ニューラルネットワークにおける適応的二次最適化手法

本川　哲哉†　　手塚　太郎††

† 筑波大学情報学群知識情報・図書館学類　〒 305–8550 茨城県つくば市春日 1-2
†† 筑波大学 図書館情報メディア系　〒 305–8550 茨城県つくば市春日 1-2
E-mail: †s1511567@u.tsukuba.ac.jp, ††tezuka@slis.tsukuba.ac.jp

あらまし　現在，ニューラルネットワークにおける重みパラメータ最適化では，確率的勾配降下法（SGD）をはじめとする一次最適化手法を用いることが一般的である．しかしながら，一次最適化には収束速度の遅さや，学習率のような調整の難しいハイパーパラメータが多いというデメリットが存在する．近年この問題を解決するために，損失関数の重みに関するヘッセ行列に基づく二次最適化によって重みパラメータの最適化を改善する試みが活発化している．本研究では Hessian-free optimizaion（HF）と呼ばれる二次最適化手法を用いて，一次最適化で取り入られている適応的な工夫を施すことを試みる．適応的一次最適化手法にもいくつか種類が存在するが，本研究では Momentum 法と Adam という手法の計算的工夫を適用した．

キーワード　ニューラルネットワーク，適応的最適化，二次最適化，ヘッセ行列

1 はじめに

1.1 背　　景

近年，深層学習は様々な機械学習タスクにおいて大きな成功を収めている．ニューラルネットワークとは，入力層・中間層・出力層という 3 種類の層から成るネットワークである．中間層を深くしたニューラルネットワークの学習に関する問題は，機械学習において最も注目されている研究領域の 1 つである．ニューラルネットワークの学習とは，ネットワーク間の重みを入力と出力が適切な関係になるよう調整することである．具体的には，まずニューラルネットワークにおける出力の予測値と，与えられたデータの正解値との相違度を損失関数として定める．そしてその損失関数を最小化する重みパラメータを見つけることで，最適化を行なっている．

現在この重みパラメータ最適化には，損失関数の一次微分，つまり勾配に基づく一次最適化を用いることが主流となっている．確率的勾配降下法（Stochastic Gradient Descent，SGD) をベースとして，適応的な工夫を取り入れた Momentum SGD，Nesterov Accelerate Gradient，AdaGrad，RMSProp，Adam のような一次最適化手法がよく用いられている．これらの手法における適応的な工夫とは，どの手法でも現在ステップの重みに関する勾配だけでなく，過去ステップにおける勾配も合わせて使用するという部分にある．これによって従来の SGD よりも効率的な学習が可能となった．しかし，一次最適化にはデメリットがある．まず，収束速度が遅いということが挙げられる．これは訓練の 1 反復における計算にかかる時間のことではなく，収束に至るまでの繰り返し回数が多くなってしまうという意味である．また，学習率のような調整の難しいハイパーパラメータが多く存在するということもデメリットして挙げられる．ニューラルネットワークではハイパーパラメータを慎重に決めないと学習に失敗してしまうことが多々ある．これらは主に，一次微分である勾配のみに基づいて最適化していることに起因する．つまり損失関数をテイラー展開した時の一次微分の項までしか考えないため，二次以降の微分値に基づく誤差が含まれてしまうのである．これを，二次微分値まで考えて最適化を行う二次最適化によって解決しようとする試みがなされている．

1.2 目　　的

そこで本研究では，損失関数の二次微分値全体で構成されるヘッセ行列に基づく二次最適化によって，より効率的な重みパラメータ最適化を目指す．現在ニューラルネットワークにおける二次最適化には様々な研究が行われている [3] [4] [5]．本研究では二次最適化手法として Hessian-free optimization [3] と呼ばれる最適化手法を用いる．特徴次元数の多い問題や重みの次元数の多いニューラルネットワークでは，ヘッセ行列とその逆行列を求めるのは計算量的な観点で難しい．Hessian-free optimization ではこの問題を明示的にヘッセ行列を求めないことで回避している．この手法は Hessian-vector product というヘッセ行列とベクトルの積の高速な計算と，共役勾配法という最適化手法に基づいている．本研究ではこの Hessian-free optimization をもとに，パラメータ更新部分に Momentum 法と Adam の適応的な工夫を取り入れた手法を提案する．

2 関連研究

現在，ニューラルネットワークにおける重みパラメータ最適化に関する研究は活発に行われている．中でも確率的勾配降下法（SGD) をはじめとする一次最適化手法の研究は古くから盛んである．過去の勾配を用いる Momentum SGD [11] や，Nesterov Accelerate Gradient [9] のような手法はシンプルな実装が可能で，理論的にも強力なため今でもよく用いられる．特に近年では，過去の勾配を使用すると共に，学習率をステップ数ごとに適応的に調整する適応的一次最適化手法が注目されて

いる．代表例として AdaGrad [8]，RMSProp，Adam [12] のような適応的一次最適化手法がニューラルネットワークの訓練にはよく用いられる．ニューラルネットワークにおける目的関数が形成するような高次元のパラメータ空間の場合，停留点の多くは鞍点と呼ばれる勾配が非常に小さい点となる．適応的一次最適化手法はこの鞍点を効率的に抜け出す手法として有効だと言われている．現在のところ，適応的一次最適化手法は理論解析には乏しいが，様々な損失関数に対して何らかのチューニングなしに最適化が可能な手法として注目されている．

高次元空間の鞍点問題を解消する別のアプローチとして，自然勾配法 [15] という最適化手法が近年再注目されている．これは情報幾何学に基づいて定義される，フィッシャー情報行列と呼ばれる行列を勾配の再スケーリング要素として使用する．これは KL-Divergence と呼ばれるパラメータ空間の座標系によらない確率分布における距離構造を考慮した上で，損失関数を減少させる方向を決めている．

本研究では二次最適化手法として Hessian-free optimization という手法を用いる．3.5 で詳しく説明するが，Hessian-free optimization の主なアイデアはヘッセ行列と勾配ベクトルの積を効率的に計算する部分にある．また，Hessian-free optimization には分散化に対応した手法も存在する [18]．ニューラルネットワークにおける二次最適化手法には他にも多くの手法が存在する．最近注目が集まっている手法として，ヘッセ行列を Kronecker Factorization と呼ばれる行列におけるブロック対角近似テクニックを用いる手法がある．これはブロック対角近似によってヘッセ行列の計算コストが減らせるだけではなく，Kronecker Factorization の効果によって逆行列の計算もしやすくなるという利点がある．Kronecker Factorization を用いる代表手法として KFAC [4] や KFRA [5] がある．KFAC では，フィッシャー情報行列を Kronecker Factorization で近似して自然勾配法の枠組みで最適化を行う．また，Hessian-free optimization や他の多くの二次最適化手法ではヘッセ行列を正定値行列にするために，対角成分に小さな正の値を足すというテクニックをとることが多い．それに対して，Saddle-free newton method [13] という手法では負の固有値を無視するために，ヘッセ行列の対角成分全ての絶対値を取った行列を勾配の再スケーリング要素として用いる．また Saddle-free newton method の重みパラメータ最適化部分では信頼領域法という制約付き最適化を適用している．パラメータが信頼区間内にいるときは二次最適化を実行し，信頼区間外では一次最適化に切り替えるという工夫である．さらに，Saddle-free newton method と Hessian-free optimization を組み合わせた Saddle-free Hessian-free optimization [14] という手法も存在する．

3 研 究 手 法

3.1 ニューラルネットワーク

はじめにニューラルネットワークの定式化を行い，通常の勾配降下法に基づく重みパラメータ最適化の説明をする．全体で $K+1$ 層のニューラルネットワークを考えると，入力層が第 0 層，出力層が第 K 層，それ以外が中間層となる．$K+1$ 層のニューラルネットワークのノードにおける全ての重み行列を $W_1, ...W_K$，バイアスベクトルを $b_1, ..., b_K$ とする．これらがニューラルネットワークにおいて学習すべきパラメータである．入力 x が与えられた時のニューラルネットワークの出力 $f(x, \theta) = a_K$ は以下の計算を再帰的に適用することで求められる．

$$
\begin{aligned}
z_i &= W_i a_{i-1} + b_i \\
a_i &= \phi_i(z_i)
\end{aligned}
$$

ここで，入力として $a_0 = x$ とし，a_i はニューラルネットワークの活性と呼ばれる．また $\phi(\cdot)$ は活性化関数と呼ばれており，シグモイド関数やハイパボリックタンジェント関数のような非線形関数がよく用いられる．最近では，ニューラルネットワークを深層にした時に生じる勾配消失問題を解消するために有効な方法として，relu 関数という活性化関数が用いられることが多い，

学習に用いるデータセットを S とし，単一のデータを $(x, y) \in S$ とすると，上で計算した $a_K = f(x, \theta)$ を用いて正解値である y との相違度を損失関数によって測る．最適化する目的関数は以下のように損失関数のデータセット S 全体の平均で与えられる．

$$
h(\theta) = \frac{1}{|S|} \sum_{(x,y)\in S} L(y, f(x, \theta))
$$

ニューラルネットワークにおける損失関数としては，平均二乗誤差やクロスエントロピーと呼ばれる関数がよく用いられる．

学習フェーズでは損失関数の重みにおける勾配ベクトルが必要になる．勾配ベクトルを求めるためにはまず，出力に最も近い重みによる勾配を求める．さらにその勾配に基づいて微分の連鎖則を再帰的に適用することで，入力に近い重みによる勾配も求める誤差逆伝播法と呼ばれるアルゴリズムが用いられる．本論文でアルゴリズムを記述する擬似コード内では，簡単のため損失関数の微分に関する表記を以下のように定める．

$$
Dv = \frac{\partial L(y, \hat{y})}{\partial v}
$$

Algorithm 1 誤差逆伝播法による損失関数 L の勾配計算

input: $a_0 = x$, θ mapped to $(W_1, W_2, ..., W_K, b_1, b_2, ..., b_K)$
for all i from 1 to K **do**
  $z_i \leftarrow W_i a_{i-1} + b_i$
  $a_i \leftarrow \phi(z_i)$
end for
$Da_K \leftarrow \frac{\partial L}{\partial a_K}$
for all i from K to 1 **do**
  $Ds_i \leftarrow Da_i \odot \phi'(z_i)$
  $DW_i \leftarrow Ds_i a_{i-1}^T$
  $Db_i \leftarrow Ds_i$
  $Ds_{i-1} \leftarrow W_i^T Ds_i$
end for
output:
$D\theta$ as mapped from $(DW_1, DW_2, ..., DW_K, Db_1, Db_2, ..., Db_K)$

最後に，上のアルゴリズムによって得られた重みの勾配を現在の重みから引くことでパラメータの更新を行う．SGD において t 回目の反復における更新すべきパラメータをまとめて θ_t，学習率を α とすると，以下のような更新式になる．

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} h(\theta_t)$$

3.2 Momentum 法

ニューラルネットワークの重みパラメータ最適化において SGD はよく使われる手法ではあるが，学習が遅くなる場合がある．これを解決する最もシンプルな手法に Momentum 法というものが存在する [11]．本論文では Momentum 法を使用する SGD のことを Momentum SGD と呼ぶ．これは SGD での学習を高速化するための工夫が設計されており，特に損失関数の曲率が高い場合や勾配に含まれるノイズが大きい場合に大きな効果を発揮する．Momentum 法の主なアイディアは指数関数的に減衰する過去の勾配の移動平均を保存して，継続的にその勾配方向に進むという部分にある．Momentum SGD の効果を図 1 に図示する．

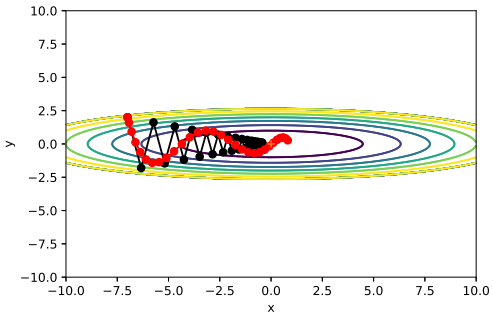

図 1　Momentum 法による効率的な最適化の様子

図 1 において黒線が SGD の軌跡で，赤線が Momentum SGD の軌跡である．Momentum 法を SGD に適用するとより効率的に最適化が行えることが確認できる．定式化として，Momentum 法では速度の役割を果たす変数 v を導入する．これはパラメータ空間上でパラメータが進む方向と速さを含む速度ベクトルである．この速度ベクトルは指数関数的に減衰する負の勾配の平均で調整される．「Momentum」という名前は物理学の現象に由来するもであり，負の勾配がパラメータ空間でニュートンの運動法則に従って他の粒子を動かす運動量のことを表している．物理学において，運動量とは「質量」と「速さ」の積である．Momentum 法を用いた学習アルゴリズムでは，単位質量を仮定し，速度ベクトル v は粒子の運動量とも考えられる．ハイパーパラメータである学習率 $\alpha \in [0, 1]$ は，これまで蓄積されていた勾配の寄与が指数関数的に減衰する速度を調整する．更新式は以下のように与えられる．

$$v_{t+1} = \alpha v_t - \epsilon \nabla_{\theta_t} h(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_t$$

3.3 Adam

現在，ニューラルネットワークの重みパラメータ最適化によく用いられる一次最適化手法の 1 つに，Adam（Adaptive moment estimation）と呼ばれる手法がある．Adam は適応的最適化手法と呼ばれる一次最適化手法の 1 つであり，Momentum 法と同様に過去の勾配を用いて最適化を行う．Momentum 法とは異なる部分として，各ステップごとに適応的に学習率を変化させ，最適化を行う方向を変えるという工夫がある．これは，パラメータ空間において損失関数の値が大きく変化する方向とそうではない方向が存在するために，ニューラルネットワークの訓練の前に適切な学習率を決めることが難しいという問題を解決する．Adam では単に過去の勾配を使用するわけではなく，過去の勾配の重み付き 1 次モーメントと重み付き 2 次モーメントを使用し，勾配に対して標準化を施してからパラメータの更新を行なっている．また重み付けは Momentum 法と同様で，指数関数的に減衰する重みを用いる．標準化によって勾配の統計的な分散が減らせるため，SGD や Momentum SGD など他の一次最適化手法よりも安定し，より効率的な最適化が可能となる．

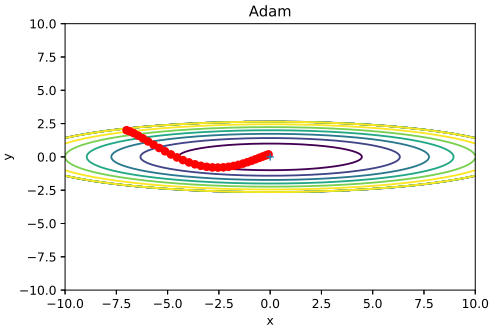

図 2　Adam による効率的な最適化の様子

Algorithm 2 Adam

Require: α: learning rate

Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay for the moment estimates

Require: $h(\theta)$: Stochastic objective function with parameters θ

Require: θ_0: initial parameter vector

$m_0 \leftarrow 0$ Initialize 1st moment vector

$v_0 \leftarrow 0$ Initialize 2nd moment vector

$t \leftarrow 0$ Initialize timestep

while θ_t not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta h_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t{}^2$

$\hat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$

$\hat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon}$

end while

3.4 二次最適化

二次最適化では，実数値ベクトルのパラメータ $\theta \in R^n$ に関する二回以上微分可能な目的関数 $h : R^n \to R$ の制約なし最適化を考える．Hessian-free optimization のような二次最適化手法は，古典的なニュートン法（Newton-Raphson 法とも呼ばれる）から導出される．これは，目的関数の局所二次近似モデルを反復的に最適化する．最も単純な方法として，現在のパラメータ θ_k が与えられると，k 回目の反復において，勾配および曲率の情報を使用して形成された目的関数 $h(\theta_k + \delta)$ の局所二次近似モデル $M_k(\delta)$ を最適化するという方法がある．局所二次近似モデルとはより正確に，以下のような定式化ができる．

$$M_k(\delta) \equiv h(\theta_k) + \nabla h(\theta_k)^T \delta + \frac{1}{2} \delta^T B_k \delta$$

ここで B_k は曲率行列（curvature matrix）と呼ばれ，標準的な二次最適化ではヘッセ行列が用いられる．

次に二次最適化手法の最も基礎的なアルゴリズムであるニュートン法について説明し，ニューラルネットワークにおける適用を議論する．ニュートン法では式（1）の臨界点での解を求める手続きを逐次更新で行う．探索方向をヘッセ行列の逆行列と勾配ベクトルの積とし，それを重みパラメータから引くことで更新を実行する．H を重みにおける損失関数のヘッセ行列，g を勾配ベクトルとすると次のような更新式を得る．

$$\theta_{t+1} = \theta_t - H^{-1} g$$

ニューラルネットワークにおける目的関数のように，鞍点のような多くの問題を抱える非凸関数では，ニュートン法による適切な最適化が期待できない．例えば鞍点付近でヘッセ行列の固有値が全て正ではない場合，ニュートン法による更新は間違った方向へ進んでしまう可能性がある．また，計算量的な観点からもニューラルネットワークへのニュートン法の適用は難しい．ニュートン法におけるパラメータ更新では損失関数のヘッセ行列が必要なので，各重みに関する二次微分を計算する必要がある．これは 3.1 節で説明した，誤差逆伝播法による勾配の計算と同様に，ニューラルネットワークにおけるヘッセ行列も誤差逆伝播法によって正確な計算が可能であることが示されている [7]．しかしながら，ニューラルネットワークにおける重みパラメータの数を k とすると，ニュートン法では $k \times k$ 行列の逆行列が必要となり，$O(k^3)$ もの計算複雑性が生じる．また更新毎に重みパラメータが変化するので，訓練の反復のたびにヘッセ行列とその逆行列を計算する必要が生じる．そのため，ニュートン法を適用できるニューラルネットワークはパラメータの数が非常に少ないモデルに限られる．

3.5 Hessian-free optimization

ここでは本研究で使用する二次最適化手法の Hessian-free optimization について説明する．Hessian-free optimization は，元々は数値計算の分野における最適化手法として使われていた Truncated-newton method という手法 [16] の 1 つである．Martens（2010）によってニューラルネットワークにおける重みパラメータ最適化への適用が考えられた [6].

通常のニュートン法ではヘッセ行列を正確に計算しなければならないため，パラメータ数の多いニューラルネットワークにそのまま適用するのは難しい．これは空間計算量，時間計算量のどちらのコストも高いことから言える．そこで Hessian-free optimization ではパラメータの更新方向を決めるのに共役勾配法（Conjugate Gradient Method）という最適化手法を用いている．

3.5.1 共役勾配法

共役勾配法は，訓練の反復時に降下する方向を勾配の共役方向に従うことで，ヘッセ行列の逆行列計算を効率的に回避する手法である．このアプローチの主なアイディアは，勾配に関連づけられた方向へ反復的に直線探索を行う勾配降下法の欠点を注意深く検証した結果から得られている．訓練の t 回目の反復後に，直前の探索方向を δ_{t-1} とすると，現在の探索方向 δ_t は以下のように定式化できる．

$$\delta_t = \nabla_\theta h(\theta) + \beta_t \delta_{t-1}$$

この β_t を選択する際にヘッセ行列と勾配ベクトルとの積が必要となってくる．

共役勾配法は理論的にも非常に強力な最適化アルゴリズムである．t 回の反復後，$K_t(A, r_0) \equiv \text{span}\{r_0, Ar_0, A^2 r_0, ..., A^{t-1} r_0\}$ として定義される Krylov 部分空間上の任意の凸二次関数 $q(x)$ において最適解が見つかることが証明されている [17]．ここで，$r_0 = Ax_0 - b$ であり，x_0 は初期値とする．Hessian-free optimization では前処理付き共役勾配法（Preconditioned Conjugate Gradient method）と呼ばれる共役勾配法の 1 種を使用する．これは係数行列であるヘッセ行列に対角化などの前処理を施すことで，ヘッセ行列の固有値分布を改善するという目的に基づいている．反復法の収束速度は係数行列の固有値分布に依存している．固有値の分布に変動が少なく，最大最小固有値の比（条件数）が 1 に近いほど収束が速くなる．

Algorithm 3 前処理付き共役勾配法（PCG）

inputs: b, A, x_0, P

$r_0 \leftarrow Ax_0 - b$
$y_0 \leftarrow$ solution of $Py = r_0$
$p_0 \leftarrow -y_0$
$i \leftarrow 0$

while solution is not satisfactory **do**
  $\alpha \leftarrow \frac{r_i^T y_i}{p_i^T A p_i}$
  $x_{i+1} \leftarrow x_i + \alpha_i p_i$
  $r_{i+1} \leftarrow r_i + \alpha_i A p_i$
  $y_{i+1} \leftarrow$ solution of $Py = r_{i+1}$
  $\beta_{i+1} \leftarrow \frac{r_{i+1}^T y_{i+1}}{r_i^T y_i}$
  $p_{i+1} \leftarrow -y_{i+1} + \beta_{i+1} p_i$
  $i \leftarrow i + 1$
end while

output: x_i

3.5.2 Hessian-vector product

共役勾配法ではヘッセ行列と勾配ベクトルの積，つまり最終的にはベクトルのみが必要となる．これをそのまま行列ベクトル積として扱い，ニューラルネットワークで誤差逆伝播が出来るような計算的工夫を取り入れる．そうすることで，ヘッセ行列とその逆行列の明示的な計算を避けることが可能となる．以下ではその計算的な工夫を説明する．

ヘッセ行列と勾配ベクトルの積（Hessian-vector product，Hv と略記する）を効率的に計算するために，ヘッセ行列が勾配ベクトルに関するヤコビ行列であることに着目する．つまり，Hv は勾配ベクトルに関する $\nabla h(\theta)$ の方向微分として扱うことができる．

$$H(\theta)v = \lim_{\epsilon \to 0} \frac{\nabla h(\theta + \epsilon v) - \nabla h(\theta)}{\epsilon}$$

上の方程式は実装上は有限差分を取った数値計算アルゴリズムとして扱うことができる．しかしながら，ニューラルネットワークのような高次元の非線形関数を扱う場合，有限差分は数値誤差の問題が避けられず一般的には向かない．そこで有限差分を利用せずに，安定的に方向微分を計算する方法を説明する [1]．最適化理論の分野でニューラルネットワークや他の関連モデルのために発見した Pearlmutter（1994）の説明に従う．この方法は「前方微分（forward-differentiation）」として知られている．勾配を求めるための誤差逆伝播法の考え方と同様に，計算グラフ内の全てのノードに対して微分の連鎖則を繰り返し使うことで前方微分の値を求めることができる．以下のように R_v という演算子を導入して定式化する．

$$R_v x = \lim_{\epsilon \to 0} \frac{x(\theta + \epsilon v) - x(\theta)}{\epsilon} = \frac{\partial x}{\partial \theta} v$$

R_v 演算子は線型性，積の微分，連鎖則という通常の微分の規則が成り立つ．

線型性：$R_v(x + y) = R_v x + R_v y$

積の微分：$R_v(xy) = (R_v x)y + x(R_v y)$

連鎖則：$R_v(\gamma(x)) = (R_v x) J_\gamma(x)$

$J_\gamma(x)$ は $\gamma(x)$ のヤコビ行列とする．$Hv = R_v(\nabla h(\theta))$ であることに注意すると，これらの規則を計算グラフに適用することでニューラルネットワークにおける Hv の誤差逆伝播法が導ける．以下，アルゴリズム中では R_v 演算子を R と略記する．

Algorithm 4 ニューラルネットワークにおける Hv の誤差逆伝播法

input: v mapped to $(RW_1, ..., RW_K, Rb_1, ..., Rb_K)$
$Ra_0 \leftarrow 0$

for for all i from 1 to K **do**
  $Rz_i \leftarrow RW_i a_{i-1} + W_i Ra_{i-1} + Rb_i$
  $Ra_i \leftarrow Rz_i \phi_i^{'}(s_i)$
end for

$RDa_K \leftarrow R(\frac{\partial L}{\partial a_K}) = \frac{\partial}{\partial a_K} \frac{\partial L}{\partial a_K} Ra_K = \frac{\partial^2 L}{\partial a_K^2} Ra_K$

for all i from K downto 1 **do**
  $RDs_i \leftarrow RDa_i \odot \phi_i^{'}(s_i) + Da_i \odot R(\phi_i^{'}(s_i)) = RDa_i \odot \phi_i^{'}(s_i) +$
  $Ra_i \odot \phi^{''}(s_i) \odot Rs_i$
  $RDW_i \leftarrow RDs_i a_{i-1}^T + Ds_i Ra_{i-1}^T$
  $RDb_i \leftarrow RDs_i$
  $RDa_{i-1} \leftarrow RW_i^T Ds_i + W_i^T RDs_i$
end for

output: Hv as mapped from $(RDW_1, ..., RDW_K, RDb_1, ..., RDb_K)$

3.5.3 一般ガウスニュートン行列

Hessian-free optimization では，さらなる計算効率化のためにヘッセ行列を一般ガウスニュートン行列（The generalized Gauss-Newton matrix）で近似する．以下で一般ガウスニュートン行列 G の定式化を行う．

$$h(\theta) = \frac{1}{|S|} \sum_{(x,y) \in S} L(y, f(x, \theta))$$

$$\nabla h(\theta) = \frac{1}{|S|} \sum_{(x,y) \in S} J_f^T \nabla L(y, f(x, \theta))$$

$$H(\theta) = \frac{1}{|S|} \sum_{(x,y) \in S} J_f^T H_L J_f + \sum_{i=1}^{m} [\nabla_f L(y, f(x, \theta))]_i H_{[f(x,\theta)]_i}$$

上記のヘッセ行列の式における第 2 項を無視した以下のような行列を，一般ガウスニュートン行列と定義する．

$$G \equiv \frac{1}{|S|} \sum_{(x,y) \in S} J_f^T H_L J_f$$

例えば損失関数 L が平均二乗誤差の時は，出力層におけるヘッセ行列が $H_L = I$（単位行列）となるので一般ガウスニュートン行列 G はヤコビ行列の積 $J^T J$ として表される．この時の最適化アルゴリズムはレーベンバーグマーカート法（Levenberg-Marquardt algorithm）として知られている．

ここまでで一般ガウスニュートン行列がヘッセ行列の近似

であることは説明した．しかしながら，実際に Hessian-free optimization において一般ガウスニュートン行列が有用であるためには行列ベクトル積 Gv を効率的に計算するアルゴリズムが必要となる．古典的なガウスニュートン行列を乗算するアルゴリズムは，数値最適化の文脈ではよく知られている [1]．これらの方法は以下で説明するアプローチを用いて，一般ガウスニュートン行列について Schraudolph（2002）によって一般化された [2].

$G \equiv \frac{1}{|S|}\sum_{(x,y)\in S} J_f^T H_L J_f$ と定めたように一般ガウスニュートン行列は 3 つの行列の積のデータセット S に関する平均として表すことができる．この行列とベクトル v との積 Gv を計算するには，それぞれの行列との積を逐次的に計算すれば良い．まず，出力におけるヤコビ行列とのベクトル積 $J_f v$ を計算する．これは $R_v(f(x,\theta))$ を計算することに等しい．次に損失関数の出力におけるヘッセ行列 $H_L = \frac{\partial^2 L}{\partial a_K^2}$ は通常簡単に求まるので，それと先ほど求められたヤコビ行列ベクトル積 $J_f v$（つまりベクトル）との積を計算する．最後に誤差逆伝播法によって，ヤコビ行列の転置行列 J_f^T と今求めた行列ベクトル積 $H_L J_f v$ との積を計算すれば単一データ点 (x, y) におけるガウスニュートン行列が求まる．データセット全体に対してこの計算をして平均を取れば一般ガウスニュートン行列の計算が可能となる．

Algorithm 5 ニューラルネットワークにおける Gv の誤差逆伝播法

input: v mapped to $(RW_1, ..., RW_K, Rb_1, ..., Rb_K)$
$Ra_0 \leftarrow 0$

for for all i from 1 to K **do**
  $Rz_i \leftarrow RW_i a_{i-1} + W_i Ra_{i-1} + Rb_i$
  $Ra_i \leftarrow Rz_i \phi_i^{'}(s_i)$
end for

$RDa_i \leftarrow \frac{\partial^2 L}{\partial a_K^2} Ra_K$

for all i from K downto 1 **do**
  $RDs_i \leftarrow RDa_i \odot \phi_i^{'}(s_i)$
  $RDW_i \leftarrow RDs_i a_{i-1}^T$
  $RDb_i \leftarrow RDs_i$
  $RDa_{i-1} \leftarrow W_i^T RDs_i$
end for

output: Gv as mapped from $(RDW_1, ..., RDW_K, RDb_1, ..., RDb_K)$

3.5.4 Damping

Gv の計算的な工夫によって，Hessian-free optimization に共役勾配法をより効率的に適用することが可能となった．しかしながら，ニューラルネットワークに二次最適化を適用する上で，ヘッセ行列の正定値性という非常に大きな問題が残されている．ヘッセ行列が正定値の時，局所二次近似をした損失関数が狭義の凸関数となるため凸最適化が可能となる．凸最適化が可能ということは，本来の目的関数における局所最適解が近似関数においては大域最適解になることを意味する．一般にニューラルネットワークのように非常にパラメータ数の多い非凸な目的関数を考えるとき，そのヘッセ行列が正定値であるという保証はない．二次最適化ではこの問題を回避するために，曲率行列に対して Damping という工夫を施す．特に本研究では，Tikhonov Damping という Damping 手法を用いる．これは曲率行列 B に対して λI を足すことで，固有値の値を正にしようとする手法である．つまり Hessian-free optimization においては，Gv の代わりに $Gv + \lambda Iv = (G + \lambda I)v$ を用いることに相当する．G に負の固有値が存在してもそれが 0 に近い場合，非常にうまく機能することが知られている．またこのような Damping は一般的な機械学習における正則化として機能することも分かっている [6].

Hessian-free optimization では λ の決め方として，Levenberg-Marquardt (LM) heuristic が有効であるとされている [6]．LM heuristic では以下のように定義される縮小比率 reduction ratio）ρ を用いて λ の調整を行う．

$$\rho \equiv \frac{h(\theta_k + \delta_k) - h(\theta_k)}{M_k(\delta_k) - M_k(0)}$$

縮小比率 ρ では δ_k によるパラメータ更新後と更新前に評価される目的関数の差 $h(\theta_k+\delta_k)-h(\theta_k)$ と，ニューラルネットワークの局所二次近似モデルによって予測される減少量 $M_k(\delta_k)-M_k(0)$ の比率を測定する．ρ が 1 よりもはるかに大きい場合，二次近似モデルは予測の減少量を過大に評価してしまっているので，λ を増加させるべきである．逆に ρ が 1 に近い場合，二次近似が δ_k 付近ではかなり正確である可能性が高いため，λ を小さくすることが可能になる．LM heuristic では具体的に以下のような更新規則を定めている．

if $\rho > \frac{3}{4}$ then $\lambda \leftarrow \frac{2}{3}\lambda$
if $\rho < \frac{1}{4}$ then $\lambda \leftarrow \frac{3}{2}\lambda$

Algorithm 6 Hessian-free optimization

inputs: θ_1, λ
$\delta_0 \leftarrow \vec{0}$
$k \leftarrow 1$
while solution is not satisfactory **do**
  Select a mini-batch $S^{'} \subset S$ of training cases for the gradient
  $b \leftarrow -\nabla h(\theta_k)$ on $S^{'}$
  Select a mini-batch $S^{''} \subset S$ of training cases for the curvature
  Compute a preconditioner P at θ_k
  Compute a damping matrix D_k
  Define $A(v) \equiv G(\theta_k)v + \lambda D_k v$ on $S^{''}$
  Choose a decay constant $\xi \in [0, 1]$
  $\delta_k \leftarrow \mathrm{PCG}\ (b, A, \xi\delta_{k-1}, P)$
  Update λ with the Levenberg-Marqurt method
  Choose/Compute a learning rate α
  $\theta_{k+1} \leftarrow \theta_k + \alpha\delta_k$
  $k \leftarrow k + 1$
end while

3.6 Adaptive Hessian-free optimization

本研究では，Hessian-free optimization のパラメータ更新部分に Momentum 法と Adam における計算的な工夫を取り入れた 2 種類の最適化手法を提案する．つまり，アルゴリズム (6) 中のパラメータ更新部分，$\theta_{k+1} \leftarrow \theta_k + \alpha\delta_k$ で Momentum 法の指数関数的に減衰する速度ベクトル v を導入する工夫と，Adam の標準化に相当する計算的な工夫を取り入れる．Momentum 法の工夫では，降下方向である δ の一次モーメントのみを保持しておき，速度ベクトル v によって過去の δ の寄与を調整する．Adam の工夫では，降下方向である δ の履歴情報を保持しておき，その一次モーメントと二次モーメントを計算し，それらを用いてパラメータの更新を行う．これにより δ の統計的な分散を減少させ，より安定した最適化を期待する．また，パラメータ更新部分には Momentum 法と Adam 以外の適応的一次最適化手法の工夫を取り入れることも可能である．

4 研究手法の評価実験

実験には学習モデルとして，入力層・中間層 1 層・出力層という 3 層の浅いニューラルネットワークを使用した．中間層のユニット数は 300 個に設定した．データセットには，MNIST 手書き数字データセット，CIFAR-10 データセットと CIFAR-100 データセットを用いて画像分類タスクを行なった．以下にデータセットの詳細を記す．

4.1 データセット

MNIST

0 から 9 の 10 クラスにラベル付けされた，28 × 28 サイズの 60,000 枚の手書きの数字が描かれたモノクロ画像データセット．50,000 枚を訓練用に用いて 10,000 枚をテスト用に用いた．

CIFAR-10

犬や飛行機など 10 クラスにラベル付けされた，32 × 32 サイズの 60,000 枚のカラー画像データセット．50,000 枚を訓練用に用いて 10,000 枚をテスト用に用いた．

CIFAR-100

100 クラスにラベル付けされた，32 × 32 サイズの 60,000 枚のカラー画像データセット．50,000 枚を訓練用に用いて 10,000 枚をテスト用に用いた．

4.2 実験方法

それぞれのデータセットごとに，一次最適化手法として SGD，MomentumSGD，Adam，二次最適化手法として HF，HF+Momentum，HF+Adam という合計 6 種類の最適化手法を用いて訓練損失に関する学習曲線の収束の様子を比較した．通常，機械学習タスクでは訓練誤差ではなく汎化誤差をどれだけ減らせるかが重要となる．しかしながら，汎化性能をどれだけ向上させられるかは最適化手法の性質よりもモデルの性質に依存する場合の方が多いため本研究の実験では訓練損失にのみ注目することにした．

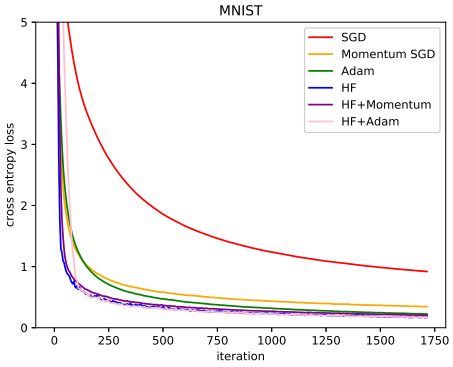

図 3　MNIST での訓練損失の減少

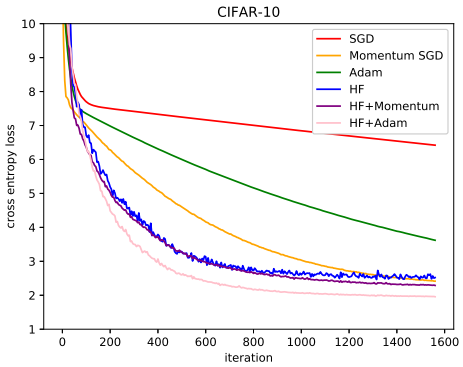

図 4　CIFAR-10 での訓練損失の減少

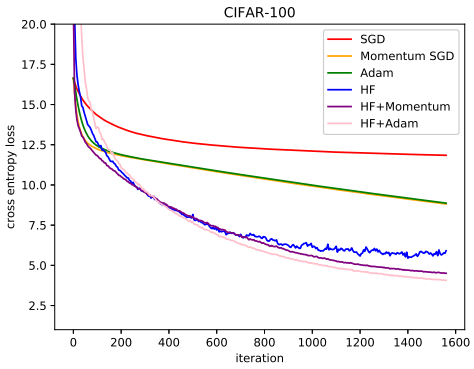

図 5　CIFAR-100 での訓練損失の減少

4.3 実験結果

実験の結果，どのデータセットでも二次最適化手法の方が，訓練損失が収束に到るまでの反復回数においては大幅に改善されているいることが分かる．二次最適化手法どうしの比較をしてみると，MNIST では分かりにくいが，CIFAR-10 と CIFAR-100 においては AdaptiveHF が HF よりも最終的な損失関数の値が低くなっていることが分かる．

5 結　　論

本研究では，Hessian-free optimization という二次最適化手法を用いて，そのパラメータ更新部分に適応的一次最適化手

法の計算的な工夫を取り入れた手法を提案した．3 種類の画像データセットを用いて分類タスクの実験を行なった結果，訓練損失を損失を最小化するという意味では AdaptiveHF が有効であることが示唆された．CIFAR-10 と CIFAR-100 においては，最終的な損失関数の値が AdaptiveHF の方が HF よりも小さくなっているだけではなく，HF の値よりも安定していることが分かる．今回の実験では，画像分類タスクに使用したモデルとして中間層 1 層の浅いフィードフォワードニューラルネットワーク（FNN）を用いた．これは実装上，FNN 以外の複雑なニューラルネットワークモデルに HF を適用することが難しいためである．一般に画像分類のようなタスクでは，浅い FNN による汎化性能は期待できない．今後の課題として，畳み込みニューラルネットワーク（CNN）や再帰型ニューラルネットワーク（RNN）の重みパラメータ最適化に HF をはじめとする二次最適化手法を用いるということが挙げられる．これによって汎化誤差を改善することが可能になり，より実用的なアプリケーションとして Adative HF を用いることができるようになる．また，本研究では Hessian-free optimization のパラメータ更新部分 $\theta_{t+1} \leftarrow \theta_t + \alpha\delta_t$ に適応的な工夫を加えたが，一般ガウスニュートン行列と勾配ベクトルとの積 Gv を計算する部分にも同様の工夫を取り入れることが考えられる．この工夫は過去の曲率行列を用いることで，現在の曲率行列のみで損失関数を局所二次近似したモデルよりも安定した近似が可能になり，効率的な最適化につながるのではないかと考えられる．二次最適化手法として，Hessian-free optimization はヘッセ行列の逆行列計算を Hv の効率的な計算によって回避した手法である．KFAC や KFRA のような逆行列計算自体を近似的に計算する手法で，過去の曲率行列を使用するという Adaptive な計算的工夫を取り入れるということも効率的な最適化を行う手法として期待できる．

6 謝　　辞

本研究に取り組むに当たって，ご指導してくださった筑波大学図書館情報メディア系手塚太郎准教授に心より感謝いたします．日頃から様々な観点からアドバイスを頂き，本論文を執筆することができました．また，筑波大学図書館情報メディア系若林啓助教授には手塚若林研の合同ゼミや合同合宿でアドバイスを頂き，大変参考になりました．ありがとうございます．幸運なことに筑波大学情報学群知識情報・図書館学類の手塚研究室，若林研究室の友人や先輩にも恵まれました．友人や先輩との議論によって，大変有意義な研究生活を送ることができました．本当にありがとうございます．また、本研究は JSPS 科研費 JP16K00228,JP16H02904 の助成を受けたものです．感謝いたします。

文　　献

[1] J. Nocedal and S. J. Wright, “Numerical Optimization”, 1999.

[2] N. N. Schraudolph, “Fast curvature matrix-vector products for second-order gradient descent”, Neural Computation, 2002.

[3] J. Martens, “Deep learning via Hessian-free optimization”, In ICML, 2010.

[4] J. Martens and R.Grosse, “Optimizing neural networks with kronecker-factored approximate curvature”, 2015.

[5] A. Botev, H. Ritter, D. Barder, “Practical Gauss-Newton Optimization for Deep Learning”, In ICML, 2017.

[6] J. Martens and I. Sutskever, “Training Deep and Recurrent Networks with Hessian-Free Optimization”, Neural Networks: Tricks of the Trade, p.479-535, 2012.

[7] C. M. Bishop, “Exact Caluculation of the Hessian Matrix for the Multi-layer Perceptron”, Neural Computation, p.494-501, 1992.

[8] J. Duchi, E. Hazan, Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”, Journal of Machine Learning Research, 12, 2121–2159, 2011.

[9] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)”, Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543– 547, 1983.

[10] I. Sutskever, J. Martens, G. Dahl, G. Hinton, “On the importance of initialization and momentum in deep learning”, 2013.

[11] N. Qian, “On the momentum term in gradient descent learning algorithms”, Neural Networks : The Official Journal of the International Neural Network Society, 12(1), 145–151, 1999.

[12] D. P. Kingma and J. L. Ba, “Adam: a Method for Stochastic Optimization”, ICLR, 2015.

[13] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization”, 2014.

[14] A. Martin, “Saddle-free Hessian-free Optimization”, NIPS Workshop on Nonconvex Optimization for Machine Learning, 2016.

[15] S. Amari, “Natural Gradient Works Efficiently in Learning”, Neural Computation, 1998.

[16] G. Stephen, “A survey of truncated-Newton methods Stephen”, 1999.

[17] J. Shewchuk, “An introduction to the conjugate gradient method without the agonizing pain”, 1994.

[18] H. Xi, M. Dheevatsa, S. Mikhail, T. Martin, “Distributed Hessian-Free Optimization for Deep Neural Network”, 2016.