平成26年度 茨城大学工学部情報工学科卒業研究論文

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

平成27年2月10日提出

茨城大学 工学部情報工学科

11T4056H 永田 純平

指導教員： 新納 浩幸 准教授

平成 26 年度 茨城大学工学部情報工学科卒業研究

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

著者：永田 純平 (11T4056H)
指導教員： 新納 浩幸 准教授

論文要旨

近年、Deep Learning の研究が活発であり、様々なタスクで優れた成績を残している。Deep Learning は多層ニューラルネットを用いた機械学習手法である。Deep Learning の中には様々な手法があり、Maxout+Dropout もその一つである。Maxout はニューラルネットの活性化関数であり、隠れ層の出力の中から最大のものをノードの入力として決定する。Dropout はニューラルネットの学習テクニックの 1 つで、中間層のノードの半分を無視して学習を行うテクニックである。Dropout を使用することで過学習を抑制し、ネットワークの汎化性能を向上させることができる。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でき、識別のタスクでの state of the art となっている。非常に高い精度を示す Maxout+Dropout であるが、自然言語処理の分野では、我々の知る限り適用事例はない。それは、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 が存在する。そして Pylearn2 の中には Maxout、Dropout がともに実装されている。Pylearn2 は新しいツールであり、現在も開発改良中で更新がこまめに行われている。そのため、Pylearn2 の利用法を示し、本ツールの実装例を利用し Maxout+Dropout を文書分類のタスクへの適用が本論文の目的である。

実験では Pylearn2 を使用し、Maxout+Dropout を文書分類のタスクに適用した。Maxout+Dropout を用いたネットワークの構造は三層構造で、第一層と第二層は Maxout 層、出力層である第三層は Softmax 層となっており、学習された分類器を利用して 2 クラスの文書分類を行った。

実験の結果、Maxout+Dropout を用いた手法は正解率約 95%得ることができた。比較実験として行った SVM や Naive Bayes の正解率と遜色ない結果を得ることができたが、この手法は 非常に長い実行時間を要し、実際にデータをそのままの次元数で学習を行うことは現実的ではない。そのため、実行時間短縮のために特異値分解を用いてデータの次元数を圧縮してから実験を行った。その結果実行時間の大幅な短縮に成功したが、正解率が約 55%と正解率の低下が顕著にみられた。

Pylearn2 ではネットワークを構築するパラメータが多く、パラメータによってネットワークの構造も変わってくるため、適したパラメータの設定や、特異値分解とは異なる次元圧縮の手法の使用を試みることで結果の向上が見込まれる。パラメータの調整と多値分類への拡張が今後の課題である。

目 次

図 目 次

表 目 次

第1章　序論

1.1　概要

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

1.2 本論文の構成

本論文では、最初に今回の研究の主な目的である Maxout と Dropout の理論を説明し、第 3 章で今回使用するツールの Pylearn2 の紹介と利用法を説明する。第 4 章と第 5 章ではそれまでに説明した Maxout+Dropout の手法を Pylearn2 を文書分類のタスクに適用した実験内容と結果、実験結果を経た考察を行う。

最後に第 6 章では本研究を終えての結論と今後の課題について述べる。

第2章　Deep Learning

2.1　ニューラルネット

2.1.1　Multi Layer Perceptron(MLP)

MLP はニューラルネットの一種であり、下位の層の出力が上位層の入力になるように階層的に積み重ねたものをいう。

MLP では上位のノード y_j の出力を次の関数で表すことができる。

$$y_j = f(\sum_i x_i W_i j + b_j) \tag{2.1}$$

x_i は入力、$w_i j$ は入力のノード i と次の層のノード j との間の重みで, b_j はバイアスである。それら活性化関数 (activation function)$f(\cdot)$ に与えることで出力を計算することができる。

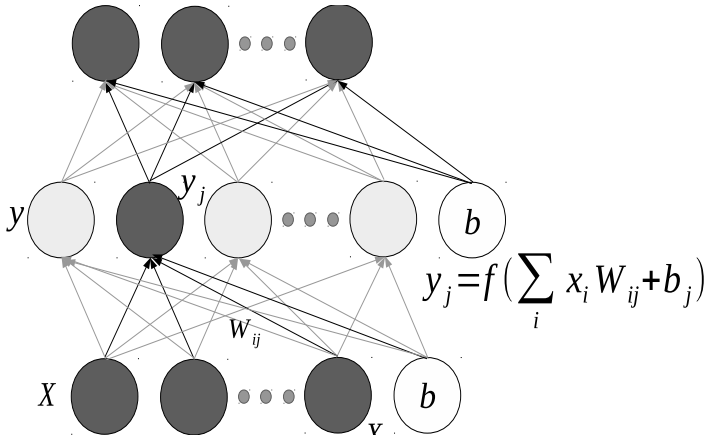

図 2.1: ニューラルネット

図 2.1 から分かるように上位層のノード y_j は下層 x の出力とバイアス b から成っている。

ニューラルネットを用いた学習の流れは、ネットワークを通して推定、出力値と正解データとの誤差を計算し、誤差から重みなどのパラメータを更新する。この流れを誤差が収束するまで繰り返す形となる。

パラメータの更新には誤差逆伝播法 (Back Propagation) で行う。誤差逆伝播法は下位層から上位層へ学習を進めていく方向とは逆方向に値を伝播させる手法で、学習から得られた識別結果と教師データとの誤差をもとにパラメータを下位層に向かってパラメータの更新を行う。

まず、ネットワークの出力 z_i と教師データ d_i との誤差を計算する。誤差の取り方にも二乗誤差やヒンジロスなどいくつか種類があり、タスクや出力によって異なる。ニューラルネットを用いて分類問題に取り組む場合、ネットワークの各出力をシグモイド関数にかけることで出力を 0〜1 に変換し、それを確率とするのが一般的である。そのため確率の誤差をとるためにはエントロピー誤差がよく用いられる。

$$E = Loss(z_i, d_i)^1 \tag{2.2}$$

続いてパラメータの更新を行う。出力全体の誤差をとる場合、各教師データに対して式 2.4 のようにパラメータを更新する。

$$E_N = \sum_i^N Loss(z_i, d_i) \tag{2.3}$$

$$W = W - \frac{\partial E_N}{\partial W} \tag{2.4}$$

この手法は最急降下法といい、比較する教師データが少なければ有効だがデータ数が多い場合、一度に全てのデータで誤差を求めるのはマシンへの負担が大きくさらに過学習を引き起こしやすい。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過学習という。

そのため、最急降下法に対して確率的勾配降下法 (stochastic gradient descent) を用いると i 番目のデータに注目し、式 2.5 のようにパラメータを更新する。

$$W \leftarrow W - \rho \frac{\partial E_i}{\partial W} \tag{2.5}$$

ρ は学習率を表す。教師データを使って誤差をとる場合、一度の更新で 1 つのデータを参照するため、更新の動作を向上させることができる。このように一度の学習で一つのデータの更新を行う学習をオンライン学習という。

さらに、更新速度の向上のためにモーメント法も用いられる。これは学習率 ρ が低く設定してあると誤差の収束に時間を要するため、momentum 係数 μ を用いることで重みの調節を行う。モーメント法を用いた確率的勾配降下法の更新式を式 2.6 に示す。

$$W \leftarrow \mu W - \rho \frac{\partial E_i}{\partial W} \tag{2.6}$$

[1]Loss は誤差を表す。

2.1.2 Deep Neural Network(DNN)

Deep Learning はニューラルネットワークを多層に重ね合わせた機械学習の手法である。Deep Learning で使用するネットワークには、MLP を多層にしたネットワークを DNN や、Restricted Boltzman Machines(RBM) を多層化した Deep Belief Networks,Deep Boltzman Machines など様々な種類があり、今回は DNN を使用する。

通常機械学習では、入力データからの特徴抽出を行い、抽出した特徴から学習、推定を行うが、Deep Learning では特徴抽出までモデルが行う。これは Deep Leaning の主な特徴の一つである。

既存のニューラルネットワークを組み合わせて学習を行うため、DNN では基本的に 2.1.1 で説明した学習のアルゴリズムで機械学習が行われる。

Deep Learning ではパラメータの更新の際、確率的勾配降下法とともにミニバッチ学習と併用してよく用いられる。全ての学習データを一度の学習で使用するバッチ学習に対して、ミニバッチ学習は指定されたサイズにデータを分割し、分割したグループのデータの更新を行う手法である。Deep Learning ではミニバッチごとにランダムに出力データを分割したデータを確率的勾配降下法でパラメータの更新を行う。この学習の流れを図 2.2 に示す。

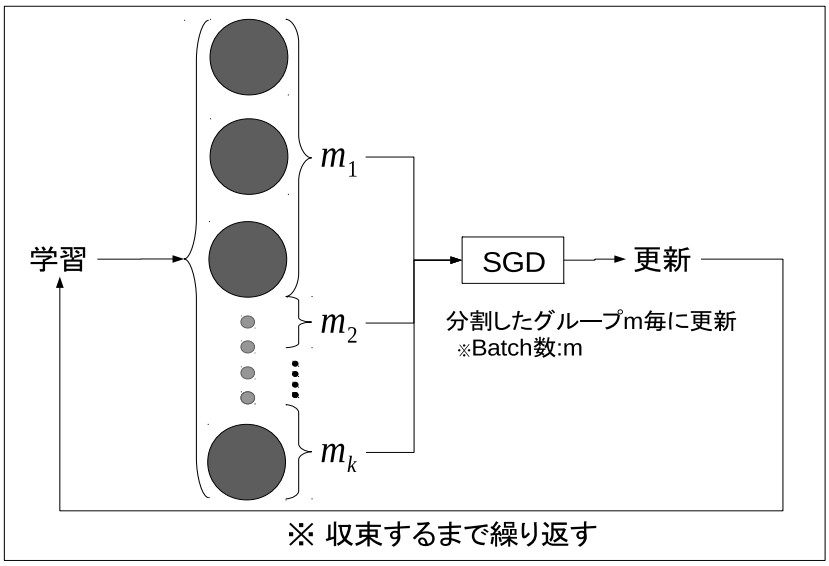

図 2.2: Deep Learning でのパラメータ更新

こうした学習の結果作成された分類器を使用して、文書分類を行う。

2.2 活性化関数

活性化関数とはニューラルネットでの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。この節では一般的な活性化関数と今回の研究で用いる活性化関

数を紹介する。

2.2.1 従来の活性化関数

ニューラルネットの活性化関数として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) などもよく用いられている。

＜シグモイド関数＞

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-ax}} \tag{2.7}$$

＜ ReLU ＞

$$f(x) = max(0, x) \tag{2.8}$$

各関数の特徴として、シグモイド関数は非線形関数であり、Saturation を引き起こすことで勾配が消滅してしまい、ReLU は正の値であると勾配が 1 となり、Saturation を引き起こすことがなく学習が早いという特徴がある。

2.2.2 Maxout

Maxout はニューラルネットの新しい活性化関数である。

Maxout は層間に存在する隠れ層から次の層への複数の出力をまとめて関数に掛け合わせることで次の層への入力値を決定する。

＜ Maxout ＞

$$h_i(x) = max_{j \in [1,k]} Z_{ij} \tag{2.9}$$

$$Z_{ij} = x^{\mathrm{T}} W_{...ij} + b_{ij} \tag{2.10}$$

式 2.9 からわかるように隠れ層の出力の中から最大のものを次の層のノードへの入力値とする。

図 2.4 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。

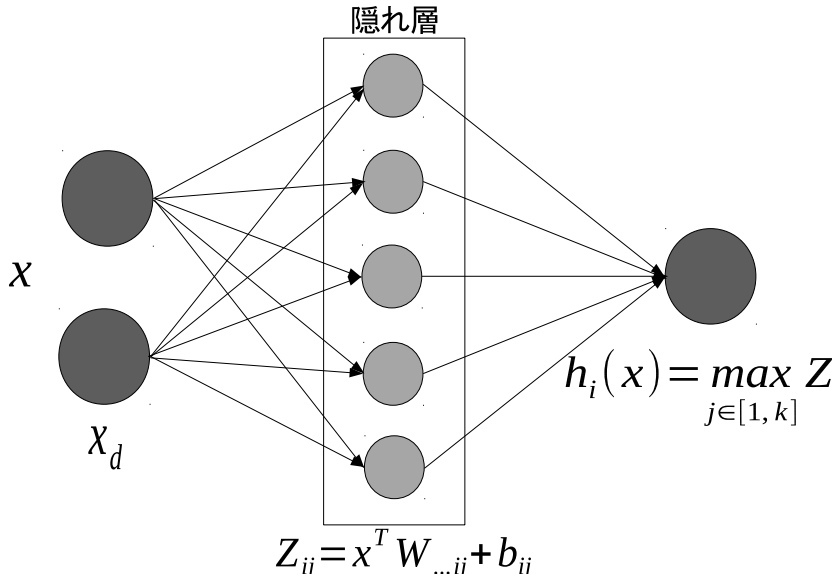

図 2.3: Maxout

2.2.3 Softmax

最後に Softmax 関数 に関して説明を行う。

Softmax はニューラルネットの出力に関連する重要な活性化関数である。ノードの出力値を確率分布にすることで、正規化を行っており、出力のノード数は、識別するクラス数であり、出力の確率分布の中から最大のものを識別クラスとする。前に紹介したシグモイド関数の多変量版であり、入力を K 次元のベクトルを a とすると Softmax 関数 は以下のように表す。

$$\sigma(a_i) = \frac{\exp(a_i)}{\sum_{k=1}^{K} \exp(a_k)} (i = 1, ..., K) \tag{2.11}$$

$$\sum_{k=1}^{K} \exp(a_k) = 1 \tag{2.12}$$

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。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。

2.3 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

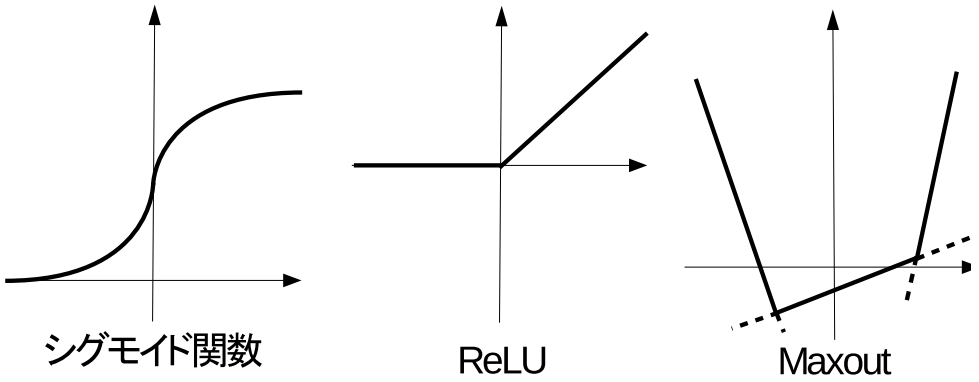

図 2.4: 活性化関数

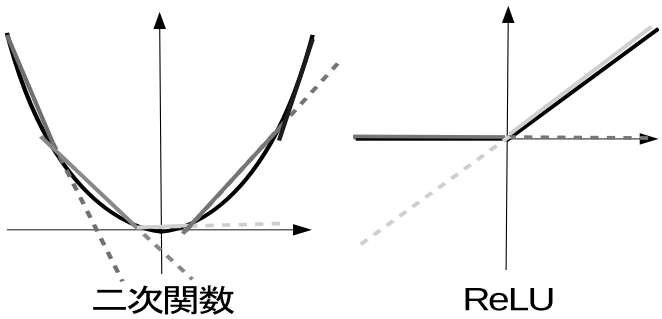

図 2.5: Maxout による関数の近似

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。

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 に似た手法に Dropconnect という学習テクニックがある。Dropout は各層のノードの半分の出力を 0 とするが、Dropconnect は各ノード間の結合の一部をランダムに選択しパラメータを 0 とする手法である。

Dropout は各ステップでの負担を減らすことで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。また、Dropconnect でも同様の効果を得ることができる。

その反面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

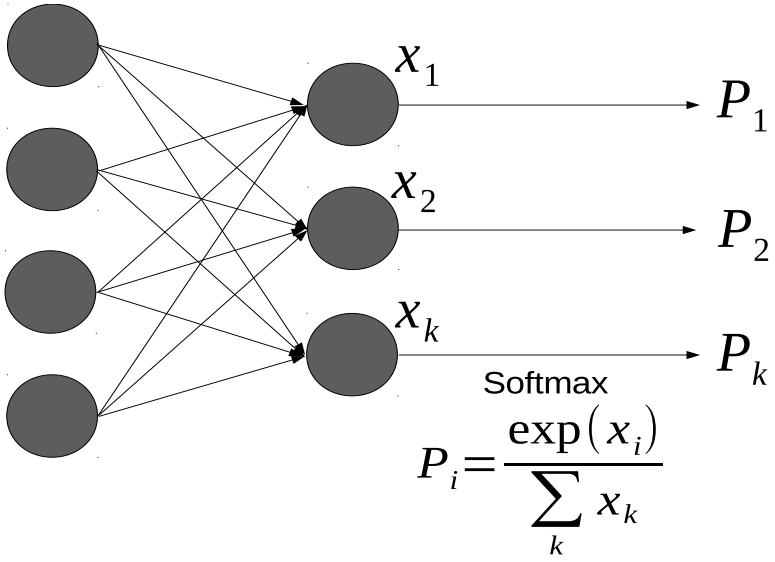

図 2.6: Softmax

が遅くなるという難点がある。

今回の研究で使用する Maxout+Dropout を用いた手法は、様々なタスクで優れた成績を残しており現段階での最も優れた手法とされている。

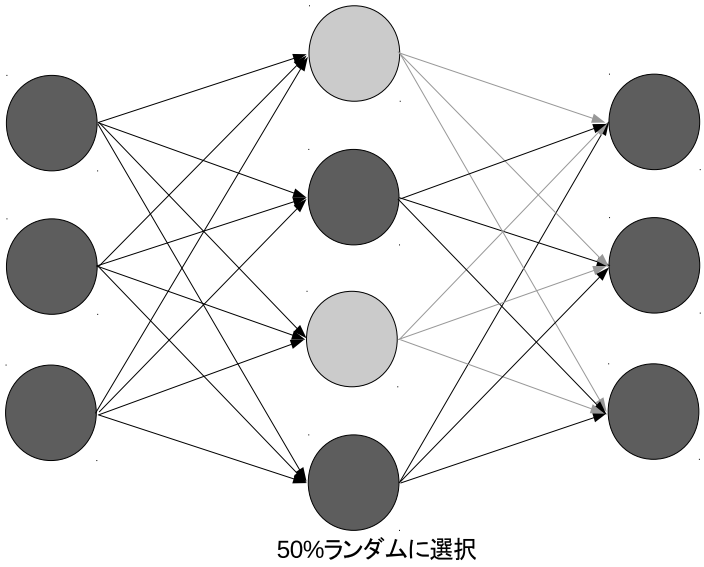

図 2.7: Dropout

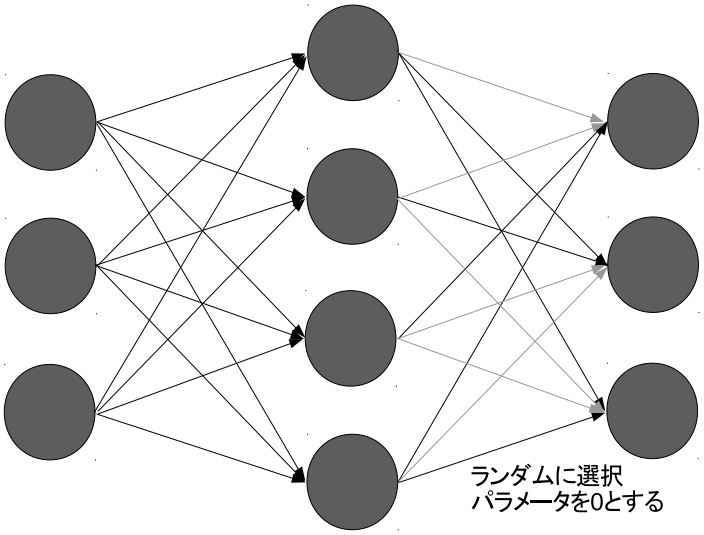

図 2.8: Dropconnect

第3章　Pylearn2

3.1　Pylearn2 の紹介

今回の研究では、Deep Learning のライブラリの Pylearn2 を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点があげられる。

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。プログラムを実行する際コマンドライン引数として学習に関する詳細が記述している YAML ファイルを指定する。このように Pylearn2 では学習プログラムと、学習の詳細を示す 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することができるという点も Pylearn2 を用いる利点の１つである。

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、それぞれのタスクの評価を行うプログラムも Pylearn2 内で実装されている。

3.2　YAML

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おり、Pylearn2 を使用するうえで理解が必要な重要なファイルである。

YAML ファイルでネットワークを構造を記述する際、ファイル内の dataset , model , algorithm に着目する。dataset では入力データ、model では学習器、algorithm では学習方法がそれぞれ記入されているため、ここを書き換えることで学習の詳細を設定することができる。また、学習した結果を出力する pkl ファイルも YAML ファイルで指定する。

今回の実験で使用する YAML ファイルは表 3.1 のように設定した。

YAML ファイル内で入力データ、学習器、学習方法などそれぞれ設定するが、Deep Learning は多層にネットワークが構築されておりモデルが複雑であるため、その分パラメータが非常に多く存在する。

表 3.1: YAML ファイル

	mnist_pi	mnist_pi_continued
dataset	AdultDataset	AdultDataset
model	MLP	push_monitor
algorithm	SGD	SGD

層の数や入力の次元数や隠れ層の次元数などのネットワークの構造に関するものや、バッチサイズ、学習率、更新回数など学習の最適化に関するパラメータなどがあり、与えるパラメータによっては学習結果に影響を与えるため、適したパラメータの設定も Pylearn2 を利用するうえで重要な作業である。さらにパラメータ以外にも判別器や活性化関数の選択、Stochastic Gradient Descent(SGD) などの更新の手法の選択、学習の収束の判断なども考慮する必要がある。

Pylearn2 での学習器と学習法に関する重要なパラメータ[1]を表 3.2 に示す。

表 3.2: パラメータリスト

	パラメータ		
dataset	start	int	start から検査を開始
	stop	int	stop で検査を終了
model	nvis	int	入力の次元数
	nhid	int	隠れ層の次元数
	num_units	int	層のノード数
	num_pieces	int	次の層への接続の数
	n_classes	int	識別クラス数 (出力次元数)
algorithm	batch_size	int	バッチサイズ
	learning_rate	float	学習率
	init_momentum	float	momentum 係数

ネットワークでの学習を通じて学習器のモデル構築を行う際、Train データを用いて分類器を作成するが、Train データの一部を利用して分類器の汎化性能の評価を行う。そうして作成された学習器を用いて Test データの識別に利用される。学習 (train)、検査 (valid)、識別 (test) のエラー率はそれぞれ train_y_misclass , valid_y_misclass , test_y_misclass を見ることができ、そのエラー率によって精度の評価が可能となる。

この過程に関しては YAML ファイルの monitoring_dataset に記述されており、記述内容を以下に示す。

[1] http://deeplearning.net/software/pylearn2/library/index.html

monitoring_dataset

```
monitoring_dataset:
            {
                'train' : *train,
                'valid' : !obj:adult_dataset.AdultDataset {
                              which_set: 'adult/train.csv',
                              start: 1000,
                              stop: 1187
                          },
                'test'  : !obj:adult_dataset.AdultDataset {
                              which_set: 'adult/test.csv',
                          }
            },
```

3.3 ラッパー

Pylearn2 は対応するデータセットが多いと上記で示したが、もちろん対応していないデータセットも多く存在する。対応していないデータセットを Pylearn2 で使用する際には、使用するデータセットを Pylearn2 に対応するデータの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 に対応していないため、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 3.3 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 3.3 に示す。

表 3.3: 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 の一般的な利用法として、YAML ファイルの作成、使用するデー

タセットが対応していなければラッパーの作成、学習プログラムの実行、評価といった手順となる。

第4章　文書分類の実験

文書分類 (Text Classification) とは文書をいくつかのクラスに分類するタスクである。分類にはなにも情報がない状態から分類を行う教師なし文書分類と事前に与えられたデータをもとに分類を行う教師あり文書分類があるが、今回は教師あり文書分類の実験を行う。

教師ありでの文書分類は、与えられたデータから特徴を抽出し規則性を見出だす。見出した規則性から分類器を作成し、分類器を用いて未知のデータの分類を行い結果を返すというのが教師ありのアプローチである。

4.1　実験

今回の実験は Maxout+Dropout を用いた手法を文書分類のタスクに適用し実験を行う。

Maxout+Dropout は機械学習による文書分類であり、文書データを機械学習に適用するには素性を要素とするベクトル表現で表す必要がある。そのため今回実験で使用するデータは 20newsgroup の文書データを使用し、このデータは libSVM 形式でベクトル表現されている。

今回の実験で使用するプログラムリストを表 4.1 に示し、プログラムの実行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
```

実行手順の中で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。

1 度目の学習の YAML ファイルに記述されたネットワークの構造を表 4.2 に、学習方法を表 4.3 に示す。

表 4.2 からわかるように、今回の実験で使用するニューラルネットワークの構造は 3 層構造となっている。1 層と 2 層は Maxout、3 層は Softmax といった構造となっており、第 3 層の Softmax がこのネットワークの出力層となっている。ネットワークの出力層に Softmax を使用しているが今回の実験では 2 クラスでの文書分類を行う。

表 4.1: 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

また、本手法の精度評価のための比較実験として、データの次元数を圧縮し、同じ Maxout+Dropout を用いて実験を行うアプローチと、同じデータを用いて様々な手法で実験を行うアプローチ、二つのアプローチで比較実験を行った。

次元数を圧縮する理由として、Deep Learning のデメリットに長い実行時間を要する点があり、実行時間の短縮のためにデータの次元圧縮を事前に行った。次元圧縮の手法には特異値分解を、圧縮する次元数はオリジナルのデータは 60000 次元に対し、100 次元と 1000 次元に圧縮した。

その他に、手法の精度比較のため 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オリジナルのデータを使用する。

4.2 特異値分解

今回の実験ではデータの次元圧縮に特異値分解を使用する。特異値分解は線形代数の行列分解の手法の一つであり、特異値分解などを用いてベクトル表現された文書単語行列の次元を圧縮する技術を自然言語処理の分野では Latent Semantic Analysis(LSA) と呼ばれる。

A をランク r、$m \times n$ の行列とすると、特異値分解によってこの行列は以下のように分解することができる。

$$A = U \sum V^{\mathrm{T}} \tag{4.1}$$

U は $m \times m$ の直交行列、$\sum$ は $m \times n$ の対角上に r 個の特異値が並んだ対角行列、V は $n \times n$ の直交行列である。

Python では特異値分解は以下のように実装することができる。

表 4.2: ネットワーク

model		**MLP**
	パラメータ	
layer1 Maxout	layer_name	'h0'
	num_units	240
	num_pieces	5
	irange	.005
	max_col_norm	1.9365
layer2 Maxout	layer_name	'h1'
	num_units	240
	num_pieces	5
	irange	.005
	max_col_norm	1.9365
layer3 Softmax	layer_name	'y'
	num_units	240
	irange	.005
	max_col_norm	1.9365
	n_classes	2
	nvis	*

*入力データの次元数を入力する

特異値分解

```
import numpy as np
X, Y, Z = np.linalg.svd(A)
```

X,Y,Z がそれぞれ式 4.1 の U,$\sum$,V^{T} を示す。LSI による次元圧縮法には、ランク数の削減による次元圧縮と文書行列の次元削減による次元圧縮の 2 つのアプローチがあり、今回の実験では次元削減による次元圧縮で次元圧縮を行った。

4.2.1 ランクの削減による次元圧縮

式 4.1 より $A = U \sum V^{\mathrm{T}}$ に分解できることが分かったが、分解した各行列の k+1 列目以降を削除することで A の行列に近似することができ、ランクを r から k に削減することができる。

$$A \simeq A_{(k)} = U_k \sum\nolimits_k V_k^{\mathrm{T}} \tag{4.2}$$

ランク削減による LSI は文書行列は m 次元のまま k 次元の空間に射影することで次元圧縮を行う。

表 4.3: 学習方法

algorithm		**SGD**
	パラメータ	
	batch_size	100
	learning_rate	.1
learning_rule Momentum	init_momentum	.5
cost : Dropout	input_include_probs	{ 'h0' : .8 }
	input_scales	{ 'h0': 1. }
termination_criterion MonitorBased	channel_name	"valid_y_misclass"
	prop_decrease	0.
	N	100
update_callbacks ExponentialDecay	decay_factor	1.000004
	min_lr	.000001

4.2.2 次元削減による次元圧縮

次元削減による次元圧縮は、文書行列の次元数はそのままで次元圧縮を行います。

$$A_{(k)} = V_k^{\mathrm{T}} A \tag{4.3}$$

もとに文書行列 A に特異値分解した V^{T} を k 以降の列を削除した行列 V_k^{T} を左からかける。そうすることで A を k 次元に次元圧縮することができる。

4.3 比較実験

今回の実験では比較実験としてサポートベクターマシーン (SVM)、最大エントロピー法、Naive Bayes の 3 つの手法を用いて比較実験を行った。ここではその 3 つの手法の紹介をする。

4.3.1 サポートベクターマシーン (SVM)

SVM は教師ありの機械学習の手法のひとつであり、2 クラスに識別する境界を決定する。未知データに対しての識別の誤差を小さくするために、教師あり学習を通して事前に与えられたデータのマージンを最大化するような境界を求める。マージンとは識別面から最も近いデータとの距離をいい、距離はユークリッド距離によって求められる。

$$y = sign(f(x)) \tag{4.4}$$

$$f(x) = W^{\mathrm{T}}x + b \tag{4.5}$$

$$sign(f(x)) = \begin{cases} 1 & f(x) \geq 0 \\ -1 & f(x) \leq 0 \end{cases} \tag{4.6}$$

マージン $1/||W||$ を最大とするには、制約条件式 4.7 の下、

$$\text{制約条件} : y_i(W^{\mathrm{T}}x + b_i) - 1 \geq 0 \tag{4.7}$$

$$\text{目的関数} : L(W) = \frac{1}{2}|||W||^2 \tag{4.8}$$

目的関数式 4.8 を最小化する W と b を求めればよい。

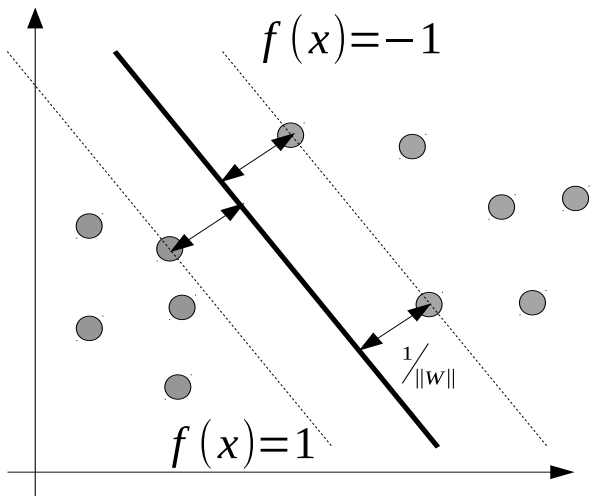

図 4.1: SVM

SVM では完全に線形に分離することが可能であれば適用可能であり、線形分離できない場合にはソフトマージンで境界を決定する。ソフトマージン SVM はこれまでの SVM の条件を緩く設定したもので、誤分類されたものにはペナルティを設けることで誤分類を許容することで対応することができる。

また、SVM は 2 クラスの分類を行うこモデルであり、SVM を多クラスの分類問題へ適用させるには、複数の SVM の分類器を作成する必要がある。

SVM を用いて K クラスの多クラスの分類問題を解く場合、あるクラスとそれ以外のクラスの二値分類器を $K-1$ 個作成して分類を行う一対他分類器と、${}_K\mathrm{C}_2$ 個のクラスの対毎に二値分類器を作成して分類を行う一対一分類器、適当に二値でのクラスの組み合わせを $K-1$〜$K(K-1)/2$ 設け、二値分類器を作成して分類を行う誤り訂正出力符号などのアプローチがある。

こうすることで SVM に限らず二値分類器を多値分類問題への適用をすることができる。

4.3.2 最大エントロピー法

最大エントロピー法は素性関数で表された制約のもと、エントロピーを最大にする手法である。

最大エントロピー法には、文脈述語関数と素性関数が必要である。入力となる文脈の集合を X、出力のクラスの集合を Y として、入力ベクトル $x \in X$ と出力クラスを $y \in Y$ と考えると、素性関数 $f(x,y)$ は x と y はが特定の条件を満たすときとなり、それ以外の場合 0 となる。

$$f_{b,class} = \begin{cases} 1 & b(x) = true, y = class \\ 0 & otherwise \end{cases} \tag{4.9}$$

最大エントロピー法を用いて分類問題を解く際、条件つき確率 $P(y|x)$ を解くことで求めることができる。

$$P(y|x) = \frac{1}{Z(x)} exp(\sum_i \lambda_i f_i(x,y)) = \frac{1}{Z(x)} exp(\lambda^T f(x,y)) \tag{4.10}$$

$$Z(x) = \sum_{y \in Y} exp(\lambda^T f(x,y)) \tag{4.11}$$

最大エントロピー法を用いた分類モデルを最大エントロピー分類器といい、識別モデルの一種である。最大エントロピー分類器の特徴として、素性とクラスの組み合わせが自由で、コーパスが少なくても機能するという特徴がある。

4.3.3 Naive Bayes

Naive Bayes を用いて分類問題を解く場合、文書 D が与えられたとき、クラス C に属する確率 $P(C|D)$ を求めればよい。この確率を求めるには、ベイズの定理を用いる。

ベイズの定理とは

$$P(C|D) = \frac{P(C)P(D|C)}{P(D)} \tag{4.12}$$

$P(C)$ クラス C が得られる確率 (事前分布)、$P(D|C)$ クラス C が与えられた場合に文書 D が得られる確率 (事後分布)

$$P(C) = \frac{\text{クラス } C \text{ と判定された文書数}}{\text{全文書数}} \tag{4.13}$$

$$P(D|C) = \frac{\text{文書 } D \text{ の数}}{\text{クラス } C \text{ に属する文書数}} \tag{4.14}$$

この式 $P(D)$ はクラスに依存しておらずどのクラスでも値は変化しないため、定数として扱うことができるため無視することができる。

文書 D は単語群 $D = \{W_1, W_2, W_3, ..., W_N\}$ で表現することで各単語の確率の席で近似することができる。

$$P(D|C) \simeq P(W_1|C)P(W_2|C)\cdots P(W_N|C) = \prod_i^N P(W_i|C) \tag{4.15}$$

$$P(W_i|C) = \frac{\text{単語}\, W_i\text{の数}}{\text{クラス}\, C\, \text{の語彙数}} \tag{4.16}$$

これらを式 4.12 に代入すると、式 4.17 に示すことができる。

$$P(C|D) = \frac{P(C)P(D|C)}{P(D)} \simeq P(C)\prod_i^N P(W_i|C) \tag{4.17}$$

Naive Bayes を用いて分類を行う場合、分類するクラスは最も確率の高いクラスを選択すればよい。

$$class = \arg\max_C [P(C)\prod_i^N P(W_i|C)] \tag{4.18}$$

Naive Bayes 分類器は式 4.18 を用いて分類を行う分類器であり、事前確率と素性とクラスの組み合わせから素性の貢献度を測ることができ、最大エントロピー分類器と比べて大きなコーパスを必要とする。

4.4 実験結果

本研究での実験結果を表 4.4 に示す。

表 4.4: 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

第5章　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[1] 、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。

[1] 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

第6章　結論

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。

謝辞

本研究を進めるにあたり、多岐にわたりご指導ご協力を頂いた指導教員の新納浩幸准教授には心より感謝申し上げます。

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

[4] 永田純平, 新納浩幸, 佐々木稔, 古宮嘉那子, "文書分類をタスクとした Pylearn2 の Maxout+Dropout の利用", 言語処理学会第 21 回年次大会,to appear., (2015)