

# Web 工学で応用するための Deep Learning 利用法と知見の体系化

2013 年度卒業論文  
東京大学工学部システム創成学科知能社会システムコース  
03120929 黒滝 紘生

指導教官: 松尾 豊 准教授

2013 年 2 月 日

# 概要

近年機械学習の分野において、Deep Learning と呼ばれるアルゴリズム群が優れた成果を納めている。Web 工学でも、Deep Learning を応用することによる発展が期待される。

しかし、Deep Learning は歴史の浅い発展途上の技術であり、どのアルゴリズムを定番とすれば良いのか、試行錯誤の段階にある。これは、各アルゴリズムの改良点が次々と見つかったことに加え、各アルゴリズムの得手不得手や、学習性能が高くなる原理の詳細など、解明されていない部分が多いことが、主な原因である。アルゴリズムが開発途上で確定できていないため、公開されているライブラリも、現状では、開発用途や実験的なものが多くなっている。そもそも有力なアルゴリズムに対応する実装が用意されていない場合や、問題に応じて自らアルゴリズムの細部を調整しなければならない場合もある。標準と言える公開ライブラリが確立していない状況なので、Web 工学など応用分野に Deep Learning を適用したいと考えても、プログラム開発に長い時間がかかってしまい、開発における大きな障壁となっている。

さらに、現在の Deep Learning 技術では、他のアルゴリズムに比べて学習にかかる時間が長いことが多く、ハードウェア性能が低いマシンでは、アルゴリズムを実用的な時間で実行すること自体が容易ではない。実行時間の長さをカバーするため、GPU を用いて演算をスピードアップさせる手法が確立されつつあるが、特殊なプログラミングが要求され、開発における障壁の 1 つとなっている。また、ノート PC の大部分など、並列演算に利用可能な GPU を搭載していない PC を使っている場合には、ライブラリが GPU を利用しているために、却ってその実行が不可能になってしまうこともある。

以上に挙げた原因により、Deep Learning 技術に関心を持っていても、まず実際の問題に Deep Learning を試行すること自体が困難であり、応用技術開発のハードルは更に高くなっている。特に、国内での研究開発は遅れており、早急なキャッチアップが必要である。このような現状を踏まえ、本研究では、Web 工学における応用を見据えつつ、Deep Learning を様々な問題に応用するための方法論を整理する。具体的な目標としては、Deep Learning の特徴である高い学習性能を確実に利用できて、その上で出来る限り、実行時間の短さ、実行プログラムの使いやすさ、アルゴリズムの調整・改良の容易さを兼ね備えた方法を確立する。

# 目次

概要	i
第 1 章 はじめに	1
1.1 Web 工学と機械学習	1
1.2 深層学習の台頭	2
1.3 深層学習の課題と、研究の目的	3
第 2 章 関連研究	7
2.1 Web 工学と機械学習	7
2.2 機械学習で利用される、代表的な分類器	8
第 3 章 深層学習のアルゴリズム	11
3.1 Deep Belief Network	11
3.2 深層学習モデルのバリエーションと詳細	13
第 4 章 深層学習の実装における技術	16
4.1 評価基準	16
4.2 既存ソースコード使用の利点	16
4.3 Graphics Processing Unit の利用による高速化	18
4.4 数値計算ライブラリ Theano の利点	18
4.5 機械学習ライブラリ Pylearn2 の利点	18
4.6 ハードウェアの構成	20
第 5 章 Facebook の顔画像認識と深層学習技術	21
5.1 背景	21
5.2 ソーシャルメディアにおける、プロフィール画像	23
5.3 提案手法	24
5.4 問題設定と、実験方法の詳細	25

5.5	結果 . . . . .	27
<b>第 6 章</b>	<b>Web 工学への深層学習技術の応用</b>	<b>28</b>
6.1	利用したデータセット . . . . .	28
6.2	使用したライブラリ . . . . .	31
6.3	使用したハードウェア . . . . .	31
6.4	ベンチマーク実験の詳細 . . . . .	32
<b>第 7 章</b>	<b>考察と提言</b>	<b>34</b>
7.1	5 章の実験に関する考察 . . . . .	34
7.2	6 章の実験に関する考察 . . . . .	35
7.3	深層学習の利用法に関する提言 . . . . .	37
<b>第 8 章</b>	<b>おわりに</b>	<b>39</b>
8.1	研究の成果 . . . . .	39
8.2	今後の課題 . . . . .	39
<b>謝辞</b>		<b>41</b>
<b>参考文献</b>		<b>42</b>

# 目次

1.1	ディープラーニングで画像を認識する流れ . . . . .	3
1.2	猫を認識するニューロン (教師無し Youtube ビデオより学習された。) . . . . .	4
1.3	上 : 入力画像の中で、ニューロンが最も強く反応した 48 枚 下 : 計算上、最も ニューロンが強く反応する画像 . . . . .	4
3.1	DropConnect の模式図 . . . . .	14
3.2	Maxout Network の構造図 . . . . .	15
3.3	Maxout Unit が凸関数を近似する様子 (1 次元の場合) . . . . .	15
4.1	既存コードの利用における、完全オリジナルコードの作成に対する利点 . . . . .	17
4.2	既存ソースコード利用のリスクと、対処方法 . . . . .	17
4.3	Pylearn2 の実験計画例 . . . . .	19
5.1	ユーザ属性推定の必要性 . . . . .	22
5.2	Facebook におけるプロフィール画像の例 . . . . .	23
5.3	提案手法 . . . . .	24
5.4	今回の実験の模式図 . . . . .	26
5.5	実験で使った、画像から入力ベクトルへの変換方法 . . . . .	26
5.6	プロフィール画像からの顔画像抽出方法 . . . . .	26
5.7	実験で学習されたフィルター . . . . .	27
6.1	MNIST データの例 . . . . .	29
6.2	CIFAR10 データの例 . . . . .	31
7.1	Maxout Network による MNIST 分類実験で、誤差が論文より大きくなった原因 . . . . .	36

# 表目次

5.1	Facebook 顔画像の分類結果 . . . . .	27
6.1	MNIST の分類誤差ランキング . . . . .	30
6.2	CIFAR10 分類誤差のランキング . . . . .	31
6.3	ハードウェア性能の比較 . . . . .	32
6.4	Maxout Network による MNIST(2 次元) 分類の結果 . . . . .	32
6.5	Maxout Network による MNIST(1 次元) 分類の結果 . . . . .	33
6.6	Maxout Network による MNIST(1 次元) 分類の実験詳細 . . . . .	33



# 第 1 章

## はじめに

### 1.1 Web 工学と機械学習

近年 Web 工学の領域においては、広告の効率アップや文章の分析、検索エンジンの性能向上などが求められている。このため、ユーザの性格や行動の分析、ネットワーク構造の未来予測、文章の意味や感情の理解といった技術が必要とされている。

例えば、Web 上で物を販売しているサイトにおいて、ユーザが購入しそうな商品を勧め、売り上げを増大させることを考える。ユーザが今まで見た商品、購入した商品を分析して、ユーザの求めている商品のジャンルを知ったり、Web ショッピングにおける性格を知ることが出来れば、ユーザが買いたい商品を先回りして、広告表示することが出来る。ユーザに対して、より購入してくれそうな広告を多く見せて、広告の効率をアップさせることで、サイトからの収益が上がる事が期待できる。

Web 工学では、Web 上のデータを元に知識を学習し、次に得た知識を用いて予測や分類を行う、というアルゴリズムが必要になっている。例えば、ユーザ行動やソーシャルネットワーク構造の傾向を学習して、将来の動向を予測したり、ユーザに送られたメールの意味を理解できるようになって、迷惑メールとそうでないものを分類できるようになる、といった具合である。こういった知識学習のためには、機械学習 (マシンラーニング) と呼ばれる手法が広く用いられている。

機械学習とは、対象となる知識を数学的なモデルで表し、より適切なモデルを求めて微修正を重ねることで、知識を洗練させていくアプローチである。使用する数学的なモデルの大筋は、経験を基にあらかじめ決めておく。ユーザ行動、リンク構造、メールの文章といった元データは、全て数値に変換されて、モデルに入力される。モデルは、入力データから予測や識別を実際に行いつつ、より良い結果が得られるように、モデル内の係数を調整していく。最終的に、多種多様なデータに対して、適切な予測や識別を行うことが出来る数学的モデルを得ることが出来るのである。



## 1.2 深層学習の台頭

機械学習における大きなポイントの1つに、元データをどのような数値データに変換するか、という問題がある。

変換された数値データのことを、特徴量、あるいは素性 (feature) と呼ぶ。機械学習は、「生データから素性への変換」「素性をモデルに入力」「モデルによる数値計算で、結果を出力」「結果がより正確になるよう、モデルを修正」というプロセスの繰り返しで成り立っている。このうち、「素性への変換」だけは、画像・文章・リンク構造といった、データの種類の依存している。いったん素性が数値の形で得られれば、残りのプロセスは、データの種類の依存せず、全て入力数値とモデルの関係だけで解決することができる。つまり、機械学習のプロセスは、データの種類の依存する素性変換と、汎用的に使えるモデル学習の部分に分かれているのである。

機械学習の性能を上げるためには、素性への変換部分を工夫する方法と、使用する数学的モデルを洗練する方法の2つがある。素性への変換部分の改良は、対象となるデータの種類の強く依存している。例えば、画像データを素性数値に変換する場合、単純な RGB 画素データをそのまま使っても良いが、SIFT 特徴量や SURF 特徴量、フィッシャーベクトルなど、より画像の特徴を捉えた特徴量を用いることが定石となっている。音声データに対しては、時間領域や周波数領域の波形をそのまま用いても良いが、メル周波数ケプストラム係数と呼ばれる特徴量も用いられる。問題とデータ種別に応じて、利用する特徴量を工夫することにより、機械学習の識別精度や実行時間などの性能が向上することが知られていた。このとき、特徴量はそれぞれのデータの専門家による、謂わば職人芸によって作られていた。

2000 年代半ばに、Deep Learning と呼ばれる一群の手法が台頭した。Deep Learning とは機械学習の手法の1つで、人間の脳の構造に類似した、多層ニューラルネットワーク構造をモデルに用いて、学習を行わせる方法である。Deep Learning は、画像認識や音声認識、化合物の生成予測といった分野で優れた性能を示し、注目を浴びるようになった。図 1.1 は、多層ニューラルネットワークが人間の脳の思考回路である、神経細胞のつながりを模倣する様子を表している。(日経ビジネス 2013 年 4 月号より一部抜粋)

Deep Learning の特徴の一つとして、生の入力データから自動的に素性を作り、抽象表現を習得する働きがあると考えられている。例えば人間の顔画像データを学習させた場合、多層ニューラルネットワークを構成する複数のレイヤーのうち、入力に近い低レイヤーのニューロンは、画像のエッジ部分に対して強く反応し、出力に近い高レイヤーのニューロンでは、目口鼻、さらに顔全体など、より抽象度の高い要素に対して反応することがわかった。これは、見方を変えれば、低レイヤーでは中間表現 (素性に当たる) を抽出しており、低レイヤーで得た素性を高レイヤーに入力することにより、

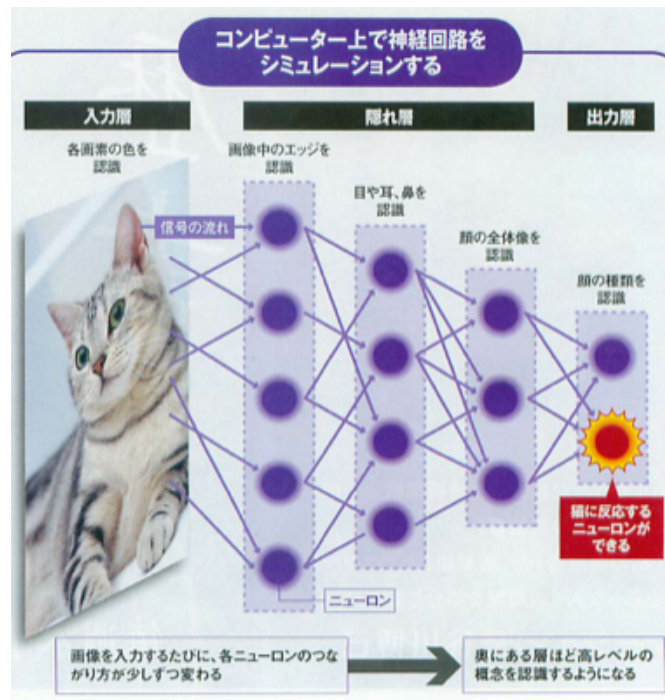


図 1.1 ディープラーニングで画像を認識する流れ

より高レベルな抽象的概念に対しても優れた識別精度を実現している、と考えられる。どのような素性を使えば良い結果が出るのか、素性の抽出方法自体を同時に機械学習していることから、Deep Learning は Representation Learning(表現学習) とも呼ばれている。

Google の Deep Learning 研究グループは 2012 年、Youtube のビデオによる学習を長時間行わせることで、ニューラルネットワークの 1 つのニューロンが、猫の画像 (1.2、Google の研究紹介記事より引用<sup>\*1</sup>) を認識するようになったと発表し、大きな話題を呼んだ [23]。この Youtube ビデオ学習の間、「この画像は猫という名前のものである」「猫を認識できるよう学習を行いなさい」といった、認識ラベルの付け方における指針は一切学習器に与えられなかった (教師無し学習)。それにも関わらず、多層ニューラルネットワークは、自然と「この (我々の言葉でいう猫の) 画像は覚えておくべきである」というように、何を認識すると学習効率が良いのか、何を素性とすれば学習がしやすいのか、という情報自体を学習することが出来た。また、同じ論文にて、この多層ニューラルネットワークが人間の顔を学習できることも示された (1.3、[23] より引用)。

### 1.3 深層学習の課題と、研究の目的

Deep Learning が高い識別性能を持つことがわかり、Deep Learning を身近な問題に適用して、良い成果を得たいという機運が高まっている。例えば、Web 工学の分野では機械学習が大きな役割を

<sup>\*1</sup> <http://googleblog.blogspot.jp/2012/06/using-large-scale-brain-simulations-for.html>



図 1.2 猫を認識するニューロン (教師無し Youtube ビデオより学習された。)

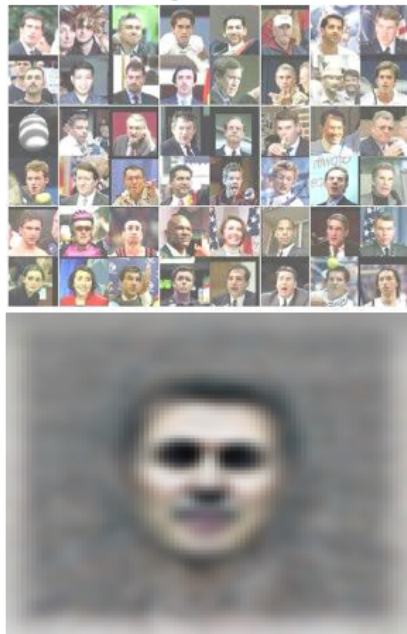


図 1.3 上：入力画像の中で、ニューロンが最も強く反応した 48 枚 下：計算上、最もニューロンが強く反応する画像

果たしており、この学習プロセスに Deep Learning を組みこむことで、学習精度が向上したり、より多様な情報を扱えるようになる可能性がある。出来るだけ簡易に、Deep Learning を様々な問題に応用するための方法論が求められている。

しかし、Deep Learning は歴史の浅い発展途上の技術であり、どのアルゴリズムを定番とすれば良いのか、試行錯誤の段階にある。これは、各アルゴリズムの改良点が次々と見つかったことに加え、学習性能が高くなる原理や、各アルゴリズムの得手不得手など、解明されていない部分が多いことが、主な原因である。アルゴリズムが開発途上で確定できていないため、公開されているライブラリも、現状では、開発用途や実験的なものが多くなってしまっている。実験的なライブラリでは、一部の種類のデータにのみ適用されることを想定して書いている場合があり、他の種類のデータを扱うためには、データ変換用のソースコードを記述しなければならないケースが多い。そもそも有力なア

ルゴリズムに対応する実装が公開されていない場合もあり、この場合、アルゴリズムの部分も含めて全ての実装を用意しなければならない。また、問題に応じて自らアルゴリズムの細部を調整しなければならない場合もある。例えば、学習の繰り返し回数 (エポック数) や、どの種類のレイヤーを何回重ねるべきか (レイヤー構造)、1 つのレイヤーに含まれる学習素子 (ニューロン) の個数はどうか、などである。これらはソースコードの作成者が経験的に手作業で調整しているケースが多く、標準と言える公開ライブラリが確立していない状況なので、Web 工学など応用分野に Deep Learning を適用したいと考えても、プログラム開発に長い時間がかかってしまう。開発における大きな障壁となっている。

さらに、現在の Deep Learning 技術では、他のアルゴリズムに比べて学習にかかる時間が長いことが多く、ハードウェア性能が低いマシンでは、アルゴリズムを実用的な時間で実行すること自体が容易ではない。実行時間の長さをカバーするため、GPU を用いて演算をスピードアップさせる手法が確立されつつあるが、特殊なプログラミングが要求され、開発における障壁の 1 つとなっている。また、ノート PC の大部分など、並列演算に利用可能な GPU を搭載していない PC を使っている場合には、ライブラリが GPU を利用しているために、却ってその実行が不可能になってしまうこともある。

以上に挙げた原因により、Deep Learning 技術に関心を持っていても、まず実際の問題に Deep Learning を試行すること自体が困難であり、応用技術開発のハードルは更に高くなっている。特に、国内での研究開発は遅れており、早急なキャッチアップが必要である。

本研究では、このような現状を踏まえ、Deep Learning 技術を実際の問題に応用するための方法論を確立すると共に、実装における障壁を出来る限り取り除くことを目指す。特に Web 工学における応用を目標とする。

2 章では、従来の Web 工学における機械学習の応用方法と、Deep Learning 出現以前に良く用いられていた、機械学習による識別器学習方法について、俯瞰する。

3 章と 4 章では、現在の Deep Learning 技術について紹介する。3 章では、Deep Learning を構成する個々のアルゴリズムについて、その原理と詳細を述べる。4 章では、実際に Deep Learning アルゴリズムを利用する上で、どのように実装を進めれば、Deep Learning の特徴である高い学習性能を確実に利用できるのか、また、現実の問題を解決するときに重要となる、実行時間の短さ、実行プログラムの使いやすさ、アルゴリズムの調整・改良の容易さなどは、どのようにすれば確保できるのか、といった Deep Learning を使っていく上でのノウハウを記述する。

5 章と 6 章では、4 章までの内容を実際の問題に応用することで、その有用性を確かめる。5 章では、ソーシャルメディア上のプロフィール画像を分類させるタスクに、Deep Learning を応用する。6 章では、Deep Learning のプログラムを、汎用的なベンチマークにかけることで、その性能を測

定・比較する。5章と6章の結果を受けて、7章では Deep Learning を利用していく上でのベストプラクティスを提案する。

## 第 2 章

# 関連研究

### 2.1 Web 工学と機械学習

この節では、Web 工学における課題をいくつか例示し、それらの課題を解決するために、機械学習がどのように用いられてきたかを概観する。

#### 2.1.1 Recommendation System

主に Web ショッピングを含むサイトや、Web 広告の配信において求められる技術である。Web サイトを閲覧しているユーザに対し、そのユーザが購入したいと商品を予測して、Web 上の広告などの形で推薦する。適切な広告を表示することにより、ユーザの購買行動を促進することが出来る。Recommendation System の研究は、1992 年の Tapestry システムに始まる [10]。また、少なくとも 90 年代の終わりには、Amazon.com, CDNOW, eBay, Levis, GroupLens など、様々なサイトにて Recommendation System は利用されていた [32]。Recommendation System を実現するための、機械学習のテクニックとしては、大きく 2 種類が挙げられる [18]。1 つは、Content Filtering と呼ばれ、ユーザや商品の属性や購買傾向を学習することことで、推薦を行う。もう 1 つは Collaborative Filtering と呼ばれており、ユーザや商品の属性を扱う代わりに、購入や評価といった、ユーザの過去の行動を基にして、推薦を行う。

#### 2.1.2 Link Prediction

グラフにおいて、現在のノード間接続から未来の接続を予測するソーシャルグラフの予測、タンパク質の反応予測 (PPI) に有効である [加筆]

#### 2.1.3 Sentiment Analysis

ユーザの感想データを手に入れられるようになってきた  
事実だけでなく、他の人がどのような感情を抱いているのかを分析したい

opinion mining という語も広義には似た内容を指している

2001-2003 に研究が開始された

common neighbors

Jaccard 係数 SVD による次元削減

Stanford によるデモに言及

## 2.1.4 Learning to Rank

多様なランキング素因を組み合わせ、ランキング関数を作成

→どの組み合わせが有効か、機械学習する

## 2.2 機械学習で利用される、代表的な分類器

機械学習のプロセスは、「入力データを、数学的モデルで使える素性に変換する」「素性を数学的モデルに入力して、出力値を得る」「出力を見ながら、モデルを修正する」という行程に大きく分けられる。データの分類問題を機械学習で解く場合、モデルによる出力値が分類結果に対応するよう、モデルを学習させることになる。この場合、モデルのことを分類器とも呼ぶ。

機械学習において、素性への変換部分は、データの種類の大きく依存する。一方、分類器に用いる数学的モデルと、モデルの改修法、つまり学習法は、汎用的に使うことができる。あるいは、画像や音声、文章といったデータの多様性を、素性という一般的な数値に落とし込むことで吸収して、汎用的分類モデルでも学習できるようにしている。

Deep Learning、あるいは Deep Neural Network(多層ニューラルネットワーク) は、汎用的分類モデルの一種である。ここでは、Deep Learning の他にどのような分類器が存在するのか、代表的なものを述べる。

### 2.2.1 Support Vector Machine

Support Vector Machine(SVM) は、データを 2 つのクラスに分類する能力を持っている。

カーネルマジックとマージン最大化により、優れた精度を出している。SVM は、画像認識などに既に実用化されている。

SVM は、広くその信頼性が認められたモデルの 1 つであり、ライブラリの利用方法も確立している。例えば、libsvm<sup>\*1</sup>や liblinear<sup>\*2</sup>は使用が容易なライブラリとしてよく知られている。これらのライブラリを使うと、簡単な所定の方式に沿って入力データファイルを用意し、CUI 上で 2,3 回の操作

<sup>\*1</sup> <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

<sup>\*2</sup> <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

をするだけで、SVM による分類を行わせることができる。このとき、利用者が自分でプログラムを書く必要は全くない。プログラムを書かないで済むと、手軽に利用することができ、またバグを起こす危険性が非常に少なく安全に使うことができる。

Deep Learning については、このようなライブラリはまだ存在していないため、Deep Learning の代表的なアルゴリズムについて、プログラム無しで利用できるようなライブラリの整備が望まれる。

## 2.2.2 ニューラルネットワーク

ニューラルネットワークは、人間の脳の構造を模倣した数学的モデルである。人間の脳は、ニューロンと呼ばれる神経細胞が大量に接続されて出来ている。ニューロンが電気信号を伝達することで、様々な脳の働きが行われている、と考えられている。プログラム上で表現されるニューラルネットワークには、人間の脳の動作との相違点もあり、脳が行っている計算を正しくシミュレートしているとは言い難い面もある [44] が、機械学習のモデルとしては広く使われてきた。

### ニューラルネットワークの伝達方式

以後、機械学習におけるニューラルネットワークのニューロンを、慣例に従ってユニットと呼ぶことにする。ニューラルネットワークでは、ユニットからユニットへの接続が網目のように広がっている。ユニットは入出力の機能を持つ。これは、生体におけるニューロンが、他のニューロンから化学的な刺激を受け取り、受け取った刺激に応じて自らも他のニューロンを刺激する構造を真似ている。ユニットは入力を受け取ると、活性化関数 (activation function) を使って入力値を変換し、接続先のユニットへ出力する。このとき、ユニットからユニットへの結線に、重み (weight) と呼ばれる係数を付与して、出力を変化させることが普通である。これは、生体のニューロン同士の接続が、脳の学習につれて強固になり、刺激がより伝わりやすくなっていくことに対応させている。

### パーセプトロン

パーセプトロンとは、ニューラルネットワークの一種である。複数のユニットをまとめた層 (レイヤー) が、いくつか重なって出来ており、値の伝達は入力レイヤーから出力レイヤーへの一方向に限られているものを指す (feed-forward)。狭義には、単層かつ活性化関数にヘビサイド関数を用いた 2 クラス分類モデルのみを指すこともある。

始めに提唱されたのも、入力層と出力層の 2 層から成る、単層パーセプトロンと呼ばれるモデルだった [33]。このモデルは、後に線形関数しか近似できないことがわかり、いったん下火になった [27]。例えば、単層パーセプトロンでは、非線形関数である XOR 関数を学習させることが出来なかった。

しかし、隠れ層を追加し、活性化関数にシグモイド関数などの非線形関数を用い、さらにバック



プロパゲーションという方法で学習を行わせることにより、非線形関数を近似可能となることがわかり、再び有用な識別モデルとして脚光を浴びた [34]。これを単層パーセプトロンと区別して、多層パーセプトロン (Multi Layer Perceptron, 以下 MLP) とも呼ぶ。このモデルは 2 クラス分類モデルの範囲を逸脱しており、元々のパーセプトロンとはやや異なるものだが、慣例的にこのように呼ばれている。

バックプロパゲーションは教師有り学習の一種である。出力層におけるモデルが出した推定値と、教師データの結果が異なっている場合、教師データからの推定誤差 (error) が減少するように、モデルのパラメータを修正する。MLP の場合、モデルのパラメータとはユニット間の伝達にかかる重み係数のことである。パラメータの修正値の仕方にはバリエーションがあるが、最もよく使われるのは Stochastic Gradient Descent (以下 SGD) と呼ばれる方法である。これは確率変数に拡張された一種の再急降下法である。直感的に言えば、微分によって、その場で推定誤差が最も急速に減少するパラメータの修正方向を算出し、その方向に向かってパラメータを変化させる。SGD を少し変化させ、複数のデータによる誤差を一度に処理する Batch Gradient Descent (BGD) や、SGD の 1 次精度に対し、2 次精度までの情報を使う共役勾配法 (Conjugate Gradient Method, CG) も存在する。

バックプロパゲーションを行うためには、活性化関数が微分できることが重要である。シグモイド関数は、元々使われていたヘビサイド関数に形が似ている上に、微分が容易という点でバックプロパゲーションとの親和性が高く、有利である。しかし、シグモイド関数のデメリットとして、入力と重みの積が大きくなるにつれて、誤差への反応が小さくなってしまい、学習の進行が遅くなるという問題を抱えていた。この問題は、特に隠れ層を 2 層以上にしたとき顕著になった。このため、多層ニューラルネットワークを学習させる方法は長い間課題となっていた。

## 第 3 章

# 深層学習のアルゴリズム

Deep Learning は、多層ニューラルネットワーク (MLP) のうち、普通隠れ層が 2 つ以上のものという。2006 年に Hinton らによって、このような Deep な構造を学習させる方法が発見され、発展への道が開けた [12, 13]。従来よりも多くのレイヤーを扱うことで、より複雑な関数を学習できるようになった。Deep Learning は、数々の分類タスクにて、従来手法を大きくしのぐ成果を収め、注目を浴びている。例えば第 1 章で触れたように、2012 年には、Google が Youtube ビデオの静止画を使って多層ニューラルネットワークの教師なし学習を行わせ、結果として、猫を認識するユニットや人を認識するユニットを学習させることに成功したと共に、ImageNet Large Scale Visual Recognition Competition(ILSVRC) というデータセット [8] の分類にて、state of the art の結果を残した。この章では、Deep Learning で使われているアルゴリズムの詳細について述べる。

### 3.1 Deep Belief Network

最初に MLP の学習のブレイクスルーとなったのは、2006 年の Hinton らによる Deep Belief Nets(DBN)[12, 13] である。このモデルについて解説する。

#### 3.1.1 unsupervised pretraining と finetuning

この研究では、MLP の重みをランダムに初期化した後、すぐバックプロパゲーション学習にかけるのではなく、各レイヤーの重みをあらかじめ教師無し (unsupervised) 学習で調整して、隠れ層が効率の良い素性を学習できるよう仕込んでおくというアイデアが使われた。この事前学習のことを、pretraining と呼び、バックプロパゲーションにて学習する段階の方は、finetuning と呼ぶ。pretraining の段階では、入力側から 1 レイヤーずつ学習を行い、学習済のレイヤーは固定して動かさない (Greedy Layer-wise pretraining)。finetuning の段階では、全てのレイヤーをバックプロパゲーションにて同時に変化させていく。

unsupervised pretraining をどのような基準で行うかが問題だが、この研究では、Restricted

Boltzman Machine(以下 RBM) というモデルを用いている。RBM とは、ニューラルネットワークの一種で、可視変数層と隠れ変数層の 2 つのレイヤーから成り立っている。DBN においては、可視変数=入力データと考えておけばよい。RBM モデルの学習を進めることで、入力データ→隠れ変数への推論モデルを得ることができる。これが Deep Learning の最大の特徴の 1 つである、「抽象度の高い素性への変換方法自体を学習する」ことに対応している。DBN では、RBM を 1 レイヤーずつ学習させながら、RBM 同士を「前のレイヤーの隠れ層 = 次のレイヤーの入力層」となるようにつなげていく。これにより「入力層→抽象度の低いレイヤー→... →抽象度の高いレイヤー→出力」という、複数の隠れ層をもつ Deep な構造を作り、上手く pretraining させることに成功している。RBM を積み重ねた最も上のレイヤーに、普通の線形レイヤーを 1 つ置いて、「最も抽象度の高い素性→出力値」の推測を行わせることが多い。なお、finetuning におけるバックプロパゲーションでは、RBM にあった「隠れ層→入力層」の方のつながりは、最終レイヤーにおける誤差に影響しない。よって、以前からある feed-forward な MLP と同様に学習させることができる。

### 3.1.2 Restricted Boltzman Machine

DBN の pretraining に用いられる Restricted Boltzman Machine[38] とは、ニューラルネットワークを用いた生成モデルの一種である。まず生成モデルについて説明する。

クラス分類問題を確率的アプローチで解く方法は、生成モデル (generative model) と識別モデル (判別モデル、discriminative model) の 2 つに大別できる [?]\*<sup>1</sup>。クラス分類問題では、まずクラス事後確率  $p(C_k|x)$  を各クラス  $k$  に対して求める。基本的には入力  $x$  に対して最もクラス事後確率が大きくなったクラスに分類することになる\*<sup>2</sup>。識別モデルでは、クラス事後確率  $p(C_k|x)$  のみを直接学習するのに対し、生成モデルでは、まずクラスの条件付き密度  $p(x|C_k)$  を学習する。これと、別途求めたクラスの事前確率  $p(C_k)$  を用いることで、クラス事後確率は

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{p(x|C_k)p(C_k)}{\sum_k p(x|C_k)p(C_k)} \quad (3.1)$$

によって求められる。これは、入力とクラスの同時確率  $p(C_k, x)$  を求めることと等価である。生成モデルの利点は、クラス分類のモデルだけでなく、入力データの性質に関する情報をも同時に得られる点にある。しかし、一般に識別モデルよりも複雑で困難な問題を解く必要が生じるため、常に生成モデルが用いられるわけではない。

さて、前述したように、RBM は可視層と隠れ層の 2 レイヤーで構成されるニューラルネットワークである。入力層の各ユニットの値を  $v_j$ 、隠れ層の各ユニットの値を  $h_i$  とおく。これらをまとめた

\*<sup>1</sup> 確率モデルを介さず、入力からクラスを得る識別関数を直接導くことも出来る。

\*<sup>2</sup> 医療における誤診断など、誤分類のタイプに応じて重要度が異なる場合、損失関数 (loss function) を導入することもある。

ベクトルをそれぞれ  $\mathbf{v} = \{v_1, v_2, \dots\}$ ,  $\mathbf{h} = \{h_1, h_2, \dots\}$  とおく。これは、例えば  $\mathbf{v}$  が画像や音声などの入力データをベクトル化したもので、 $\mathbf{h}$  がこれらの変数の関係を説明する変数だと考えれば良い。ただし、 $\mathbf{h}$  が即  $\mathbf{v}$  のクラス分類に対応するとは限らないことに注意する。RBM の学習を進めると、 $p(\mathbf{v}|\mathbf{h})$  と  $p(\mathbf{h}|\mathbf{v})$  との、2 方向の条件付き確率のモデルが同時に得られる。

RBM の学習は、自由エネルギーと呼ばれる従属変数を最小に行われる。ここでは、最も簡単かつ重要な、 $v_j, h_i \in \{0, 1\}$  のケースについて述べる。 $W$  を  $\mathbf{v}$  と  $\mathbf{h}$  の間の重み、 $b$  と  $c$  を、それぞれ  $\mathbf{v}$  と  $\mathbf{h}$  に対応するバイアス項とする。このとき、RBM における自由エネルギー  $F(v)$  は、

$$F(v) = -b'v - \sum_i \log(1 + e^{(c_i + W_i v)}) \quad (3.2)$$

となる。この自由エネルギーを、SGD などのアルゴリズムを用いて最小化することで、目標となる生成モデルを得ることができる。なお、自由エネルギーは Energy-Based Model と呼ばれるアルゴリズム群

### 3.1.3 DBN の成果

この研究では、評価実験として MNIST の手書き数字認識データセットを用いている。彼らは各レイヤーのユニット数が”784-500-500-2000-10”という構造の DBN によって、学習を行った。そして、Permutation invariant と呼ばれるタスクにて、当時の state of the art を塗り替えたことにより、注目された。ここで、Permutation invariant とは、データを単なる 1 次元の値の集まりと見なし、2 次元の画像という事前情報の利用を禁止した状態で、分類実験を行うことである。具体的には、画像的変形による水増しや、2 次元の畳み込み (Convolutional Layer の節にて詳述) などが当てはまる。Permutation invariant の制約下で良い成果を出した DBN は、画像の性質を利用した Convolutional Net に比べたとき、画像以外の 1 次元のデータに対しても適用しやすいアルゴリズムだと考えられる。

## 3.2 深層学習モデルのバリエーションと詳細

この節では、Deep Learning のアルゴリズムが、どのような要素の組み合わせで出来ているのか、紹介する。2 章の最後で紹介した多層パーセプトロン (MLP) に、どのような工夫を加えることで学習が可能になったのか、使われている技術を列挙する。

### Multi Layer Perceptron (MLP)

後述する Maxout Network では、単純な MLP 構造を用いながら、いくつかのテクニックによって精度良い学習を実現している。

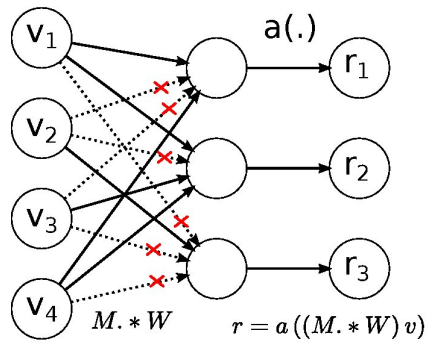


図 3.1 DropConnect の模式図

(MLP, SDA, DBM, CNN)

### 3.2.1 Dropout と robustness の確保

Dropout とは、2012 年に Hinton 氏らによって提案された、過学習を防ぐための技術である [14]。過学習とは、機械学習全般において使われる用語で、学習モデルが訓練中に見たデータへの適応を重視し過ぎたために、未知のデータをうまく識別できなくなる現象のことを指す。Dropout では、過学習を防ぐため、各レイヤーからの出力をランダムに消去してしまう。これによって各ユニットには、他のユニットに頼らず自力で学習をする必要が生じ、より様々な入力に対応しやすい堅固 (robust) なモデルが獲得される、と考えられている。

DropConnect は、Dropout をさらに一般化した手法である。DropConnect では、各ユニットからの出力ではなく、ユニット同士のつながりをランダムにカットしている。図 3.1 は、DropConnect の著者による紹介 Web ページ<sup>\*3</sup>に掲載されている画像の引用である。ニューラルネットワークにおけるニューロン同士の接続が、ランダムに切断されている様子を表している。

### 3.2.2 活性化関数

各ユニットに入力された値は、ある一定の関数によって増幅変換される。これは人間の神経細胞を真似た仕組みで、神経細胞の化学物質による活性化にちなんで、活性化関数 (activation function) と呼ばれる。この項では、ニューラルネットにおける様々な活性化関数を紹介する。

#### sigmoid と hyperbolic tangent

シグモイド関数は、以下の式で表される。

<sup>\*3</sup> <http://cs.nyu.edu/~wanli/dropc/>

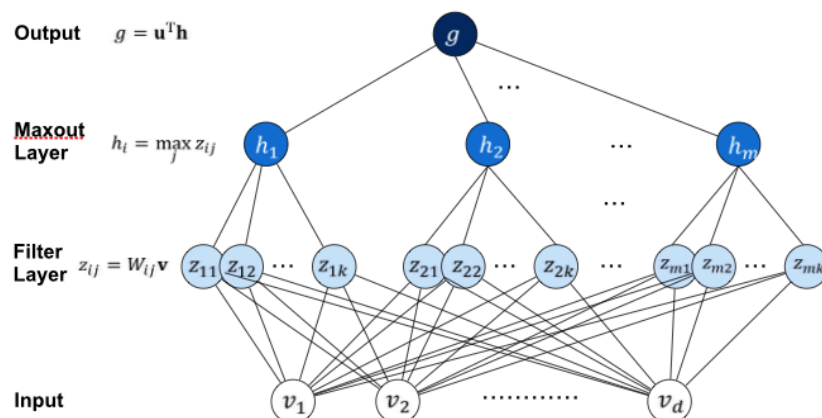


図 3.2 Maxout Network の構造図

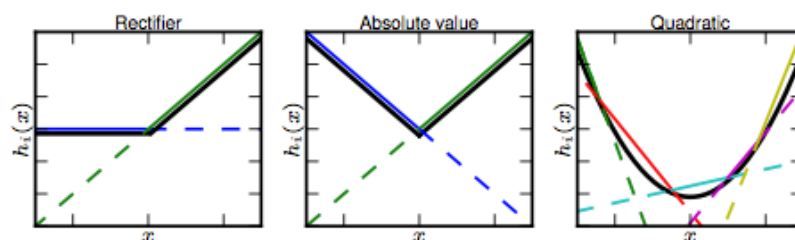


図 3.3 Maxout Unit が凸関数を近似する様子 (1次元の場合)

rectifier

Maxout Unit

Maxout Network とは、バックプロパゲーションを伴う単純な多層 MLP 構造に、3.2.1 で述べた Dropout 法を組み合わせ、さらに Maxout Unit という特殊な活性化関数を併用したニューラルネットワークである [11]。Maxout Network の 1 レイヤー分の構造を、図 3.2 に示す。

Maxout Unit では、複数の線形ユニットの出力について、max を取っている。これにより、任意の凸関数を近似することが出来る。図 3.3 は、Maxout Unit が凸関数を近似する様子を、1次元の入力について示している。そして、複数の maxout unit の線形和を取ることは、複数の近似された凸関数の線形和を取ることである。これによって、最終的に任意の連続関数を近似できることが証明されている。

## 第 4 章

# 深層学習の実装における技術

この章では、Deep Learning のアルゴリズムを実装して、実際の問題に適用するに当たって、どのようにすれば良い結果が得られるか、紹介する。

### 4.1 評価基準

Deep Learning のアルゴリズムを使用するための具体的な手段を選ぶにあたり、次のような判断基準を設けた。優先順位は、1 → 2 → 3 の順に高い。基準 1. Deep Learning が注目された大きな理由である、高い識別精度を再現できる。基準 2. 学習にかかる時間が、他のアルゴリズムに比べて、極端に長くない。基準 3. 利用にあたって必要なプログラミング量が出来るだけ少なく、バグが混入しにくい。基準 1 は最優先目標である。今回 Deep Learning を選択した理由は、Web 工学のタスクにおいて、高い分類精度を実現するための最も有力な方法だと思われるからである。つまり、例えば Deep Learning のアルゴリズムとして正しいプログラムだったとしても、分類精度が従来の分類器より劣っていれば意味はない。従来の分類器をそのまま使い続ければよいことになってしまう。基準 2 と 3 は、現実の問題を Deep Learning で解決する際に、必要となってくる視点である。プログラムの作成や、学習の実行にかかる時間が、あまりにも長くなってしまうと、実際のビジネスや、刻一刻と変化する Web サービスに対して応用するのは非現実的だろう。

### 4.2 既存ソースコード使用の利点

Deep Learning のプログラムを実行するために、まず大きく分けて、「自分でソースコードを全て書く」方法と、「主に既存のソースコードを利用する」方法の 2 つが考えられる。今回は、既存ソースコードをベースに使うことを選択した。以下、その理由を説明する (図 4.1 参考)。

「既存のソースコードを利用する」場合のメリットとして、「開発期間は基本的にゼロで済む」「新たにバグが混入する危険性が無い」「自分の改造コードが、他のライブラリ利用者によって使ってもらえるチャンスが大きい」という点が挙げられる。デメリットとして、「ソースコード中にブラック

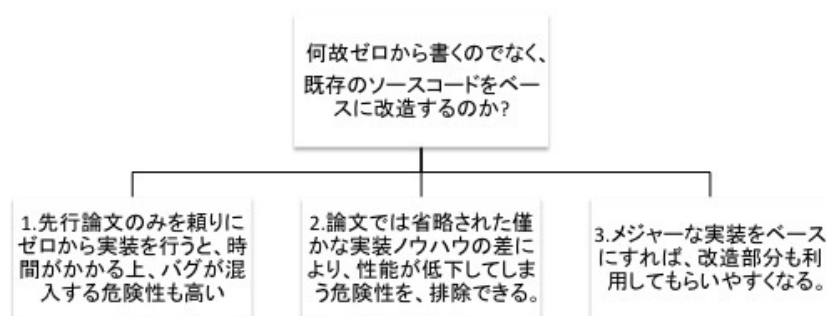


図 4.1 既存コードの利用における、完全オリジナルコードの作成に対する利点

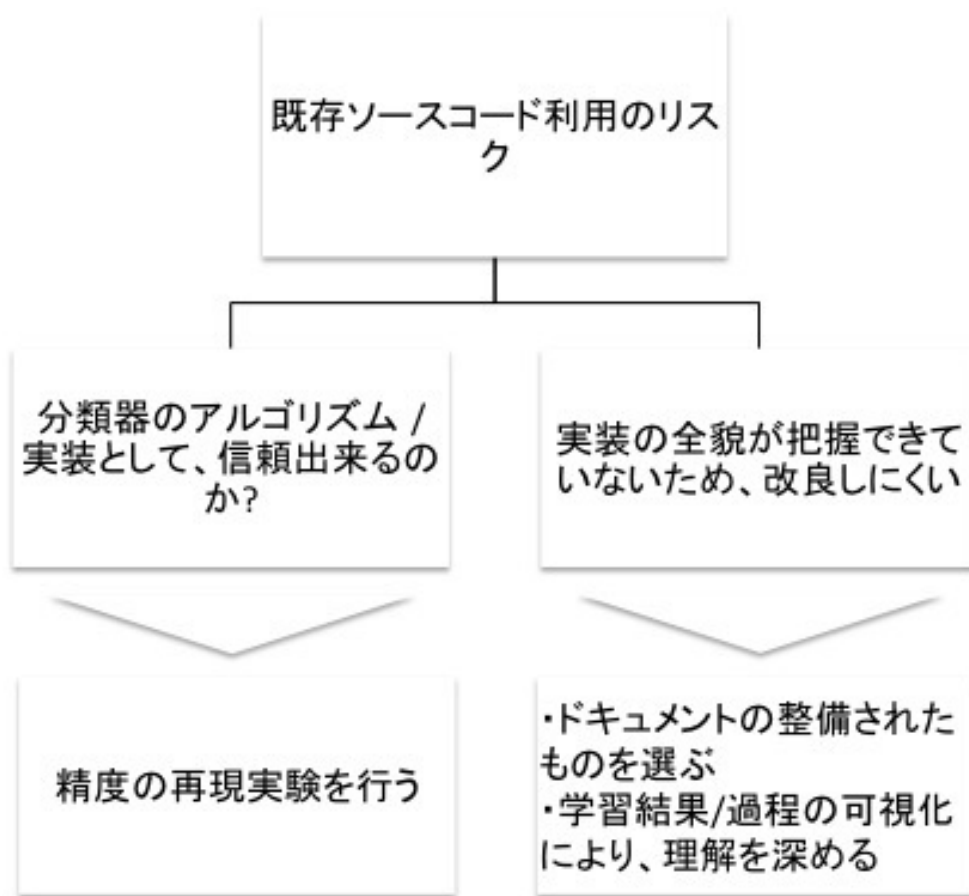


図 4.2 既存ソースコード利用のリスクと、対処方法

ボックスが増え、改造にかかる時間が短くなる」「全く新しいアルゴリズムを実装する場合、ゼロからスタートの方が早く書けるケースもある」などが想定される。しかし、基準3の「プログラミング量が少なくバグのリスクが低い」という点において優れているため、今回は既存のソースコードを探して利用していくことにした。既存ソースコードを利用する際に生じるリスクについては、図 4.2 のように対処できる。

なお、「自分でソースコードを全て書く」場合のメリットとデメリットは、上記「既存のソースコード」の場合の逆となる。自分で新たなコードを書くので時間がかかり、バグの混入リスクも大きい。



また、ソースコードを公開した場合の、ライブラリとしての信頼度も、ゼロから築かなければならない。ただし、コードの詳細部分を改造する段階では、自分の手で書いたコードを使う方が、より深い理解を得やすく、確実に素早い実装ができる可能性もある。

### 4.3 Graphics Processing Unit の利用による高速化

一般に、Deep Learning の研究においては、GPGPU による並列計算が有効とされている。並列計算のプログラム構成にもよるが、100 倍近く早くなることもある。また、Deep Learning の実装によっては、はじめから GPU で高速化することを前提に、GPU 専用のコードを書いている場合があるこの場合、そもそも GPU を搭載したマシンを使わないと、コンパイルや実行が全く出来なくなってしまう。

CPU の計算能力は頭打ちになっている。GPU は many core による高速演算ができる。NVIDIA は CUDA という GPU による演算の開発環境を開発した。

### 4.4 数値計算ライブラリ Theano の利点

Theano は、python 上で記述される数式処理/数値計算ライブラリである。Theano は、数式のコンパイルと実際のデータによる数値計算の 2 段階で動作する。数式は文字式で記述される。このときデバッグの難しい数値計算の誤差絡みの部分、0div など危険な処理を、自動的に解析してくれる。ニューラルネットワークの演算過程では、非常に小さい数値が出現することがあり、また、文字式を分析することで、計算グラフも最適化してくれる。プログラマは、プログラム上の些末な計算テクニックに囚われることなく、数式の本質的な部分の記述に集中することができる。また、微分も自動で行ってくれる。ニューラルネットワークでは、様々な文字の微分が必要になり、多層にすると爆発的に式が複雑になる。これを手計算による文字式の微分をしてから書いていくと、層を増やす度に手計算が必要になり、非常に煩雑で、開発効率が落ちてしまう。Theano の自動微分と自動最適化は、pylearn2 の内部記述の簡略化を大いに助けているおいて、多層のニューラルネットワークを使うときでも、文字式に対する大量の手計算と、大量のプログラムの式を書くことなく、簡単に微分やニューラルネットワークを拡張することができる。

### 4.5 機械学習ライブラリ Pylearn2 の利点

今回は、Deep Learning を使うための既存ソースコードとして、モントリオール大学の LISA Lab. が提供している、pylearn2 というライブラリを選んだ。pylearn2 は、python で記述された、Deep Learning など機械学習アルゴリズムを使うためのライブラリである。LISA Lab. を中心とする開発者によって、ソースコードの開発がほぼ毎日行われている。以下、pylearn2 を選択した理由を記す。

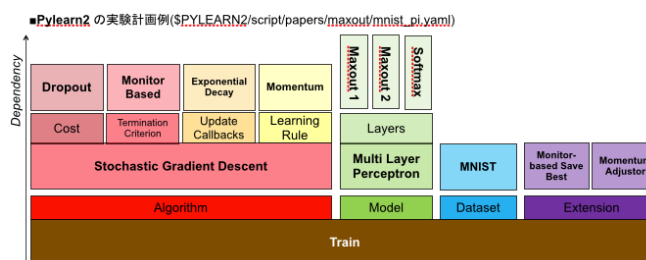


図 4.3 Pylearn2 の実験計画例

### 4.5.1 高精度アルゴリズム Maxout Network の実装

Pylearn2 には、Maxout Network という Deep Learning のアルゴリズムの一種が実装されている。このアルゴリズムは、画像認識タスクにおいて非常に高い精度を実現しており、基準 1 を満たす上で都合が良い。Maxout Network の解説 [過去のスライドをもとに加筆]

### 4.5.2 Theano の利用

pylearn2 のプログラムは、前述した Theano を全面的に利用して記述されている。Theano を用いたことで、可読性が大きく上昇している。また、何らかの理由で、pylearn2 の部分的拡張が必要になったときでも、[過去のスライドを基に加筆]

### 4.5.3 高度な拡張性と可読性

pylearn2 のソースコードは、拡張性や可読性を非常に強く意識して書かれている。基本的な使い方を記したチュートリアル Web ページが存在し、主なソースファイルには、詳細なドキュメントも記述されている。また、モジュール化が丁寧なので、自分で書く部分が少なくて済む。例えば、データセットを変更する場合は、Dataset クラスのみを書き下せばよく、他の部分に対してコードを書く必要がほとんどない。図 4.3 に、pylearn2 において機械学習のアルゴリズムがどのようにモジュール化されているかを示す。

また、プログラムを改造する場合でも、数値パラメータを変更するだけであれば、設定ファイルのみの変更で完結させることができる。この設定ファイルは、YAML(YAML Ain't a Markup Language)形式に少し独自拡張を加えた形式になっている。

プログラム本体と、数値の設定ファイルが分離されていることで、プログラミングが不得手な人でも、比較的簡単に設定を変更することができる。

## 4.6 ハードウェアの構成

pylearn2 を通して Deep Learning のアルゴリズムを実行する、と決めたところで、実際に Deep Learning を利用するためのマシンを用意する必要がある。まず、Deep Learning に限った話ではないが、機械学習のタスクは長時間の計算を必要とする場合が多く、プログラムの実行時間が 1 日以上かかることも稀ではない。普段使用している PC が使えなくなると困る場合は、機械学習を動かしておくための専用サーバを用意することが望ましい。GPU は必ずしも搭載していなくとも良いのだが、実践上 Theano や CUDA の性能を最大限に生かすためには、GPU を搭載しなければならない。また、CUDA を用いて GPU マシン専用にかかれたアルゴリズムを動作させるためには、GPU が必須となる。GPU は、NVIDIA 製で、CUDA に対応していなければならない。ただし、グラフィック出力の機能はなくても構わない。2013 年 1 月現在、NVIDIA の Web ページに掲載されている GPU に関しては、全て CUDA に対応しているため、店舗で新品の GPU を購入する場合は基本的に問題は生じないと思われる。NVIDIA の GPU の中で、どれを選ぶかも問題になる。NVIDIA のホームページでは、GPU のうち Tesla というシリーズが GPGPU に特化しており、非常に高精度/高速な演算を行うことができる、と述べられている。しかし、非常に高価な上、Tesla シリーズは一般的なグラフィックカードではなく、サーバ全体の購入を基本としている。販売員の方によれば、Tesla は主に商用の大規模データや、ミスの許されない長時間科学計算などに用いることを想定して作られている。今回の目的は、Web 工学の一般的タスクにおける Deep Learning 技術を確立するという、いわば実験的な用途であり、Tesla の利用はオーバースペックだと思われる。NVIDIA の GPU は、Tesla を除くと、Quadro と GeForce という 2 つのシリーズに分かれている。店舗の販売員の方に詳しい話を伺ったところ、Quadro はコンピュータグラフィックの出力機能に注力したシリーズであり、GeForce はより計算性能重視という傾向をもっている。つまり、Deep Learning の計算など GPGPU に用いる場合、同じ価格帯ならば、GeForce シリーズの GPU を用いた方が、費用対効果が大きくなると考えられる。また、販売員の方によれば、GeForce シリーズの中でも Titan という機種は、Tesla 用の部品の中で品質チェックに漏れてしまったものを流用しており、Tesla とほぼ同様の、非常に高い性能を発揮することが出来る、とのことだった。しかし、今回利用する中で最も計算量が多い Maxout Network について、元の発表論文によれば、GeForce シリーズの GTX 580 という、少し世代が前の GPU が用いられている。Deep Learning を実行させる上で、GPU が存在すること自体は非常に重要だが、必ずしも最新スペックの GPU が必要というわけではないことがわかる。今回の構成では、知の構造化センターの中山浩太郎先生のアドバイスもあり、Titan ではなく、同じ GeForce シリーズの GTX 760 という GPU を搭載することにした。

## 第 5 章

# Facebook の顔画像認識と深層学習技術

この章では、Deep Learning を画像認識の問題に適用した応用例を挙げる。

ソーシャルメディアのユーザ属性を推定することは、マーケティングやネット広告の最適化において有益である。ここでは、公開プロフィールの一種であるプロフィール画像に着目する。入力となるプロフィール画像と、推定する目標であるユーザ属性のセットを用いて、Deep Learning を行わせることで、より精度の高いユーザ属性推定を目指す。

### 5.1 背景

ソーシャルメディアのユーザプロフィールから、そのユーザの属性を推定するためのアルゴリズムが求められている。

マーケティングや、ネット広告の最適化などに応用可能で、ビジネス面において大きな貢献が期待できる。

ユーザ属性の推定元としては、名前 / 日記 / ツイートなどのテキストデータ、プロフィールなどの画像、友達関係などが挙げられる。[29]

テキストデータは量が豊富だが、ソーシャルメディアの種類によっては、「友達のみ公開」などの設定がされている。ネット広告を載せる企業の視点から見ると、常に自由にアクセスできるとは限らない。

一方プロフィール画像は、ほぼ全てのソーシャルメディアにおいて、全員に公開されている。ネット広告企業からでも、必ずチェックできる。

画像からユーザ属性を推定できれば、より多くの場面で、効率の良いマーケティングが可能になる。これはテキストベースの方法にはない、大きなメリットである。



図 5.1 ユーザ属性推定の必要性

しかし、プロフィール画像を元データとした研究はまだメジャーではない。

最終的に、ソーシャルメディアのプロフィール画像が与えられれば、ユーザ属性を推定できる状態を目指す。

その過程として、今回の領域プロジェクトでは、「Facebook」より「プロフィール顔画像」に限定してのユーザ推定を試みる。

公開プロフィールからユーザ属性を推定するというアプローチによる過去の研究としては、Twitter のプロフィールやツイート内容、フォロー関係などから、ユーザ属性を割り出したものがある [29]。この研究では、「スターバックスが好きか」「支持政党は民主党と共和党のどちらか」「どの民族 (ethnicity) に属しているか」といった情報を推定している。

なお、この研究においては、プロフィール写真はユーザ属性を推定する参考になりにくいとされている。彼らに拠れば、ソーシャルメディアのユーザーは、必ずしもプロフィール画像にユーザ本人の画像を用いるわけではない。むしろ、有名人の画像が使われていて、本人の属性推定においてノイズとなってしまうケースもある。

また、別の研究 [28] では、Twitter のデータより、Twitter のユーザ構成の傾向について、基本的なデータを収集している。具体的には、位置情報より居住地の傾向を、first name から性別の傾向を、last name から宗教の傾向を割り出している。なお、この研究でもプロフィール画像データは用いられていなかった。

H. Andrew Schwartz らは、公開プロフィールという制限こそ課していないが、75000 名のユーザ有志より、7000 万個の単語や言葉、話題の例を収集して分析にかけ、同時に収集した年齢・性別・性格テストのデータとの相関性を学習させることや、言葉遣いからこれらのユーザ属性を精度良く推測させることに成功した [36]。彼らは、年代や性格ごとに特徴的な言葉遣いを可視化しており、可視化

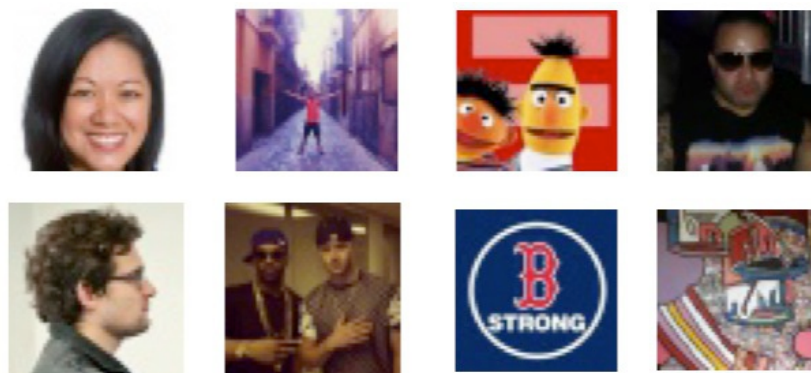


図 5.2 Facebook におけるプロフィール画像の例

の結果は Web サイトにて見ることができる\*1。

## 5.2 ソーシャルメディアにおける、プロフィール画像

図 5.2 に、実際に Facebook にて利用されているプロフィール画像の例を挙げる。これはごく一部の例であるが、これらを見るだけでも、以下の障害となりそうな点が考えられる。

- **人間が写っているとは限らない。**人気キャラクターやブランドのロゴ、絵画が含まれている。入力データのバリエーションが増えることで、学習が難しくなることが予想される。
- **2人以上の人間が映っていることがある。**例えば、カップルの写真やクラスの集合写真がプロフィール画像として使われていたとき、誰がユーザ本人であるかを推定するのは極めて困難だと想像できる。
- **本人の画像かどうかわからない。**先述した研究 [29] でも述べられている通り、少なくとも画像自体の中に、本人である証拠はない。有名人に関しては、有名人の顔写真のデータベースを別途作成することで対応できる可能性があるが、あまり知られていない他人の写真だった場合、対処は難しい。
- **顔が鮮明でないことがある。**例え本人が写っていても、顔の拡大写真ではなく、全身が映されていて、顔がぼやけてしまっていることがある。

以上をまとめると、「プロフィール画像＝ユーザの顔」という前提が成り立たないという問題点が浮き彫りになる。この前提が成り立たない場合、「画像＝ユーザの顔→ユーザの年齢や性別」という一段階の推論ではなく、「画像→本人か?→本人でなかったら、誰の容姿なのか、またはキャラクターや

\*1 <http://wwbp.org/>

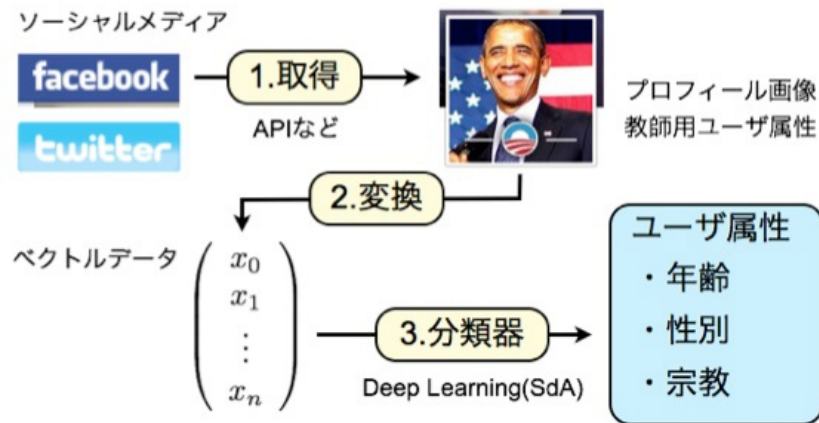


図 5.3 提案手法

ブランドなのか→その有名人やキャラクターを好むユーザは、どのような年齢や性別が多いか?」という多段階の推論が必要となり、学習難易度が高くなると考えられる。

今回は、

### 5.3 提案手法

図 5.3 は、今回の提案手法を表している。提案手法は、次の 3 ステップより成る。

1. **ソーシャルメディアより、プロフィール画像を得る。** この方法はソーシャルメディア毎に大きく異なるが、各メディアが提供している API や、閲覧者用の Web ページより、様々なユーザのプロフィール画像を取得する。また、このときユーザ属性の教師データを取得できる場合もある。
2. **プロフィール画像を、ベクトルデータに変換する。** 基本的には、画像の各画素値そのものを、入力ベクトルとして使用する。これは、可能な限り取得したままのデータを加工せずに入力することで、素性の採り方自体を学習できる Deep Learning の特徴を活かせるからである。ただし、対象データによっては、画像の特徴量を付加することで精度が上昇する [4] ことが知られており、工夫の余地もある。
3. **ベクトルデータから、分類器を使って、ユーザ属性を識別する。** 機械学習により、画像のベクトルデータからユーザ属性を推定できるような、クラス識別モデルを取得する。Deep Learning が応用されるのは、この部分である。

## 5.4 問題設定と、実験方法の詳細

今回の実験では、ソーシャルメディアのプロフィール画像からユーザ属性を推定する例として、Facebook のプロフィール画像のうち、人間の顔画像を 1 つだけ含んでいるものを対象として、ユーザの性別を推定出来るかどうか、学習を行わせた。以下に、実験方法の詳細を記す。

### 5.4.1 画像の取得と加工

Facebook の公開 API を用いて、サイズ 200x200(px) のプロフィール画像を 11028 件取得した。画像と同時に、ユーザの性別データも収集した。

取得した画像からは、条件に合わないものを除外すると同時に、顔の特徴が失われない範囲で、学習速度を上げるための前処理を施した。まず、各画像を RGB カラーからグレースケールに変換した。次に、人間の顔画像がちょうど 1 人分入っているかどうか、判定を行った。これには画像処理ライブラリの OpenCV に含まれている、顔画像識別のプログラム<sup>\*2</sup>を用いた。識別器には、LBP Cascade 法 [?] を使った。判別の結果、サイズ 20x20[px] 以上で正面を向いた 1 人分の顔画像が検出できた場合のみ、その画像を使用した (図 5.2)。さらに、検出した顔領域が中心で一定のサイズを占めるように、画像を切り抜き、30x30[px] に解像度を落とした。また、より画像の特徴がわかりやすくなって学習が進むよう、色調を平均化補正し、コントラストが際立つようにした。

以上の行程により、男性 2180 件、女性 1566 件、計 3746 件の画像データ及び、対応する性別のデータが集まった。

### 5.4.2 ベクトルデータへの変換

平面的な広がりを持つ画像データから、画素値 [0.0-1.0) を要素とする 900 次元のベクトルデータへの変換を施した。画像の各画素と、ベクトルの順番の関係は、図 5.5 のようになっており、この図における画素の番号と、ベクトルの要素番号が対応している。

### 5.4.3 ユーザの分類

入力がここまでの変換で得たベクトルデータ、出力がユーザの性別の、クラス分類問題として考える。分類器に学習を行わせ、ユーザを精度よく分類することを目指す。学習モデルとしては、Deep Neural Network の一種である Stacked Denoising Autoencoder(以下 SDA) と、従来手法としての Support Vector Machine(以下 SVM) の 2 つを用いた。

---

<sup>\*2</sup> [http://docs.opencv.org/trunk/modules/objdetect/doc/cascade\\_classification.html](http://docs.opencv.org/trunk/modules/objdetect/doc/cascade_classification.html)



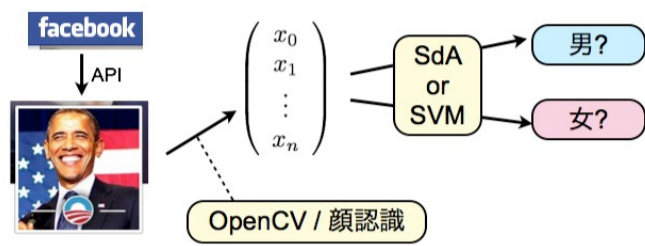


図 5.4 今回の実験の模式図

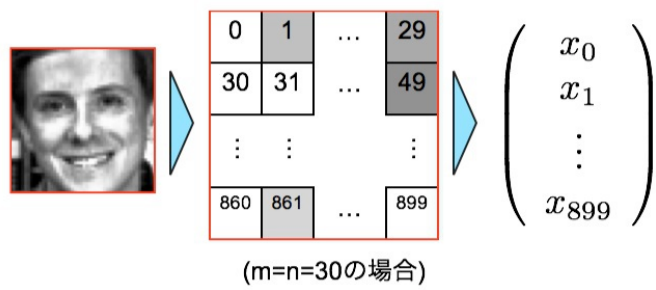


図 5.5 実験で使った、画像から入力ベクトルへの変換方法

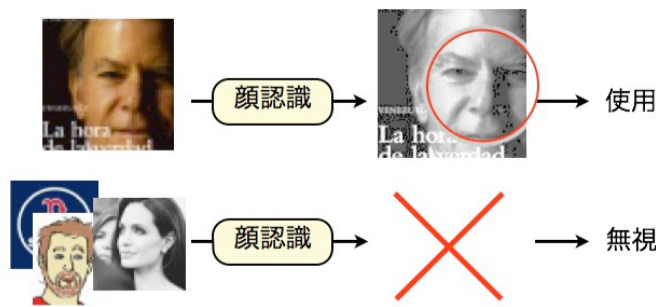


図 5.6 プロフィール画像からの顔画像抽出方法

SDA の実装には、Web サイト Deep Learning Tutorial<sup>\*3</sup>にて提供されているものを用いた。  
corruption rate は 30% とした。また、SVM の実装には、専用ライブラリの libsvm<sup>\*4</sup>を用いた。

<sup>\*3</sup> <http://www.deeplearning.net/tutorial>  
<sup>\*4</sup> <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

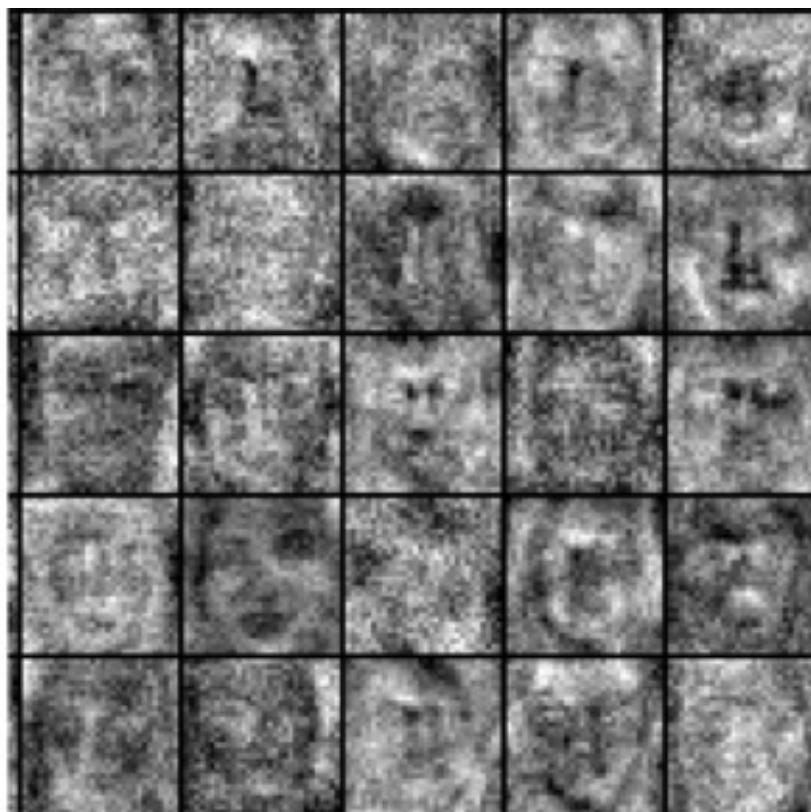


図 5.7 実験で学習されたフィルター

## 5.5 結果

分類精度は、表 5.1 の通りとなった。Deep Learning の SDA の方が、少し精度が低い結果となってしまった。

なお、図 5.7 に、今回の実験で SDA が学習した、1 レイヤー目のフィルターを示す。SDA は精度こそ僅かに劣っているものの、今回の方法でも顔を学習すること自体には成功していることがわかる。

## 第 6 章

# Web 工学への深層学習技術の応用

この章では、第 4 章で得られた Deep Learning 用の構成を用いて、実際に機械学習のタスクを実行する。分類精度、実行時間、消費メモリ量を調べることにより、Deep Learning を Web 工学の問題に適用する際の参考とする。

### 6.1 利用したデータセット

実験の前提知識として、この節では、実験に使用したデータセットについて記す。機械学習の分野では、分類精度のベンチマークを取るために、様々なデータセットが提供/提案されてきている。同じデータセットに対して、様々な分類モデルやアルゴリズムを用いて分類実験を行うことにより、どの手法が優れているのか比較することが出来る。

ここでは、画像認識のデータセットを用いて、Deep Learning プログラムのベンチマークを行う。画像データを用いる理由は、1 つには、画像認識が Deep Learning が最も高い分類性能を実現している分野だからである。加えて、画像データや画像から抽出された素性は、可視化が比較的容易なことが多い。可視化することで、学習過程を目で見て確かめることが出来るため、アルゴリズムの分析を行いやすい、という利点がある。

#### 6.1.1 MNIST

MNIST database<sup>\*1</sup>とは、手書き数字を画像分析によって認識するベンチマークタスクである。このデータセットは、National Institute of Standards and Technology(NIST) が提供する手書き数字のデータにサイズの規格化処理を加え、数字の書き手毎に整理したものである。データはサイズ 28x28 ピクセルの白黒画像 70000 件より構成される [24]。それぞれの画像は、0~9 のいずれかの数字 1 文字に該当している。画像データを認識プログラムに入力して、どの数字に該当するか判別させることで、画像識別の精度を競うことになる。図 6.1 に、MNIST の画像データの一部を示す ([24]

---

<sup>\*1</sup> <http://yann.lecun.com/exdb/mnist/>



図 6.1 MNIST データの例

より引用した)。

MNIST の 70000 件のデータは、60000 件のモデル訓練用データと、10000 件の精度測定用データに分かれている。まず 60000 件のデータを使って識別モデルに学習を行わせた後、未知の 10000 件のテストデータを実際に識別してみることで、精度を測定する。テストデータを識別している最中に、更なる学習を行うことは許されない。分類器は、過去に学習に使ったデータだけ良く識別できても意味がなく、むしろ未知のデータこそ精度良く分類出来なければならない。モデルが過去のデータに特化してしまうと、かえって未知のデータに対する識別精度が低下することもある。この低下現象は過学習 (over fitting) と呼ばれ、機械学習における落とし穴の一つとなっている。MNIST に限らず、機械学習用のデータセットにて、訓練データとテストデータをあらかじめ分けておくことは一般的であり、学習アルゴリズムが過学習を防いでいるかどうか、判定するために有効な方法の 1 つとして知られている。

MNIST は、様々な画像認識アルゴリズムの作者によって使用されている。MNIST の配布 Web サイトには、MNIST を利用した論文のリストが、分類誤差によるランキング形式で掲載されている。表 6.1 に、MNIST データセットにおける各アルゴリズムの分類誤差を、ランキング形式の Web サイト<sup>\*2</sup>より再構成して示す。

### 6.1.2 CIFAR10

CIFAR10<sup>\*3</sup>は、写真を画像分析によって識別するタスクである [19]。入力データは 60000 件のカラー画像で、どの画像も、飛行機、自動車、鳥、猫、鹿、犬、蛙、馬、船、トラックの 10 種類のクラスのどれかに属しており、各クラス均等に 6000 枚ずつの画像が割り振られている。画像データは、Google や Flickr などによる Web 検索で集められた画像を、人手でラベリングして作られている。

<sup>\*2</sup> [http://rodrigob.github.io/are\\_we\\_there\\_yet/build/classification\\_datasets\\_results.html](http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html)

<sup>\*3</sup> <http://www.cs.toronto.edu/~kriz/cifar.html>

表 6.1 MNIST の分類誤差ランキング

手法	発表学会/雑誌/年	分類誤差
DropConnect [41]	ICML 2013	0.21%
Multi-column Deep Neural Networks [4]	CVPR 2012	0.23%
Deep Big Simple Neural Nets [3]	Neural Computation 2010	0.35%
Energy-Based Model [30]	NIPS 2006	0.39%
Convolutional Neural Networks [37]	Document Analysis and Recognition 2003	0.40%
<b>Maxout Networks</b> [11]	ICML 2013	0.45%
COSFIRE filters [1]	PAMI 2013	0.52%
Multi-Stage Architecture [15]	ICCV 2009	0.53%
Deformation Models [17]	PAMI 2007	0.54%
A trainable feature extractor [22]	Journal Pattern Recognition 2007	0.54%
Invariant Support Vector Machines [7]	Machine Learning 2002	0.56%
Sparse Coding [21]	TNN 2008	0.59%
Unsupervised learning of invariant feature hierarchies [31]	CVPR 2007	0.62%
shape contexts [2]	PAMI 2002	0.63%
Receptive Field Learning [16]	CVPR 2012	0.64%
Sparse Activity, Sparse Connectivity [40]	JMLR 2013	0.75%
Convolutional Deep Belief Networks [25]	ICML 2009	0.82%
Deep Encoder Network [26]	2009	0.94%
Deep Boltzmann Machines [35]	AISTATS 2009	0.95%
Deep Belief Networks [6]	2008	1.12%
Convolutional Neural Networks [37]	2003	1.19%
neural networks [13]	2006	1.20%
Deep learning via semi-supervised embedding [42]	2008	1.50%

また、サイズを 32x32 ピクセルに縮小されている。

MNIST と同じように、60000 件のデータは、50000 の訓練データと 10000 件のテストデータに分割されている。まず、分類器にこれらの訓練データを読み取らせて、学習を行わせる。その後、テストデータのうちいくつかを正しいクラスに分類することができるのか、分類精度を競うことになる。

CIFAR10 についても、データの例を 6.2 に、分類誤差のランキング Web サイト\*<sup>4</sup>より、再構成した表を 6.2 に示す。

\*<sup>4</sup> [http://rodrigob.github.io/are\\_we\\_there\\_yet/build/classification\\_datasets\\_results.html](http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html)

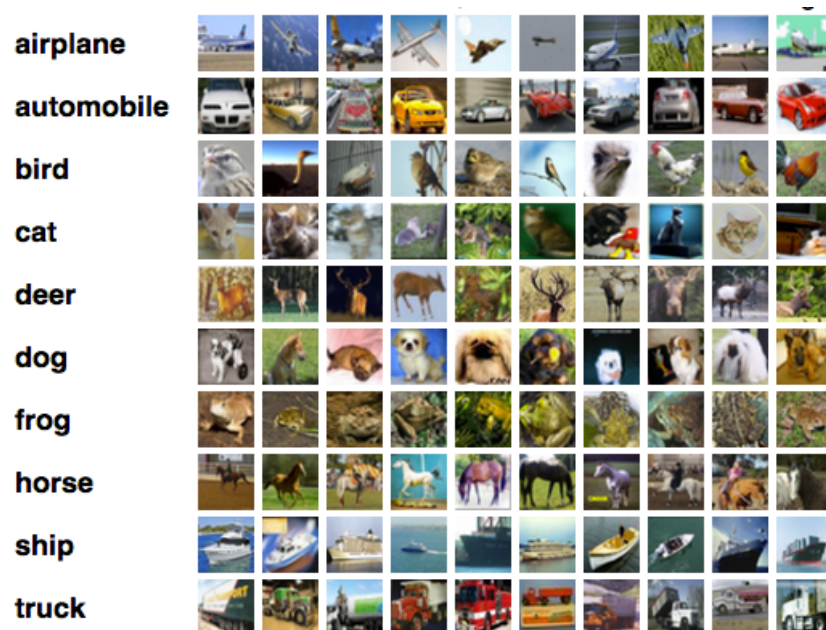


図 6.2 CIFAR10 データの例

手法	発表学会/雑誌	分類誤差
DropConnect [41]	ICML 2013	9.32 %
Maxout Networks [11]	ICML 2013	9.35 %
Bayesian Optimization [39]	NIPS 2012	9.50 %
Deep Convolutional Neural Networks [20]	NIPS 2012	11.00 %
Multi-Column Deep Neural Networks [4]	CVPR 2012	11.21 %
Deep Convolutional Neural Networks [43]	arXiv 2013	15.13 %
Dropout [14]	arXiv 2012	15.60 %
Sum-Product Networks [9]	NIPS 2012	16.04 %
Single-Layer Networks [5]	AISTATS 2011	20.4 %

表 6.2 CIFAR10 分類誤差のランキング

## 6.2 使用したライブラリ

この章の実験には、`pylearn2` という Deep Learning のライブラリを用いた。

## 6.3 使用したハードウェア

この章の実験では、GPU を搭載したサーバマシンを用いた。表 6.3 に、今回の実験で使用したマシンの主な使用パーツ及び性能を列挙した。また、Maxout Network の実装ソースコードには、実験

	GPU	CUDA core	CPU	クロック数	メモリ容量
今回	GTX760	1152	Intel CPU Core i7 4770S	3.1 GHz	32GB
論文	GTX580	512	Intel Xeon CPU E5620	2.40GHz	(記載無し)

表 6.3 ハードウェア性能の比較

表 6.4 Maxout Network による MNIST(2 次元) 分類の結果

手法	データセット	元論文の誤差	実験誤差	増加分
Maxout Network	MNIST(2 次元)	0.45%	0.51%	+0.06%

マシンのパーツ性能を書いたファイルが付属している\*5。このファイルの記述内容も、比較のために  
列挙した。今回の実験に使ったマシンの方が、ハードウェア性能的には優れていることがわかる。

6.4 ベンチマーク実験の詳細

前節で挙げた画像分類のベンチマークセットに対し、Deep Learning の様々なアルゴリズムによっ  
て分類を行わせ、その性能を測定した。

6.4.1 Maxout Network による、MNIST の分類タスク (2 次元データとして扱う)

Maxout Network を分類器として用い、MNIST の分類タスクを行わせた。モデル構造としては、  
Convolutional Layer を 2 層重ねた上に、全てのニューロンが単純に接続された Maxout Layer を 1  
層付け加え、最後の層にてロジスティック回帰を行って分類結果を出した。実験を行った結果、分類  
誤差は 0.51% となった。これは、元の論文が主張している分類誤差より、僅かに悪い結果となった。

6.4.2 Maxout Network による、MNIST の分類タスク (1 次元データとして扱う)

前項と同じタスクを、「MNIST のデータは、2 次元の画像データではなく、1 次元のベクトルデー  
タである」という条件の基で行わせた。つまり、2 次元データに対する Convolutional Layer 技術を  
敢えて使わない状態で、どれだけの精度を Maxout Network が実現できるのか、実験した。この条  
件でも Maxout Network が良い性能を出すことが出来れば、1 次元のデータ、例えば言語データや  
音声データにおいても、Maxout Network が応用できる可能性が高くなる。

また、何らかの原因でランダム性が発生していないか確かめるため、この実験は同じ条件で 10 回  
行なった。乱数のシードは、元論文の実験が行われたときと同じ値で固定した。

実験の結果、誤差は 10 回とも共通で、1.16% となった。Maxout の元の論文やはり少し低い精度に

\*5 [pylearn2/scripts/papers/maxout/notes](#)

表 6.5 Maxout Network による MNIST(1 次元) 分類の結果

手法	データセット	元論文の誤差	実験誤差	増加分
Maxout Network	MNIST(1 次元)	0.94%	1.16%	+0.12%

表 6.6 Maxout Network による MNIST(1 次元) 分類の実験詳細

	平均実行時間 (秒)	平均最大使用メモリ (MB)
pretraining	2532.8	779.3
finetuning	756.7	792.5
全体	3289.5	792.5

なってしまった (表 6.5)。なお、平均実行時間は 55 分、最大メモリ使用量は 793MB だった (表 6.6)。

### 6.4.3 Maxout Network による、CIFAR10 の分類タスク

データセットを変更し、MNIST ではなく、CIFAR10 を Maxout Network を用いて分類させようと試みた。データは 2 次元画像として扱った。つまり、Convolutional Layer の使用は可能とした。レイヤー構造は、Convolutional Layer 3 層 - Maxout Layer1 層 - 識別用 Softmax Layer1 層とした。

しかし、この実験を実際に行ったところ、3 日間実行を続けたところで、残りエポック数と計算速度の比較より、全体の実行が完了するまでに 10 日以上かかることが判明した。これは Web 工学における応用に使う学習アルゴリズムとして、試行錯誤しながら使うものとしては現実的でない実行時間である。また、実験スケジュールの都合もあり、実験の中断を余儀なくされてしまった。



## 第 7 章

# 考察と提言

この章では、5 章と 6 章の結果に対する考察を行うと共に、考察結果を踏まえ、Deep Learning の実問題における効率的な応用方法について、提言を行う。

### 7.1 5 章の実験に関する考察

#### 7.1.1 全体の結果のまとめ

5 章の実験では、ソーシャルメディアのユーザ属性推定において、プロフィール画像からの推定手法を模索した。実験の結果、SDA、SVM 共に、Facebook のユーザプロフィールにて 1 人分の顔画像が使われているという条件ならば、ある程度ユーザの性別を識別できることがわかった。

#### 7.1.2 SDA にて良い精度が出なかった理由の考察

識別自体にはある程度成功したとはいえ、本来画像認識や多段階の推論で効果を発揮すると思われる Deep Learning を使用したにも関わらず、従来から用いられている単純な SVM による精度を上回ることが出来なかった。この理由はわかっていないが、考えられることとして、まず実験で用意したデータ数の不足や偏りが考えられる。例えば、Deep Learning 全般が大きな成果を挙げている MNIST のデータセットには、全体で 70000 のデータが用意されている。また、CIFAR10 でも 60000 件のデータが利用可能である。比べて、今回用意できたデータは 3746 件のみであり、男性と女性のデータ数には 1.3 倍ほどの開きがあった。このデータの不足と偏りが、学習傾向に歪みを与えてしまった可能性がある。

また、隠れレイヤー数、レイヤー内ユニット数など、学習モデルを形成する様々なハイパーパラメータの調整を行うことで、学習制度を向上させることができる可能性がある。SVM の場合、grid search と呼ばれる方法によってパラメータを調整している。しかし、この方法はハイパーパラメータが増加するにつれて指数関数的にかかる時間が増大してしまう。パラメータが 1~3 個程度で済む SVM では有効な方法だが、パラメータ数が非常に多い SDA や他の Deep Learning の方法では、現

実的ではない。ハイパーパラメータを自動的に調整する方法が必要である。

### 7.1.3 今後の発展について

今回の実験では、まず最も確実性の高いところから検証するということで、単体の顔プロフィール画像のみを識別のターゲットとした。しかし、キャラクターやブランドロゴなどの非実写アイコンや、人間の横顔、複数人が映った画像からでもユーザ属性が推定できれば、より応用での確実性が増す。性別だけでなく、ユーザの好みや交友関係の傾向も推測できれば、これも貴重な情報となる。人間の行動や思考は、テキストだけでなく、ファッションや交友関係など非言語的な特徴にも顕れる。こういった情報も学習できるようになれば、システムの利用価値が高くなっていくと思われる。

Facebook は、本人画像の使用率が比較的高いコミュニティとして一般に知られている。有名人の画像利用問題が考えられるにも関わらず、ある程度精度良く識別が出来たのも、対象が Facebook だったからということと考えられる。Facebook だけでなく、Twitter など、本人画像以外がプロフィール画像に使われやすいソーシャルメディアでも、ユーザ属性が推定出来れば、応用が容易になると考えられる。

## 7.2 6 章の実験に関する考察

### 7.2.1 全体のまとめ

pylearn2 における Maxout Network は、元論文に記されている精度こそ再現できなかったものの、MNIST の分類タスクにおいて State of the Art に近い精度を実現することが出来た。Maxout Network を利用することで、少なくとも画像認識のタスクにおいて精度を向上させることが出来ると考えられる。

### 7.2.2 分類誤差が大きくなってしまう原因の考察

今回の実験では、MNIST の 1 次元版、2 次元版共に、元の論文や pylearn2 ソースコードの付属テキストに書いてあった分類誤差よりも、大きい誤差しか再現できなかった。また、この誤差が大きくなってしまう問題は、複数回の実験を行っても、解決しなかった。

分類精度が悪くなってしまった理由として、当初、図 7.1 に挙げた 3 つの原因が考えられた。しかし、複数回実行しても全く同じ結果が出たため、乱数のシードを変更しない限り、内部的には全く同じ演算が成されていると考えられる。よって、「重みのランダム初期化」、つまり「ニューラルネットワークの接続の重みがランダムに決まっており、たまたま分類に不利な重みからスタートしたため、元の論文よりも悪い結果が出てしまった」という仮説は否定される。残る可能性は、「ソースコードのバージョン」と「ハードウェア構成の違い」であるが、GPU を始めとする各パーツの性能は、元論

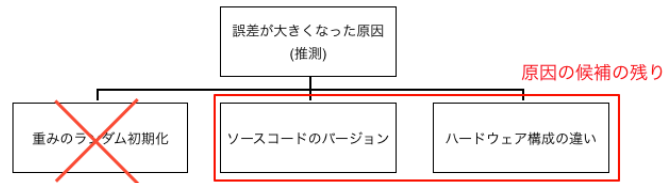


図 7.1 Maxout Network による MNIST 分類実験で、誤差が論文より大きくなった原因

文の実験時に使われたものよりも、今回の実験で使ったものの方が高いため、「ハードウェアの性能」に原因を求めるのも難しい。残る可能性は「ソースコードのバージョン」である。これは、`pylearn2` が依存しているライブラリの、`numpy` や `scipy`、`Theano` また `pylearn2` 自体などがアップデートされたことにより、内部的に細かい計算方法が変化し、分類誤差の再現性が下がってしまった、という仮説である。

実験の再現性については、Maxout Network の元実験を行ったメンバーも、

All of `pylearn2`'s dependencies (`theano` / `scipy` / `numpy` / etc.) make reproducibility very difficult. As such, we are not currently going to make any effort to ensure that all results are widely reproducible on a variety of platforms. Instead, we will record the platform on which they were verified to be correct.

と記している (`pylearn2` のソースコード<sup>\*1</sup>より引用)。彼らの認識でも、まずソースコードやライブラリのバージョンが重要であり、またハードウェアの性能も影響すると考えられていることが読み取れる。

### 7.2.3 CIFAR10 の実行時間が非常に長くなってしまいう問題について

今回の実験では、Maxout による CIFAR10 の分類を、実行時間があまりにも長くなるため、断念してしまった。実行時間をどうすれば短縮できるのか、特にハードウェアを変えず、分類精度も犠牲にしないで、出来れば長くとも 1 日程度の実行時間で収束させるための方法がないか、検討を重ねる必要がある。その後の実験により、モデルの学習中に統計情報を収集させる部分で時間がかかってしまっている可能性があるとわかっている。精度を変えずに時間短縮をするための、大きなポイントの 1 つとして、追求を続ける必要がある。

<sup>\*1</sup> `pylearn2/scripts/papers/maxout/notes`

## 7.3 深層学習の利用法に関する提言

Maxout Network が良い精度を実現できたこと、改造のしやすさ、GPU 利用の簡便さなどを考え合わせると、現時点では、pylearn2 を通して Deep Learning を利用するのが良いと考えられる。Deep Learning を利用する上で、最も求められる高い識別性能は、Maxout Network を利用することで担保できる。Maxout Network は、画像に対しては Convolutional Layer と併用することにより、ほぼ state of the art と遜色ない精度を実現することができる。また、対象が画像ではなく、1 次元の広がりしかもたないデータの場合、Convolutional Layer を使うことは出来ないが、現時点では最も高い識別精度を出すことが出来る。この識別性能は、画像データだけでなく、他のデータを扱う際にも有用だと思われる。また、任意の連続関数を近似できるため、かつての単層パーセプトロンのように、理論的に学習できない関数が出現するという事もない。Deep Learning に識別精度を求める場合、Maxout Network を使うのが最も有効な方法の 1 つである。

Maxout の既存実装としては、論文の著者自らが提供しているライブラリの、pylearn2 を使うのが最も確実である。この実装を使えば、論文にて述べられている分類精度にかなり近い精度を、実際に実現することができる。厳密には、論文の分類精度よりも、若干性能は劣化してしまうが、著者が実験を行ったときのハードウェアと、ソースコードのバージョンを揃えずとも、かなり近い精度を再現できることも、また事実であり、実用上はあまり問題にならないと考える。

Deep Learning の大きなネックの 1 つとして、その長い実行時間が挙げられる。この問題は、実行時に GPU を利用することによってかなり緩和できるが、CUDA を用いた専用のプログラムを書く必要があり、難易度が比較して高くなってしまう。この難点は、計算アルゴリズムを書くときに、数値計算/数式処理ライブラリである Theano を利用することで、かなり楽になる。Theano で書いたプログラムは、GPU 上でも利用でき、CPU のみのマシンでも、実行速度は下がるが問題なく動かすことができる。また、科学技術計算プログラムの記述においてメジャーな python で書かれており、C 言語のプログラムに比べて記述が容易であるにも関わらず、事前に CUDA のコードを生成&コンパイルする仕組みにより、C 言語で CUDA のコードを全て記述した場合と代わらない速度で動かすことができる。

pylearn2 は Theano の機能をフルに活用して記述されており、また Space クラスや画像による可視化など、Theano に不足している部分にもフォローがされている。pylearn2 を使えば、Theano の恩恵をそのまま受けることができる。この点でも、pylearn2 は望ましいと考えられる。

pylearn2 は、規模の比較的大きいライブラリではあるが、ソースコードは再利用性を強く意識して記述されており、モジュール化が明確である。libsvm のような、テキストファイルにデータを書くだけで利用できるような仕組みは残念ながら用意されていないが、例えば入力データを差し替えるだ

けならば、Dataset クラスの内容を把握して書くだけで済む。また、アルゴリズムにて使われているハイパーパラメータを調整するだけなら、ソースコードを書く必要すらなく、設定用の yaml ファイルを変更するだけで調整を済ませることができる。この「利用する部分だけ把握すれば良い」という性質は、モジュール化の大きな利点であり、そのまま pylearn2 の大きな利点でもある。

## 第 8 章

# おわりに

### 8.1 研究の成果

この研究では、Deep Learning を Web 工学の問題に適用するにあたって、その特徴である高い精度を落とすことなく、出来るだけ簡便に応用するための方法論とノウハウを調査した。pylearn2 を用い、Dataset クラスのみを書き換えて分類すると良い。GPU を使うと大幅な高速化が見込めるが、必須ではない。

### 8.2 今後の課題

画像分類タスクにおいて有効な方法が、Web 工学のデータでも必ず有効かどうかは、未知数な部分がある。例えば、Convolutional Network は、2 次元の画像データに対しては非常に高い効果を挙げるが、そのまま文章データに応用することは出来ない。pylearn2 で言えば、Dataset クラスを作るだけで識別がうまくいくのか、文章専用の Model を構成する必要があるのか、確かめていく必要がある。また、精度を上げるためには、どのようにハイパーパラメータを調整すれば良いのか、あるいは精度を多少犠牲にしても、比較的短い実行時間で良い結果を得たい時、どのような調整を施せばよいのかは、まだわかっておらず、今後の大きな課題の一つである。Deep Learning を試す上で、大きなネックとなるのが、CUDA を用いた GPU 専用のソースコードの存在である。GPU を搭載していなかったり、使えるメモリが少ない状況下でも、Deep Learning のコードを効率良く動かすことが出来れば、Deep Learning の利便性はますます増加するだろう。現在の Deep Learning では、画像のフィルタで何が学習されているのかを、部分的に可視化することはできる。しかし、文書解析において、どのような表現を学習したのかを、人間に理解できる形でみることは難しい。言い換えれば、学習によって、分類器がどこに注目するようになったのか、文構造や、感情、文体などに対応するニューロンが存在しているのか、といった情報が、人間に理解できる形になっていない。この部分をどうやって可視化して、人間に理解できる状態にするか、そこからどのような知見を得られるのか、あるいはそもそも人間には理解出来る知識として取り出せるのかどうか、といったことを調べること

により、表現学習としての Deep Learning の側面を、さらに活かすことができると思われる。例えば、Deep Learning によって学習された、データの着目点や抽象化のポイントを、人間が真似することによって、人間の方が機械から知識を習得し、さらに発展させることも考えられる。

# 謝辞

(全て完了してから改めて書きます)

東京大学工学部システム創成学科

知能社会システムコース

松尾研究室 4 年 黒滝 紘生

平成 26 年 2 月日



## 参考文献

- [1] G. Azzopardi and N. Petkov. Trainable cosfire filters for keypoint detection and pattern recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 35, No. 2, pp. 490–503, 2013.
- [2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 24, No. 4, pp. 509–522, April 2002.
- [3] Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, Vol. 22, No. 12, pp. 3207–3220, 2010.
- [4] Dan Claudiu Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *arXiv:1202.2745v1 [cs.CV]*, 2012.
- [5] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.
- [6] George Dahl and Kit La Touche. Cs81: Learning words with deep belief networks. 2008.
- [7] Dennis Decoste and Bernhard Schölkopf. Training invariant support vector machines. *Mach. Learn.*, Vol. 46, No. 1-3, pp. 161–190, March 2002.
- [8] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. CVPR, 2009.
- [9] Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pp. 3248–3256, 2012.
- [10] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, Vol. 35, No. 12, pp. 61–70, December 1992.
- [11] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

- [12] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, Vol. 18, No. 7, pp. 1527–1554, 2006.
- [13] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313, No. 5786, pp. 504–507, 2006.
- [14] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- [15] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153, 2009.
- [16] Yangqing Jia, Chang Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3370–3377, 2012.
- [17] Daniel Keysers, Thomas Deselaers, Christian Gollan, and Hermann Ney. Deformation models for image recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 29, No. 8, pp. 1422–1435, August 2007.
- [18] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, Vol. 42, No. 8, pp. 30–37, August 2009.
- [19] Alex Krizhevsky and G Hinton. Learning multiple layers of features from tiny images. Master’s thesis, 4 2009.
- [20] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pp. 1106–1114, 2012.
- [21] K. Labusch, E. Barth, and T. Martinetz. Simple method for high-performance digit recognition based on sparse coding. *Trans. Neur. Netw.*, Vol. 19, No. 11, pp. 1985–1989, November 2008.
- [22] Fabien Lauer, Ching Y. Suen, and Gérard Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recogn.*, Vol. 40, No. 6, pp. 1816–1824, June 2007.
- [23] Quoc V Le, Marc’Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S Corrado, Jeff Dean, and Andrew Y Ng. Building high-level features building high-level features using large scale unsupervised learning. *ICML*, 2012.
- [24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to docu-

- ment recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.
- [25] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 609–616, New York, NY, USA, 2009. ACM.
- [26] Martin Renqiang Min, David A. Stanley, Zineng Yuan, Anthony J. Bonner, and Zhaolei Zhang. Large-margin knn classification using a deep encoder network. *CoRR*, Vol. abs/0906.1814, , 2009.
- [27] Marvin Lee Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge Mass., expanded ed. edition, 1988.
- [28] Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J Niels Rosenquist. Understanding the demographics of twitter users. In *ICWSM*, 2011.
- [29] Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. In *ICWSM*, 2011.
- [30] Christopher Poultney, Christopher Poultney, Sumit Chopra, and Yann” Lecun. Efficient learning of sparse representations with an energy-based model. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS 2006)*, 2006.
- [31] M. Ranzato, Fu Jie Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, 2007.
- [32] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, Vol. 40, No. 3, pp. 56–58, March 1997.
- [33] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, Vol. 65, No. 6, p. 386, 1958.
- [34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.
- [35] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.
- [36] H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, and Lyle H. Ungar. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PLoS ONE*, Vol. 8, No. 9, p. e73791, 09

2013.

- [37] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, pp. 958–, Washington, DC, USA, 2003. IEEE Computer Society.
- [38] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. 1986.
- [39] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.
- [40] Markus Thom and Günther Palm. Sparse activity and sparse connectivity in supervised learning. *J. Mach. Learn. Res.*, Vol. 14, No. 1, pp. 1091–1143, April 2013.
- [41] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Vol. 28, pp. 1058–1066. JMLR Workshop and Conference Proceedings, May 2013.
- [42] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.
- [43] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.
- [44] 川人光男. 脳の計算理論. 産業図書, 1998.