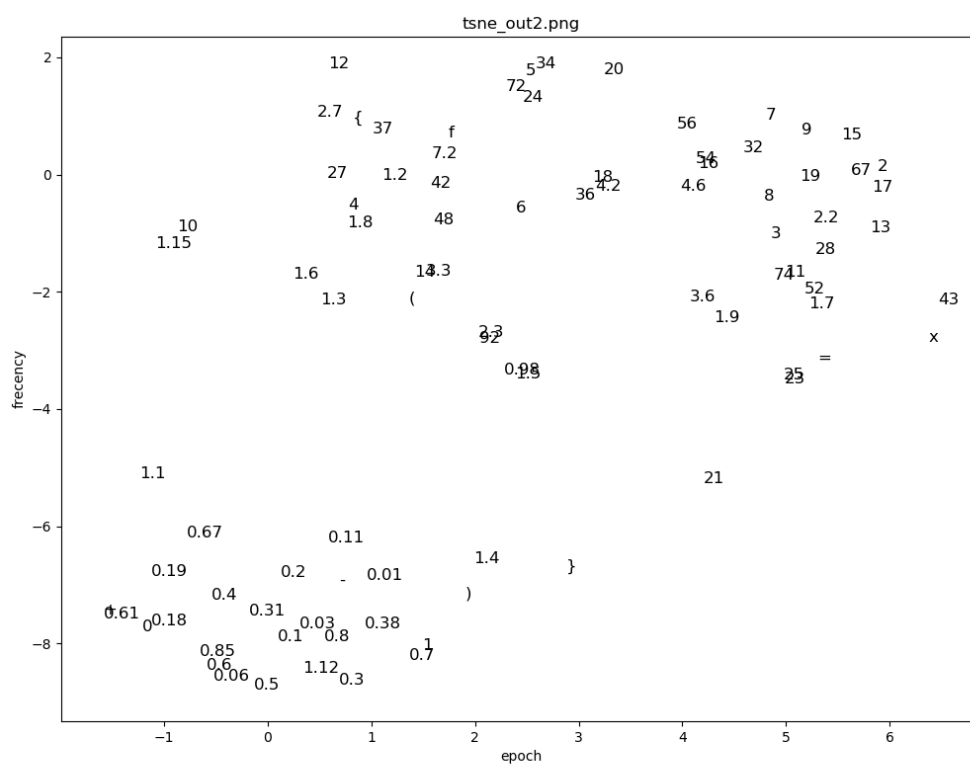


図 4.3 CBOW 89 次元を 2 次元に分散表現を圧縮したのち T-SNE で再構成しグラフ化した結果

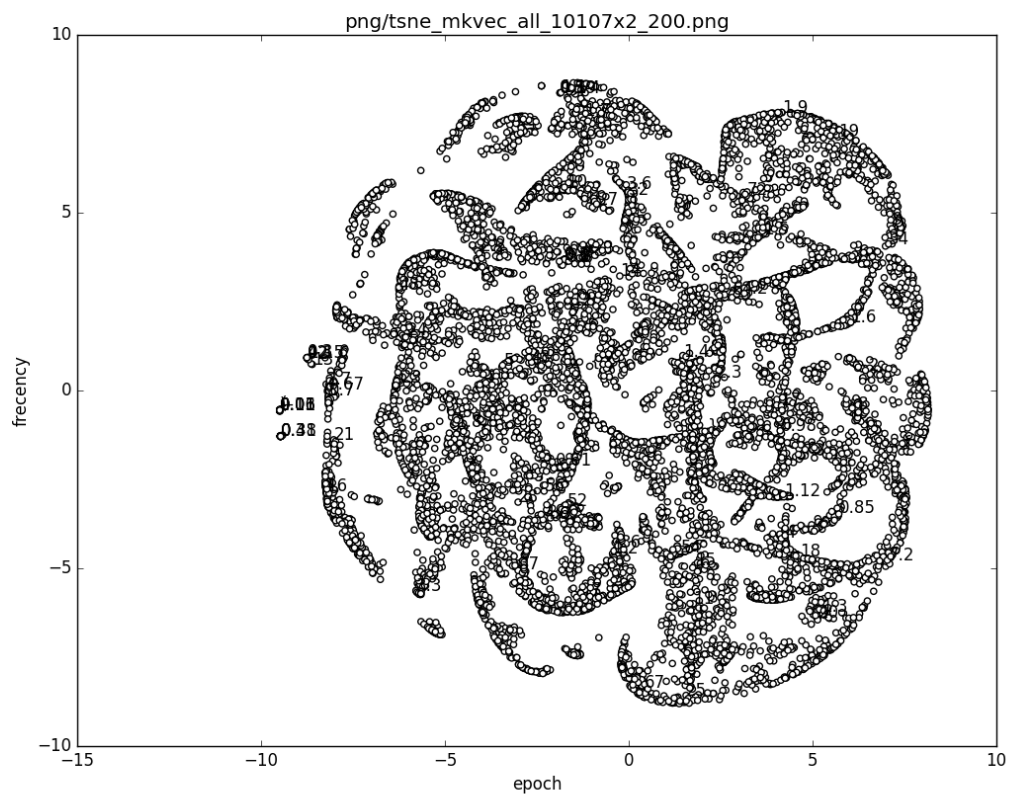


図 4.4 SkipGram 10107 次元を 2 次元に分散表現を圧縮したのち T-SNE で再構成しグラフ化した結果

4.3 正誤判定機

4.3.1 概要

プリントから回答者の問題の正誤を判定し、記録していくのが OX Discriminaer である。正誤を判定するには画像認識のニューラルネットを利用している。回答用紙の答えのところを切り取り、各問題に対して正解か不正解かを確認し、別途用意した、個人のプリント進行表 csv ファイルにプリント番号とそのプリントの問題ごとを正解なら o, 不正解なら x としてファイルに書き込んでいく。(図 4.5)

OX Discriminater 仕組み

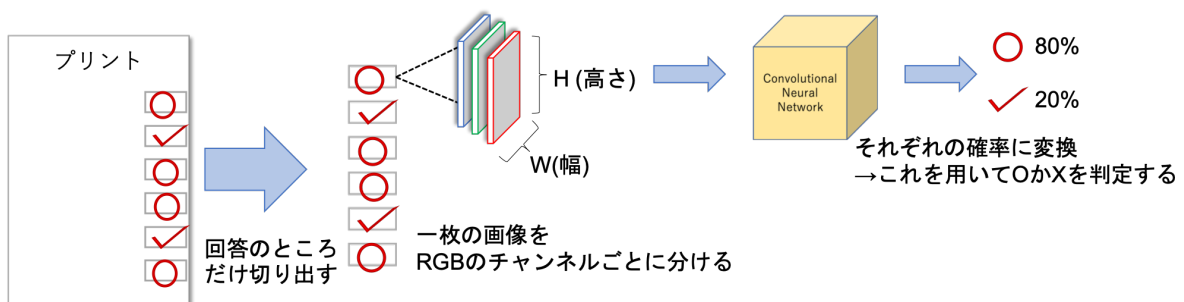


図 4.5 OX Discriminater 仕組み

第 5 章

結果とその検討

5.1 実験

以下の目的, 方法, を元の実験を行った.

計算式の特徴量抽出

- 目的. 数式のベクトル表現は可能なかどうか
- 方法. 埋め込み層, ネットワーク構成を変えながら Encoder-Decoder で復元を試みるその Encoder の出力値を数式ベクトルとしてみなし, pca,T-sne で二次元ベクトルに圧縮し図示する

実験で試したネットワーク構成は以下の通りである.

表 5.1 Encorder の構成

1	Embedding 層	入力ノード数:89, 出力ノード数:2
2	LSTM 層	入力ノード数:2, 出力ノード数:50
3	LSTM 層	入力ノード数:50, 出力ノード数:50

表 5.2 Encorder の構成

1	Embedding 層	入力ノード数:89, 出力ノード数:2
2	LSTM 層	入力ノード数:2, 出力ノード数:50
3	LSTM 層	入力ノード数:50, 出力ノード数:50
4	LSTM 層	入力ノード数:50, 出力ノード数:50
4	Affine 層	入力ノード数:50, 出力ノード数:89

5.2 実験結果

以下ネットワーク 5.1,5.2 で学習データ 10 個を学習し, Encoder の出力の式の分散表現と思われるものの分布である. 学習データは以下の 10 個である.

- $x + 5 = 8$
- $5x - 7 = 4$
- $3.6 + x = -1.4$
- $3x - 4 = -5x + 5$
- $\frac{7}{2}x + 9 = 6$
- $3(2x - 1) = 9$
- $\frac{3}{7}x + 2 = \frac{6}{7}$
- $-\frac{4}{7}x + 3 = -\frac{1}{7}x$
- $\frac{x+1}{3} + \frac{2x+1}{2} = \frac{3x-4}{2}$
- $-4x + 9 = 1$

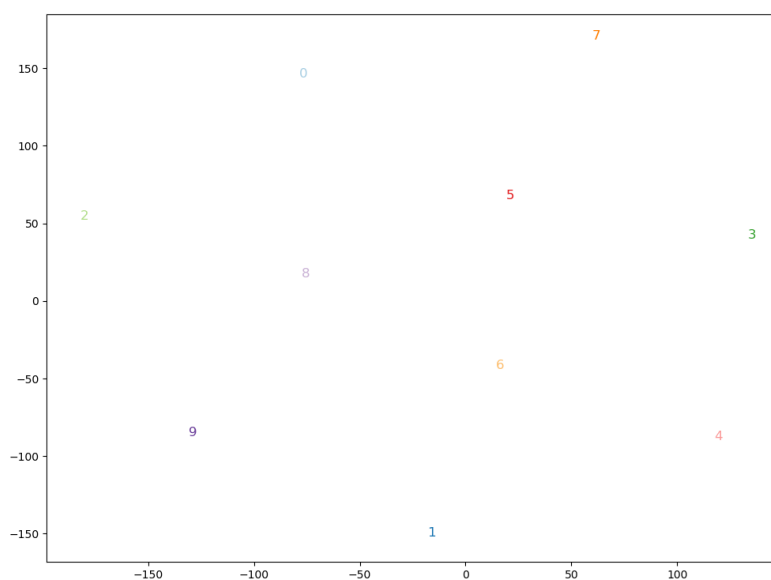


図 5.1 式ベクトルの分布 (cbow)

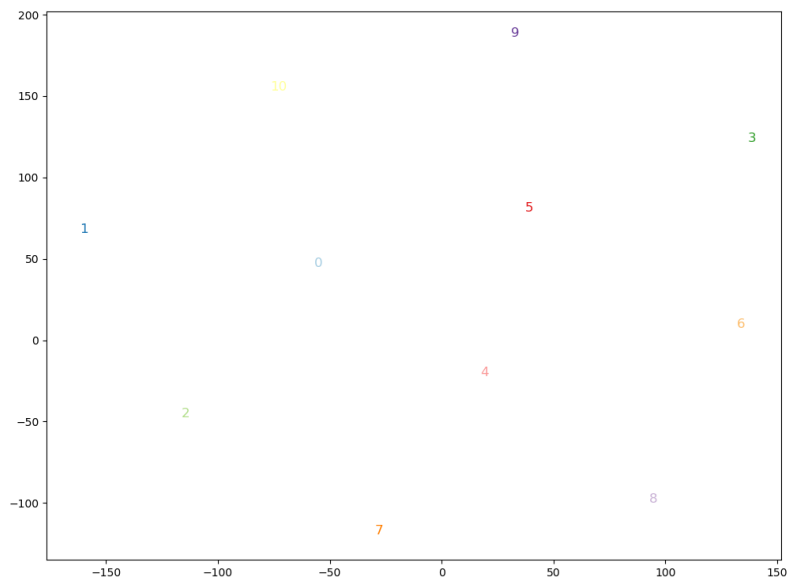


図 5.2 式ベクトルの分布 (skipgram)

5.3 考察

結果の図 5.1, 図 5.2 は結果としてどちらも特徴をとらえていない結果となった。考えられる原因として, LSTM では入力された順番の情報は残らないため前後の関係に強く依存する数式ではうまく表現できなかったように思えるよって, 今後は双方向 LSTM や Attention を用いて前後の関係と, どこに注目してエンコードするのかを強化するひち用がある。また, LSTM の隠れ層を深くすることも考えられるが, 過学習が起きてしまうため Dropout など正則化を行い, 学習を進める必要がある。

5.3.1 問題の正誤判定

研究室の同期メンバーの手書きによる訓練データ 927 件, 実際の子供たちの手書きによるテストデータ 197 件にて学習を行った。正解率 (図 5.3), 平均二乗誤差 (図 5.4) を示す。

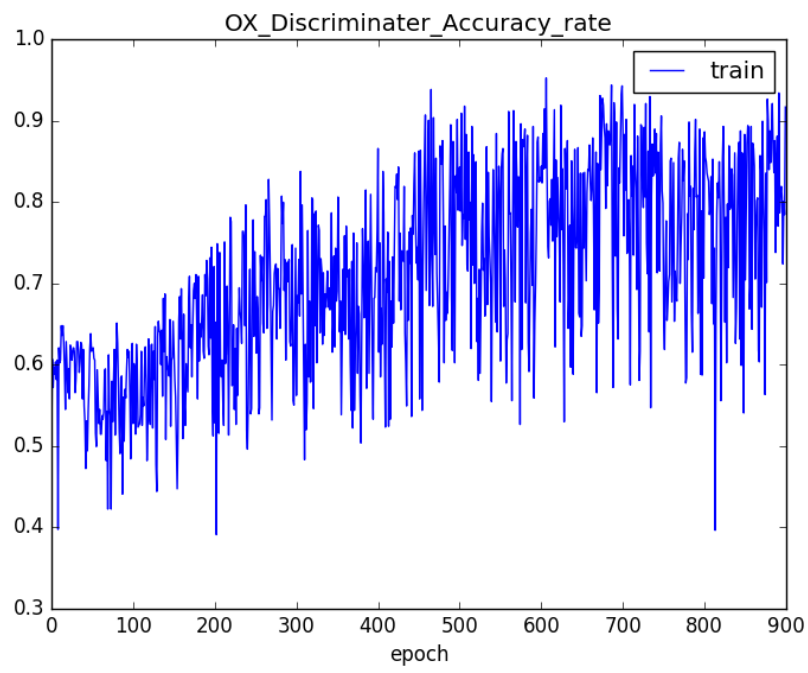


图 5.3 OX Discriminator 正解率

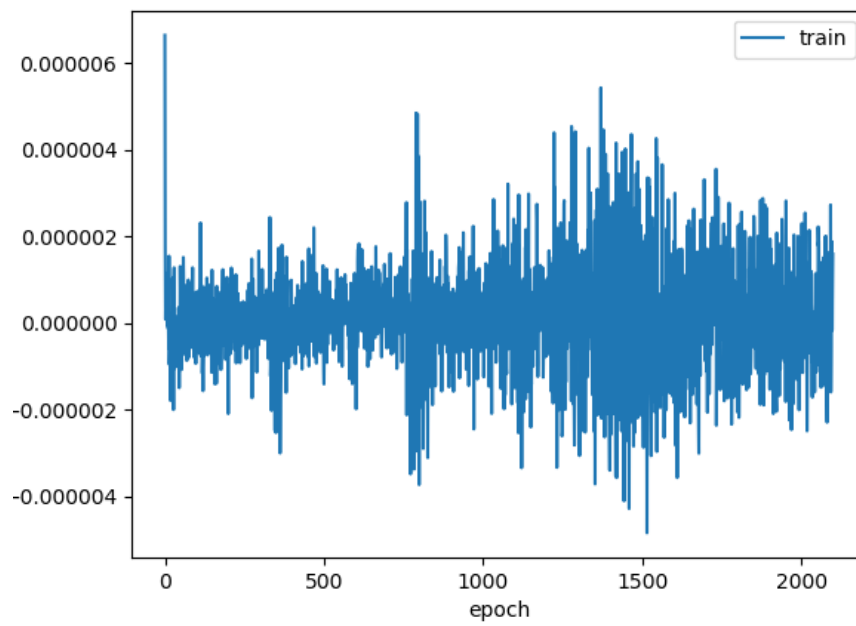


图 5.4 OX Discriminator 平均二乗誤差

第 6 章

関連研究

自然言語処理を数式を適応した例はなく，自然言語以外に Word2vec を適用しようとした研究はいくつかあり，再帰ニューラルネットを応用したものである．文献 [3] では文の特徴を Encoder-Decoder の機構を用いて学習し，Encoder の隠れ層の出力により文章の類似度を算出している．文献 [3] によると小説の一部を学習データに用いて，埋め込み層として SkipGram を適用している．gemsim の Doc2Vec との比較がなされているが，ユーザーにとって有益な評価方法の設定の難しさを述べている．

文献 [4] ではプログラムコードを埋め込み，再構成して実行できるかどうかを試している．C 言語で書かれたプログラムを一文字ごと，i-of-k 符号化して読み込んでいる．Dropout を適用させることにより性能を向上させている．今後，スペルミスによるコンパイルエラーを事前に直してくれることを目的としている．

文献 [5] では DeepLearning の技術の検索の際，ユーザーが求めている情報を推奨することを目指すとしている．評価方法として Movie Lens-100K での映画の推奨をパラメーター，ネットワークの構成を様々試しながら知見を集めている．AE は層を重ねても，性能は向上せず，各映画のユーザー平均を差し引いて正規化しても低脳は大きく変化しない．これにより，特徴抽出の重要性を筆者は述べている．

第 7 章

結論と今後の課題

本研究の結果によって計算式をベクトルとして扱うことが可能なことが示唆することができた。これにより数式データのみで最適な問題選択を可能とし、いまある個人情報と合わせるとより高い精度でシステムが提供できるように思う。本研究により、自然言語処理の分野で発展してきた技術は人口言語でも利用することができることが示せた。これは人間が作り発展させてきたものはどこか本質的には同様の特徴を持っており、その特徴をコンピュータが学ぶことができたことではないだろうか。

しかしながら、未だ問題点はある。計算問題には簡約な問題ほどパターンがなくなり、式から有用な情報が取り出せなくなる恐れがある。また、多くの子供が苦手とする文章題は本研究では扱っていないが、Dec2Vec など文章をベクトル化する手法は提案されており、それを応用することにより可能性は広がるだろう。また、本研究で扱った一次方程式の特徴量を使って別の分野を分類し、個人のまちがえた問題の情報を元に転移学習を行うとより、包括的なアダプティブラーニングシステムの構築が行えるようになり自分で学ぶ際の道しるべとなることを期待している。

謝辞

この研究を進めるにあたって、私の興味の赴くまま進めるなか、様々な角度からご指導いただいた千代教授に深く感謝申し上げます。また研究が息詰まる中、会話を通して気づきを与えてくれた研究室メンバーに感謝申し上げます。最後に、毎週たくさんの計算問題を宿題として解いてきてくれた私の塾の生徒たちに感謝をするとともに必ず、この経験を元に世の中に良い教育システムを提供することを約束し、謝辞といたします。

参考文献

- [1] Tomas Mikolov , Kai Chen , Greg Corrado , Jeffrey Dean : JeffreyDeanEfficient Estimation of Word Representations in Vector Space (2013).
- [2] ベネッセ教育総合研究所：第6回学習指導基本調査 DATA BOOK（高校版）(2016).
- [3] 福田 清人 森 直樹 松本 啓之亮：LSTM を用いた文の分散表現の獲得手法に関する一考察 言語処理学会 第24回年次大会 発表論文集 (2018).
- [4] 岡田龍治, 河原崎徳之:Deep Learning を用いた人工言語の学習と獲得について The 31st Annual Conference of the Japanese Society for Artificial Intelligence (2017).
- [5] 川上 和也, 松尾 豊：Deep Collaborative Filtering Deep Learning 技術推薦システムへ応用 The 28th Annual Conference of the Japanese Society for Artificial Intelligence, (2014).