気になる機械学習

総合情報基盤センター 教授 布村 紀男

ここ数年、ビッグデータ活用に併せて人工知能(AI)の話題が各方面で取り上げられています。画像認識を始め、音声認識、自動翻訳、さらには自動運転に AI 技術が使われます。今後、この AI 技術が社会に及ぼすインパクトは大きいと予想されています。AI 技術に関連する機械学習(Machine Learning)や深層学習(Deep Learning) という語をよく見かけ、専門外の筆者も気になっています。そこで、少し手を動かして、機械学習の入口の扉を開けてみたいと思います。

キーワード：機械学習, 深層学習, 教師あり学習, 強化学習, 教師なし学習, word2vec

1．はじめに

人工知能(AI)技術の歴史は、1940 年代から研究が始まり、2 度のブームと冬の時代を経て、2000 年代以降に再びブームになり、今日、衰退することなく急速に普及しています。急速に普及している理由として、データサイズの大規模化と取得が容易になったことやコンピュータ性能の向上、さらに機械学習アルゴリズムの発展が挙げられます。特に深層学習を使って自動で特徴を抽出できるようになったことは劇的です。このような背景から、深層学習向けのフレームワークやライブラリが複数の会社や団体から提供されています[11-13]。本稿では、Web 上での情報も多く、誰でもすぐに試せる自然言語処理に関する word2vec [8,9]を取り扱います。

2．機械学習

機械学習とは、人が機械的に学習することではなく、人が自然に行っている学習能力と同様な機能を機械（コンピュータ）で実現しようとする AI 技術です。人が大規模で複雑なデータを分析してルールやパターンを見つけてモデル構築する代わりに、機械学習はデータから知識を引き出し、より効率的な方法を提供することで、予測モデルの性能を向上させます。

機械学習には「教師あり学習」,「教師なし学習」、「強化学習」の3種類があります。教師あり学習は、ラベル付けされた訓練データからモデルを学習し、未知のデータや将来のデータを予測できるようにします。分類や回帰分析が行われます。強化学習は、教師あり学習に関連し、環境とのやり取りに基づき性能を向上させることをめざします。環境のやり取りはエージェントが行い、報酬や状態を反映させます。教師なし学習は、答えや報酬がなくてもデータ構造を調べることでグループ分けやクラスタリングにより、意味のある情報を抽出することができます。また、教師なし学習では次元削減(圧縮)が行われます。

初期の機械学習のアルゴリズムは、生物の脳神経回路網からモデル化された人工ニューロンに基づく、パーセプトロンのアルゴリズムです。図-1 は 2 つの信号を入力とする 2 入力パーセプトロンモデルを表します。x_1、x_2は入力信号、y は出力信号、w_1、w_2は重みを表します。図の○はニューロンを意味します。

入力信号はニューロンに送られるとき、w_1x_1、w_2x_2としてそれぞれに重みが乗算されます。送られてきた信号の和が計算され、その和がある閾値を越えた場合に 1 を出力するモデルです。一方、最近の機械

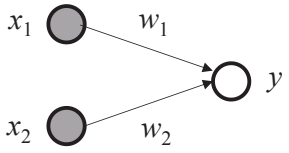

図-1 パーセプトロンモデル

学習アルゴリズムでは、多層ニューラルネットワークモデルが用いられています。(図-2)

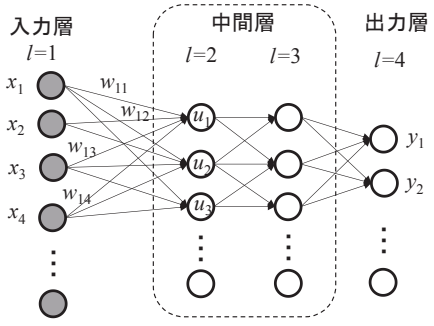

図-2 ニューラルネットワークモデル

３．word2vec

word2vec [8]は、単語の分散表現を計算するためのツールです。単語の分散表現とは、その単語の意味を低次元の密な実数値ベクトルで表現したものです。簡単に言えば、単語列の前後並びから単語のベクトルを学習するモデルで、教師なし学習です。アルゴリズムの詳細はオリジナル論文[9]を参照してください。word2vec の手法は、skip-gram(SG) モデルと continuous bag of word(CBOW)モデルが実装されています。一般的に学習データがあまり大きくない場合は前者、一方、大きい場合は後者の方が分散表現は良いようです。また、類義語と対義語の区別が苦手なことや出現頻度が下がると正しいペアを作ることが難しくなることが指摘されています[10]。

４．環境構築

筆者の手元には手頃な PC が無かったので、まずは安価で小型の Rasberry Pi3 の利用を考えました。しかし、標準実装の 1GB ではメモリ不足なため、仕方なく、現役を退いた小型サーバ機(HP ProlLiont MicroServer)上でテスト環境を構築しました。ハードウェアと主なソフトウェアは、次のとおりです。

[ハードウェア]
プロセッサ: AMD Athlon II Neo N36L Dual-Core
メモリ：8GB
ハードディスク: 100GB
[ソフトウェア]
OS: ubuntu 16.04 LTS (64 ビット)
日本語形態要素解析: MeCab ver.0.996
スクリプト言語: ruby ver. 2.2.3
テキスト変換: wp2txt ver.0.8.0
日本語コード変換: nkf ver.2.1.3

４．１ word2vec のインストール

・事前作業

word2vec 導入の前に subversion と build-essential をインストールします。

```
$ sudo apt-get install subversion build-essential
```

・word2vec ソースコードの入手

subversion を使って、ホームディレクトリにソースコードをダウンロードします。

```
$ cd
$ svn checkout
http://word2vec.googlecode.com/svn/trunk
word2vec
```

・word2vec のコンパイル

```
$ cd word2vec
$ make
```

４．２ 日本語形態素解析システム MeCab の導入

学習用の訓練データとして、日本語文書を準備するには、語と語の間に空白で区切る、単語分割、いわゆる「分かち書き」をする必要があります。今回は MeCab[14]を使用します。ubuntu Linux では、以下の手順で MeCab および周辺環境ソフトウェアの導入を行いました。

```
$ sudo apt-get install mecab libmecab-dev
mecab-ipadic-utf8 git make curl xz-utils file
```

続いて、MeCab の辞書 mecab-ipadic-NEologd を導入します。

まずは github からクローンを取得します。

```
$ git clone --depth 1
https//github.com/neologd/mecab-ipadic-
neologd.git
```

次にディレクトリ移動し、インストールします。

```
$ cd mecab-ipadic-neologd
$ ./bin/install-mecab-ipadic-neologd -n
```

辞書のインストール先を確認します。

```
$ echo `mecab-config --dicdir`"/mecab-ipadic-
neologd"
```

辞書のインストール PATH が表示されます。

```
/usr/lib/mecab/dic/mecab-ipadic-neologd
```

４．３ wp2txt のインストール

訓練データで用いる日本語 wikipedia の dump ファイルをテキスト形式へ変換するためのツール wp2txt[15]を導入します。

```
$ sudo apt-get install rbenv ruby-build
```

・設定

環境設定ファイル ~/.bashrc に次の 2 行を追加します。

```
export PATH="$HOME/.rbenv/bin:$PATH
eval "$(rbenv init -)"
```

・設定の反映

```
$ source ~/.bashrc
```

・rbenv でインストール

```
$ rbenv install 2.2.3
$ rbenv local 2.2.3
$ rbenv global 2.2.3
$ rbenv exec gem install wp2txt bundler
$ rbenv rehash
```

４．４ 訓練データの入手とクレンジング（洗浄）

・訓練データとして、日本語 wikipedia の dump ファイルと青空文庫のドストエフスキー「カラマーゾフの兄弟」上巻の 2 種類のテキストを用いました。
http://www.aozora.gr.jp/cards/000363/card42286.html

以下では Web サイト[16]を参考にして日本語 wikipedia の場合について記述します。

1. 日本語 wikipedia の dump ファイルを入手

```
$ curl
https://dumps.wikimedia.org/jawiki/latest/jawiki-latest-pages-articles.xml.bz2 -o jawiki-latest-pages-articels.xml.bz2
```

2. wp2txt を使って圧縮 dump ファイルを展開し、テキスト形式に変換

```
$ rbenv exec wp2txt --input-file jawiki-latest-pages-articles.xml.bz2
```

3. cat コマンドで変換された複数のテキストファイルを 1 つにまとめます。

```
$ cat jawiki-latest-pages-articles.xml*.txt > jawiki-dump.txt
```

4. dump テキストファイルを MeCab で分かち書きにします。

```
$ cat jawiki-dump.txt | mecab -d
/usr/lib/mecab/dic/mecab-ipadic-neologd -Owakati -b 81920 > jawiki-wakati.txt
```

５．word2vec による機械学習の実行と評価

分かち書きされた訓練データのテキストファイルを word2vec に入力し、機械学習させます。比較的大規模なデータなので CBOW モデルを用いました。パラメータとして、ベクトル次元数 200、ウィンドウ幅 5、その他はデフォルト値を使用しました。

```
$ word2vec -train jawiki-wakati.txt -output jawiki-train.bin -size 200 -window 5 -binary 1
```

CPU は低速なため、待つこと約 8 時間、学習が完了し、出力ファイル jawiki-train.bin が作成されます。このファイルを使って学習した成果を試します。

・学習したモデルを単語の距離で評価

単語ベクトルの類似度は word2vec に付属の distance を使います。このツールでは、2 つの単語ベクトルのなす角 θ の $\cos\theta$ で距離を評価します。出力数値が 1 に近いほど距離が短く、類似性があることを意味します。図-3 に地名 (a)「東京」と(b)「富山」を調べた場合の結果、上位 20 個を示します。

```
$ distance jawiki-train.bin
```

(a) 東京

順位	語	距離
1	大阪	0.824149
2	名古屋	0.773900
3	横浜	0.747458
4	京都	0.744903
5	関西	0.713791
6	新宿	0.711502
7	神戸	0.710877
8	札幌	0.707497
9	神奈川	0.703759
10	銀座	0.703213
11	福岡	0.700529
12	関東	0.694355
13	浅草	0.690322
14	仙台	0.689733
15	新潟	0.688747
16	日本橋	0.688551
17	静岡	0.681967
18	広島	0.679848
19	八王子	0.677199
20	池袋	0.675877

(b) 富山

順位	語	距離
1	金沢	0.825594
2	福井	0.763193
3	静岡	0.699248
4	新潟	0.694198
5	長野	0.693281
6	浜松	0.680235
7	岐阜	0.671190
8	秋田	0.659375
9	山形	0.655197
10	敦賀	0.650514
11	高岡	0.649387
12	魚津	0.647037
13	氷見	0.638468
14	和歌山	0.638207
15	鳥取	0.637717
16	群馬	0.635681
17	栃木	0.635302
18	山梨	0.630615
19	名古屋	0.629754
20	高知	0.629738

図-3 距離の評価

東京の場合は上位の大阪、名古屋、横浜、京都から隣接県よりも大都市圏という類似性が連想されます。一方、富山の場合は隣接県が上位にあります。例外的か？ 3 位に静岡、6 位に浜松が選ばれています。

・word-analogy でベクトル計算から類似度を評価

word-analogy では、man、king、woman の 3 単語を問い合わせると、ベクトル空間上で king－man＋woman の演算を行い、単語を探し出し、queen と返すことで有名です。このツールを使って、類似度を評価しました。ここでは、(a) 男 王様 女 と (b) 過去 地球 未来 について入力した結果を図-4 に示します。

(a)男　王様　女

順位	語	距離
1	お姫様	0.670001
2	シンデレラ	0.605138
3	お姫さま	0.595396
4	女王様	0.590689
5	姫君	0.572743
6	魔女	0.565778
7	花嫁	0.563741
8	女王	0.563054
9	人魚	0.551140
10	わたし	0.550871
11	蜜蜂	0.545814
12	天使	0.542975
13	ママ	0.542548
14	白雪姫	0.538775
15	王子	0.538658
16	アリス	0.537621
17	ひとりぼっち	0.536430
18	淑女	0.536068
19	人魚姫	0.535842
20	パパ	0.535776

(b)過去　地球　未来

順位	語	距離
1	宇宙	0.682902
2	人類	0.579001
3	火星	0.575456
4	惑星	0.565249
5	太陽系	0.561297
6	銀河系	0.550485
7	銀河	0.542897
8	太陽	0.537393
9	宇宙へ	0.537015
10	世界	0.520607
11	星	0.515386
12	木星	0.504622
13	新しい世界	0.503219
14	月面	0.490279
15	ラーメタル	0.489819
16	デジタルワールド	0.48745
17	地球の危機	0.483831
18	地球環境	0.480584
19	時空	0.479461
20	ツフル	0.477158

図-4 類似性の評価

(a)は見事、「王様」-「男」+「女」≈「お姫様」の結果が得られました。しかし下位の順位に目を向けると、関連性のない語も選ばれています。一方、(b)では、宇宙が1位に選ばれています。すなわち、「地球」-「過去」+「未来」≈「宇宙」で、何となく意味合いが想像できそうです。単なるテキスト文章からの機械学習による演算結果なので驚きです。

・学習モデルの比較テスト

青空文庫からテキストデータ「カラマーゾフの兄弟」について、SGとCBOW学習モデルによる比較テストを行いました。word2vecのパラメータは、ベクトル次元数 200、ウィンドウ幅 10、階層的ソフトマックス処理有りで固定しました。図-5にdistanceツールを使い「国家」に近い距離の単語を調べた結果を示します。

(a) SG

順位	語	距離
1	教会	0.887842
2	同化	0.855127
3	上級	0.820857
4	下級	0.801086
5	社会	0.789021
6	裁判	0.773324
7	またがる	0.753092
8	基礎	0.747027
9	別個	0.745852
10	使命	0.745454
11	キリスト教	0.743138
12	である	0.741561
13	包含	0.740400
14	求む	0.731620
15	永久不変	0.728374
16	一定	0.724099
17	異教	0.723305
18	登っ	0.721655
19	段階	0.719961
20	区別	0.719858

(b) CBOW

順位	語	距離
1	教会	0.824149
2	キリスト教	0.773900
3	社会	0.747458
4	包含	0.744903
5	処罰	0.713791
6	本質	0.711502
7	役回り	0.710877
8	Moor	0.707497
9	論拠	0.703759
10	国法	0.703213
11	裁判	0.700529
12	異教	0.694355
13	永久不変	0.690322
14	定め	0.689733
15	基礎	0.688747
16	現今	0.688551
17	監視	0.681967
18	投げこま	0.679848
19	犯人	0.677199
20	刑罰	0.675877

図-5 単語距離の評価

両モデルで1位に「教会」が選ばれています。訓練データに利用した「カラマーゾフの兄弟」では「国家と教会」もテーマに含んでいます。大きくない訓練データではCBOWよりもSGモデルが良いと言われていますが、このケースでは1位に関して差がないことがわかります。また、SGモデルでは2位「同化」、3位「上級」、4位「下級」、一方CBOWモデルでは「キリスト教」、「社会」、「包含」を選出され、違いがみられます。この原因はよくわかりません。

6．おわりに

気になっていた機械学習について、word2vecに手を出してみて、なぜAI技術の注目度が高いのか、何となくわかった気がします。試した結果は学習モデル以前に、日本語テキストの場合、分かち書きの品質が影響していると考えられます。この影響を考慮して、余力があれば、自分のメールボックスに蓄積されたメール文書でもword2vecしてみようかと思います。

参考文献

[1] 大関真之:「機械学習入門」ボルツマン機械学習から深層学習まで オーム社 (2016).

[2] 齋藤康毅:「ゼロから作るDeep Learning」Pythonで学ぶディープラーニングの理論と実践 オライリー・ジャパン (2016).

[3] 岡谷貴之:「深層学習」 機械学習プロフェッショナルシリーズ 講談社 (2015).

[4] Sebastian Raschka:「Python機械学習プログラミング 達人データサイエンティストによる理論と実践」インプレス (2016).

[5] 新納浩幸:「Chainerによる実践深層学習」 オーム社 (2016).

[6] 三好健文:「カメラ眼付き人工知能コンピュータの実験」トランジスタ技術 2016 8 pp.92-104 CQ出版社.

[7] 「ラズパイに ON! Google 人工知能」インターフェース 2017年3月号 CQ出版社

[8] word2vec https://code.google.com/p/word2vec

[9] T.Mikolov, K.Chen, G.Corrado, and J. Dean. "Efficient Estimation of Word Representatons in Vector Space. arXiv preprint arXir.1301.3781,2013 http://arxiv.org/pdf/1301.3781.pdf

[10] 西尾泰和:「word2vecにより自然言語処理」オライリー・ジャパン 電子書籍 epub (2014).

[11] TensorFlow https://www.tensorflow.org

[12] Chainer http://chainer.org

[13] Caffe http://caffe.berkeleyvision.org/

[14] MeCab http://taku910.github.io/mecab/

[15] wp2txt http://wp2txt.rubyforge.org/index-old.html

[16] http://yoshipc.net/word2vec-wiki/