研究ノート

階層型ニューラルネットワークの学習に対する online/batch ハイブリッド型準ニュートン法の有効性に関する研究

阿部俊和¹, 坂下善彦², 二宮 洋³

¹ 湘南工科大学大学院工学研究科電気情報工学専攻 〒251-8511 藤沢市辻堂西海岸 1-1-25

^{2,3} 湘南工科大学工学部情報工学科 〒251-8511 藤沢市辻堂西海岸 1-1-25

E-mail: ¹10T2002@sit.shonan-it.ac.jp, ³ninomiya@info.shonan-it.ac.jp

On the Study of Hybridized online/batch quasi-Newton Training for Feedforward Neural Networks

Toshikazu Abe¹, Yoshihiko Sakashita², Hiroshi Ninomiya³

¹ Graduate School of Electrical and Information Engineering, Shonan Institute of Technology

1-1-25 Tsujido-Nishikaigan, Fujisawa, Kanagawa 251-8511, Japan

^{2,3} Department of Information Science, Faculty of Engineering, Shonan Institute of Technology

1-1-25 Tsujido-Nishikaigan, Fujisawa, Kanagawa 251-8511, Japan

E-mail: ¹10T2002@sit.shonan-it.ac.jp, ³ninomiya@info.shonan-it.ac.jp

Abstract Various techniques based on the gradient descent method have been studied as training algorithms for neural networks. Neural network training poses data-driven optimization problems in which the objective function involves the summation of loss terms over a set of data to be modeled. For a given training data set, the gradient-based algorithm operates in one of two modes: online (stochastic) or batch. In this paper, a robust training algorithm is proposed, combining “online” mode with “batch” one. The validity of the proposed algorithm is demonstrated through computer simulations compared with the previous quasi-Newton based training methods.

Keywords: feedforward neural network, quasi-Newton method, online training algorithm, batch training algorithm

1. はじめに

ニューラルネットワークの学習問題に対する最も有効な解法の1つに BFGS 公式を用いた準ニュートン法に基づく勾配学習アルゴリズムがある[1]。また、この手法に関する研究に、学習時のデータの与え方に着目したオンライン学習及びバッチ学習がある[1][2]。これらの手法の特徴として、以下の点が挙げられる。バッチ学習はすべての学習データをネットワークへ入力後に重みを更新する。このため、収束は速いが、1回の反復に多くの計算量が必要となり、特に、機械学習のような冗長で膨大な学習データを扱う場合、計算時間が問題となる。一方、オンライン学習は、収束は遅いが、1回の反復に要する計算量が少なく、また、局所最適解を抜け出す能力がバッチ学習と比較して高くなる[1]。さらに、オンライン学習とバッチ学習を組み合わせることで、マイクロ波回路モデリング等の非線形性の強い関数を学習することを可能にしたアルゴリズムが提案された[3][4]。この手法は、1回の反復で用いる学習

データ数を、反復が進むに従って徐々に増加させることで収束性を高めている。つまり、学習初期段階ではオンライン学習で局所解の回りを探索し、最終的にバッチ学習に切り替えるメカニズムを導入している。このメカニズムは準ニュートン法以外の勾配法(最急降下法、共役勾配法、記憶制限準ニュートン法、自然勾配法)に応用され、実験的ではあるが、その有効性が示されている[5]。ただし、準ニュートン法への応用が最も効果的であった。一方、準ニュートン法において、学習データの与え方を1つのパラメータにより表現することを可能にしたアルゴリズムが提案された[6][7]。この学習法はオンライン学習とバッチ学習の誤差関数をパラメータで関連付けることでオンラインからバッチへとアルゴリズムを変化させる手法である。さらに、このメカニズムがシミュレーテッドアニーリング(SA)に類似していることが示されている。これらの手法は、バッチ学習やオンライン学習と比較して非線形性の強い関数の学習に有効であった。しかしながら、複数回

の計算機シミュレーションにおいて、誤差の大きな局所解に収束してしまう場合が存在した。また、SAと同様に、最適解への高い収束性を得るためには、膨大な反復が必要であった。

本研究では、[3]及び[6][7]のオンラインからバッチへの変化のメカニズムに着目し、それらを組み合わせた、新たな学習アルゴリズムを提案する。つまり、本研究では、オンラインからバッチへの変化のメカニズムの二重化により、SAの様な多くの反復を必要とせず、局所解を抜け出す能力を持つアルゴリズムを実現する。いくつかのベンチマーク問題に対する計算機シミュレーションにより、提案手法の有効性を示す。

2. 階層型ニューラルネットワークと勾配学習

2.1 階層型ニューラルネットワーク

本研究では入出力関係として(1)の関係を持つ階層型ニューラルネットワーク(LNN)を考える。

$$\mathbf{o}_p = f_{NN}(\mathbf{w}, \mathbf{x}_p) \quad (1)$$

ここで、 $\mathbf{x}_p$ は p 番目の入力ベクトルを、 $\mathbf{o}_p$ は $\mathbf{x}_p$ に対するLNNの出力ベクトルを示す。また、LNNの重みベクトルを $\mathbf{w}$ とする。中間層数、及び、それらのニューロン数は任意とし、中間層ニューロンの入出力関係はシグモイド関数とする。さらに、 $\mathbf{x}_p$ に対する教師信号ベクトルを $\mathbf{d}_p$ とすると、平均二乗誤差関数は(2)で定義される。

$$E(\mathbf{w}) = \frac{1}{P_{T_r}} \sum_{p \in T_r} E_p(\mathbf{w}) = \frac{1}{P_{T_r}} \sum_{p \in T_r} \frac{1}{2} \|\mathbf{d}_p - \mathbf{o}_p\|^2 \quad (2)$$

ここで、 T_r は学習データセットを示し、学習データ数を P_{T_r} とする。また、 $E_p(\mathbf{w})$ は p 番目の学習パターンに対する誤差関数を示し、 $E_p(\mathbf{w}) = 1/2 \|\mathbf{d}_p - \mathbf{o}_p\|^2$ とした。学習はLNNの重み $\mathbf{w}$ に関して(2)を最小にする無制約最適化問題となる。

2.2 勾配学習

勾配学習では一般に(3)に示す反復式により(2)を最小にする重みベクトル $\mathbf{w}$ を求める。

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{g}_k \quad (3)$$

ここで、 k は反復回数を示す。 $\mathbf{g}_k$ は勾配ベクトルを示し、バッチ学習では、勾配ベクトルは(2)の勾配、 $\mathbf{g}_k = -\partial E(\mathbf{w}_k)/\partial \mathbf{w}_k$ を用いる。オンライン学習では、一般に、学習パターン毎の誤差 $E_p(\mathbf{w})$ の勾配ベクトル $\mathbf{g}_k = -\partial E_p(\mathbf{w}_k)/\partial \mathbf{w}_k$ を用いることになる。また、学習係数 α_k は勾配アルゴリズムによって異なり、最急降下法の場合は正数(スカラー)となり、準ニュートン法の場合は正定値行列となる。

本研究では勾配アルゴリズムとして準ニュートン法を用いる。従来、準ニュートン法に基づく学習では(2)の勾配を用いたバッチ学習であった[1]。特にBFGS公式に基づく準ニュートン法(以下、BFGS)は現在の無制約非線形計画法の中では最も有効な手法の1つである[8]。一方、BFGSの苦手とする機械学習のようなパターン数が冗長で膨大な問題に対して、オンライン準ニュートン法(以下、oBFGS)が提案された[3]。さらに、oBFGSをより複雑な入出力特性を持つ問題の学習が可能なアルゴリズムへと改良した、改良型オンライン準ニュートン法(Improved oBFGS, ioBFGS)が提案された[3]。ioBFGSでは1回の反復に用いる学習データ数を、反復が進むにつれて増加させることで、オンラインからバッチへと学習を変化させた。また、この変化のメカニズムを1つのパラメータにより表現し、その収束特性をさらに向上させた、パラメータ化オンライン準ニュートン法(Parameterized oBFGS, poBFGS)が提案された[6][7]。これらの手法は、オンライン学習(oBFGS)もしくはバッチ学習(BFGS)と比較して非常に高い収束性を得られる学習法である。さらに、poBFGSのオンラインからバッチへの学習変化とSAの類似性が示された[7]。しかしながら、その変化の過程がSAと同様な温度制御を必要とするpoBFGSは、常に最適解を得る為には、膨大な反復回数を要した。

3. ハイブリッド型準ニュートン法

3.1 poBFGS [6][7]

本節では、まず、提案手法で繰り返し用いられる、poBFGSに関して簡単に説明する。poBFGSでは、パラメータ μ を用いて学習をオンラインからバッチへと変化させるメカニズムを導入する。つまり、poBFGSでは、 k 回目の反復に対して誤差関数(4)を導入した。

$$E_{po}(\mathbf{w}^k) = (1 - \mu)E(\mathbf{w}^k) + \mu E_{p(k)}(\mathbf{w}^k) \quad (4)$$

ここで、 $p(k)$ は k 回目の反復に用いる p 番目の学習データを、 $E_{p(k)}(\mathbf{w}^k)$ はその誤差関数を示す。(4)に対して、 μ を k の増加に伴って1から0に近づけることで、poBFGSは、オンラインからバッチへと変化する勾配学習法となった。さらに、 μ をSimulated Annealing(SA)[9]における山登り確率とみなすことで、同様のクーリングスケジュールを用いて1から0に変化させる。つまり、 μ を、SAと同様に j 番目のメトロポリスループ[9]に対して、温度パラメータ T^j を用いて次式のように定義する。

$$\mu^j = \exp \left(\frac{E_{best}(\mathbf{w}) - E_{worst}(\mathbf{w})}{E_{best}(\mathbf{w})} \cdot \frac{1}{T^j} \right) \quad (5)$$

ここで、 $E_{best}(\mathbf{w})$ 及び $E_{worst}(\mathbf{w})$ は、それぞれ、同一メ

トロポリスループ内で得られた $E(\mathbf{w})$ の最小値, 及び, 最大値とする。ここで, メトロポリスループとは温度 T^j が一定での反復過程のことである[9]。また, j 番目のメトロポリスループ内では, T^j を一定とすることで, パラメータ μ^j を固定した誤差関数(4)に対して複数回の反復を実行することになる。従って, poBFGS は, 適当なクーリングスケジュールを用いて温度 T^j を冷却することにより, オンライン学習からバッチ学習へと変化する勾配学習法となる。

3.2 提案アルゴリズム (hBFGS)

本研究では, ioBFGS 及び poBFGS のメカニズムを組み合わせることで, 膨大な反復回数を費やすことなく, その学習能力を実現する手法を提案する。提案手法では, まず, ioBFGS で提案されている1回の反復に用いる学習データ数を徐々に増加させる手法に着目し, その更新過程で poBFGS による学習を適用することを考える。つまり, poBFGS ではバッチ学習とオンライン学習の誤差関数を1つのパラメータ μ を用いて(4)に示すように結合している。これに対し, 提案手法では, poBFGS のバッチ学習の誤差関数, 具体的には(4)の $E(\mathbf{w}^k)$ を, ioBFGS で用いられた学習データの部分集合に対する誤差関数に置き換える。従って, 提案手法は, 学習データの増加とパラメータによる制御を組み合わせることで, オンライン学習からバッチ学習への変化を実現する手法である。提案手法は以下の手順に従ってオンラインからバッチへのメカニズムを実現する。学習の初期では, 学習の対象となる関数(学習対象関数)の全学習データセット T_r をある一定間隔(seg)で選択した, 学習データの部分集合に対して poBFGS を用いて学習を行う。次に, この部分集合を用いて学習したLNNを初期値として, 選択間隔を狭めることで学習データを増加させた部分集合を用いた, より詳細な学習対象関数を, 再び poBFGS を用いて学習する。この操作を, 学習データ数を徐々に増加させながら繰り返す。最終的には選択間隔をゼロとし, 学習対象関数のモデル化に必要なすべてのサンプル点を含んだ学習データによるバッチ学習に変化させる。以下に, 具体的なアルゴリズムを示す。

全学習データセット T_r を(6)のように定義する。

$$T_r = \{T_{r,0}, T_{r,2}, \dots, T_{r,(P_{T_r}-1)}\} \quad (6)$$

学習開始時は(6)のデータを, seg 間隔毎に選択して学習データの部分集合(7)を用意する。

$$T_r^{\text{hBFGS}(1)} = \{T_{r,0}, T_{r,\text{seg}}, T_{r,(\text{seg} \times 2)}, \dots, T_{r,(P_{T_r}-\text{seg})}\} \quad (7)$$

(7)のデータセットを用いた誤差関数を, poBFGS のバッチ学習の誤差評価と置き換える。つまり, (4)の $E(\mathbf{w}_k)$ を, $T_r^{\text{hBFGS}(1)}$ による誤差関数 $E_{m(1)}(\mathbf{w}_k)$ に置き換えて, このステップにおける誤差関数 $E_{h(1)}(\mathbf{w}_k)$ を, パラメータ $\mu_{(1)}$ を用いて以下のように定義する。

$$E_{h(1)}(\mathbf{w}_k) = (1 - \mu_{(1)})E_{m(1)}(\mathbf{w}_k) + \mu_{(1)}E_{p(k)}(\mathbf{w}_k) \quad (8)$$

ここで,

$$E_{m(1)}(\mathbf{w}) = \frac{1}{(P_{T_r}/\text{seg})} \sum_{p \in T_r^{\text{hBFGS}(1)}} E_p(\mathbf{w}) \quad (9)$$

である。 $E_{h(1)}(\mathbf{w}_k)$ に対して, poBFGS と同様に, $\mu_{(1)}$ を1から0に変化させながら学習を行う。これにより, オンライン学習から部分集合 $T_r^{\text{hBFGS}(1)}$ のバッチ学習へ変化させる。このステップを hBFGS(1)と表す。次のステップ, hBFGS(2)では, 選択データの間隔をseg/2とし, 学習データ数を2倍にした部分集合(10)を用意する。

$$T_r^{\text{hBFGS}(2)} = \{T_{r,0}, T_{r,(\text{seg}/2)}, T_{r,\text{seg}}, \dots, T_{r,(P_{T_r}-(\text{seg}/2))}\} \quad (10)$$

(10)のデータセットによる誤差関数 $E_{m(2)}(\mathbf{w}_k)$ を用いた, $E_{h(2)}(\mathbf{w}_k)$ を用意する。この誤差関数に対して, hBFGS(1)で得られたLNNを初期値として, $\mu_{(2)}$ を1から0へ変化させながら poBFGS を実行する。従って, hBFGS(2)の初めにパラメータ $\mu_{(2)}$ は1に初期化される。この操作をc回繰り返すと, hBFGS(c)での学習データセットは

$$T_r^{\text{hBFGS}(c)} = \{T_{r,[(\text{seg}/c) \times i]} | i = 0, 1, \dots, [(P_{T_r}/\text{seg}) \times c]\} \quad (11)$$

となり, 誤差関数は

$$E_{h(c)}(\mathbf{w}_k) = (1 - \mu_{(c)})E_{m(c)}(\mathbf{w}_k) + \mu_{(c)}E_{p(k)}(\mathbf{w}_k) \quad (12)$$

と定義される。ここで $E_{m(c)}(\mathbf{w}_k)$ は以下である。

$$E_{m(c)}(\mathbf{w}) = \frac{1}{[(P_{T_r}/\text{seg}) \times c]} \sum_{p \in T_r^{\text{hBFGS}(c)}} E_p(\mathbf{w}) \quad (13)$$

従って, c回目の学習データの更新後, $T_r^{\text{hBFGS}(c)}$ による誤差関数 $E_{m(c)}(\mathbf{w})$ を用いて定義された $E_{h(c)}(\mathbf{w})$ に対して, $\mu_{(c)} = 1$ の時, オンライン学習となり, $\mu_{(c)} = 0$ の時, $T_r^{\text{hBFGS}(c)}$ のバッチ学習となる。つまり, $E_{h(c)}(\mathbf{w})$ に対して, $\mu_{(c)}$ を1から0に変化させることで, poBFGS のメカニズムを学習データの部分集合に対して導入することになる。さらに, この部分集合の学習データ数を徐々に増加させていくことで, ここでも, オンライン学習からバッチ学習へのメカニズムを導入することになる。これにより, 従来の poBFGS では, バッチ学習の誤差関数がすべての学習データを含んだ $E(\mathbf{w}_k)$ であったため, パラメータ μ を非常にゆっくりと変化させる必要があった。つまり, 多

くの反復が必要であった。しかし、提案手法では、部分集合の更新制御を導入することで、 μ の変化を poBFGS と比較して、急激に変化させる制御が可能となる。結果として、十分に多くの反復回数を伴った poBFGS と同様なオンラインからバッチへの変化のメカニズムが保たれることになる。ここで、hBFGS(c)において、 $\mu_{(c)}$ は、poBFGS と同様に、 j 番目のメトロポリスループにおける温度 $T_{(c)}^j$ を用いて次式のように表現される。

$$\mu_{(c)}^j = \exp\left(\frac{E_{m(c)}^{best}(\mathbf{w}) - E_{m(c)}^{worst}(\mathbf{w})}{E_{m(c)}^{best}(\mathbf{w})} \cdot \frac{1}{T_{(c)}^j}\right) \quad (14)$$

ここで、同一のメトロポリスループ内では温度 $T_{(c)}^j$ は変化しない。従って、 $\mu_{(c)}^j$ も変化しない。また、 $E_{m(c)}^{best}(\mathbf{w})$ 及び $E_{m(c)}^{worst}(\mathbf{w})$ は、それぞれ、同一メトロポリスループ内の誤差関数 $E_{m(c)}(\mathbf{w})$ の最小値、及び、最大値とする。(14)より、 $T_{(c)}^j$ が大きいときは $\mu_{(c)}^j$ は 1 となり、アルゴリズムはオンライン学習となる。また、徐々に $T_{(c)}^j$ を下げていくことにより、 $\mu_{(c)}^j$ は 0 に近づき $T_r^{hBFGS(c)}$ に対するバッチ学習となる。 $T_{(c)}^j$ を制御するクーリングスケジュールは、一般的な SA と同様に、クーリングパラメータ ρ を用いて

$$T_{(c)}^j = \rho^j \times T_{(c)}^1 \quad (15)$$

とする[6][7]。さらに、 j 番目のメトロポリスループ内の $\mathbf{w}$ の更新回数 $N_{(c)}^j$ は

$$N_{(c)}^j = (P_{Tr}/(P_{Tr} - 10))^j \times P_{Tr} \quad (16)$$

とする[6][7]。また、hBFGS(c)の初めに、 $T_{(c)}^j$ 、 $\mu_{(c)}^j$ 及び $N_{(c)}^j$ は、それぞれ、 $T_{(c)}^1 = k_{max} \times E(\mathbf{w}_k)$ 、 $\mu_{(c)}^1 = 1$ 及び $N_{(c)}^1 = P_{Tr}$ に初期化される。ここで、hBFGS(c)における反復回数 k 、メトロポリスループの反復回数 j 、及び、パラメータ $\mu_{(c)}^j$ の関係を図 1 に示す。図 1 の $k_{(c-1)}$ は hBFGS(c-1)が終了した時点での反復回数である。以上より、 $\mu_{(c-1)}^j$ が 0 となつて、hBFGS(c-1)ステップが終了すると、メトロポリスループの反復 j が $j=1$ に初期化される。同時に、 $T_{(c)}^1 = k_{max} \times E(\mathbf{w}_k)$ 、 $\mu_{(c)}^1 = 1$ 及び $N_{(c)}^1 = P_{Tr}$ として、新たな学習データの部分集合 $T_r^{hBFGS(c)}$ に対する poBFGS ルー

プ、つまり、hBFGS(c)が開始されることになる。従って、従来の poBFGS では反復回数を増やすことで実現していた変化のメカニズムを、提案手法ではオンラインからバッチへの変化を二重化することで実現した。これにより、パラメータ μ の変化の過程を短くすることが可能となり、結果として、少ない反復回数でも最適値を求めることが可能な学習法となる。本稿では、提案手法をハイブリッド型準ニュートン法(hBFGS)と呼ぶ。hBFGS をアルゴリズム 1 に示す。アルゴリズム 1 の BFGS 公式に基づく近似行列 $\mathbf{H}_k$ の更新式は以下で与えられる。

$$\begin{aligned} \mathbf{H}_{k+1} &= \mathbf{H}_k - ((\mathbf{H}_k \mathbf{y}_k) \mathbf{s}_k^T + \mathbf{s}_k (\mathbf{H}_k \mathbf{y}_k)^T) / \mathbf{s}_k^T \mathbf{y}_k \\ &\quad + (1 + \mathbf{y}_k^T \mathbf{H}_k \mathbf{y}_k / \mathbf{s}_k^T \mathbf{y}_k) \mathbf{s}_k \mathbf{s}_k^T / \mathbf{s}_k^T \mathbf{y}_k \end{aligned} \quad (17)$$

$$\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k \quad (18)$$

$$\begin{aligned} \mathbf{y}_k &= \frac{\partial E_{h(c)}(\mathbf{w}_{k+1})}{\partial \mathbf{w}_{k+1}} - \frac{\partial E_{h(c)}(\mathbf{w}_k)}{\partial \mathbf{w}_k} + \lambda_k \mathbf{s}_k \\ &= \mathbf{y}_k + \lambda_k \mathbf{s}_k \end{aligned} \quad (19)$$

ここで、 T は転置を示す。 λ_k は $\mathbf{H}_k$ の正定値性を保証するための係数であり、以下のように決定する[3]。

$$\begin{aligned} \lambda_k &= \eta \|\partial E_{h(c)}(\mathbf{w}_k) / \partial \mathbf{w}_k\| \\ &\quad + \max\{-\mathbf{y}_k^T \mathbf{s}_k / \|\mathbf{s}_k\|^2, 0\} \\ \left\{ \begin{array}{ll} \eta = 2 & \text{if } \|\partial E_{h(c)}(\mathbf{w}_k) / \partial \mathbf{w}_k\| > 10^{-2} \\ \eta = 100 & \text{if } \|\partial E_{h(c)}(\mathbf{w}_k) / \partial \mathbf{w}_k\| < 10^{-2} \end{array} \right. \end{aligned} \quad (20)$$

また、提案手法の Line search(アルゴリズムの Step6)では、各反復で誤差関数が増えるため、以下に示す Modified Armijo's criterion を用いて η_k を求めた[3]。

<Line search using Modified Armijo's criterion>

$0 < \xi < 1$ であるような定数に対して、

$$\begin{aligned} E_{h(c)}(\mathbf{w}_k + \eta_k \mathbf{H}_k \mathbf{g}_k) \\ \leq E_{h(c)}(\mathbf{w}_k) + \xi \cdot \eta_k \cdot \partial E_{h(c)}(\mathbf{w}_k) / \partial \mathbf{w}_k^T \cdot \mathbf{H}_k \mathbf{g}_k \end{aligned} \quad (21)$$

を満たす $\eta_k \geq 0$ を選ぶ。本研究では、 $\xi = 10^{-4}$ とした。

4. シミュレーション結果

本章では 3 層の LNN(3LNN)を用いて BFGS, oBFGS, ioBFGS 及び poBFGS に対して hBFGS の有効性をシミュレーションにより示す。例題には、関数近似問題のベンチマーク(22)-(24)を用いる[3][6]。

k	$k_{(c-1)} + 1, k_{(c-1)} + 2, \dots, k_{(c-1)} + N_{(c)}^1$	$k_{(c-1)} + N_{(c)}^1 + 1, \dots, k_{(c-1)} + N_{(c)}^1 + N_{(c)}^2 \dots$	$k_{(c-1)} + \sum_{i=1}^{j-1} N_{(c)}^i + 1, \dots, k_{(c-1)} + \sum_{i=1}^j N_{(c)}^i \dots$
j	1	2	j
$\mu_{(c)}^j$	1 $\dots$ $\mu_{(c)}^1$ $\dots$ 0	1 $\dots$ $\mu_{(c)}^2$ $\dots$ 0	1 $\dots$ $\mu_{(c)}^j$ $\dots$ 0

図 1 hBFGS(c)における反復回数 k 、メトロポリスループの反復回数 j 、及び、パラメータ $\mu_{(c)}^j$ の関係

Fig. 1 Relation between j -th Metropolis loop and parameter $\mu_{(c)}^j$ at the k -th iteration in hBFGS(c)

アルゴリズム1: hBFGS

1. $k = 1, i = 1, j = 1, c = 1, \mu_{(c)}^j = 1, N_{(c)}^j = P_{Tr}$;
2. Initialize $\mathbf{w}_1$ and $\mathbf{H}_1$ by uniform random numbers and the unit matrix $\mathbf{I}$, respectively;
3. Initialize $T_{(c)}^1 = k_{max} \times E(\mathbf{w}_1)$;
4. **While** ($k < k_{max}$ or $\|E(\mathbf{w}_k)/\partial \mathbf{w}_k\| > \varepsilon$)
5. Calculate $\mathbf{g}_k = -\partial E_{h(c)}(\mathbf{w}_k)/\partial \mathbf{w}_k$;
6. Execute line search to decide the step size η_k ;
7. Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \mathbf{H}_k \mathbf{g}_k$;
8. Calculate $\mathbf{g}_{k+1} = -\partial E_{h(c)}(\mathbf{w}_{k+1})/\partial \mathbf{w}_{k+1}$;
9. Update $\mathbf{H}_{k+1}$ by using BFGS formula of (17)-(19);
10. if ($i == N_{(c)}^j$) then
 - Update $T_{(c)}^j, \mu_{(c)}^j, N_{(c)}^j$ using (15), (14) and (16), respectively;
 - and $j = j + 1, i = 1$;
 - if ($\mu_{(c)}^j == 0$ and $c < seg$) then
 - $c = c + 1, j = 1, \mu_{(c)}^j = 1,$
 - $T_{(c)}^j = k_{max} \times E(\mathbf{w}_k)$ and $N_{(c)}^j = P_{Tr}$;
12. $k = k + 1, i = i + 1$;
13. **EndWhile**

$$y(x_1, x_2) = 1.9(1.35 + e^{x_1} e^{-x_2} \times \sin(13(x_1 - 0.6)^2) \sin(7x_2)) \quad (22)$$

$$y(x_1, x_2) = 1.33356(1.5(1 - x_1) + e^{2x_1 - 1} \sin(3\pi(x_1 - 0.6)^2)) + e^{3x_2 - 0.5} \sin(4\pi(x_2 - 0.9)^2) \quad (23)$$

$$y(x) = 1 + (x + 2x^2) \sin(-x^2) \quad (24)$$

ここで、(22)、(23)に対して、[3]と同様に $0 \leq x_1, x_2 \leq 1$ として学習データを与えることとし、学習データ数 P_{Tr} はともに 440 とする。中間層のニューロン数(H)は(22)及び(23)に対して、それぞれ 16 及び 5 とした。(24)に対

しても[6][7]と同様に $-4 \leq x \leq 4$ として学習データを与え、学習データ数及びHは、それぞれ、 $P_{Tr} = 400$ 及び $H = 7$ とした。特に、(24)はその非線形特性に対して、非常に少ない中間層ニューロン数で実現しようとする困難な問題の 1 つである[6]。シミュレーション結果を表 1 に示す。ここで、アルゴリズムのパラメータは、以下の様に決定した。すべての例題に対して、最大反復回数(k_{max})は 2×10^5 とする。poBFGS 及び hBFGS におけるクーリングパラメータ ρ と、hBFGS 及び ioBFGS の学習データの分割パラメータ seg を表 1 のように設定する。ここで、[3]と同様に、 seg は部分集合の学習データ数の初期値 $P_{Tr,s} (= P_{Tr}/seg)$ を用いて示した。これらの値は複数回の実験により最も平均学習誤差が小さくなった値に設定した。各手法に対して、異なる初期値により 50 回のシミュレーションを実行し、(2)に示す学習誤差の平均値(Ave)、最良値(Best)及び最悪値(Worst)を用いて評価する。同時に、学習終了時の反復回数の平均値(k_{ave})及び各手法が 1 回の反復に要した平均計算時間 $time(ms)$ に関しても評価する。表 1 より、学習誤差に関しては以下の結果を得た。まず、(22)及び(23)に対しては、提案手法は BFGS、ioBFGS、poBFGS と同様の結果を得た。また、oBFGS は、他の手法と比較して常に大きな学習誤差となってしまう、このような問題に対してはあまり有効な手法ではないことが分かる。一方、(24)に対しては、hBFGS は他の手法と比較して非常に有効な手法であることが分かる。つまり、BFGS、ioBFGS、poBFGS、及び、hBFGS は、50 回のシミュレーションにおいて、いくつかの初期値で同じ最良値を得た。しかしながら、提案手法以外のアルゴリズムでは局所解となる最悪値が大きな値となり、また、局所

表 1 シミュレーション結果

Table 1 Simulation results

	Ex.(22)			Ex.(23)			Ex.(24)		
	$P_{Tr,s}$	Ave	$k_{ave}(\times 10^5)$	$P_{Tr,s}$	Ave	$k_{ave}(\times 10^5)$	$P_{Tr,s}$	Ave	$k_{ave}(\times 10^5)$
	ρ	Best/Worst	time(ms)	ρ	Best/Worst	time(ms)	ρ	Best/Worst	time(ms)
BFGS	—	0.787	2.00	—	2.79	1.74	—	1.91	1.62
	—	0.511/1.17	0.805	—	0.608/23.3	0.315	—	0.186/18.3	0.371
oBFGS	44	3.44	2.00	44	10.2	2.00	40	54.5	2.00
	—	1.88/11.5	0.231	—	0.791/23.4	0.0620	—	52.9/59.0	0.0713
ioBFGS	44	0.853	2.00	44	2.00	2.00	40	2.43	1.93
	—	0.603/1.47	1.14	—	0.450/3.74	0.370	—	0.186/13.1	0.424
poBFGS	—	0.846	1.96	—	1.98	1.91	—	1.04	1.67
	0.7	0.541/1.83	0.824	0.7	0.609/23.3	0.287	0.7	0.186/13.2	0.370
hBFGS	44	0.753	2.00	44	2.11	2.00	40	0.574	1.85
	0.2	0.501/1.37	1.09	0.2	0.602/3.74	0.364	0.4	0.186/2.74	0.394

解への収束回数も hBFGS と比較して多くなった。結果として、hBFGS の平均値は他の手法と比較して小さな値となった。つまり、hBFGS は局所解への収束がほとんど見られなかった。さらに、(24)に対する各手法の反復回数 k と各反復における学習誤差の平均値 $E_{ave}(\mathbf{w}_k)$ の関係を図2に示す。図2より、hBFGS 以外の手法は、各反復で T_r の学習誤差 $E(\mathbf{w})$ を最小化するように学習が動作するため、 $E(\mathbf{w})$ の平均値($E_{ave}(\mathbf{w}_k)$)の変動はあまり見られない。一方、hBFGS は、hBFGS(c)ステップの反復内では $T_r^{\text{hBFGS(c)}}$ の誤差関数 $E_{m(c)}(\mathbf{w})$ を最小化するようにアルゴリズムは動作する。この為、各反復で $E_{ave}(\mathbf{w}_k)$ は激しく変動することになる。つまり、 $E_{m(c)}(\mathbf{w})$ を最小にする $\mathbf{w}$ の更新が、必ずしも $E(\mathbf{w})$ を小さくする方向へ動くとは限らないことを示している。従って、1回の poBFGS ループ毎に誤差関数が異なることになり、探索過程も大きく変動する。この変動が hBFGS の局所解への収束を避けながら最適解を探索する能力を向上させていると予想される。

学習の平均反復回数(k_{ave})に関しては、学習中に勾配がゼロに近づくことに起因する学習の停滞、いわゆるプラトー[3]を越えて最適解の探索を続けることを可能にするため、アルゴリズム 1 の学習終了条件 ε を $\varepsilon = 10^{-10}$ と小さな値に設定した。この為、各アルゴリズムは多くの試行で k_{max} まで学習を繰り返すことになり、 k_{ave} も大きな値となった。特に、常に探索を続ける oBFGS は $k_{ave} = k_{max}$ となった。一方、BFGS や poBFGS は最適解や局所解を見つけると k_{max} まで学習を繰り返すことなく早期に学習が終了する場合があります、 k_{ave} も他の手法と比較して小さな値となった。また、ioBFGS や hBFGS は探索を続ける能力が比較的高く、 k_{max} に近い平均反復回数となった。

1回の反復に要する平均計算時間 time(ms)に関しては、oBFGS が最も高速な手法であった。一般に、勾配法に基づく学習では各反復での勾配計算において必要となる学習パターン数が計算時間に最も影響する。従って、oBFGS が高速であることは各反復で用いる学習パターン数が少ないことに起因している。BFGS と poBFGS は各反復で全学習パターンを必要とするため、ほぼ同じ計算時間となった。一方、ioBFGS や hBFGS は学習初期では学習パターンの部分集合のみで勾配を計算できるにもかかわらず、他の手法と比較して多くの計算時間を必要とした。ioBFGS は1反復で常に2回の勾配計算を必要とし、かつ、学習パターン数も反復が進むに従って増加するアルゴリズムであることが原因であると考えられる。また、hBFGS は、アルゴリズム 1 の Line

図2 k と $E_{ave}(\mathbf{w}_k)$ の関係

Fig. 2 Relation between iteration count k and $E_{ave}(\mathbf{w}_k)$

search(ステップサイズの決定)に多くの時間が必要であることが原因である。つまり、poBFGS は学習の初期ではステップサイズの決定のために多くの時間を必要とする手法である。この為、繰り返し poBFGS を用いる hBFGS は、他の手法と比較して、1回の反復に若干多くの時間を要した。しかしながら、oBFGS と他の手法との差と比較すると、その差は僅かである。さらに、(24)の問題に対して、hBFGS と同様の学習誤差を得るためには、poBFGS を用い、最大反復回数 k_{max} を $k_{max} = 1 \times 10^6$ 、クーリングパラメータを $\rho = 0.9$ に設定する必要があった[6][7]。つまり、十分に時間をかけてオンラインからバッチへ学習を変化させる必要があった。具体的には、同様の学習誤差を得た poBFGS の結果[6][7]と比較すると、hBFGS($k_{max} = 2 \times 10^5$)の平均計算時間が 72.9(s)であったのに対し、poBFGS($k_{max} = 1 \times 10^6$)では 318.3(s)必要であった。

結果として、従来の poBFGS は、オンラインからバッチへ学習を変化させるメカニズムを、多くの反復を繰り返すことで、十分ゆっくりと実現していた手法であった。一方、提案手法は、poBFGS と同様のメカニズムを、部分集合の更新とパラメータの制御を組み合わせ、オンラインからバッチへの変化を二重化させることで実現した。これにより、hBFGS は反復回数を少なく制限しても、初期値に対するロバスト性を損なうことなく、poBFGS と同様のメカニズムを実現できる手法であることが分かる。

5. まとめ

本研究では、ニューラルネットワークの学習法として、オンラインからバッチへの変化のメカニズムを導入した新たな学習アルゴリズム、ハイブリッド型準ニュートン法(hBFGS)を提案した。提案手法では、オンラインからバッチへの変化のメカニズムを導入した従来の2つの学習法、ioBFGSとpoBFGSのそれぞれの有効性に着目した。さらに、これらを組み合わせることで、従来法と比較して学習結果の収束性を損なうことなく、反復回数を大幅に低減する、つまり、計算時間の大幅な短縮を実現することが可能なアルゴリズムを構築した。提案手法の有効性を計算機シミュレーションを用いて、他の準ニュートン法に基づくアルゴリズムと比較することにより示した。

今後の課題として、パラメータの決定法に関する議論が挙げられる。本稿ではシミュレーションに用いたパラメータは実験的に決定した。しかしながら、最適化手法におけるパラメータの設定は非常に重要な問題である。このため、今後は、各パラメータのアルゴリズムの収束性に対する影響、及び、それぞれのパラメータの関連性についても考察する予定である。さらには、オンラインからバッチへの変化のメカニズムとSAとの関連性及び大域収束性に関する解析も今後の課題としたい。

謝辞

本研究の一部は科研費基盤研究(C)(23500189)の助成を受けたものである。ここに謝意を表す。

参考文献

- [1] S. Haykin: Neural Networks and Learning Machines, 3rd, Pearson, 2009.
- [2] N. N. Schraudolph, J. Yu and S. Gunter: A stochastic quasi-Newton method for online convex optimization, Proc. 11th Intl. Conf. Artificial Intelligence and Statistics, March 2007.
- [3] 二宮 洋: 改良型オンライン準ニュートン法によるニューラルネットワークの学習, 信学論 A, Vol. J93-A, No.12, pp.828-832, 2010. 12.
- [4] H. Ninomiya: Microwave neural network models using improved online quasi-Newton training algorithm, Journal of Signal Processing, Vol.15, No.6, pp.483-488, Nov. 2011.
- [5] 阿部俊和, 坂下善彦, 二宮 洋: 改良型オンライン準ニュートン法に対する学習データの与え方の影響に関する考察, 信学技報, Vol.110, No.465, NLP2010-192, pp.163-168, 2011. 3.
- [6] H. Ninomiya: Parameterized online quasi-Newton training for high-nonlinearity function approximation using multilayer neural networks, Proc. IEEE&INNS/IJCNN'11, pp.2770-2777, July 2011.
- [7] 二宮 洋: パラメータ化オンライン準ニュートン法による階層型ニューラルネットワークの学習, 信学論 A, Vol.J95-A, No.8, pp.698-703, 2012. 8.
- [8] J. Nocedal and S. J. Wright: Numerical Optimization, 2nd, Springer Series in Operations Research, 2006.
- [9] A. Corane, M. Marechesi, C. Martini and S. Ridella: Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, ACM Trans. Math. Soft., Vol.13, No.3, pp.262-280, Sept., 1987.

阿部俊和 平 22 湘南工科大学工学部情報工学科卒, 平 24 同大学院工学研究科修士課程了。同年, (株) 第一コンピュータサービス入社。在学中, ニューラルネットワークの研究に従事。情報処理学会会員。

坂下善彦 1971 年東北大・工・応物学科卒, 静岡大学大学院電子科学研究科電子応用工学専攻卒。三菱電機株式会社入社。数値制御システムの開発に従事。その後, 同社情報技術総合研究所に所属。文書処理, ユーザインタフェース, 分散システム, ソフトウェア開発方式, 等の研究に従事。2001 年湘南工科大学情報工学科。ネットワーク環境における自律協調機構, 疎・密結合分散並列処理, ソフトウェア工学, 等の研究に従事。博士(工学)。電子情報通信学会, 情報処理学会, IEEE, ACM 各会員。

二宮 洋 平 5 静岡大工学部
光電機械工学科卒，平 10 同大
大学院電子工学研究科博士課
程了。博士(工学)。同年，湘南
工科大学情報工学科講師。現在，
同大准教授。これまでに，最適
化手法及びニューラルネット
ワークの学習に関する研究に

従事。平 7-10 日本学術振興会特別研究員(DC1)。平 17-21
情報処理学会論文誌編集委員。平 18-19 カールトン大学
客員研究員。 IEEE, IEICE, IPSJ 各会員。

(受付 2012 年 4 月 4 日，再受付 2012 年 5 月 25 日)