バックプロパゲーションの高速学習アルゴリズム

A study on fast learning algorithm of back propagation

－ 追記学習 －

Incremental learning algorithm

益岡 竜介
Masuoka Ryusuke
(株)富士通研究所
FUJITSU LABORATORIES LIMITED

1. 序

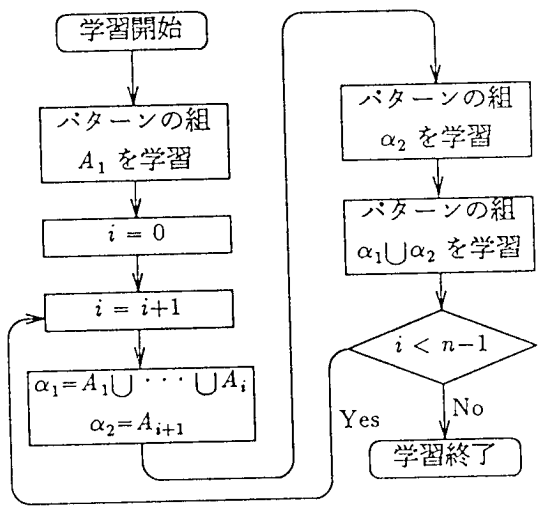
いろいろな問題に対するバックプロパゲーションの有効性は示されつつあるが，実用上で問題になるのは，学習するデータが増えた場合に最初から学習をやり直さなければいけないことや，その学習にかかる計算量の多さである．そこで，学習データを幾つかの組に分け，その組ごとに新規の学習と復習を繰り返すというアルゴリズムを開発した．このアルゴリズムを追記学習と名付けた．追記学習のアルゴリズムにより，学習のデータの追加が容易になり，しかも計算量を減らすことができた．本文では，追記学習アルゴリズムの説明，なぜこの形のアルゴリズムになったかの理由，このアルゴリズムを実際に使って行ったアルファベットフォントの認識実験の内容とその結果を述べる．

2. 追記学習アルゴリズムの目的

学習データの追加に対応可能とする．学習の高速化を目指す．

3. 追記学習アルゴリズムの概要

学習データを幾つかの組に分ける．例えば，大量のデータを小さなデータの組に分ける．あるいは今までの学習データに新たな学習データが増えた場合に，今までの学習データと新たな学習データをそれぞれを１つづつの組にしたりする．そのようにして，得られたデータの組を $\{A_1, A_2, \cdots, A_n\}$ とする．このとき第１図が追記学習のアルゴリズムとなる．つまりある学習データの組 $\alpha_1 = A_1 \cup \cdots \cup A_i$ を学習している状態のネットワークに，更に別の学習データの組 $\alpha_2 = A_{i+1}$ を追加して学習させたい場合に次のようにするということである．まず，α_2 を学習させる．そして，次に $\alpha_1 \cup \alpha_2$ を学習させて，α_1 と α_2 の両方の学習が完了した状態にする．最初の組 A_1 以外は「新規学習フェーズ」と「復習フェーズ」の繰り返しである．



第１図 追記学習のアルゴリズム

4. なぜ追記学習アルゴリズムか

このアルゴリズムにたどり着くには，いろいろな試行錯誤があった．当研究室で，このようなアルゴリズムを考え始めた最初の動機は，ある学習データを学習したネットワークに後から別の学習データを

追加したいということであった．当研究室で行っていた具体的なアプリケーションは，ニューラルネットワークにロボットの行動制御をやらせるというもので，行動パターンをニューラルネットワークで，学習させていた．しかし，このような行動パターンの組でよいだろうとある行動パターンの組を学習させても，実際に動かして見ると，学習させたい別の行動パターンが増える．今までは，前の行動パターンの組と新しい行動パターンの組を合わせた1つの学習データ（行動パターンの組）として，全くの初期状態のネットワークにその学習データの組を学習させるしかなかった．しかし，最初からネットワークを学習し直すというのは非常に不経済である．そこで，少し乱暴だが，既に前の学習データを学習しているネットワークに新しい学習データだけを学習させて，ネットワークの状態を調べてみた．すると前の学習データも，新しい学習データも覚えていた．すなわち追記学習のアルゴリズムは，学習データ α_1 を学習している状態のネットワークに，直接別の学習データ α_2 を学習させるアルゴリズムからスタートした．

```
pattern <A>
0 0 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 1 1 1 1 1 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0
->
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

第2図 学習データ（アルファベットのフォント）のデータの一部

では，このアルゴリズムを一般の場合に適用できないかということで，このアルゴリズムをアルファベットのフォントの認識に適用してみたが，全くうまく行かなかった．しかし考えて見れば，このアルゴリズムが一般に適用できないのは明らかである．α_1 を学習した後に α_2 を学習すれば，α_1 を忘れてしまうのは，当然のことである．特にアルファベットのフォントの認識のデータなどは（第2図参照），教師信号が，２６ユニット中１ユニットだけが出力1で，それ以外は0である．よってアルファベットのフォントの認識の学習では，一部分のデータを学習した後では，学習したデータに対応した出力ユニット以外のユニットの出力は0に近いものになる．{A, ..., M} を学習させた後に，{N, ..., Z} を学習させると，A を入力しても A に対応する出力ユニットの出力はほとんど0となっていた．ロボットの行動の学習の際の学習データを調べて見ると，この学習データでは，第二の学習データの組 α_2 のほとんどのデータが第一の学習データ α_1 のどれかと非常に似ていた．そのために α_2 を学習した後でも，α_1 の学習結果が壊されずに残っていたようだ．

しかし，一般の学習データに対しても後から追加可能である高速なアルゴリズムがあることが望ましい．そこで色々なアルゴリズムを試して見た．まず，α_1 を学習させて，その状態のネットワークにすぐに $\alpha_1 \cup \alpha_2$ を学習させてみた．しかしこの方法は，α_1 の影響が大きすぎるのか，α_2 の学習に時間がかかり，最初から $\alpha_1 \cup \alpha_2$ を学習させるのに比べて効率が悪い．その他にも色々なアルゴリズムを試してみた．結論としてデータの組数に関係のない一般的なアルゴリズムであり，効率良く学習出来る方法として，上に述べた追記学習のアルゴリズムを得た．

5. なぜ追記学習がうまく行くか

ちゃんと最終的に全ての学習データを学習をするかどうかは，最後に全ての学習のデータセットを合わせたものを学習させるので，大丈夫である．α_1 を学習した後に α_2 を学習する時のように前のデータを忘れてしまっているといったことはない．

効率の面で考察してみる．上に，「α_1 を学習した後に α_2 を学習すれば，α_1 を忘れてしまうのは，当然のことである．」と書いた．それでは，α_1 を学習して，次に α_2 をした後に，$\alpha_1 \cup \alpha_2$ を学習するのは，まず α_1 を学習させて，その状態のネットワークにすぐに $\alpha_1 \cup \alpha_2$ を学習させるよりも非効率ではないかと思われる．しかし，実験結果はそうではなかった．先に α_1 を学習して，次に α_2 を学習すると，α_1 を忘れてしまった後でも α_1 の記憶の痕跡といったものは残っているらしい．

追記学習のアルゴリズムで，α_1 を学習している状態のネットワークから，α_2 を学習させると，最初から α_2 を学習させるのとは違い，α_1 の影響が残っている．よって追記学習のアルゴリズムで α_2 の学習が終わった段階では，α_1 の影響が残っていて，α_2 に関しては完全に収束している状態なので，

この状態のネットワークに $\alpha_1 \bigcup \alpha_2$ を学習させると非常に少ない回数で学習が収束する．追記学習のアルゴリズムで学習回数がかかるのは，新しい学習データを学習する新規学習フェーズであり，復習フェーズの学習は非常に早く済む．学習回数のかかる部分では，学習データの数が少ないので，学習一回当たり（すなわち前回の重み更新から次の重み更新まで）に要する計算時間は短い．また学習データが多く，学習一回当たりに要する計算時間が長くかかる部分では，学習回数が減る．これらの理由により，学習に要する時間を短くでき，効率良く学習できる．

更に，追記学習のアルゴリズムが効率の良いその他の理由としては，分割したデータの学習のときには，大きなデータの時には収束しないような大きな学習定数を取ることができる．つまり小さいデータなら小回りが効くということである．後で述べる実験では，学習定数を学習パターン数に反比例させた．

6. 追記学習アルゴリズムの長所

上以外の追記学習のアルゴリズムの長所は，少しづつ学習データを追加しながら，学習データを増やしていける点である．学習データを追加しながら，学習の状態を監視していれば，うまく学習が行かなくなった時点で，学習のやり方を変えることも可能である．小さなデータに分けて，大体仕上げておいて，最後に全部のデータで復習するというものである．

また追記学習のアルゴリズムを使えば最初から全てのデータを揃える必要がないというのも大きな長所である．後から学習データが増えることが分かっていても，即，その時点の学習データで学習を始めることができる．学習データが増えた時点で，今までの学習データを１つの組とし，新たな学習データを別の１つの組とし，上の追記学習のアルゴリズムを実行すれば良い．以上のことは，学習データが増えるということが何度あっても対応可能である．

7. 実験

7.1. 追記学習の方法

以下に追記学習の方法を具体的に実験例を用いて説明する．実験はニューラル・ネットワークにバックプロパゲーション法を用いて，８ドット×８ドットのアルファベット２６文字を認識させるものである．具体的には以下のようにアルファベットを提示していく．（４分割の場合を説明した）

(a) AからZまでのパターンを４つのパターンの組に分ける．

$A(1)$={A,B,C,D,E,F,G}
$A(2)$={H,I,J,K,L,M,N}
$A(3)$={O,P,Q,R,S,T}
$A(4)$={U,V,W,X,Y,Z}

(b) $A(1)$ のパターンを提示する．
(c) $A(2)$ のパターンを提示する．
(d) $A(1)$, $A(2)$ のパターンを合わせて提示する．
(e) $A(3)$ のパターンを提示する．
(f) $A(1)$, $A(2)$, $A(3)$ のパターンを合わせて提示する．
(g) $A(4)$ のパターンを提示する．
(h) $A(1)$, $A(2)$, $A(3)$, $A(4)$ のパターンを合わせて提示する．

7.2. 実験の設定

同じ状態のネットワークからそれぞれ通常の学習と追記学習でアルファベットの認識を学習させて，学習の速さを比較した．

7.2.1. ネットワークの構造

使ったネットワークの構造は以下のようである．

入力層 64 ユニット
中間層 15 ユニット
出力層 26 ユニット

閾値は，中間層と出力層に入れた．

7.2.2. 重みの初期値と学習

重みと閾値の初期値は，$[-0.1, 0.1]$ からランダムに取った．学習は，パターン提示以外は，通常のバックプロパゲーションを用いて，閾値も重みと同様に学習させた．

7.2.3. 学習定数

学習定数は以下のように取った．

(A) learning rate

学習定数はパターン数に反比例して取った．

$$\epsilon = \frac{5.2}{p}$$

ただし，p は学習させるパターン数である．
例えば，$A(1) \bigcup A(2)$ を学習させている時は，p=14 であり，ϵ=5.2/14 である．

(B) momentum

一律 $\alpha = 0.4$ とした．

第３図提示回数と総計算量の表

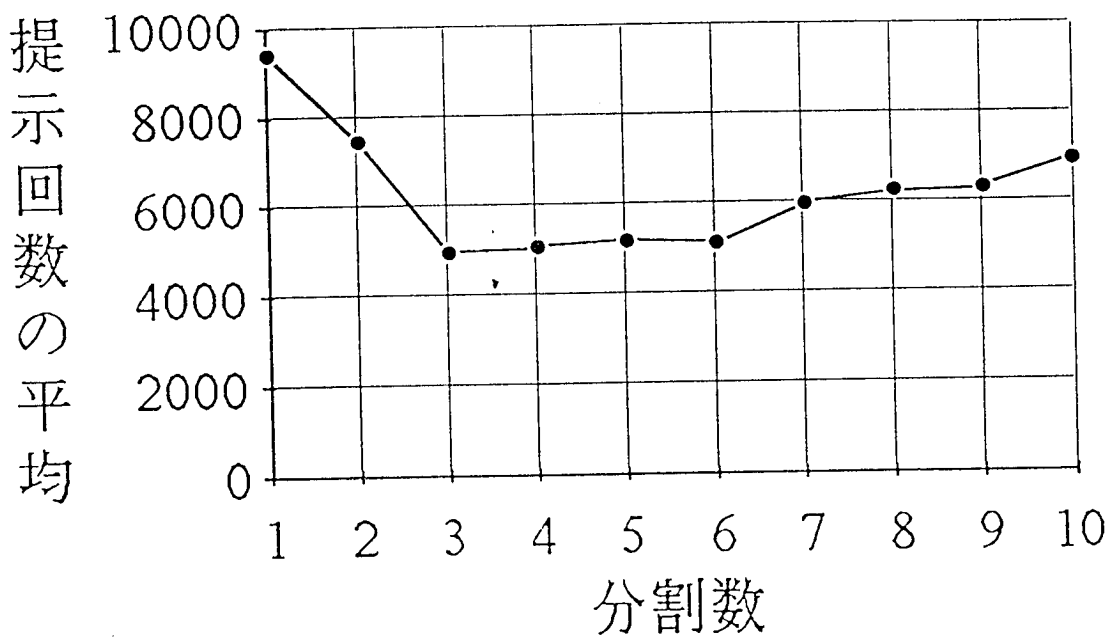
追記学習：[A]分割数による提示回数の違い

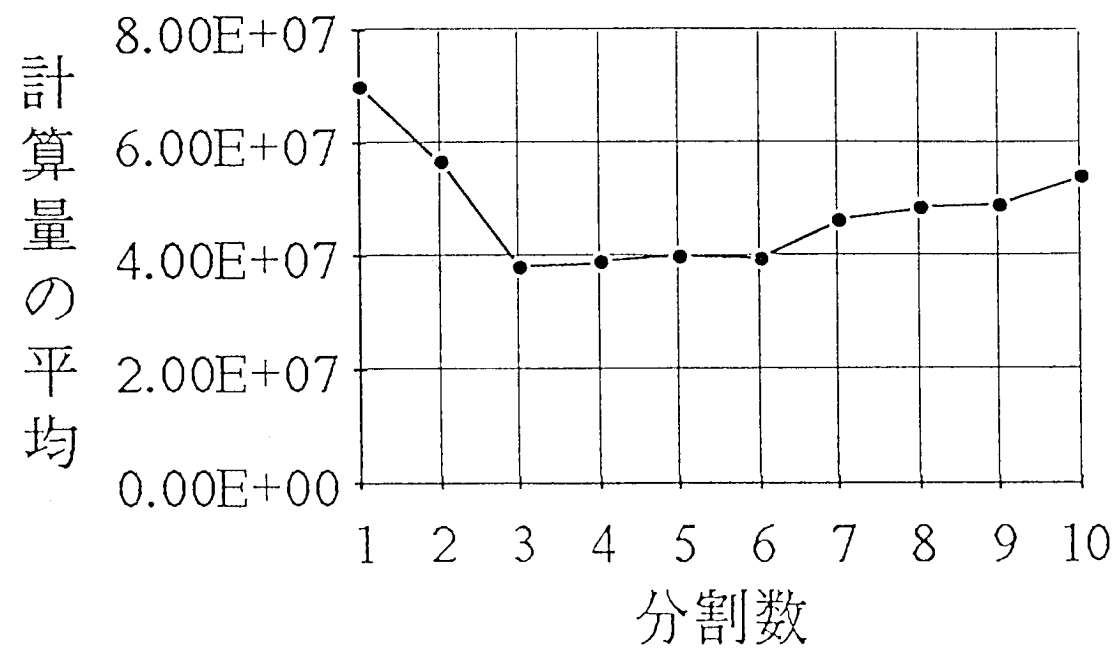
seed	1分割	2分割	3分割	4分割	5分割	6分割	7分割	8分割	9分割	10分割	13分割	26分割
100000	8398	6825	4516	4476	4760	5513	5552	6123	6858	6917	9922	72201
110000	9698	5928	4466	4145	5470	5193	6758	6387	4608	6268	15424	69344
120000	9724	7137	5857	5376	5300	4582	6233	5913	6905	6042	13268	34168
130000	9230	6591	3933	5323	5607	4514	5477	5935	5868	6242	13428	166422
140000	10530	7891	5670	5633	5242	4781	5921	6791	6506	7072	18402	139868
150000	9490	8268	5829	5821	4917	4762	5425	5549	6426	7699	13470	69014
160000	8814	9607	5670	4476	5030	4900	6479	5955	7613	8677	42364	69659
170000	10244	6188	4268	4727	4372	5677	5696	6195	6169	6879	47488	69227
180000	9048	9295	5079	5929	5072	5583	5830	5674	6474	7138	7758	46958
190000	8996	6903	4049	4503	5771	5324	6427	7797	5608	6387	11072	109766
平均	9417	7463	4933	5040	5154	5082	5959	6231	6303	6932	19259	84662
最大値	10530	9607	5857	5929	5771	5677	6758	7797	7613	8677	47488	166422
最小値	8398	5928	3933	4145	4372	4514	5425	5549	4608	6042	7758	34168
分散	620	1198	733	613	394	407	454	619	776	755	13175	39283

追記学習：[B]分割数による総計算量の違い

seed	1分割	2分割	3分割	4分割	5分割	6分割	7分割	8分割	9分割	10分割	13分割	26分割
100000	62691070	52125411	34815324	34647692	36979612	42706687	43117448	47580481	53154782	53810715	77536242	579175015
110000	72395570	45292988	34608566	32297687	42422714	40447571	52546526	49900917	36022124	48526160	119501052	558458908
120000	72589660	54482311	45251107	41718864	41139968	35894802	48695239	46202435	53829419	47365938	103038860	260566060
130000	68901950	50373037	30510095	41156513	43455049	35279326	42779491	46300753	45792680	48643654	106547028	1262371486
140000	78606450	60300097	43722686	43849443	40992046	37476851	46282699	52796937	50936270	55240640	142685834	1069828432
150000	70842850	63267412	45504327	45546471	38685547	37305698	42675075	43908039	50267166	59965065	105199062	556066078
160000	65796510	73505081	44251266	35109504	39276786	38556716	50634777	46824125	59513963	67546175	325564820	560742973
170000	76471460	47334040	33256328	36634613	34232992	44246383	44361896	48147065	48025307	53362693	365484252	557610541
180000	67543320	70959005	39242605	45851075	39742684	43581481	43944534	44185682	50414910	55340854	60487462	357407018
190000	67155140	52629785	31417979	34893545	44327065	41224968	49640121	60530999	43595836	49472489	87265892	851558830
平均	70299398	57026916	38258028	39170540	40125446	39672048	46467780	48637743	49155245	53927438	149331050	661378534
最大値	78606450	73505081	45504327	45851075	44327065	44246383	52546526	60530999	59513963	67546175	365484252	1262371486
最小値	62691070	45292988	30510095	32297687	34232992	35279326	42675075	43908039	36022124	47365938	60487462	260566060
分散	4632859	9170801	5702344	4759412	2887124	3060331	3455270	4675200	6058976	5881919	100764760	294376902

第４図 提示回数と総計算量のグラフ





7.2.4. 学習の収束判定

学習の収束判定は，教師信号（0 と 1 にとった）からの差が各ユニットで 0.4 未満になったときとした．

7.2.5. データの構造

入力信号は各アルファベットについて 8 × 8ドットのフォントのデータを 0, 1 の 2 値としたものを与えた．教師信号は各アルファベットにつきそれぞれ異なる一つのユニットだけが 1 でその他は全て 0 としたものを与えた．データの数は A から Z までの 26 個である．データの例は，第2図にある．

8. 実験結果

この追記学習では，学習一回ごとに提示するパターン数が異なるために，単純に学習回数を比較したのでは，計算に必要なリソースを測る事が出来ない．ここでは，通常の学習との比較のために，二つの量による比較の表を用意した．

[A] 提示回数

計算に必要なリソースを測るもっと簡単な指標としては，パターン提示回数がある．これは一回ごとに提示するパターン数を足しあわせていったものである．表中の [A] 提示回数は，学習が完了するまでのパターン提示回数を表にしたものである．

[B] 総計算量

浮動少数点の足し算と掛け算の回数としての計算量(sigmoid 関数の計算に 30 回要するとした）は，一回の重みの更新に関して

$$7251p+5564$$

要する．ただしここで p は提示するパターン数である．表中の [B] 総計算量は，学習が完了するまでの計算量を全て足しあわせたものを表にしたものである．

表中 seed とあるのは，乱数の発生用の seed である．この値でネットワークの最初の重みが定まる．つまり seed が同じなら，ネットワークの最初の状態は同じである．

分割の仕方は，1分割が全く分割しない場合で，A から Z を1つの組にして提示する通常のバックプロパゲーションの学習方法である．

$A(1)$={A,B,C,D,E,F,G,H,I,J,K,L,M}
$A(2)$={N,O,P,Q,R,S,T,U,V,W,X,Y,Z}

が2分割の場合であり

$A(1)$={A,B,C,D,E,F,G,H,I}
$A(2)$={J,K,L,M,N,O,P,Q,R}
$A(3)$={S,T,U,V,W,X,Y,Z}

が3分割の場合である．他の分割数の場合も同様で，分割数で 2 6 を割り切れない場合は，分割数を g としたとき，$A(1)$ から $A(26-g\times\lfloor 26/g\rfloor)$ の各組に $\lfloor 26/g\rfloor+1$ 個のアルファベットを割り当て，それ以外の組に $\lfloor 26/g\rfloor$ 個のアルファベットを割り当てた．第3図が，各分割における提示回数と，総計算量の表であり，第4図が，それらをグラフにしたものである．

これらの表及びグラフから読み取れるのは，以下のことである．同じ分割数でも重みの初期値によって，学習にかかる提示回数や総計算量にばらつきがあることが分かる．しかし1分割から3分割までは，分割数の多い方が，提示回数や総計算量の平均，最大値，最小値の全てに渡って減っている．3分割までは明らかに高速化の効果があることが分かる．その効果は，このアルファベット認識実験の場合に1分割と3分割の場合を比較して，約2倍である．4分割から，6分割まででは，3分割の場合と提示回数や総計算量は余り変わらないが，3分割の場合と比べて分散が小さくなった．4から6組ぐらいに分けた方が，3分割の場合などより，重みの初期値に影響されずに学習が安定して行われる．更に分割数を増やしていくと，徐々に学習の効率が悪くなる．1 3分割以上では，全く分割しない1分割の場合と比べても学習の効率が悪い．

9. 結論

追記学習のアルゴリズムの概要や，このアルゴリズムを得ることになった過程を説明した．アルファベットのフォントの認識の学習を用いて，通常の学習アルゴリズムと追記学習アルゴリズムの効率の比較をする実験を行った．その結果，後者は後から別の学習データが追加可能でありながら，通常の学習より効率が良いという実験結果を得た．追加可能でありながら，効率の良いアルゴリズムが得られた結果，実際のアプリケーションにおいて学習が容易となり，学習を柔軟に行えるようになった．更に組み分けの仕方などを工夫することにより，いっそう学習の効率を上げ得る可能性がある．

10. 謝辞

日頃から御指導頂く棚橋純一部門長，林弘部長ならびに人工知能第二研究室の皆様に感謝します．