CNN の畳み込み層の相違による 文章のランク分類精度の評価

椎名研究室
I13I006 井上 佳祐

1. はじめに

人工知能を使ったロボットや自動返答ボットなどが数多く登場した．SoftBank の Pepper[1]や Microsoft の女子高生 AI りんな[2]などが，その典型的な例である．このように人工知能という技術が広く使われるようになった背景として，近年のインターネットの普及やコンピュータの性能の向上が挙げられる．それと同時に，ディープラーニング[3]を含め機械学習を簡単に行うための便利なツールやライブラリなどが容易に利用できるようになった．ディープラーニングの中でも畳み込みニューラルネットワーク(Convolutional Neural Network，以下 CNN)は，画像データの分類に広く用いられている．近年ではコメントデータのような文章の分類など自然言語処理に用いる研究[4]も行われている．本研究では，CNN によるコメントのランク分類を行うことを目的としている．また，CNN の構成の最適化について考察する．

2. 実験データ

本研究では，楽天市場の商品レビューデータ(2011 年 7 月 28 日)[5]を利用した．このデータには 17 個の項目が登録されており，評価ポイントとレビュー内容の 2 つの項目のデータを学習および評価に利用した．評価ポイントは 0～5 の 6 段階評価である．本研究では 2 分割されているデータのうち，1297697 件を学習データに，1216726 件を評価データとした．購入者が投稿したレビューデータのコメントをその商品に対する購入者の評価ランクで分類する．

3. ニューラルネットワークの実装

自然言語処理における CNN で問題なのは，ニューラルネットワークへの文章の入力方法である．画像はピクセルの集合であり縦×横の行列であると考えられるので，文章もこれに類似した行列に変換する．したがって，埋め込み層で単語の埋め込み表現を学習し，それで畳み込みを行う．この分類処理の CNN による実装を以下に示す．

(1) コメント文の前処理(トークン化)
(2) CNN の定義
　入力層，埋め込み層，畳み込み層，
　プーリング層，全結合層，出力層
(3) 目的関数の定義
(4) データの学習

全体的な CNN の処理の流れは，図 1 である．

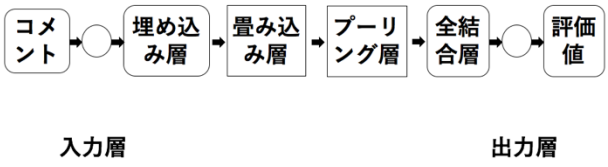

図 1：定義する CNN

3.1 コメント文の前処理

埋め込み層で単語の埋め込み表現を学習するために文を形態素に分割し，単語をトークンにし，あらかじめ決定しておいた単語番号に変換する．

3.2 埋め込み層

埋め込み層では，トークン化された入力文の埋め込み表現を学習する．これはフレームワークが提供している埋め込み表現の学習用の関数を使うことで，任意の大きさのベクトル表現が得られる．単語ベクトルの例を図 2 に示す．

学習データ

[4 1 3 2]
[9 6 7 2]
[8 4 1 3]

⇨

[0.22253558, -0.51343656, 0.39483854, -1.01296079, 0.74450827, 0.03340778,
0.63655376, 0.29914647, 1.26631176, -0.44816419, -0.10227863, 0.76881063,
0.93215591, 0.63805532, 1.07391071, -0.52140832, 0.02097545, -0.86970377, -
0.3341319 , 1.69880629, 1.17793298, 1.1131103 , 1.1095798 , -0.03460694, -
0.49842513, -1.38973546, 1.0281918 , 2.53226948, -1.05585551, 0.74964237, -
0.45200208, 0.46049953, 0.12066962, -0.57276702, -0.73867673, 0.37622529, -
0.43154648, 0.01532916, 0.49568811, 0.61219311, 0.29185149, -0.24250962, 0.6957947 ,
-0.26319307, 0.04157272, -1.65584898, 0.13503219, 0.24356173, -0.92708266, -
0.63016993, 0.72835582, 2.88588667, 1.19045007, -0.92559004, -2.13895416, -
0.49224392, 1.4457078 , -1.41968906, -1.80346417, -1.42850816, -0.66523618,
1.09498549, -0.32863444, -0.31957778, 1.40867913, -0.81215465, 0.67264998,
0.13475539, 0.54513305, 0.38563031, -0.38309678, -0.29069212, -0.22043755, -
0.36124042, 1.27557719, -0.30965707, 1.04994857, -1.99358869, 1.26344717, -
1.53390515, -1.74759626, 0.97374189, -0.14662105, 0.55295181, -1.34445226,
0.84889925, 0.48737845, -1.19684994, 2.91381645, 2.30349541, -0.56967443,
0.31953144, -0.5177117 , 1.85178065, 0.82115096, -0.19061778, -2.09569383,
1.97779095, -1.2134403 , -0.04439155, -0.00703091, 0.60047144, 0.66870439, -
0.40702832, -2.66448641, -0.67620909, -2.90645075, 2.66200376, -0.30617481,
1.77124357, 0.31038299, 0.79812115, 0.72735864, -1.19478667, 0.51105452, 0.25091335,
0.51274854, -0.00474253, -0.69179523, -1.74236977, -1.39979017, 0.1123848 ,
0.16439371, -1.57108748, 1.31287861, 0.08910379, -2.49765635, -0.94409096]

図 2：「スマホ」の単語ベクトル

3.3 畳み込み層

畳み込み層へは，埋め込み層から出力された単語のベクトル表現を縦に並べ，文の単語数×単語ベクトルの大きさの行列を作る．1 つの文ごとにひとまとめにして入力する．

次の畳み込み層の処理では，行列に対して指定した高さと幅のフィルタごとに畳み込みを行う．フィルタの幅は，入力行列の各行が 1 つの単語トークンとなっているため，それに対応させ，埋め込み表現の次元数と同じ大きさに設定する．また，フィルタの高さを任意に設定する．畳み込み層は，畳み込みで得られるデータを特徴マップとしてベクトルを出力する．

3.4 プーリング層

プーリング層ではマックスプーリングのルールに従い，畳み込み層の各フィルタから出力された特徴マップの行列から最大値を取得し，次元削減後，それを結合して 1 つのベクトルにする．これが文章の特徴ベクトルとなる．この特徴ベクトルは，文章の長さが変わっても常に同じ次元数のベクトルを作る．

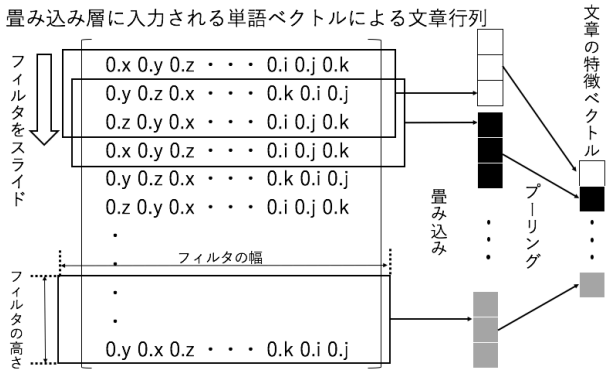

図 3：畳み込みとプーリングイメージ

3.5 出力層

出力層では，全結合層のからの出力に，ソフトマックス関数を適用させて分類する各クラスに属する確率を出力する．畳み込み層・プーリング層を通って全結合層から出力される文章の行列ベクトル$(x_1, x_2, \cdots x_M)$が評価値kである確率を次に表すソフトマックス関数を用いる．

$$\mathrm{P}_k(x_1, x_2, \cdots x_M) = \frac{e^{f_k(x_1, x_2, \cdots x_M)}}{\sum_{k'=1}^{K} e^{f_{k'}(x_1, x_2, \cdots x_M)}}$$

3.6 目的関数の定義

ニューラルネットワークの学習は，学習時に出力層から出力される確率値と教師ラベルに属する確率値を大きくするように重みを学習する．その指標となるのが目的関数である．出力層でソフトマックス関数を使用しているため，目的関数に交差エントロピーを用いた．データ$X_n = (x_1, x_2, \cdots x_M)$が教師ラベル$t_{k',n}$に属する確率値$P(X_n)$を最大化するには，次式に表す目的関数 E を最小化する．

$$\mathrm{E} = -\sum_{n=1}^{N}\sum_{k'=1}^{K} t_{k',n} \log P_{k'}(X_n)$$

3.7 データの学習

定義したニューラルネットワークの学習を複数回繰り返し行う．データの学習は，目的関数の微分による勾配を利用したアルゴリズムに従って目的関数の値を最小化する．

4. 精度および考察

ニューラルネットワークの実装とアルゴリズムに対して，畳み込み層でのフィルタの高さ(図 4)と畳み込み層とプーリング層の並列数(ノード数)(図 5)を変えた 7 つのモデルを評価した．

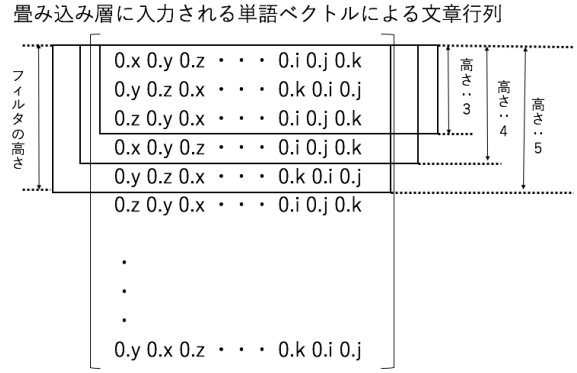

図 4：畳み込み層のフィルタの高さイメージ

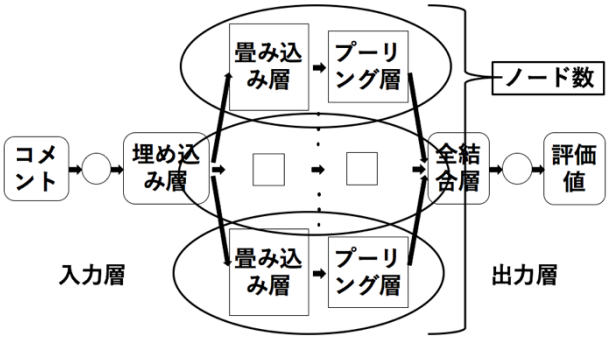

図 5：ノード数イメージ

4.1 畳み込み層 1 ノードの CNN

畳み込み層とプーリング層が 1 ノードで高さが 2～6 とそれぞれ違う畳み込みフィルタを用いた CNN を図 6 に示す．また，その精度を，表 1～5 に示す．

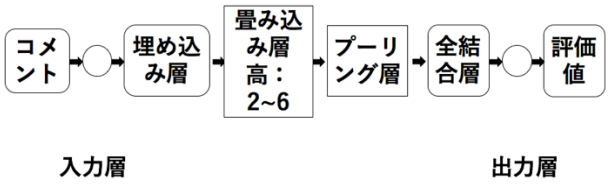

図 6：畳み込み層 1 ノードモデル

表 1：1 ノードモデル(高さ 2)の精度

回数	損失	精度
1	0.7374	0.6852
2	0.7286	0.6876
3	0.7308	0.6866
4	0.7490	0.6812
5	0.8002	0.6663

表 2：1 ノードモデル(高さ 3)の精度

回数	損失	精度
1	0.7335	0.6869
2	0.7223	0.6903
3	0.7240	0.6898
4	0.7442	0.6840
5	0.7887	0.6703

表 3：1 ノードモデル(高さ 4)の精度

回数	損失	精度
1	0.7303	0.6867
2	0.7186	0.6912
3	0.7200	0.6911
4	0.7381	0.6864
5	0.7760	0.6751

表 4：1 ノードモデル(高さ 5)の精度

回数	損失	精度
1	0.7304	0.6881
2	0.7195	0.6906
3	0.7169	0.6925
4	0.7352	0.6880
5	0.7721	0.6764

表 5：1 ノードモデル(高さ 6)の精度

回数	損失	精度
1	0.7259	0.6891
2	0.7151	0.6927
3	0.7164	0.6926
4	0.7369	0.6875
5	0.7805	0.6736

5 回の学習における各回の損失と精度を見てみると，2,3 回学習時での精度が最も高く，それを境に損失も大きくなり精度も低下している．最も高い精度と 5 回学習時の精度を比較すると，およそ 1.5～2%も精度が低下している．また，5 回学習時の精度が 5 回の中でも一番低く，過学習していると考えられる．このモデルでの学習回数は 2,3 回が適切でありそれ以上学習する必要はない．また，フィルタの高さを 2～6 と変えて学習したが，高さを高くし，より多くの単語を一度に畳み込むほど精度が高くなっている事が分かる．しかし，高さ 5 と高さ 6 の精度はそこまで差がみられないため，これ以上高くしても効果は見られないと考える．しかし，高さ 6 のほうが高さ 5 よりも 1 回すくない学習回数で同程度の精度に到達している．

4.2 畳み込み層 2 ノードの CNN

畳み込み層とプーリング層を並列に 2 ノードにし，高さ 3 と高さ 4 の畳み込みフィルタを用いた CNN を図 7 に示す．また，その精度を，表 6 に示す．

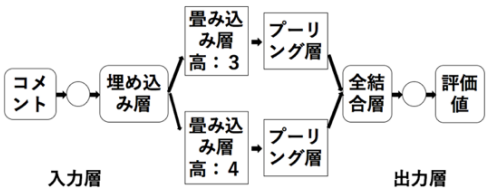

図 7： 畳み込み層 2 ノードモデル

表 6：2 ノードモデルの精度

回数	損失	精度
1	0.7272	0.6884
2	0.7182	0.6921
3	0.7222	0.6908
4	0.7462	0.6837
5	0.7917	0.6729

5 回の学習における各回の損失と精度を見てみると，1 ノードの時と同様に 2 回学習時での精度が最も高く，それを境に損失も大きくなり精度も低下している．1 ノードの高さ 3 と高さ 4 での精度と比較すると，0.1%程度であるが，それぞれ 1 ノードずつで分類するよりも高い精度が得られているので，ノードを複数個にすることで，精度が向上する可能性が考えられる．

4.3 畳み込み層 3 ノードの CNN

畳み込み層とプーリング層を並列に 3 ノードにし，高さ 3 と高さ 4 と高さ 5 の畳み込みフィルタを用いた CNN を図 8 に示す．また，その精度を，表 7 に示す．

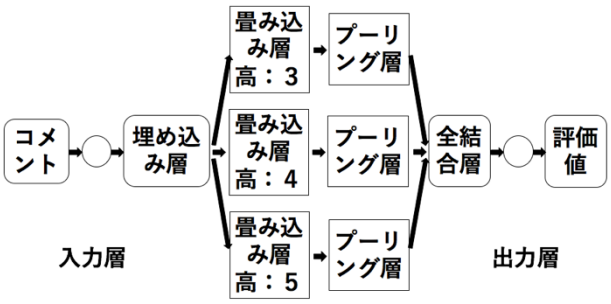

図 8： 畳み込み層 3 ノードモデル

表 7：3 ノードモデルの精度

回数	損失	精度
1	0.7174	0.6908
2	0.7123	0.6935
3	0.7300	0.6897
4	0.7910	0.6779
5	0.8835	0.6620

5 回の学習における各回の損失と精度を見てみると，1 ノード・2 ノードの時と同様に 2 回学習時での精度が最も高く，それを境に損失も大きくなり精度も低下している．このモデルでは，1 回目の学習で既に精度が 69%を超えてきており，実験に用いた CNN のなかで一番速い．また精度もどの CNN よりも高い値が得られており，ノード数の複数化による精度向上に多少期待できる．

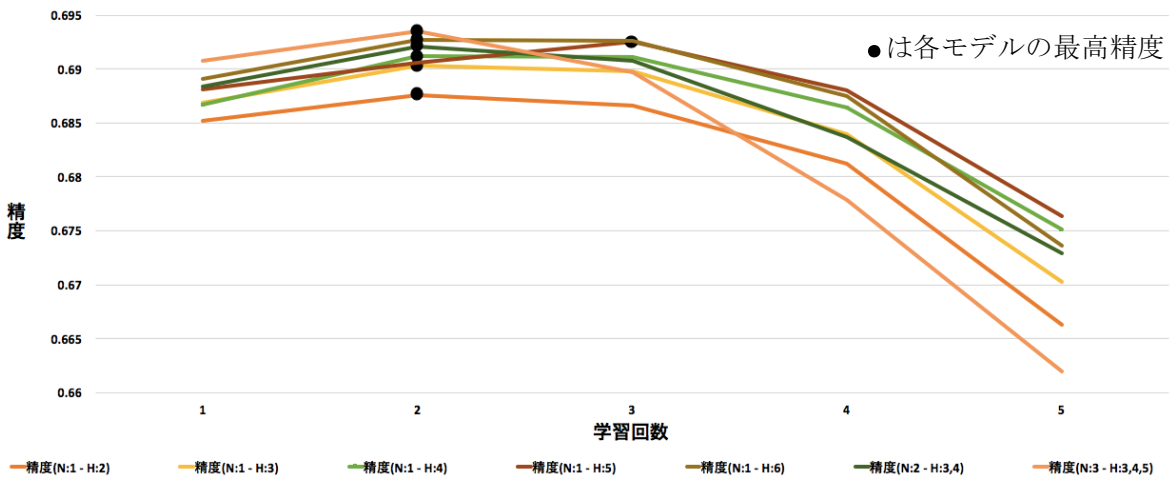

図 9： 学習回数ごとの精度変化グラフ

5. 実験評価

5.1 7 つのモデルの総合評価

7 つのモデルにおいて学習回数ごとの精度を図 9 のグラフ(N:ノード数，H:フィルタの高さ)に示す．どの種類のモデルでも類似した値の変化になっているが，中でも 3 ノードのモデル(N:3 – H:3,4,5)は学習を重ねるに連れて，精度の低下度合いが大きい事がわかる．

5.2 CNN 構成の最適化についての考察

CNN によるコメント分類での精度向上の最適化は次の手順で行う．(1) 1 ノードの高さが小さいフィルタで学習する．(2) 精度が低下するまで学習し，精度が悪くなったら 1 つ前のモデルを使い局所最適化を行う．(3) フィルタの高さを大きくし，同様に学習する．(4) フィルタの高さを大きくしても精度に変化が見られなければ，ノード数を増やす．(5) 1 ノードの時の精度を参考にフィルタの高さを設定し同様に学習する．この手順を 2 ノード，3 ノードと繰り返すことで多少の精度向上は見込めると考えられる．

6. 結論

実験結果とその評価から，今回の実験データでより高い精度を得るためには，畳み込み層の高さを変え，ノードを複数並列にした CNN を用いるのがよい．また，学習回数は 2,3 回にするのがよい．しかし，畳み込みを行う処理が多ければ多いほど，より時間がかかるので精度と学習時間との兼ね合いが大切である．

7. まとめ

本稿では，CNN での自然言語処理として，商品レビューのコメントデータの評価分類を行う CNN の実装方法を示した．このようなニューラルネットワークはパラメータの数が多く学習には膨大な時間が必要である．本来ならばフィルタ数や次元数などをもっとチューニングすべきであるが，時間の都合上その点が疎かになってしまっている．学習時間を短くするには学習データをもっと小さいデータにし，学習回数を少なくすれば良いと考えられる．今後は，少ない学習データでも同程度の精度が得られるかどうか，パラメータやノード数の違いによる精度への影響を評価する必要がある．

参考文献

[1] Life with Pepper | ソフトバンク，http://www.softbank.jp/robot/special/pepper/，（2017/01/01 参照）
[2] りんな，http://rinna.jp/，（2017/01/01 参照）
[3] NTT コムウェア | ディープラーニング，https://www.nttcom.co.jp/research/keyword/dl/，（2017/01/01 参照）
[4] Yoon Kim, Convolutional Neural Networks for Sentence Classification, EMNLP(2014), 1746-1751
[5] 国立情報学研究所，http://www.nii.ac.jp，（2017/01/01 参照）
[6] 新納浩幸，Chainer による実践深層学習，オーム社（2016）
[7] 浅川伸一，Python で体験する深層学習，コロナ社（2016）
[8] 斎藤康毅，ゼロから作る DeepLearning，オライリー・ジャパン（2016）
[9] 中井悦司，TensorFlow で学ぶディープラーニング入門，マイナビ（2016）
[10] 清水亮，はじめての深層学習プログラミング，（2017）
[11] TensorFlow, https://www.tensorflow.org/，（2017/01/01 参照）
[12] Keras Documentation, https://keras.io/，（2017/01/01 参照）