文書分類をタスクとした Pylearn2 の Maxout+Dropout の利用

永田純平　新納浩幸　佐々木稔　古宮嘉那子
茨城大学 工学部情報工学科
11t4056h@hcs.ibaraki.ac.jp, {shinnou, msasaki, kkomiya}@mx.ibaraki.ac.jp

1 はじめに

本論文では Deep Learning のライブラリ Pylearn2 で実装されている Maxout+Dropout 利用して文書分類のタスクを行う。上記ライブラリを実際に利用しその利用方法を紹介することが本論文の目的である。

近年 Deep Learning の研究が進んでおり、様々なタスクで優れた成績を残している [1]。Deep Learning は多層ニューラルネットの総称であり、Deep Learning の中には様々な手法が存在する。本研究で扱う Maxout+Dropout もその１つである。Maxout はニューラルネットの活性化関数であり、Dropout は過学習を抑制するニューラルネットの学習テクニックである。この２つを組み合わせた Maxout+Dropout は精度の高い分類器を学習することが可能であり、現在、識別のタスクに対する帰納学習手法の中で state of the art となっている [2]。

非常に高い精度を出す Maxout+Dropout ではあるが、自然言語処理の研究分野では、我々の知る限り、適用事例はない。その原因は手軽に使えるツールがなく、自然言語処理のタスクに対して Maxout+Dropout が現実的な計算機資源で実行できるのかどうかが不明であることの２点が考えられる。

一方、Deep Learning のツールとして Pylearn2[1] が存在する [3]。そして Pylearn2 の中には Maxout+Dropout が実装されている。本論文ではこの実装例を利用して、Maxout+Dropout を文書分類のタスクへの適用を試みる。

Pylearn2 は非常に有用性の高いツールであるが、学習データを与えることで学習器ができあがり、その学習器にテストデータを与えれば識別結果が得られる、というようにツールの中身、実行過程をブラックボックス化して利用できるものではない。そのために本論文では、上記ツールの具体的な利用方法を示す。これによって上記に述べた Maxout+Dropout が利用されない２つの原因を解消することができると考えている。

[1]http://deeplearning.net/software/pylearn2/
https://github.com/lisa-lab/pylearn2

2 Maxout と Dropout

2.1 Maxout

Maxout はニューラルネットの新しい活性化関数の１つである。活性化関数とは、ニューラルネットで次の層へと情報を伝播する際、伝播する値を計算する関数である。従来、活性化関数にはシグモイド関数が用いられることが一般的であったが、最近では Rectified Linear Unit(ReLU) なども用いられている。

＜シグモイド関数＞

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-ax}} \tag{1}$$

＜ ReLU ＞

$$f(x) = max(0, x) \tag{2}$$

Maxout は層間に存在する隠れ層から次の層への複数の出力からまとめて値を決定するもので、その中の最大値を次の層のノードへの入力とする。

＜ Maxout ＞

$$h_i(x) = max_{j \in [1,k]} Z_{ij} \tag{3}$$

$$Z_{ij} = x^{\mathrm{T}} W_{...ij} + b_{ij} \tag{4}$$

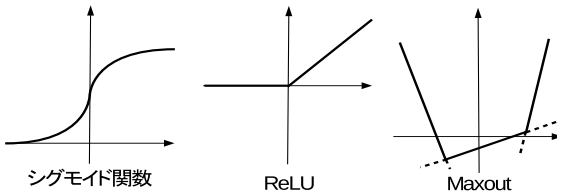

図 1: 活性化関数

図１のグラフを見てわかるように ReLU は二種類の線形関数からなるが、Maxout は特定の型を持っておらず何種類もの線形関数で構成することができる区分線形関数である。活性化関数を学習することで図３で示すような ReLU や二次関数など様々な関数に近

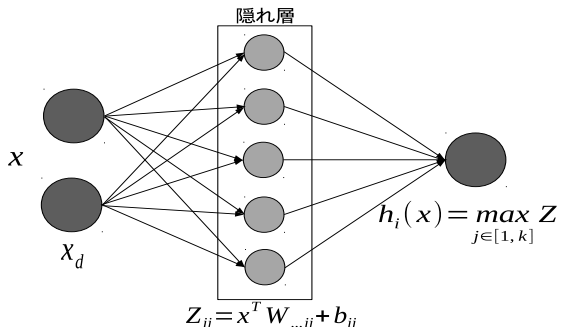

図 2: Maxout

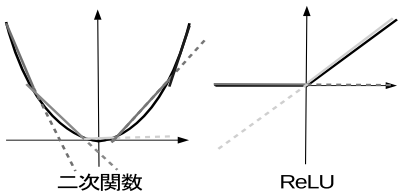

図 3: Maxout による関数の近似

似することができるため、Maxout は ReLU よりも表現力が高く、勾配が消滅しないという特徴を持っている。

Maxout の関数の近似に関しては、Maxout $h_i(x)$ は隠れノードが十分にあれば、任意の凸関数を近似することができ、二種類の Maxout $h_1(x) - h_2(x)$ からなるネットワークは任意の関数を近似することができると定理されている。そのため 2 層以上の Maxout からなるネットワークと 1 層以上の Maxout と 1 層以上の Softmax からなるネットワークは任意の関数を近似することができる。よって上記の定理に沿ったネットワークが構成されると Maxout は様々な関数に近似することができるため万能の活性化関数といえる。

2.2 Dropout

Dropout はニューラルネットでの学習テクニックの 1 つで、ネットワーク内で次の層へと情報を伝播する際に、情報を伝播する層のノードの中から 50%を Mini-batch 毎にランダムに選択し、選択されたノードを無視して情報を伝播するという学習テクニックである。無視をするというのはそのノードの出力を 0 とするということであり、無視する割合として一般的に 50%という割合が用いられているが、実際にはその割合は任意である。しかし、割合が大きくなればなるほどノード間の関係が疎となり割合が小さければ小さいほどノード間の関係が密となるため、50%が適当であるとされている。

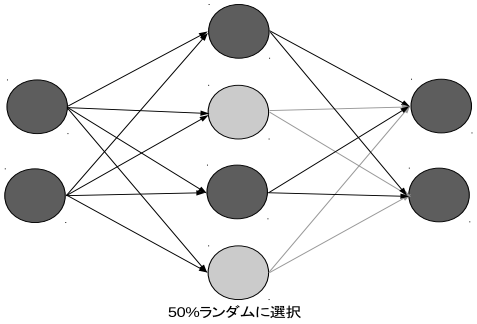

図 4: Dropout

Dropout のアルゴリズムとして、学習時には各層のノードの半分をランダムに無視して学習を行い、学習後の推定時には学習したパラメータを 1/2 にしてすべてのノードを使って推定を行う。

Dropout には過学習を抑制する効果があり、アンサンブル学習と似た効果が得られるためネットワークの汎化性能が向上する。過学習とは学習をしすぎることで、既知のデータへの識別精度が向上する分、未知データの識別精度が低下することをいい、Dropout は各学習ステップでの負担を減らすことで過学習を抑制することができている。その反面、Dropout はサンプリングに時間を要し、ノードの半分を無視することでパラメータを半分しか使用できないため、更新する回数を多く設定しなければ学習が進行せず学習が遅くなるという難点がある。

3 Pylearn2

Deep Learning のライブラリ Pylearn2 を今回の実験ではツールとして利用する。

Pylearn2 は Python で実装されており、Theano という計算ライブラリをベースとしている。また、現在も開発改良中で更新がこまめに行われている。

Pylearn2 を用いる利点として、学習アルゴリズムが豊富に実装されており、MNIST や CIFAR といった様々なデータセットに対応しているなど、利用するうえでの利便性が非常に高い。

次に、Pylearn2 の利用方法を紹介する。Pylearn2 の学習プログラムは train.py である。このプログラムを実行する際コマンドライン引数として YAML ファイルを指定する。YAML ファイルには学習アルゴリズムや、使用するデータセット、学習モデルなど学習するネットワークの構造、学習過程での詳細が記述されている。このように Pylearn2 では学習プログラムと、ネットワークの構造を示す YAML ファイルとがそれぞれが独立しているため、それらを個別に開発できるという点も Pylearn2 を用いる利点の 1 つである。

YAML ファイルでネットワークを構造を記述するには、ファイル内の dataset , model , algorithm に着目する。dataset では入力データの指定、model では学習器の設定、algorithm では学習法の設定を行う。今回の実験で使用する YAML ファイルは表 1 のように設定した。

表 1: YAML ファイル

	mnist_pi	mnist_pi_continued
dataset	AdultDataset	AdultDataset
model	MLP	push_monitor
algorithm	SGD	SGD

前述の学習アルゴリズムのプログラムは YAML ファイルに記述されている内容に従って学習を行い、学習した結果を pkl ファイルとして出力する。学習結果である pkl ファイルを用いて実行結果の評価を行う。評価する点に関してはタスクによって異なるが、それぞれ評価を行うプログラムも Pylearn2 内で実装されている。

Pylearn2 は対応するデータセットが多いと上記で示したが、対応していないデータセットを使用する際には、使用するデータセットをプログラムに対応するフォーマットに変換するラッパーを作成する必要がある。ラッパーを作成しデータセットを適したフォーマットに変換することで、対応していないデータセットを Pylearn2 で利用することが可能となる。

今回の実験では libSVM 形式のデータセットを使用したが、このデータのフォーマットは Pylearn2 に対応しておらず、libSVM 形式から csv 形式に変換するラッパーを作成した。また、データセットやラッパーも YAML ファイル内で呼び出して使用している。

表 2 に示すように、libSVM 形式は要素が 0 の index は表示しておらず値を持つ index をコロンで区切っている。csv 形式は値をコンマで区切っている。それぞれのデータフォーマットと変換例を表 2 に示す。

表 2: 変換例

libSVM	label index0:value0...indexN:valueN				
	label	index:value			
	13	0:2 2:1 ... N:1			
csv	label,value0,value1, ... ,valueN				
	label	(0)	(1)	(2)	... (N)
	13,	2,	0,	1,	..., 1

以上より、Pylearn2 の利用法として、YAML ファイルの作成、ラッパーの作成、学習プログラムの実行、評価といった手順となる。

4 文書分類の実験

今回の実験は Maxout+Dropout を用いた二値分類での文書分類を行う。実験で使用したデータは 20newsgroup からとってきた文書データを使用し、このデータ libSVM 形式でベクトル表現されている。ネットワークの構造は 3 層構造で 1 層と 2 層は Maxout、3 層は Softmax という構造となっている。Softmax は出力に関連する活性化関数で、ノードの出力値を確率分布にするために正規化を行っており、シグモイド関数の多変量版である。K 次元のベクトルを a とすると Softmax 関数は以下で表せる。

$$\sigma(a_i) = \frac{\exp(a_i)}{\sum_{k=1}^{K} \exp(a_k)} (i = 1, ..., K) \quad (5)$$

$$\sum_{k=1}^{K} \exp(a_k) = 1 \quad (6)$$

softmax 関数にベクトルを入力すると同次元のベクトルを出力として返すが、各要素の合計値は 1 となるため出力されるベクトルは離散確率分布となる。そのため Softmax 関数は多クラスの分類問題にも対応することが可能である。

今回の実験使用でするプログラムリストを表 3 に示す。

表 3: プログラムリスト

プログラム名	説明
train.py	学習を行うプログラム、引数に YAML ファイルをとる
mnist_pi.yaml	学習のモデルの構成を設定し構築するプログラム
mnist_pi_continued.yaml	
print_monitor.py	学習結果の pkl ファイルを用いて評価する 引数として pkl ファイルと test_y_misclass をとる

*test_y_misclass は分類した結果のエラー率を示す

実行のコマンド手順を以下に示す。

実行コマンド手順

```
train.py mnist_pi.yaml
train.py mnist_pi_continued.yaml
print_monitor.py mnist_pi_continued.pkl -
| grep test_y_misclass
```

学習を二度行っているが、最初の学習の YAML ファイルは、多層のニューラルネットの構造で学習を行い、二度目の学習では、一度目の学習結果の pkl ファイルを用いて再度学習することで精度を向上させている。

本手法の精度の評価するために比較実験を行った。オリジナルのデータをそのままの次元数で実行すると長い実行時間を要するため、使用するデータを任意の次元数に次元圧縮をしてから実験を行った。次元圧縮の手法には特異値分解を用い、次元数は 100 次元と 1000 次元に圧縮した。また、オリジナルのデータの次元数は 60000 次元である。その他に、SVM、最大エントロピー法、Naive Bayes を比較実験として行った。これらは次元圧縮せずにそのままの次元数のデータを使用する。本研究での実験結果を表 4 に示す。

表 4: 実験結果

手法		正解率
SVM		0.90
最大エントロピー法		0.94
Naive Bayes		0.98
Maxout + Dropout	次元数	
	100 次元	0.54
	1000 次元	0.55
	60000 次元 *オリジナルデータ	0.95

5 考察

実験結果から、Maxout+Dropout を用いた手法は正解率約 95 パーセントを記録し他の手法と遜色ない結果が得られた。しかし、他の手法との正解率を比較すると、高い正解率を残した反面、画期的な手法と呼べるまでには至らなかった。加えて実行時間に約 1 日もの時間を要するため[2] 、オリジナルデータをそのままこの手法に適用するのは現実的ではない。実行時間の短縮のためにデータの次元圧縮を行って実験を行った際には、大幅に時間の短縮には成功したが正解率が約 55%を記録、正解率の悪化が顕著に現れた。今回の実験は二値分類の文書分類であるため、約 55%という正解率は非常に悪い結果といえる。100 次元と 1000 次元に次元を圧縮して実験を行った場合にも、100 次元と 1000 次元での結果の間にも大した差が見られなかったため、特異値分解を用いた次元圧縮は、次元圧縮の手法としては適していないといえる。しかし、実際に次元を圧縮することで実行時間の短縮には成功しているため、他の次元圧縮の手法を使用し、適当な手法を見つけることができればより結果の向上が見込まれる。

本研究では Pylearn2 をツールとして利用したが、YAML ファイルで学習モデルを構成する際、学習回数や中間層の数、ノード数などユーザが設定しなければならないパラメータの数が多く、パラメータによってもネットワークの構造が変わってくるため、適したパラメータの設定など今後さらなる検討が必要である。

[2] ここでの利用計算機は OS:ubuntu 14.04 LTS メモリ:4GB プロセッサ:Intel Core 2 Duo CPU P8700 @ 2.53Hz × 2

6 おわりに

ここでは Pylearn2 の利用、Maxout と Dropout の理論、Maxout+Dropout の文書分類への適用について述べた。

Pylearn2 は環境構築やツールの理解に時間を要する。そのためこのツールを利用するうえでのユーザーの負担は大きい。しかし Pylearn2 は現在も開発改良中であるため、今後新たな技術の実装、機能性と実用性の向上などが期待できる。今回の実験では二値分類での文書分類を行った。今後は汎用性の向上のため二値分類から多値分類への拡張が必要である。実験で使用したモデルの構造として最後の層に Softmax を用いているため、多値への対応は可能であるが、二値での分類問題に取り組む現段階でも長い実行時間を要し、メモリを大きく使用するため、現段階では困難であった。これは筆者の Pylearn2 の理解が十分でないことが原因かもしれない。今後 Pylearn2 の更なる理解を深めこのツールおよび Maxout+Dropout の有用性を示していきたい。

参考文献

[1] Quoc V. Le,Marc 'Aurelio Ranzato,Rajat Monga,Matthieu Devin,Kai Chen,Greg S. Corrado,Jeff Dean,Andrew Y. Ng. "Building High-level Features Using Large Scale Unsupervised Learning" ICML-2011, (2011)

[2] Ian J. Goodfellow,David Warde-Farley,Mehdi Mirza,Aaron Courville,Yoshua Bengio. "Maxout Networks" ICML-2013, (2013)

[3] Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frederic Bastien, and Yoshua Bengio. " Pylearn2: a machine learning research library ". arXiv preprint arXiv:1308.4214