博　士　論　文

慣性付準ニュートン法に基づくニューラルネットワークの学習アルゴリズムに関する研究

令和 5 年 3 月

湘南工科大学大学院
工学研究科　電気情報工学専攻　博士後期課程

20T2502

Shahrzad　Mahboubi

論文内容の要旨

近年, 情報技術の急速な発達に伴い, 様々なデータの取得および蓄積が可能となった. 従って, 蓄積されるデータの複雑 (非線形) 性とその量は日々増加しつつある. これに伴い, 大規模な強非線形データの高精度かつ高速な処理を可能とする技術の必要性も高まっている. この処理を可能とする技術の一つとしてニューラルネットワーク (NN) が注目されている. NN では学習アルゴリズムは重要であり, その多くは 1 次近似勾配手法 (1 次手法) と 2 次近似勾配手法 (2 次手法) に分類することができる. 一般的に, NN の学習には 1 次手法が用いられ, 様々な応用および改良研究が行われている. その一つとして, 慣性項の導入により学習速度を大幅に高速化した研究が挙げられる. しかし, 1 次手法では強非線形性を内包する学習データの高精度な学習に対して, 現実的な時間での処理が困難であった. このため, 強非線形データの学習には学習率に目的関数の曲率情報であるヘッセ行列を利用した 2 次手法が用いられ, その中でも準ニュートン法 (QN) が最も有効とされている. 本研究では, 1 次手法に対する慣性項の効果を 2 次手法においても同様に得られることを期待し, 2 次手法における慣性項の影響を調査する. さらに, 慣性項を用いた 2 次手法の問題点に対して解決策を提案する. 具体的には, 慣性項を用いた 2 次手法の新たなアルゴリズムとして 4 つの手法を提案する. 1 つ目の提案手法として, 慣性項を用いることで QN を高速化したネステロフの加速準ニュートン法 (NAQ) のハイパーパラメータ問題に着目し, 適応的慣性係数を導入することで問題を解決した適応的 NAQ(AdaNAQ) を提案する. 提案手法はハイパーパラメータ問題を解決すると共に, 収束の安定性と初期値に対する学習のロバスト性を向上させた. 2 つ目の提案手法として, 慣性項の影響により増加した NAQ の 1 反復における計算時間に着目し, その問題を克服した新たな準ニュートン法に基づく学習アルゴリズムとして慣性付準ニュートン法 (MoQ) を提案する. NAQ では 2 つの勾配, 通常勾配とネステロフの加速勾配の計算が各反復において必要であった. MoQ は誤差関数を 2 次関数と見なし, ネステロフの加速勾配を通常勾配の重み付の線形和として近似した. 提案手法は NAQ の学習性能を維持しながら, 高速化に成功した. 3 つ目の提案手法として, NAQ と MoQ の計算コストに着目し, 計算コストを削減した手法として記憶制限 NAQ(LNAQ) と記憶制限 MoQ(LMoQ) を提案する. LNAQ と LMoQ は任意の記憶量によりヘッセ行列の計算コストを削減することができる. 提案手法は, 従来手法の記憶制限 QN(LQN) を高速化し, NAQ と MoQ の計算コストを削減した. 4 つ目の提案手法として, NAQ と MoQ のさらなる計算コストの削減を目指し, メモリレス NAQ(MLNAQ) とメモリレス MoQ(MLMoQ) を提案する. 提案手法は, ヘッセ行列をベクトルの内積だけで更新するため, その計算コストは 1 次手法とほぼ同様である. さらに, MLNAQ および MLMoQ は従来手法のメモリレス QN(MLQN) を高速化することに成功した. 全ての提案手法の有効性は計算機実験により示した.

目 次

1 序論 **1**
1.1 研究の背景 1
1.2 研究の目的 3
1.3 論文の構成 6

2 最適化とニューラルネットワーク **9**
2.1 勾配法に基づく最適化と NN の概念 9
2.2 NN における勾配の導出 13
2.3 1 次近似勾配学習法 17
2.3.1 Gradient Descent method 18
2.3.2 Classical Momentum method 20
2.3.3 Nesterov's Accelerated Gradient method 21
2.3.4 AdaGrad 23
2.3.5 RMSprop 25
2.3.6 AdaDelta 26
2.3.7 Adam 28
2.4 2 次近似勾配学習手法 30
2.4.1 Newton method 32
2.4.2 Quasi-Newton method 33
2.4.3 Nesterov's Accelerated Quasi-Newton method 35
2.5 まとめ 38

3 適応的ネステロフの加速準ニュートン法 **39**
3.1 適応的慣性係数 39
3.2 実験 41
3.2.1 関数近似問題 43
3.2.2 マイクロ波回路モデリング問題 1：microstrip Low-Pass Filter 46
3.2.3 マイクロ波回路モデリング問題 2：Microstrip Patch Antenna 49
3.3 まとめ 51

4 慣性付準ニュートン法 **52**
4.1 Multi-Step Quasi-Newton method 53
4.2 Momentum Quasi-Newton method 54
4.2.1 MoQ の収束特性 57
4.2.2 計算コスト 60
4.2.3 適応的慣性係数 62
4.3 実験 63
4.3.1 関数近似問題 1 64
4.3.2 関数近似問題 2: Levy Function 66
4.3.3 マイクロ波回路モデリング問題 1 : microstrip Low-Pass Filter 68

4.3.4 マイクロ波回路モデリング問題 2: Microstrip Patch Antenna 72
4.3.5 分類問題 1: 8 × 8 MNIST handwritten digit dataset 74
4.3.6 分類問題 2: Three-Spirals dataset 77
4.4 まとめ 81

5 慣性付記憶制限準ニュートン法 82
5.1 Limited-Memory Quasi-Newton method 82
5.2 慣性項による記憶制限準ニュートン法の高速化 85
5.2.1 Limited-Memory Nesterov's Accelerated Quasi-Newton method 85
5.2.2 Limited-Memory Momentum Quasi-Newton method 87
5.3 計算コスト 89
5.4 実験 91
5.4.1 関数近似問題 1 92
5.4.2 関数近似問題 2: Levy Function 93
5.4.3 マイクロ波回路モデリング問題：microstrip Low-Pass Filter 96
5.5 まとめ 98

6 慣性付メモリレス準ニュートン法 99
6.1 Conjugate Gradient method 99
6.2 MemoryLess Quasi-Newton method 101
6.3 慣性項によるメモリレス手法の高速化 102
6.3.1 MemoryLess Nesterov's Accelerated Quasi-Newton method 102
6.3.2 MemoryLess Momentum Quasi-Newton method 104
6.4 計算コスト 106
6.5 実験 109
6.5.1 関数近似問題 109
6.5.2 分類問題 1: Three-Spirals detaset 111
6.5.3 分類問題 2: 8 × 8 MNIST handwritten digit dataset 113
6.5.4 分類問題 3: 28 × 28 MNIST handwritten digit dataset 115
6.6 まとめ 117

7 結論 118

付録 122

参考文献 125

研究業績 131

謝辞 135

アルゴリズムの目次

1 Gradient Descent method (GD) . . . 19
2 Classical Momentum method (CM) . . . 21
3 Nesterov's Accelerated Gradient method (NAG) . . . 22
4 AdaGrad . . . 24
5 RMSprop . . . 25
6 AdaDelta . . . 27
7 Adam . . . 29
8 Quasi-Newton method (QN) -BFGS- . . . 35
9 Nesterov's Accelerated Quasi-Newton method (NAQ) . . . 37
10 Calculation of stepsize . . . 42
11 Multi-Step Quasi-Newton method (MSQN) . . . 54
12 Momentum Quasi-Newton method (MoQ) . . . 56
13 Limited-Memory Quasi-Newton method (LQN) . . . 84
14 Direction Vector of LQN (Two-Loop recursion) . . . 85
15 Limited-Memory Nesterov's Accelerated Quasi-Newton method (LNAQ) . . . 86
16 Direction Vector of LNAQ (Two-Loop recursion) . . . 87
17 Limited-Memory Momentum Quasi-Newton method (LMoQ) . . . 88
18 Direction Vector of LMoQ (Two-Loop recursion) . . . 89
19 Conjugate Gradient method (CG) . . . 100
20 MemoryLess Quasi-Newton method (MLQN) . . . 102
21 MemoryLess Nesterov's Accelerated Quasi-Newton method (MLNAQ) . . . 104
22 MemoryLess Momentum Quasi-Newton method (MLMoQ) . . . 106

図の目次

1 勾配法に基づく学習アルゴリズム 2
2 提案手法と従来手法の関係 5
3 局所的最小解と大域的最小解 9
4 機械学習とその学習法 10
5 形式ニューロン 11
6 階層型 (深層) ニューラルネットワーク 12
7 ニューロンの入出力の関係 15
8 Beale 関数の 2D 等高線図 17
9 Beale 関数の 3D モデル 18
10 GD による Beale 関数の最適化 20
11 CM による Beale 関数の最適化 21
12 CM と NAG の重みの更新ベクトル表現 22
13 NAG による Beale 関数の最適化 23
14 AdaGrad による Beale 関数の最適化 24
15 RMSprop による Beale 関数の最適化 26
16 AdaDelta による Beale 関数の最適化 27
17 Adam による Beale 関数の最適化 29
18 LPF の構造 30
19 LPF の学習データ 31
20 LPF におけるデータポイントの一部 31
21 QN と NAQ の重みの更新ベクトル表現 37
22 1 次手法のアルゴリズムの関係 38
23 2 次手法のアルゴリズムの関係 38
24 反復回数に対する μ_k の変化 40
25 関数 (68) に対する HN = 35 と 55 の NN モデルとテストモデルの比較 44
18 LPF の構造 46
19 LPF の学習データ 47
26 LPFD = 13 mm に対する AdaNAQ の NN モデルとテストデータの比較 . . . 47
27 LPFD = 15 mm に対する AdaNAQ の NN モデルとテストデータの比較 . . . 48
28 LPFD = 17 mm に対する AdaNAQ の NN モデルとテストデータの比較 . . . 49
29 LPFD = 19 mm に対する AdaNAQ の NN モデルとテストデータの比較 . . . 49
30 MPA の構造 50
31 MoQ の重みの更新ベクトル表現 57
32 2 次手法の計算コストの比較 61
33 2 次手法のメモリ量の比較 62
34 関数 (68) に対する AdaMoQ の NN モデルとテストモデルの比較 66
35 Levy 関数 $f(x_1, x_2)$ 67
36 LPF の学習誤差と反復回数グラフ 69
37 LPF の学習誤差と時間グラフ 70

38 LPFD = 13 mm に対する AdaMoQ の NN モデルとテストデータの比較 . . . 70
39 LPFD = 15 mm に対する AdaMoQ の NN モデルとテストデータの比較 . . . 71
40 LPFD = 17 mm に対する AdaMoQ の NN モデルとテストデータの比較 . . . 71
41 LPFD = 19 mm に対する AdaMoQ の NN モデルとテストデータの比較 . . . 72
42 MPA の学習誤差と反復回数グラフ 73
43 MPA の学習誤差と時間グラフ 73
44 8 × 8 MNIST 手書き数字文字のデータセットサンプル 74
45 8 × 8 MNIST の学習誤差と反復回数グラフ 75
46 8 × 8 MNIST の学習誤差と時間グラフ 76
47 8 × 8 MNIST のテスト精度と反復回数グラフ 76
48 8 × 8 MNIST のテスト精度と時間グラフ 77
49 Three-Spirals のデータセット構造 77
50 Three-Spirals の誤差と反復回数グラフ 78
51 Three-Spirals の誤差と時間グラフ 79
52 Three-Spirals の精度と反復回数グラフ 79
53 Three-Spirals の精度と時間グラフ 80
54 Three-Spirals の Adam による検証結果 80
55 Three-Spirals の AdaMoQ による検証結果 81
56 2 次手法と記憶制限手法の計算コストの比較 90
57 2 次手法と記憶制限手法のメモリ量の比較 90
58 記憶制限手法に対する Levy 関数の学習誤差と反復回数グラフ 94
59 記憶制限手法に対する Levy 関数の学習誤差と時間グラフ 96
60 2 次と手法, 記憶制限手法そしてメモリレス手法の計算コストの比較 108
61 2 次と手法, 記憶制限手法そしてメモリレス手法のメモリ量の比較 108
62 関数 (124) に対する MLMoQ の NN モデルとテストデータの比較 111
63 Three-Spirals の学習精度と時間グラフ 112
64 Three-Spirals の MLMoQ による検証結果 113
65 8 × 8 MNIST の学習誤差と反復回数グラフ 114
66 8 × 8 MNIST の検証精度と時間グラフ 114
67 28 × 28 MNIST のデータサンプルの一部 115
68 28 × 28 MNIST の学習誤差と反復回数グラフ 116
69 28 × 28 MNIST のテスト精度と反復回数グラフ 116

表の目次

1 f_1 問題に対する中間層ニューロン数の比較結果 43
2 f_1 問題に対する AdaNAQ のシミュレーション結果 45
3 f_2 問題に対する AdaNAQ のシミュレーション結果 46
4 LPF に対する AdaNAQ のシミュレーション結果 48
5 MPA に対する AdaNAQ のシミュレーション結果 50
6 2 次手法の計算コストとメモリ量の比較 61
7 関数近似問題 (68) に対する MoQ のシミュレーション結果 65
8 Levy 関数に対する MoQ のシミュレーション結果 67
9 LPF に対する MoQ のシミュレーション結果 69
10 MPA に対する AdaMoQ のシミュレーション結果 73
11 8×8 MNIST に対する AdaMoQ のシミュレーション結果 75
12 Three-Spirals に対する AdaMoQ のシミュレーション結果 78
13 記憶制限手法の計算コストとメモリ量の比較 89
14 関数 (124) に対する記憶量 m 毎のシミュレーション結果 93
15 Levy 関数に対する記憶量 m 毎のシミュレーション結果 95
16 LPF に対する記憶量 m 毎のシミュレーション結果 97
17 メモリレス手法の計算コストとメモリ量の比較 107
18 関数近似問題 (68) に対する MLMoQ のシミュレーション結果 110
19 Three-Spirals に対する MLMoQ のシミュレーション結果 111

1　序論

本研究では, 強非線形データのニューラルネットワークによる学習を高速かつ高精度に可能とする新たなアルゴリズムとして慣性項を用いた 2 次近似勾配法を提案する. 本章では, 本研究の背景と目的について詳細に述べる.

1.1　研究の背景

近年, 情報通信技術の発達により, 様々な機器の IoT 化が進み, 多様かつ膨大なデータの取得および蓄積が可能になった. これまで全く無関係であると考えられていた蓄積されたあらゆるデータを同時に扱うことで新たな知見を得ることが様々な研究で示されてきた. さらに, 取得されるデータの量とその複雑 (非線形) な関係性は日々増加し続けている. これに伴い, より複雑な関係性 (強非線形性) を内包する大規模データの高精度かつ高速な処理を可能とする技術に対する需要も日々高まっている. 近年, この需要を満たす技術の一つとして人工知能 (Artificial Intelligence, AI) が注目され, 幅広く応用されている. AI の急速な発展におけるコア技術の一つにニューラルネットワーク (Neural Networks, NN) が挙げられる [1, 2]. NN は数多くの実世界で応用されている線形または非線形最適化問題に対して有効であることが示されている [2]. その一例として, 非線形性が強い (強非線形) 特性の入出力を持つマイクロ波回路のモデリングが挙げられる [3–6]. マイクロ波回路のモデリングを目的とする NN では, ネットワークの重みは電磁波 (Electro Magnetic waves, EM) データや物理パラメータを入力とし, その周波数応答を教師信号とする強非線形データを用いて学習される. 観測済みである周波数応答によって学習されたネットワークは, 未観測 (必要とする) データに対する周波数応答を出力することが可能となり, 結果としてモデリングを可能とする [3–6]. 一般に, 物理パラメータに対する EM の振る舞いは強非線形特性を持つ [6]. これは, 基となる数式の利用が困難なとき, または基のモデルの計算コストが高いモデリングに有効である. マイクロ波回路問題の他にも, 強非線形入出力特性を持つ関数の近似問題または画像処理 [7] や自然言語処理 [8] のような分類問題など, 多くの分野における様々な問題の解決に NN が効率的に利用されている. 実世界で応用されている強非線形特性の学習問題の多くは, 高い精度と低い誤差を最小限の時間で得るために, 大規模な NN モデルと膨大な学習データを必要とする. しかし，NN の最適化には不良条件付け問題 (ill-conditioning), 勾配の消失および爆発問題 (vanishing and exploding gradients), ハイパーパラメータの最適な選択, 膨大な計算コストなどいくつかの課題が存在する. 従って, NN のモデル開発において学習 (最適

化) アルゴリズムの選択は最も重要なステップであると考えられる.

一般的に NN の学習アルゴリズムには誤差逆伝番法 (Back Propagation, BP) に基づく勾配学習法が用いられている [2]. 勾配学習法は学習率 (Learning Rate) の形状から 1 次収束特性を持つ 1 次近似勾配法 (First-Order method, 以降, 1 次手法) と 2 次もしくは超 1 次収束特性を持つ 2 次近似勾配法 (Second-Order method, 以降, 2 次手法) に分類することができる. 従来より, その計算効率の観点から, 学習率がスカラー (Scalar) である 1 次手法が広く研究され, 応用されてきた. 近年の 1 次手法の代表的なアルゴリズムとして勾配降下法 (Gradient Descent method, GD) [1, 9], モーメント法 (Classical Momentum method, CM) [10–14], ネステロフの加速勾配法 (Nesterov's Accelerated Gradient method, NAG) [1, 9, 13–15], AdaGrad [1, 9, 16], RMSprop [1, 9, 17], AdaDelta [1, 9, 18] そして Adam [1, 9, 19] が挙げられる. しかし, 1 次手法では強非線形性を内包する学習データの高精度な学習に対して, 現実的な時間では学習が困難であった [20, 21]. このため, 強非線形データの学習には, 学習率に目的関数の曲率情報であるヘッセ行列 (Hessian) とスカラーの内積を利用した 2 次手法が有効とされている [1, 20–22]. 2 次手法の中でも特に準ニュートン法 (Quasi-Newton method, QN) が用いられ, 高速化および軽量化など様々な応用研究が行われている [20–34]. 近年の応用研究の一つとして, QN を高速化した慣性付 2 次近似勾配モデルを用いた準ニュートン法, 別名ネストロフの加速準ニュートン法 (Nesterov's Accelerated Quasi-Newton method, NAQ) が挙げられる [32–34]. 本節で明記した 1 次と 2 次手法の各アルゴリズムの位置関係を図 1 に示す.

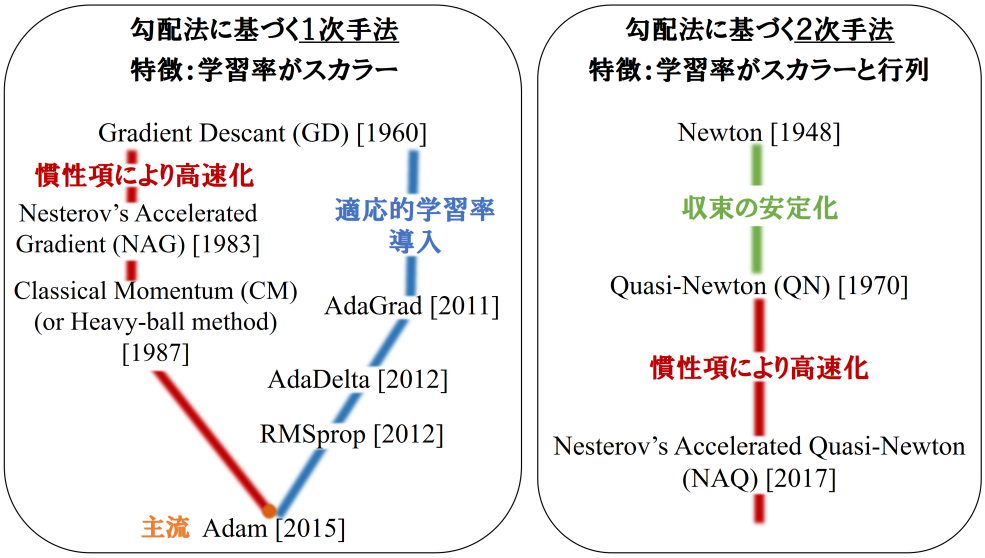


図 1: 勾配法に基づく学習アルゴリズム

1.2　研究の目的

現代の情報社会では, 取得されるデータの量とその非線形性が日々増すばかりである. これに伴い, 大規模な強非線形データに対して高精度かつ高速な処理を可能とする最適化手法の必要性も日々増加し続けている. 近年, NN の最適化手法として使用される 1 次手法に慣性項を用いることで, 学習速度を大幅に高速化した研究が, 数多く提案された [1, 11–14, 16–19]. 一方で, 強非線形問題の学習に対しては 2 次手法の QN は有効である [1, 20–22, 34]. 本研究では, 1 次手法に対する慣性項の影響を 2 次手法においても同様に得られることを期待し, 2 次手法における慣性項の影響を調査する. さらに, 慣性項を用いた 2 次手法の問題点に対して解決策を提案する. これにより, 強非線形問題の最適化に適した高精度かつ高速な学習アルゴリズムとして慣性項を用いた 2 次手法の体系化を試みる. このため, 慣性項の導入により QN の学習全体の反復回数を削減し, 高速化に成功した NAQ [32–34] に着目した. NAQ ではいくつかの問題点が存在した. 本研究では, 以下の NAQ の問題点を克服した新たな慣性項を用いた 2 次手法の提案を目指す.

1. NAQ における慣性項のモーメント係数は固定値（ハイパーパラメータ）である. さらに, 複数回の試行における学習結果は NN の重みの初期値に対して異なるため, 学習が不安定である. 従って, 最適値を求めるため手間と時間を必要とする.

2. NAQ では 1 反復において勾配計算回数が 1 回である QN と異なり, 2 回必要であり, これは QN と比較して NAQ の計算時間の増加に繋がる.

3. NAQ ではヘッセ行列の逆近似行列が利用されているため, 使用するメモリ量や計算コストが 1 次手法と比較して高く, 計算機の性能によっては学習が困難となる.

本研究では, 1 つ目の問題点である, NAQ における慣性項のハイパーパラメータ問題に対しては, 慣性項のモーメント係数に適応的慣性 (モーメント) 係数 [35, 36, 39] を導入することで, その克服を試みた. 慣性項のモーメント係数はハイパーパラメータであるため, 学習では全ての反復で固定されていた [34]. これは, 学習を重みの初期値に対して不安定化させ, 複数回の試行において同じ学習精度を得られない原因であった. 従って, モーメント係数の最適値を決めるには多くの実験, 経験と時間が必要であった. この問題を解決するため, 適応的慣性係数を NAQ に導入し, 適応的ネステロフの加速準ニュートン法 (Adaptive Nesterov's Accelerated Quasi-Newton method, AdaNAQ) として提案する [35, 36]. 適応的慣性係数 [13] は 1 次手法である NAG [13, 14] における凸最適化問題に対して提案された [13]. 本研究 [35, 36] では, 適応的慣性係数を用いた AdaNAQ を NN の学習に応用し, 収束の安定

性と重みの初期値に対する学習のロバスト性の向上を目指した.

次に, 2 つ目の問題点を克服した新たな準ニュートン法に基づく学習アルゴリズムとして慣性付準ニュートン法 (Momentum Quasi-Newton method, MoQ) を提案する [37–39]. 具体的に, NAQ では 1 反復において, 通常勾配とネステロフの加速勾配の 2 つの勾配計算が必要であった. これに対して, QN は 1 反復において, 1 度の通常勾配計算でアルゴリズムが構成されていた. つまり, NAQ では慣性項と 2 つの勾配の影響により学習全体の反復回数が減少し, 全体の計算時間が大幅に削減された. しかし, これは 1 反復における計算時間を, QN と比較して増加させてしまう問題点に繋がった. 提案手法 MoQ は, 誤差関数を 2 次関数と見なすことで, NAQ のネステロフの加速勾配を通常勾配の重み付きの線形和として近似し, 勾配計算回数を 2 回から 1 回に削減した手法である. また提案手法の学習の安定性とパラメトリック化を目指すため, 適応的慣性係数を導入した適応的慣性付準ニュートン法 (Adaptive Momentum Quasi-Newton method, AdaMoQ) も提案する [40].

3 つ目の問題点であるメモリ量と計算コスト問題を克服するため本研究では 2 つの手法を提案する. 最初の解決策として, NAQ および MoQ に, QN で用いられる記憶制限手法 (Limited-Memory strategy) [20, 21] を導入した記憶制限ネステロフの加速準ニュートン法 (Limited-Memory Nesterov's Accelerated Quasi-Newton method, LNAQ) と記憶制限慣性付準ニュートン法 (Limited-Memory Momentum Quasi-Newton method, LMoQ) を提案する [41–44]. 記憶制限手法はヘッセ行列のメモリ量を任意の次元まで制限することで計算コストを削減した手法である [20, 21]. 本研究では, この手法を NAQ と MoQ に導入することで, NAQ と MoQ の非線形問題に対する高速な学習収束を保ちながら行列計算量の削減を目指した. 次に, NAQ と MoQ に, QN で用いられるメモリレス手法 (MemoryLess strategy) [30, 31] を導入したメモリレスネステロフの加速準ニュートン法 (MemoryLess Nesterov's Accelerated Quasi-Newton method, MLNAQ) とメモリレス慣性付準ニュートン法 (MemoryLess Momentum Quasi-Newton method, MLMoQ) を提案する [45–47]. メモリレス手法は, 過去のヘッセ行列を保存せずに学習を行う手法である. 本研究では, NAQ と MoQ にメモリレス手法を導入することで, 提案手法の計算コストを 1 次手法と同程度に削減しながらも, NAQ と MoQ の QN に対する高速な収束性を維持する新たな学習アルゴリズムの提案を目指した. 本研究では, 提案手法 AdaNAQ, MoQ と AdaMoQ, 記憶制限 NAQ と MoQ およびメモリレス NAQ と MoQ を関数近似問題, 実世界で実用されているマイクロ波回路モデリング問題および分類問題に対する NN の学習に応用し, 従来手法との比較を行った上, その有効性を計算機実験により示した.

本研究の最終目標は, 2 次手法における慣性項の影響を調査し, 強非線形問題の最適化に

適した高精度かつ高速な学習アルゴリズムとして慣性項を用いた 2 次手法を体系化させることである. 従って, 上記の問題点に対する解決策を提案することで, 強非線形問題の高精度かつ高速な処理を可能とする洗練された新たな 2 次手法のアルゴリズムが提案できると信ずる. 本研究の提案手法と従来手法の関係性を図 2 に示す.

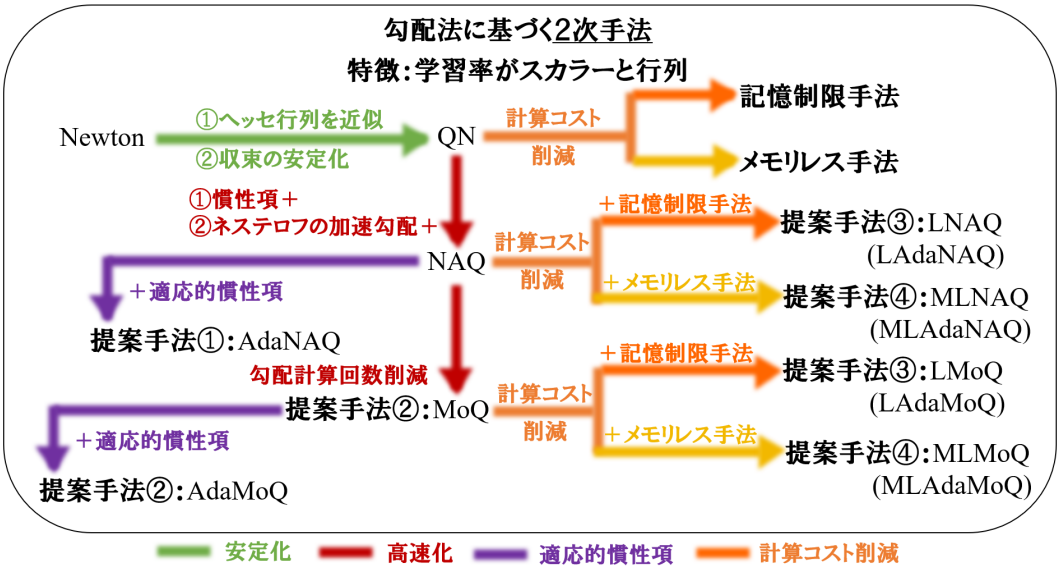


図 2: 提案手法と従来手法の関係

1.3 論文の構成

本論文は下記に示す構成となっている:

- **第 2 章:** 本章では, 最初に勾配法に基づく最適化と NN の概念に関して説明する. 次に, NN における最適化手法として使用される 1 次と 2 次 (超 1 次) の従来手法を紹介する.
- **第 3 章:** 本章では, 適応的慣性係数を用いた NAQ, 適応的ネステロフの加速準ニュートン法 (AdaNAQ) を提案する. 最後に提案手法の有効性を実証した計算機実験の結果を示す.
- **第 4 章:** 本章では, 強非線形問題の最適化に有効とされている NAQ の高速化を行った慣性付準ニュートン法 (MoQ) を提案する. さらに, MoQ の収束性に関しても示す. 最後に提案手法の有効性を実証した実験結果を示す.
- **第 5 章:** 本章では, NAQ の計算コストを削減した記憶制限 NAQ と, その高速化を目指した記憶制限 MoQ を提案する. さらに, 従来手法の記憶制限 QN の紹介に加え計算コストについても議論を行う. 最後に提案手法の有効性を実証した実験結果を示す.
- **第 6 章:** 本章では, メモリレス QN の高速化を行うと共に, 記憶制限手法と比べて, NAQ と MoQ の計算コストをさらに削減したメモリレス NAQ と MoQ を提案する. さらに, 従来手法の共役勾配法 (CG) とメモリレス QN の紹介に加え計算コストについても議論を行う. 最後に提案手法の有効性を実証した実験結果を示す.
- **第 7 章:** 本章では, 本研究の結論と今後の展望について述べる.

表記

本節では, 本論文を通して使用される数学表記および固定の変数表記を簡潔に説明する. なお, ここで定義されていない変数表記に関しては各章または節で異なる役割で使用されており，対応する章または節にて説明とその求め方が記されている.

数学表記

- a : スカラー (整数または実数)
- $\mathbf{a}$: 縦ベクトル
- a_i : $\mathbf{a}$ ベクトルの i 番目の要素
- $\mathbf{a}^{\mathrm{T}}$: $\mathbf{a}$ ベクトルの転置 (横ベクトル)
- $\mathbb{R}^d$: 実数の集合
- $[a, b]$: a と b を含む実区間
- $(a, b]$: a は含まず b は含む実区間
- $\|\mathbf{a}\|$: $\mathbf{a}$ ベクトルの L_2 ノルム
- $a_i \in \mathbf{b}$: a_i は $\mathbf{b}$ に含まれる
- $\frac{\partial a}{\partial b}$: a の b に関する偏微分
- $\mathbf{I}$: 単位行列
- $f(\mathbf{w}; \mathbf{x}, \mathbf{d})$:$\mathbf{x}$ と $\mathbf{d}$ を入力に持つ $f(\mathbf{w})$ の関数

固定変数表記

- k : 反復回数
- d : パラメータの次元数
- $|T_r| = n$: 学習データサンプル T_r の全体数
- $|T_e|$: テストデータサンプル T_e の全体数
- $\mathbf{w}$: 重みベクトル
- $\mathbf{v}$: 更新ベクトル
- ϵ : 学習終了条件
- μ : 慣性 (モーメント) 係数
- $\mu\mathbf{v}$: 慣性項
- η : 学習率
- α : ステップサイズ
- $\mathbf{g}$: 探索方向ベクトル
- $E(\mathbf{w})$: 誤差関数（目的関数, 評価関数）
- $\nabla E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$: 誤差関数 $E(\mathbf{w})$ における勾配 (通常勾配)
- $\nabla E(\mathbf{w} + \mu\mathbf{v}) = \frac{\partial E(\mathbf{w}+\mu\mathbf{v})}{\partial(\mathbf{w}+\mu\mathbf{v})}$: 誤差関数 $E(\mathbf{w} + \mu\mathbf{v})$ における勾配 (ネステロフの加速勾配)

- $\nabla^2 E(\mathbf{w}) = \frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w}^2}$: $E(\mathbf{w})$ の $\mathbf{w}$ ベクトルに関する 2 階偏微分, 別名ヘッセ行列
- $\mathbf{B}$: ヘッセ行列の近似行列（$\mathbf{B}$ 行列）
- $\mathbf{H}$: ヘッセ行列の逆近似行列（$\mathbf{H}$ 行列）
- $\mathbf{s}, \mathbf{y}$: $\mathbf{H}^{\mathrm{QN}}$ 行列の更新で使用されるベクトル
- $\mathbf{p}, \mathbf{q}$: $\mathbf{H}^{\mathrm{NAQ}}$ 行列の更新で使用されるベクトル
- $\mathbf{p}, \hat{\mathbf{q}}$: $\mathbf{H}^{\mathrm{MoQ}}$ 行列の更新で使用されるベクトル (MoQ の $\mathbf{p}$ は NAQ と同様に求まる)
- m : 記憶制限手法で使用されるメモリ量

2　最適化とニューラルネットワーク

本章では, 勾配法に基づく最適化とニューラルネットワーク (NN) の概念, と NN における勾配の導出法について説明した後, NN の学習アルゴリズムである 1 次手法と 2 次手法のそれぞれの従来アルゴリズムを紹介する. 本章で紹介する 1 次手法の従来アルゴリズムは, GD [1, 48, 49], CM [1, 11, 12, 48, 49], NAG [1, 13, 14, 48, 49], AdaGrad [1, 9, 16, 48, 49], RMSprop [1,9,17,48,49], AdaDelta [1,9,18,48,49] そして Adam [1,9,19,48,49] であり, 2 次手法の従来アルゴリズムは, Newton [20,21], QN [20,21] そして NAQ [32–34] である.

2.1　勾配法に基づく最適化と NN の概念

最適化問題 (Optimization Problem), 別名数理計画問題 (Mathematical Programming Problem) とは,「与えられた制約条件 (Restrict Condition) の下で, ある目的関数 (Objective Function, 別名: 誤差関数 (Error Function) または評価関数 (Evaluation Function)) を最小または最大にする解を求めること」である. この問題は統計学, 物理学, 脳科学そして化学などの自然科学だけでなく社会学や経済学など多種多様な分野の様々な問題に当てはまる. 最適化問題では制約条件, 誤差関数およびパラメータ (Parameter) といった要素を定義し, 数理モデルを構築した上で最適解 (Optimum Solution) を最適化手法 (Optimization Algorithm) により求める. 例として, 図 3 に示す「$f(x) = (1/4)x^4 + (2/3)x^3 - (1/2)x^2 - 2x + 1$ を最小にする $x \in \mathbb{R}^d$ を求める問題」, いわば無制約最小化問題が挙げられる. この問題は非線形な関数であ

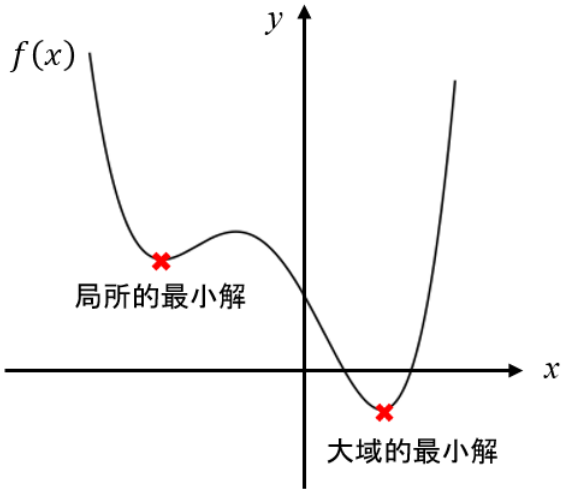


図 3: 局所的最小解と大域的最小解

り, 2 つの最小解, 局所的最小解 (Local Optimum Solution) と大域的最小解 (Global Optimum Solution) を持つ. 一般的に, 最適化手法として勾配法 (Gradient method) が用いらる. 勾配法では, 大域的最小解を求めることが最終目標となる. 勾配法は, 学習率 (Learning Rate) の形状から 1 次収束特性を持つ 1 次近似勾配法 (First-Order method, 以降, 1 次手法) と 2 次もしくは超 1 次収束特性を持つ 2 次近似勾配法 (Second-Oreder method, 以降, 2 次手法) に分類できる. 1 次手法では, 学習率がスカラーであり, 計算コストが低い観点から様々な研究で応用されている. しかし, 1 次近似手法では, 非線形性が強い問題の最適化において誤差関数と制約条件が複雑な関係を持つため, 局所的最小解, 鞍点 (Saddle points) またはプラトー領域 (Flat Regions) で停留する傾向がある. 従って, 大域的最小解を求めることが困難となり, 収束速度が低下する [1,26,50]. この問題に対して, 学習率に目的関数の曲率情報であるヘッセ行列 (Hessian) とスカラーの内積を利用した 2 次手法が有効とされている.

一方で, 近年, 各分野における様々な問題に対して分類, 予測, 判断およびモデリングを可能とする手法として機械学習 (Machine Learning) が多くの注目を集めている. 機械学習では, 最適化手法はプロセスの一環であり, 大きく教師あり学習 (Supervised Learning), 半教師あり学習 (Semi-supervised Learning), 教師なし学習 (Unsupervised Learning) そして強化学習 (Reinforcement Learning) に分類することができる. 機械学習とその学習法を図 4 に示す. 教師あり学習の学習モデルとしてニューラルネットワーク (Neural Networks, NN) が挙げられる. NN は, 機械学習がブームへと繋がったコア技術として知られている. NN は, 脳の神経細胞 (ニューロン, Neuron) を数理モデル化した技術である. ニューロンの数理モデルは提案当初, 形式ニューロン (Formal Neuron) または人工ニューロン (Artificial Neuron) と呼ばれていた [48,49,51]. 形式ニューロンを図 5 に示す. 図 5 では, $\mathbf{x}$, $\mathbf{w}$, z そして o はそれぞれ入力, 重み (Weight, 別名結合荷重) ベクトルと内部状態そして出力を示す. 形式ニューロンにおけ

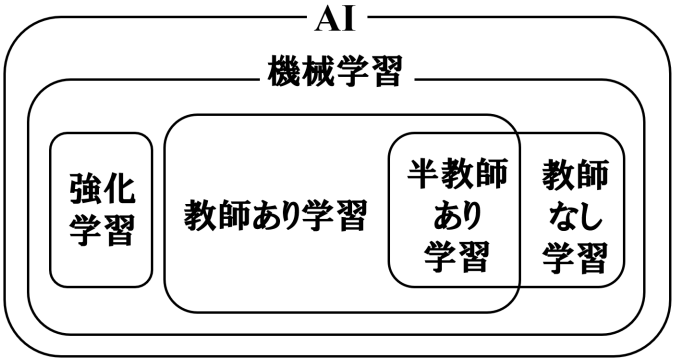


図 4: 機械学習とその学習法

る出力 o の計算法を (1) と (2) に示す.

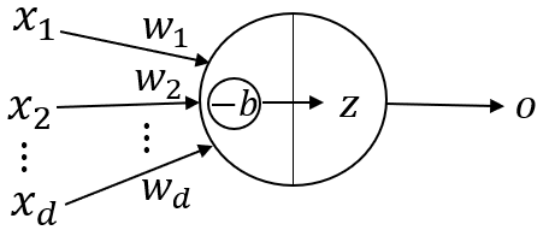


図 5: 形式ニューロン

$$\begin{aligned} z &= \sum_{l=1}^{d} \mathbf{w}_l \cdot \mathbf{x}_l - b \\ &= \sum_{l=1}^{d+1} \mathbf{w}_l \cdot \mathbf{x}_l, \end{aligned} \tag{1}$$

$$o = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases} . \tag{2}$$

(1) では, b は閾値 (bias) であり, 本研究では入力が 1 の重み, $b = w \cdot 1$ と考える. 形式ニューロンでは (2) に示すように, ニューロンの出す信号は 1(On) または 0(Off) の情報しかない. 内部状態 z から出力 o に変換する関数を活性化関数 (Activation Function) と呼び, 形式ニューロンでは階段関数 (Step Function) が使われている. [51] で提案された形式ニューロンでは, 重み $\mathbf{w}$ は固定の値として考えられていた. 従って, 入力から出力を計算する単純なシステムであり, 学習を行うことが出来ず, 様々な問題に対する応用が困難であった. 一方で, 形式ニューロンの重み $\mathbf{w}$ を可変のパラメータとして考えた手法, 単純パーセプトロン (Simple Perceptron) が提案された [48, 49, 52, 53]. パーセプトロンでは重み $\mathbf{w}$ の最適化を学習 (Learning, Training) と呼び, 反復法 (Iterative method) でその更新を行った [53, 54]. 単純パーセプトロンにおいて, 線形分離が可能な問題では, 有限回の反復で重み $\mathbf{w}$ を最適化し, 問題を解くことができることが示された [53, 54]. しかし, 単純パーセプトロンでは排他的論理和 (eXclusive OR, XOR) などのような線形分離ができない問題の学習は不可能であると指摘された [55]. この問題に対して, 多数のパーセプトロンを階層的に組み合わせた多層パーセプトロン (MultiLayer Perceptron, MLP), 別名, 階層型ニューラルネットワーク (MultiLayer Neural Networks) が提案された [56–60]. さらに, MLP の学習法として, 微分可能な活性化関数を使用し, 出力から

の誤差を逆伝播しながらパラメータの最適化を行う誤差逆伝播法 (Back Propagation method, BP) [56–60] が提案された. この手法では, 十分な数のニューロンが中間層にあれば, 線形分離ができない問題を学習することが可能になった. さらに, 生物学における小脳の働きは MLP と酷似するという研究報告 [61–63] により, MLP に対する注目が増加した. しかし, MLP では, 勾配消失問題 (Vanishing Gradient Problem) および過学習問題 (Overtraining Problem) により, その注目が弱まった. その後, 解の精度向上を目指し, 様々な研究が行われた. その中でも, 再度 NN に対する注目を強めた手法は, MLP の中間層の数をさらに深めた深層ニューラルネットワーク (Deep Neural Networks, DNN) [64] である. DNN では, 単純パーセプトロンを複数接続させてネットワークを形成した考え方であり, 入力層 (Input Layer), 中間 (隠れ) 層 (Hidden Layer) そして出力層 (Output Layer) から成る. 図 6 に DNN を示す. この図からもわかるように, DNN では出力層を除き, ニューロンの出力が次層のニューロンの入力となっている.

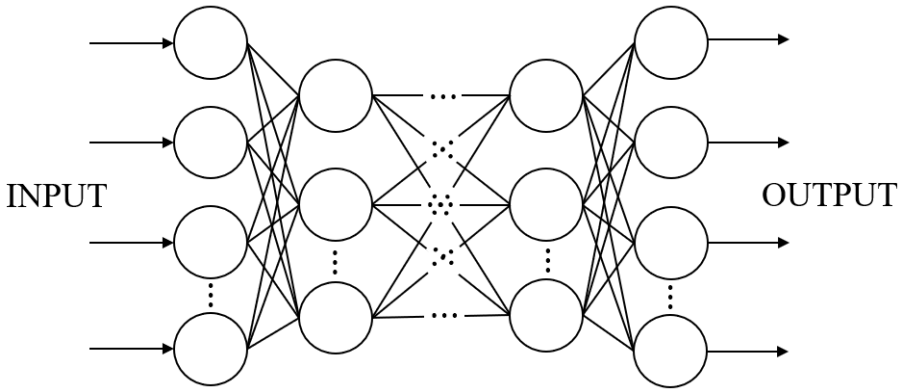


図 6: 階層型 (深層) ニューラルネットワーク

DNN では, ネットワークの層またはニューロンを増やすことで，線形分離が不可能な問題の分離が可能になった [59]. 一般的に層を増やした DNN による学習を深層学習 (Deep Learning) という. 近年, DNN を用いた新なネットワークとして, 畳み込みニューラルネットワーク (Convolutional Neural Networks, CNN), 再帰型ニューラルネットワーク (Recurrent Neural Networks, RNN) および敵対的生成ネットワーク (Generative Adversarial Networks, GAN) などのような複雑な構造を持つネットワークが数多く提案されており, アプリケーションとしても実用化されている [1, 48, 49].

本研究では, DNN(以降, NN) を用いた教師あり学習に着目する. 教師あり学習では, n 個

のサンプルを持つ学習用データセット T_r が与えられたとき, 各サンプルは入力データ $\mathbf{x}_p$ と出力ラベル (教師信号, Teach-Signal or desired-Signal) $\mathbf{d}_p$ のペア, $(\mathbf{x}_p, \mathbf{d}_p)$ から構成されている. つまり, $T_r = (\mathbf{x}_1, \mathbf{d}_1), ..., (\mathbf{x}_p, \mathbf{d}_p), p \in n$ となる. そして教師あり学習では NN が予測する出力 $\mathbf{o}_p$ と教師信号の誤差の総和が最小になるよう重み $\mathbf{w}$ の最適化を行う. 従って, NN の教師あり学習は最小化を目指した最適化問題に帰着する. NN は一般的に BP に基づく勾配降下学習法 (勾配法) を用いて学習を行う. BP では与えられた重み $\mathbf{w}$ に関して, 累積誤差関数の偏導関数 (逆伝播) に比例した量に準じて修正または更新を行う. BP を用いた勾配学習法では一般的に (3) に示す反復式により誤差関数を最小にする重みベクトル $\mathbf{w}$ を求める.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}. \tag{3}$$

ここで, k は反復回数であり, $\mathbf{v}_{k+1}$ は更新ベクトルを示す. 更新ベクトルは学習アルゴリズムに強く依存し, アルゴリズムによってその導出方法と計算コストおよびメモリ量が異なる. 従って, 問題に対して最適な学習アルゴリズムを選択することは NN の学習において重要となる.

2.2　NN における勾配の導出

本節では, BP に基づく勾配法を用いた NN における勾配の導出を紹介する. NN の学習は, ランダムな値によって初期化された重み $\mathbf{w} \in \mathbb{R}^d$ を最適解に収束させるため繰り返し更新を行う, 反復法を使用している. ここで最適解とは, (4) の誤差関数 $E(\mathbf{w})$ の累積を最小化することであり, NN の学習は, 無制約最適化問題となる.

$$\min_{\mathbf{w} \in \mathbb{R}^d} \quad E(\mathbf{w}). \tag{4}$$

BP を用いた学習はバッチ手法 (Batch strategy) と確率的 (ミニバッチ) 手法 (Mini-Batch strategy) に分類することが出来る. バッチ学習法では, 各反復で学習データセット T_r を全て用いて学習を行い, その勾配を計算する. 従って, (4) の誤差関数 $E(\mathbf{w})$ は,

$$E(\mathbf{w}) = \frac{1}{|T_r|} \sum_{p \in T_r} E_p(\mathbf{w}), \tag{5}$$

と定義される. ここで, $|T_r|$ はサンプル数, p はサンプルパターン番号を示す. バッチ学習法は非線形性が強い問題の最適化に有効な手法である. 一方で, ミニバッチ学習法では, 各反復で T_r からランダムに抽出された $X \subseteq T_r$ 個のデータを用いて学習を行い, 勾配は (6) に示

す誤差関数 $E_b(\mathbf{w})$ によって求められる.

$$E_b(\mathbf{w}) = \frac{1}{b}\sum_{p\in X} E_p(\mathbf{w}). \tag{6}$$

ここで, $b = |X|$ はミニバッチサイズであり, 任意の値に設定される. バッチ手法の場合 $X = T_r$, $b = |T_r|$ となる. ミニバッチ学習法は, 主にデータ量が多く, バッチ学習が困難な問題の最適化に対して有効とされている. 本研究では, 強非線形問題の高精度かつ高速な処理を可能とする学習アルゴリズムの開発を目指すため, バッチ学習法にのみ着目する.

次に, NN の入出力の関係を

$$\mathbf{o}_p = \mathbf{x}_p^{out} = f_{\mathrm{NN}}(\mathbf{w}; \mathbf{x}_p, \mathbf{d}_p), \tag{7}$$

と定義する. ここで, $x_{i,p}^s (1 \le s \le out)$ を p 番目の入力に対する s 層の i 番目のニューロンの出力とし, $s-1$ 層の j 番目のニューロンから s 層の i 番目のニューロンへの重みを w_{ij}^s とする. つまり, 重みベクトル $\mathbf{w}$ と s 層における i 番目のニューロンの重みベクトル $\mathbf{w}_i^s$ の関係性を (8) に示す.

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1^1 \\ \mathbf{w}_1^2 \\ \vdots \\ \mathbf{w}_i^s \\ \vdots \end{bmatrix}. \tag{8}$$

ここで, s 層の i 番目のニューロンの重みベクトル $\mathbf{w}_i^s$ の j 番目の要素, w_{ij}^s は (9) となる.

$$\mathbf{w}_i^s = \begin{bmatrix} w_{i1}^s \\ \vdots \\ w_{ij}^s \\ \vdots \end{bmatrix}. \tag{9}$$

従って, ニューロンの入出力関係は図 7 のように示すことができる. さらに, その関係性は次のように与えられる.

$$x_{i,p}^s = f(z_{i,p}^s), \tag{10}$$

$$z_{i,p}^s = \sum_j w_{ij}^s \cdot x_{j,p}^{s-1}. \tag{11}$$

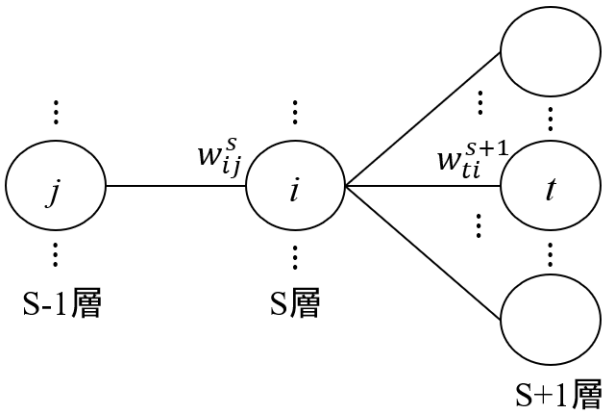


図 7: ニューロンの入出力の関係

ここで, $f(z_{i,p}^{s})$ と w_{ij}^{s} はそれぞれ活性化関数と重みベクトル $\mathbf{w}$ の要素である. 一般的な勾配降下法 (Gradient Descent method, GD) の更新式は, 学習率 η_k を用いて (12) と (13) によって定義される.

$$w_{ij}^{s}(k+1) = w_{ij}^{s}(k) + v_{ij}^{s}(k+1), \tag{12}$$

$$v_{ij}^{s}(k+1) = -\eta_k \frac{\partial E_p(\mathbf{w})}{\partial w_{ij}^{s}}. \tag{13}$$

ここで, k は反復回数, $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ は勾配であり, 連鎖律 (多変数関数の合成関数の微分, 別名: チェーンルール) を用いて (14) のように示される.

$$\frac{\partial E_p(\mathbf{w})}{\partial w_{ij}^{s}} = \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^{s}} \cdot \frac{\partial x_{i,p}^{s}}{\partial z_{i,p}^{s}} \cdot \frac{\partial z_{i,p}^{s}}{\partial w_{ij}^{s}} = \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^{s}} \cdot f'(z_{i,p}^{s}) \cdot x_{j,p}^{s-1}. \tag{14}$$

ここで, $f'(z_{i,p}^{s})$ は活性化関数の微分であり, 偏微分 $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ の解は, ニューロンが出力層 ($s = out$) の場合とそうでない場合に分けて求められる.

► **s が出力層 ($s = out$) の場合:**

s が出力層の場合, $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ が誤差関数から求められる. ここで, 平均二乗誤差 (Mean Squared Error, MSE) とシグモイド (Sigmoid) 活性化関数および交差エントロピー誤差 (Cross Entropy Error, CE) とソフトマックス (SoftMax) 活性化関数を用いた場合の $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ の解を示す.

<MSE と *sigmoid* 関数 >

MSE とシグモイド関数をそれぞれ (15) と (16) に示す.

$$E_p(\mathbf{w}) = \sum_{i=1}^{L} \frac{1}{2}(d_{i,p} - x_{i,p}^{out})^2, \tag{15}$$

$$x_{i,p}^{out} = f(z_{i,p}^{s}) = \frac{1}{1 + exp(-z_{i,p}^{s})}. \tag{16}$$

ここで，L は出力ユニットの数を示し, s 層が出力層の場合 $L = i$ となる. $d_{i,p}$ は出力 $x_{i,p}^{out}$ に対する教師信号である. 従って, (15) と (16) を用いた $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ と更新ベクトル $v_{ij}^{s}(k+1)$ はそれぞれ (17) と (18) のように求まる.

$$\frac{\partial E_p(\mathbf{w})}{\partial w_{ij}^{s}} = \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^{s}} \cdot f'(z_{i,p}^{s}) \cdot x_{j,p}^{s-1} = -(d_{i,p} - x_{i,p}^{out}) \cdot x_{i,p}^{out} \cdot (1 - x_{i,p}^{out}) \cdot x_{j,p}^{s-1}. \tag{17}$$

$$v_{ij}^{s}(k+1) = \eta_k\{(d_{i,p} - x_{i,p}^{out}) \cdot x_{i,p}^{out} \cdot (1 - x_{i,p}^{out}) \cdot x_{j,p}^{s-1}\}. \tag{18}$$

<CE と *softmax* 関数 >

CE とソフトマックス関数をそれぞれ (19) と (20) に示す.

$$E_p(\mathbf{w}) = -\sum_{i=1}^{L} d_{i,p} \log(x_{i,p}^{out}), \tag{19}$$

$$x_{i,p}^{out} = f(z_{i,p}^{s}) = \frac{e^{(z_{i,p}^{s})}}{\sum_{i=1}^{L} e^{(z_{i,p}^{s})}}. \tag{20}$$

ここで, L は出力のユニット, 別名分類クラス数であり, s 層が出力層の場合 $L = i$ となる. 勾配 $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ と更新ベクトル $v_{ij}^{s}(k+1)$ はそれぞれ (21) と (22) のように求まる.

$$\frac{\partial E_p(\mathbf{w})}{\partial w_{ij}^{s}} = \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^{s}} \cdot f'(z_{i,p}^{s}) \cdot x_{j,p}^{s-1} = -\sum_{i=1}^{L}(d_{i,p} - x_{i,p}^{out}) \cdot x_{j,p}^{s-1}, \tag{21}$$

$$v_{ij}^{s}(k+1) = \eta_k \sum_{i=1}^{L}(d_{i,p} - x_{i,p}^{out}) \cdot x_{j,p}^{s-1}. \tag{22}$$

► s が出力層以外の場合:

s 層が出力層以外の場合, GD における勾配 $\partial E_p(\mathbf{w})/\partial w_{ij}^{s}$ は (23) に示される.

$$\frac{\partial E_p(\mathbf{w})}{\partial w_{ij}^{s}} = \sum_{t} \frac{\partial E_p(\mathbf{w})}{\partial x_{t,p}^{s+1}} \cdot f'(z_{t,p}^{s+1}) \cdot w_{ti}^{s+1} \cdot f'(z_{i,p}^{s}) \cdot x_{j,p}^{s-1}. \tag{23}$$

ここで, t は $s+1$ 層目のニューロン数を示す. このように BP では層が増える毎に逆伝播成分 $(\partial E_p(\mathbf{w})/\partial x_{t,p}^{s+1}) \cdot f'(z_{t,p}^{s+1})$ を入力層の重みまで逆伝播しながら, 重みの更新を行う.

2.3　1 次近似勾配学習法

本節では, NN における 1 次近似勾配学習法に基づく 7 つの従来手法を紹介する. これらの 1 次手法は実数値関数の最小化問題に対して, スカラーである学習率 η と 1 次情報 (1 回微分) である誤差関数の勾配 $\nabla E(\mathbf{w})$ を用いて, 近似解を出力する. 従って, 計算コストが低く, NN の学習に対して一般的に用いられ, 様々な研究の対象とされている. ここで, 紹介する従来の 1 次学習法は勾配降下法 (GD) [1,48,49,51] や慣性項を用いることで GD を高速化したモーメント法 (CM) [1,9,11–14,48,49] およびネステロフの加速勾配法 (NAG) [1,9,13,14,48,49], 過去の 2 乗勾配の累計を指数関数的に減衰する平均によって決定する適応的学習率を持つ AdaGrad [1,9,16,48,49], RMSprop [1,9,17,48,49], AdaDelta [1,9,18,48,49] および慣性項と適応的学習率の両方を持つ 1 次手法の代表的な学習アルゴリズムの Adam [1,9,19,48,49] である.

本節で紹介するアルゴリズムの特性をより具体的に説明するため, 1 次手法を用いた最適化が可能である非線形関数, (9) に示す Beale 関数 [65] に対する最適化過程を示す.

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \tag{24}$$

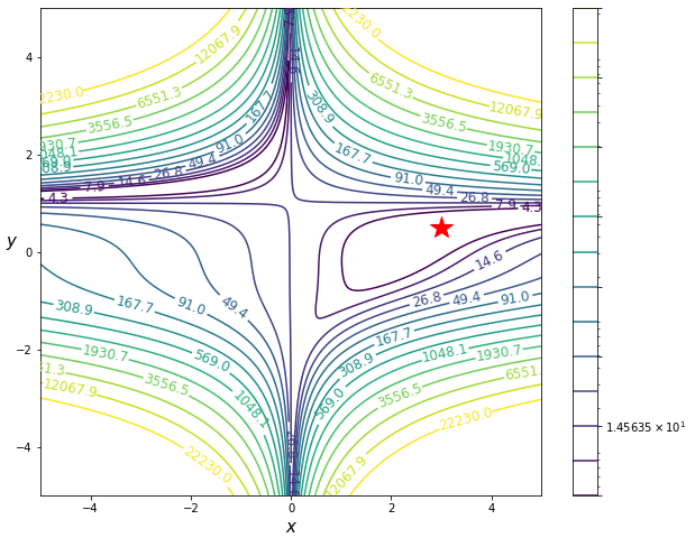


図 8: Beale 関数の 2D 等高線図

(9) は 2 つのパラメータ $-4.5 \leqq x, y \leqq 4.5$ を持つ関数であり, 本研究では初期値を $x = 7, y = 1.4$ そして最大反復回数 $k_{max} = 1,000$ とした. 大域的最小解は $x = 3, y = 0.5$ である. Beale 関数の 2 次元と 3 次元の等高線図 (Contour Plot) をそれぞれ図 8 と 9 に示す. 図 8 では赤い星印は大域的最小解である.

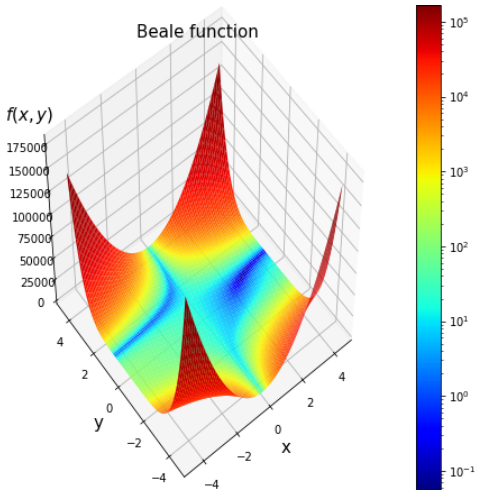


図 9: Beale 関数の 3D モデル

2.3.1　Gradient Descent method

勾配降下法 (Gradient Descent method, GD) は, 誤差関数の最小値を求めるためのもっとも古くかつ単純なアルゴリズムである [1, 48, 49, 51]. また, GD は勾配が最も降下する方向へ重み $\mathbf{w}$ を更新するため, 最急降下法 (Steepest Descent, SD) とも呼ばれている. GD では, (3) の更新ベクトル $\mathbf{v}_{k+1}$ を学習率 η_k と (25) に示す誤差関数 $E(\mathbf{w})$ の $\mathbf{w}_k$ における勾配ベクトル $\nabla E(\mathbf{w}_k)$ を用いて, (26) のように更新する.

$$\nabla E(\mathbf{w}_k) = \begin{bmatrix} \frac{\partial E(\mathbf{w}_k)}{\partial w_{1,k}} \\ \frac{\partial E(\mathbf{w}_k)}{\partial w_{2,k}} \\ \vdots \\ \frac{\partial E(\mathbf{w}_k)}{\partial w_{d,k}} \end{bmatrix}. \tag{25}$$

$$\mathbf{v}_{k+1} = -\eta_k \nabla E(\mathbf{w}_k). \tag{26}$$

(25) において d は重み $\mathbf{w}$ の次元数である. GD のアルゴリズムを Algorithm 1 に示す. Algorithm 1 では, ϵ は終了条件であり, η_k の範囲は $0 < \eta_k < 1$ とされており, 1 次手法の学習では任意の値に固定される [1, 2, 22, 26, 48, 49]. ただし, 学習率の値が大きすぎると, アルゴリズムの挙動が非常に不安定になり, 収束が困難になる可能性がある. 一方, 小さすぎると, 勾配が小さくなった領域ではアルゴリズムが停滞する可能性がある.

(9) に示す Beale 関数の最適解が求まるまでの GD による学習軌道を図 10 に示す. この問題では, 学習率 $\eta_k = 0.01$ とした. 学習結果より, 最適解が求まるまで 838 回の反復回数を必要とする.

Algorithm 1 Gradient Descent method (GD)

Require: $\epsilon, k_{max}, \eta_k$
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$
1: $k = 1$
2: **while** ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
3: Calculate $\nabla E(\mathbf{w}_k)$;
4: Update $\mathbf{v}_{k+1} = -\eta_k \nabla E(\mathbf{w}_k)$;
5: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
6: $k = k + 1$;
7: **end while**
Return: $\mathbf{w}_k$

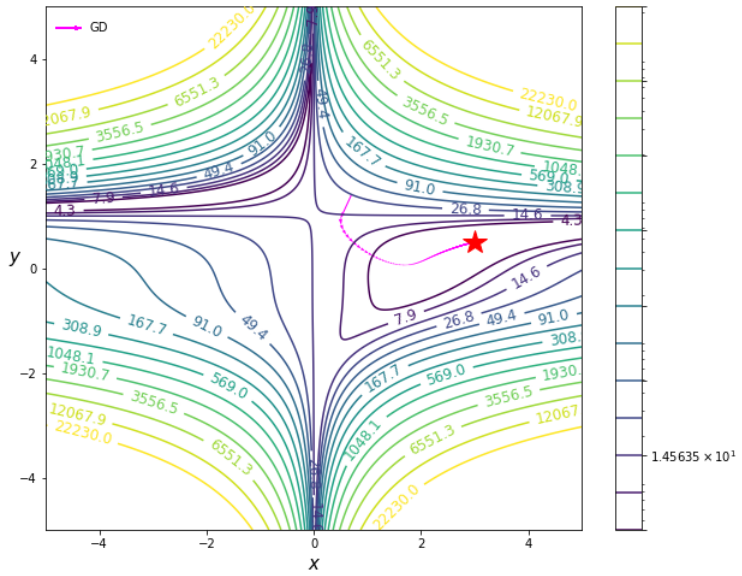


図 10: GD による Beale 関数の最適化

2.3.2　Classical Momentum method

モーメント法 (Classical Momentum method, CM) は GD に慣性項を導入することでを高速化した手法であり, 学習において継続的な減衰の方向に過去の更新ベクトルを蓄積する [1, 10, 13, 14, 48, 49]. CM は, 坂道を転がる重い (ヘビー) ボールがその軌道に沿って進行する考え方に基づいているため, この手法はヘビーボール法 (Heavy-Ball method) [11, 12] としても知られている. CM における (3) の更新ベクトル $\mathbf{v}_{k+1}$ を, 勾配ベクトル $\nabla E(\mathbf{w}_k)$ と 1 反復前の更新ベクトル $\mathbf{v}_k$ とモーメント係数 μ_k を用いて (27) により更新する.

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \eta_k \nabla E(\mathbf{w}_k). \tag{27}$$

ここで, $\mu_k \mathbf{v}_k$ は慣性項と呼び, $0 \leq \mu_k \leq 1$ はモーメント係数であり, 学習中は任意の値に固定される [14]. $\mu_k = 0$ の場合, CM と GD は一致する. 慣性項の $\mathbf{v}_k$ は, $\mathbf{v}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$ としても求めることができる. CM のアルゴリズムを Algorithm 2 に示す.

(9) に示す Beale 関数の最適解が求まるまでの CM による学習軌道を図 11 に示す. この問題では, 学習率 $\eta_k = 0.01$ そして $\mu_k = 0.9$ とした. 学習結果より, 最適解が求まるまで 379 回の反復回数を必要とする. CM は慣性項の影響により, GD と比較して, 探索方向の軌道修正が大きい一方で, より少ない反復回数で最適解を求めることができる.

Algorithm 2 Classical Momentum method (CM)

Require: $\epsilon, k_{max}, \eta_k, \mu_k$

Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$, $\mathbf{v}_1 = 0$

1: $k = 1$
2: **while** $(\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max})$ **do**
3: Calculate $\nabla E(\mathbf{w}_k)$;
4: Update $\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \eta_k \nabla E(\mathbf{w}_k)$;
5: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
6: $k = k + 1$;
7: **end while**

Return: $\mathbf{w}_k$

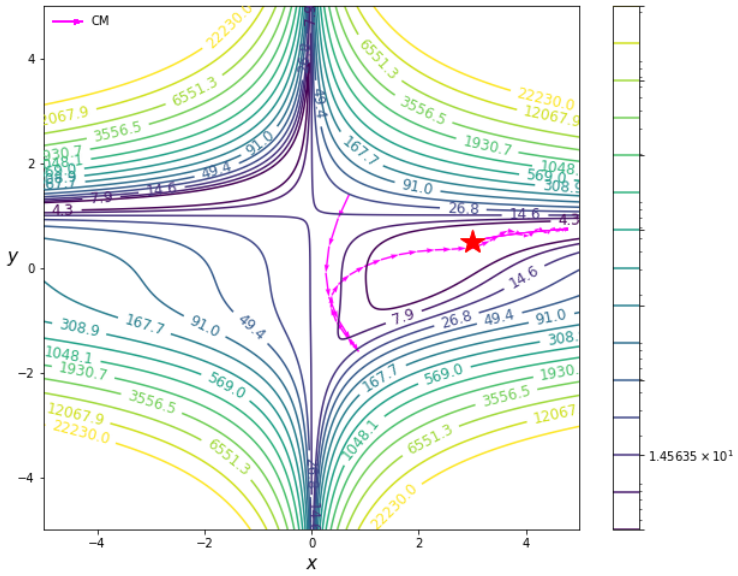


図 11: CM による Beale 関数の最適化

2.3.3　Nesterov's Accelerated Gradient method

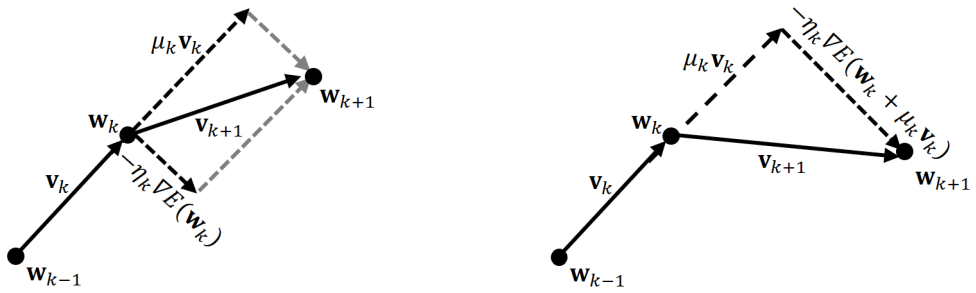
ネステロフの加速勾配法 (Nesterov's Accelerated Gradient method, NAG) は, 機械学習および最適化問題において, 注目を集めている最適化手法の一つである [1,9,13–15,48,49]. NAG は慣性項だけ移動した点, $\mathbf{w}_k + \mu_k \mathbf{v}_k$ での勾配を使用することで, CM と比較してより早い段階で誤差関数を減少させる手法であり, その収束性は保証されている [13]. CM と NAG の重みの更新ベクトル表現を図 12 に示す [34]. NAG における (3) の更新ベクトル $\mathbf{v}_{k+1}$ を (28)

に示す.

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \eta_k \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{28}$$

ここで, $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ は慣性項だけ移動した点における勾配ベクトルであり, ここでは, ネステロフの加速勾配ベクトルと呼ぶ. モーメント係数 $0 \leq \mu_k < 1$ は 1 に近い値が推奨値とされており, 一般的に, $0.8, 0.85, 0.9, 0.95$ および 0.99 に設定される [13, 14]. NAG のアルゴリズムを Algorithm 3 に示す.

(9) に示す Beale 関数の最適解が求まるまでの NAG による学習軌道を図 13 に示す. この問題では, 学習率 $\eta_k = 0.01$ そして $\mu_k = 0.9$ とした. 学習結果より, 最適解が求まるまで 236 回の反復回数を必要とする. NAG は慣性項とネステロフの加速勾配ベクトルの影響により, CM と GD と比較して, より少ない反復回数で最適解を求めることができる.



(a) CM　　(b) NAG

図 12: CM と NAG の重みの更新ベクトル表現

Algorithm 3 Nesterov's Accelerated Gradient method (NAG)

Require: $\epsilon, k_{max}, \eta_k, \mu_k$

Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$, $\mathbf{v}_1 = 0$

1: $k = 1$
2: **while** $(\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max})$ **do**
3:　Calculate $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$;
4:　Update $\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \eta_k \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$;
5:　Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
6:　$k = k + 1$;
7: **end while**

Return: $\mathbf{w}_k$

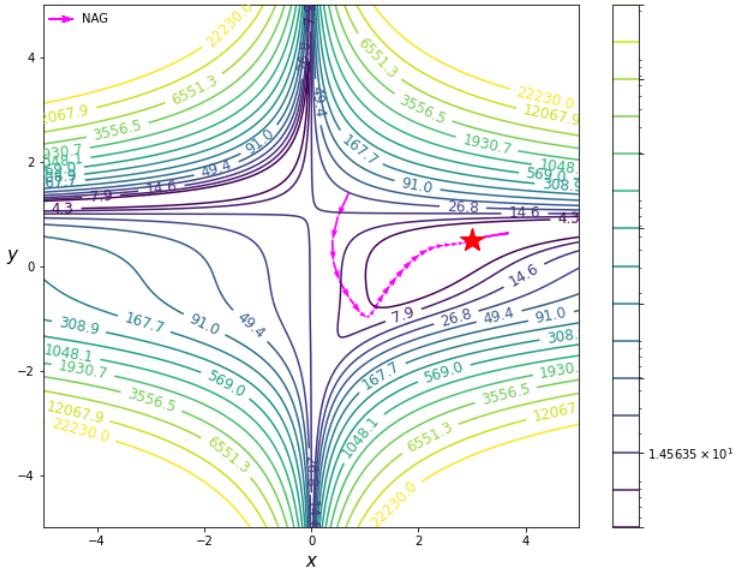


図 13: NAG による Beale 関数の最適化

2.3.4　AdaGrad

適応的劣勾配法 (Adaptive subGradient method, AdaGrad) [16] は適応的な学習率を持つ 1 次手法に基づく学習アルゴリズムの一つである. AdaGrad では過去の勾配の二乗和の平方根に比例するよう, スケーリングすることで, 重み $\mathbf{w}$ の成分ごとに学習率を適応的に調節する [1, 9, 49]. AdaGrad における (3) の更新ベクトルの各要素 $v_{k+1,i}$ を (29) に示す.

$$v_{k+1,i} = -\frac{\eta_k}{\sqrt{\sum_{l=1}^{k}(\nabla E(\mathbf{w}_l)_i)^2 + \lambda}} \nabla E(\mathbf{w}_k)_i. \tag{29}$$

ここで, $v_{k+1,i}$ および $\nabla E(\mathbf{w}_k)_i$ はそれぞれ i 番目の $\mathbf{v}_{k+1}$ と $\nabla E(\mathbf{w}_k)$ の要素である. $10^{-8} \leq \lambda \leq 10^{-6}$ であり [9, 16], ゼロによる除算を抑止する効果を持つ. 本研究では [9, 16] の推奨値 $\lambda = 10^{-8}$ とした. η_k は全ての次元で共有されたグローバルな学習率であり, その値として $\eta_k = 0.01$ が推奨値とされている [1, 9, 16]. AdaGrad の欠点として, $\mathbf{v}_k$ の成分 $v_{k,i}$ がそれぞれ狭義非減少であることが挙げられる. このため, (29) の累積和は学習率 η_k を学習の間で減らし, 収束するよりも前に, 非常に小さな値にしてしまう. 従って, バッチ学習よりもミニバッチ学習に適した最適化手法である [66]. AdaGrad のアルゴリズムを Algorithm 4 に示す.

(9) に示す Beale 関数の最適解が求まるまでの AdaGrad による学習軌道を図 14 に示す. この問題では, 実験的に学習率 $\eta_k = 0.2$ とした. 学習結果より, 最大反復回数以内に学習が収束することはできなかったが最適解に近い解を得た.

Algorithm 4 AdaGrad

Require: $\epsilon, k_{max}, \eta_k = 0.01, \lambda = 10^{-8}$
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$

$k = 1$
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
　Calculate $\nabla E(\mathbf{w}_k)$;
　for ($i = 1, 2, ..., d$) **do**
　　Calculate $G_{k,i} = \lambda + \sum_{l=1}^{k} \nabla E(\mathbf{w}_l)_i^2$;
　　Update $v_{k+1,i} = -\eta_k G_{k,i}^{-1/2} \nabla E(\mathbf{w}_k)_i$;
　　Update $w_{k+1,i} = w_{k,i} + v_{k+1,i}$;
　end for
　$k = k + 1$;
end while
Return: $\mathbf{w}_k$

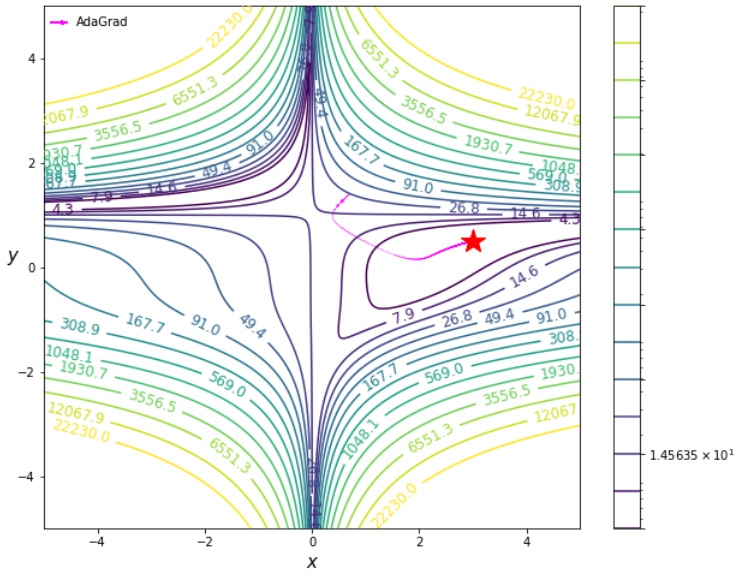


図 14: AdaGrad による Beale 関数の最適化

2.3.5　RMSprop

RMSprop [17] はミニバッチ手法を用いた Rprop [67] 学習法である. RMSprop では勾配の累計を指数関数的な重みを付けた移動平均に変更することにより, 非凸の条件下で AdaGrad の性能を改善した手法としても知られている [1,9,49]. RMSprop における (3) の更新ベクトルの各要素 $v_{k+1,i}$ を (30) に示す.

$$v_{k+1,i} = -\frac{\eta_k}{\sqrt{\theta_{k,i} + \lambda}} \nabla E(\mathbf{w}_k)_i, \tag{30}$$

$$\theta_{k,i} = \beta\theta_{k-1,i} + (1-\beta)(\nabla E(\mathbf{w}_k)_i)^2. \tag{31}$$

ここで, $v_{1,i} = 0$, $10^{-8} \leq \lambda \leq 10^{-6}$ であり [1,9,17], 本研究では [9,17] の推奨値 $\lambda = 10^{-8}$ とした. $\theta_{k,i}$ は k 反復目の i 番目の要素におけるパラメータである. ハイパーパラメータ β およびグローバルな学習率はそれぞれ $\beta = 0.9$ と $\eta_k = 0.001$ が推奨値とされている [17]. RMSprop のアルゴリズムを Algorithm 5 に示す.

(9) に示す Beale 関数の最適解が求まるまでの RMSprop による学習軌道を図 15 に示す. この問題では, 実験的に学習率 $\eta_k = 0.2$ とした. 学習結果より, 最大反復回数以内に学習が収束することはできなかったが最適解に近い解を得た.

Algorithm 5 RMSprop

Require: $\epsilon, k_{max}, \eta_k = 0.001, \lambda = 10^{-8}, \beta = 0.9$
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$

$k = 1$
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
　Calculate $\nabla E(\mathbf{w}_k)$;
　for (i = 1, 2, ..., d) **do**
　　Calculate $\theta_{k,i} = \beta\theta_{k,i} + (1-\beta)\nabla E(\mathbf{w}_k)_i^2$;
　　Update $v_{k+1,i} = -\eta_k(\theta_{k,i} + \lambda)^{-1/2}\nabla E(\mathbf{w}_k)_i$;
　　Update $w_{k+1,i} = w_{k,i} + v_{k+1,i}$;
　end for
　$k = k + 1$;
end while
Return: $\mathbf{w}_k$

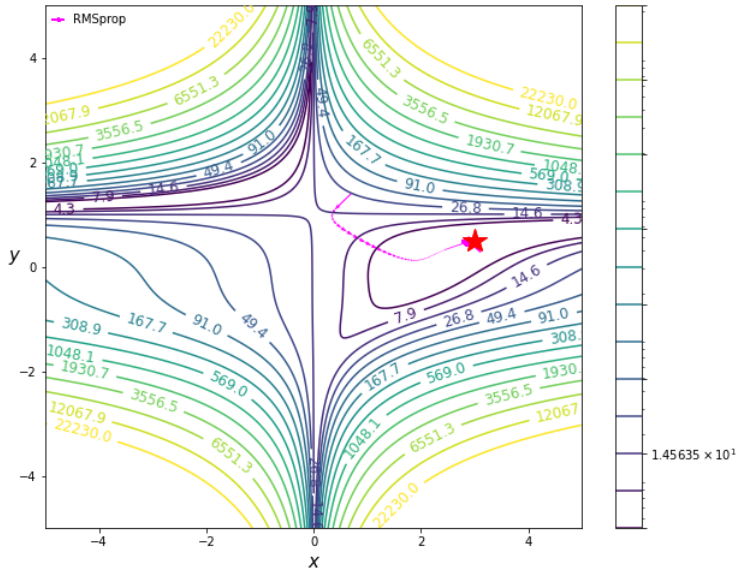


図 15: RMSprop による Beale 関数の最適化

2.3.6　AdaDelta

AdaDelta は AdaGrad の学習率に対する依存性を解消した改良手法の一つである [1,18,49]. [18] では, RMSprop の更新式を独立に得た後, 学習率に対する依存を矯正するため, 指数的に減衰する二乗和平均更新を導入した [66]. AdaDelta における (3) の更新ベクトルの各要素 $v_{k+1,i}$ を (32) に示す.

$$v_{k+1,i} = -\frac{\sqrt{r_{k,i} + \lambda}}{\sqrt{\theta_{k,i} + \lambda}} \nabla E(\mathbf{w}_k)_i, \tag{32}$$

ここで, [1, 9, 18] より λ の推奨値は $\lambda = 10^{-6}$ であり, $r_{k,i}$ と $\theta_{k,i}$ は,

$$r_{k,i} = \beta r_{k-1,i} + (1-\beta) v_{k,i}^2, \tag{33}$$

$$\theta_{k,i} = \beta \theta_{k-1,i} + (1-\beta)(\nabla E(\mathbf{w}_k)_i)^2, \tag{34}$$

である. $r_{k,i}$ と $\theta_{k,i}$ は k 反復目の i 番目の要素におけるパラメータであり, $\beta = 0.95$ が推奨値とされている [1, 9, 18]. AdaDelta のアルゴリズムを Algorithm 6 に示す.

(9) に示す Beale 関数の最適解が求まるまでの AdaDelta による学習軌道を図 16 に示す. この問題では, 実験的に $\beta = 0.6$ とした. 学習結果より, 最大反復回数以内に学習が収束することはできなかった. また, AdaGrad と RMSprop と比較して学習率が存在しないため, 探索方向も大きく変化しているものの最適解に近い解を得た.

Algorithm 6 AdaDelta

Require: $\epsilon, k_{max}, \lambda = 10^{-6}, \beta = 0.95$
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$

$k = 1$
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
　Calculate $\nabla E(\mathbf{w}_k)$;
　for ($i = 1, 2, ..., d$) **do**
　　Calculate $\theta_{k,i} = \beta\theta_{k,i} + (1-\beta)\nabla E(\mathbf{w}_k)_i^2$;
　　Calculate $r_{k,i} = \beta r_{k,i} + (1-\beta)v_{k,i}^2$;
　　Update $v_{k+1,i} = -(r_{k,i} + \lambda)^{1/2}(\theta_{k,i} + \lambda)^{-1/2}\nabla E(\mathbf{w}_k)_i$;
　　Update $w_{k+1,i} = w_{k,i} + v_{k+1,i}$;
　end for
　$k = k + 1$;
end while
Return: $\mathbf{w}_k$

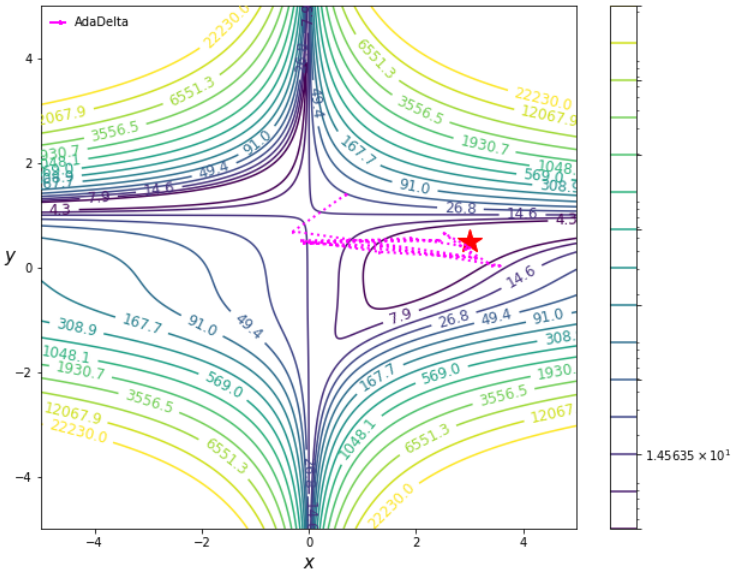


図 16: AdaDelta による Beale 関数の最適化

2.3.7　Adam

適応的モーメント推定法 (Adaptive Momentum estimation method, Adam) [19] は現在, 学習の高速化において最も代表的なアルゴリズムである. そして, この手法は 1 次手法において有効なアルゴリズムとして知られている. Adam はパラメータの成分毎に適切な学習率を用いる AdaGrad [16] と CM [14] を組み合わせた手法である [1,9,19,49]. Adam における (3) の更新ベクトルの各要素 $v_{k+1,i}$ を (35) に示す.

$$v_{k+1,i} = -\eta_k \frac{\hat{r}_{k,i}}{\sqrt{\hat{\theta}_{k,i}} + \lambda}, \tag{35}$$

ここで,

$$\hat{r}_{k,i} = \frac{r_{k,i}}{(1-\beta_1^k)}, \tag{36}$$

$$\hat{\theta}_{k,i} = \frac{\theta_{k,i}}{(1-\beta_2^k)}. \tag{37}$$

である. また $r_{k,i}$ および $\theta_{k,i}$ は (38)(39) で求まる.

$$r_{k,i} = \beta_1 r_{k-1,i} + (1-\beta_1)\nabla E(\mathbf{w}_k)_i, \tag{38}$$

$$\theta_{k,i} = \beta_2 \theta_{k-1,i} + (1-\beta_2)(\nabla E(\mathbf{w}_k)_i)^2, \tag{39}$$

ここで, λ と学習率 η_k の推奨値はそれぞれ $\lambda = 10^{-8}$ と $\eta_k = 0.001$ とされている [1,9,19]. $r_{k,i}$ および $\theta_{k,i}$ は, それぞれ勾配と二乗勾配の i 番目の要素を示す. ハイパーパラメータ β_1,β_2 は $0 \leq \beta_1, \beta_2 \leq 1$ とし, 指数移動平均および減衰率を制御する. [19] において β_1 と β_2 はそれぞれ 0.9 と 0.999 が推奨値とされている. Adam のアルゴリズムを Algorithm 7 に示す.

(9) に示す Beale 関数の最適解が求まるまでの Adam による学習軌道を図 17 に示す. この問題では, 実験的に学習率 $\eta_k = 0.1$ とした. 学習結果より, 最適解が求まるまで 409 回の反復回数を必要とする.

Algorithm 7 Adam

Require: $\epsilon, k_{max}, \eta_k = 0.001, \lambda = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$

$k = 1$
while $(\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max})$ **do**
　Calculate $\nabla E(\mathbf{w}_k)$;
　for $(i = 1, 2, ..., d)$ **do**
　　Calculate $r_{k,i} = \beta_1 r_{k,i} + (1 - \beta_1)\nabla E(\mathbf{w}_k)_i$;
　　Calculate $\theta_{k,i} = \beta_2 \theta_{k,i} + (1 - \beta_2)\nabla E(\mathbf{w}_k)_i^2$;
　　Calculate $\hat{r}_{k,i} = r_{k,i}/(1 - \beta_1^k)$ and $\hat{\theta}_{k,i} = \theta_{k,i}/(1 - \beta_2^k)$;
　　Update $v_{k+1,i} = -\eta_k \hat{r}_{k,i}(\hat{\theta}_{k,i}^{-1/2} + \lambda)^{-1}$;
　　Update $w_{k+1,i} = w_{k,i} + v_{k+1,i}$;
　end for
　$k = k + 1$;
end while
Return: $\mathbf{w}_k$

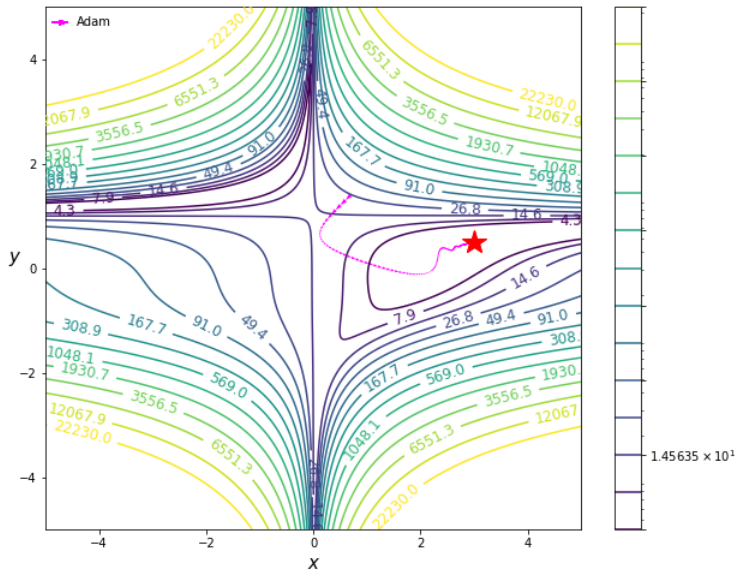


図 17: Adam による Beale 関数の最適化

2.4　2 次近似勾配学習手法

分類問題など非線形性が少ない問題に対する NN の学習では, 1 次手法は有効な最適化アルゴリズムとして近年, 様々研究で実用されている. しかし, 入出力の関係が非常に複雑な問題 (非線形性が非常に強い (強非線形) 問題) ではこれらの手法はその有効性を発揮することが困難である. 具体的には, 強非線形問題では, 一見ノイズに捉えられるデータでも, 実際にはシステムのモデルデータとして非常に重要な要素になりうる. しかし, 1 次近似手法を用いた NN では, これらの大事な要素をノイズと捉えるため, 学習の収束速度が遅くなり, ときには局所的最小解, 鞍点またはプラトー領域で停滞する傾向がある. 従って, 現実的な時間内で高精度な学習解を得ることは困難となる [1, 2, 20, 21]. 強非線形問題の一つの例として, マイクロ波回路の 1 種である microstrip Low-Pass Filter (LPF) [34, 68–71] のモデリング問題が挙げられる. NN では, 電磁波 (EM) または物理データなどの測定およびシミュレーションが行われたマイクロ波デバイスデータを用いて学習を行うことが可能である [3–6]. 学習済みの NN によって生成されたマイクロ波デバイスのモデルは, MPU の EM 物理モデルの代わりとして使用することが可能であり, これは EM/物理レベルの精度を維持しながら回路設計を大幅に高速化できる有効な手法である [3, 4]. NN によるモデリングは, デバイスレベルと回路レベルの両方において, 様々なマイクロ波回路の構成要素をモデリングするために用いられている. 例で挙げた LPF の構造を図 18 に示す. このフィルターは現代の無線システム, 特にマイクロ波および衛星通信システムにおいて欠くことのできない構成要素の一つである [72]. LPF の基板では, 誘電率そして高さはそれぞれ 9.3 F/m と 1 mm である. 幅 D は 1 mm 刻みの間隔で $12 \leq D \leq 20$ mm までの範囲のモデリングを考える. 周波数 f の範囲は $0.1 \leq f \leq 4.5$ GHz とした. 各 D に対して 221 個のサンプルが存在する. さらに, 学習の性能を測るため, D の中からいくつかのデータをテストデータとした. つまり, $D = 12, 14, 16, 18, 20$ mm を学習データ T_r とし, $D = 13, 15, 17, 19$ mm をテストデータ T_e とした. 従って, 学習データ数は $|T_r| = 1105$, テストデータ数は $|T_e| = 884$ である. この問題における学習およびテストデータは, 高周波 EM シミュレーション用の全波 3D 平面電磁界ソ

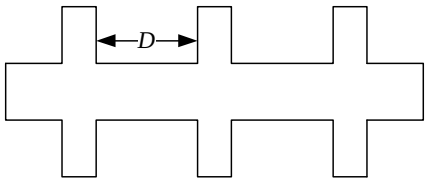


図 18: LPF の構造

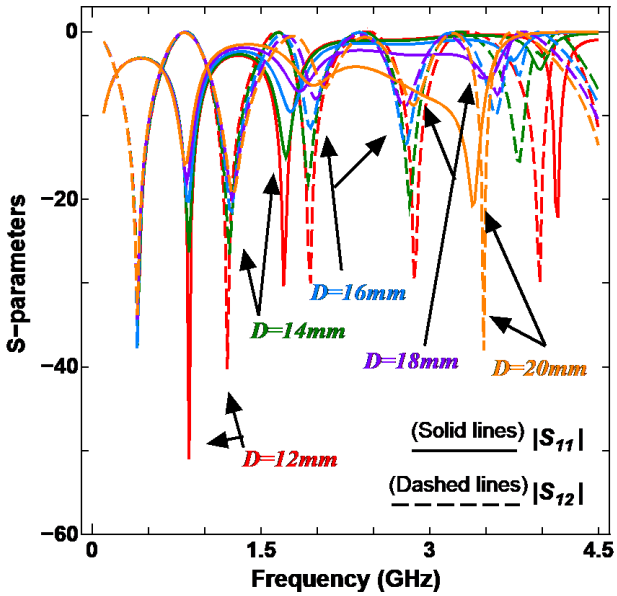


図 19: LPF の学習データ

ルバーソフトウェアである *Sonnet* [73] を用いて生成された. 学習データを図 19 に示す. 図 19 において縦軸は S パラメータの大きさであり, 横軸は周波数である. 図 19 からもわかるように, この問題は入出力の関係が非常に複雑である. さらに, 図 20 に LPF におけるデー

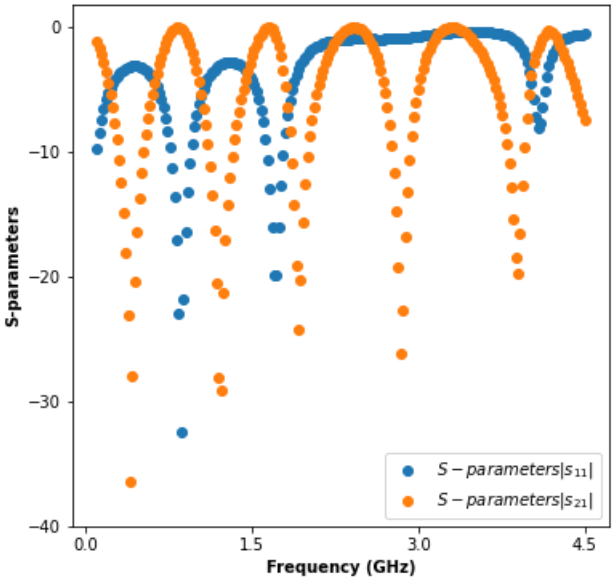


図 20: LPF におけるデータポイントの一部

タポイントの一部を示す. この図から, LPF のデータポイントには一見ノイズと捉えられる重要な要素が含まれていることがわかる. このような特性は LPF の全てのデータに存在する. 従って, LPF は強非線形モデリング問題とされている [3–6, 34]. このような強非線形問題の学習では, 2 次 (超 1 次) 手法が用いられ, 準ニュートン法 (Quasi-Newton method, QN) が有効な最適化手法として知られている. 2 次手法では, 学習率はスカラーであるステップサイズと誤差関数の曲率情報であるヘッセ行列によって決定される [20, 21]. 従って, 2 次手法では 1 反復の計算コストは 1 次手法と比較して増加するものの, 局所的最小解・鞍点またはプラトー領域を回避することが可能であるため, 収束速度は速く, 解の精度も高い. QN は, GD の高い収束性と Newton 法の速い収束速度を組み合わせた手法であり, Newton 法におけるヘッセ行列の逆行列を近似的に求める効率的なアルゴリズムである [20, 21]. 従って, 近年の最適化に対する多くの研究では QN に基づく手法が使用されている. 本節では, Newton 法 [20–22, 25, 26, 74] およびヘッセ行列を近似した, QN [20, 21] に関して紹介する. さらに, QN の応用研究として, 慣性項の導入によって QN を高速化したネステロフの加速準ニュートン法 (Nesterov's Accelerated Quasi-Newton method, NAQ) [32–34] に関しても紹介する.

2.4.1　Newton method

ニュートン法 (Newton method, Newton) は局所的に早い収束速度を期待される 2 次手法に基づくアルゴリズムである [1, 2, 20–22]. ニュートン法は (4) の誤差関数 $E(\mathbf{w})$ を $\mathbf{w}_k$ のまわりで 2 次近似することで導出される.

$$E\left(\mathbf{w}\right) \simeq \hat{E}(\mathbf{w}) = E\left(\mathbf{w}_k\right) + \nabla E\left(\mathbf{w}_k\right)^{\mathrm{T}} \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\nabla^2 E\left(\mathbf{w}_k\right)\Delta\mathbf{w}, \tag{40}$$

ここで,

$$\nabla^2 E(\mathbf{w}_k) = \begin{bmatrix} \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{1,k}^2} & \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{1,k}\partial w_{2,k}} & \cdots & \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{1,k}\partial w_{d,k}} \\ \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{2,k}\partial w_{1,k}} & \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{2,k}^2} & \cdots & \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{2,k}\partial w_{d,k}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{d,k}\partial w_{1,k}} & \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{d,k}\partial w_{2,k}} & \cdots & \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_{d,k}^2} \end{bmatrix}, \tag{41}$$

は, 誤差関数 (4) のヘッセ行列を示し, $\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_k$ である. よって, $\Delta\mathbf{w}$ に対する (40) の最適解は, $\partial\hat{E}(\mathbf{w})/\partial\mathbf{w} = 0$ とおくと, (42) となる.

$$\Delta\mathbf{w} = -\left(\nabla^2 E(\mathbf{w}_k)\right)^{-1} \nabla E(\mathbf{w}_k), \tag{42}$$

ここで, $\mathbf{w}$ の更新量 $\Delta\mathbf{w}$ を $\mathbf{w}_k$ から $\mathbf{w}_{k+1}$ への移動方向と考えると, $\Delta\mathbf{w} = \mathbf{w}_{k+1} - \mathbf{w}_k$ と考えられる. 従って, ニュートン法の更新式が (43) として得られる.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \left(\nabla^2 E(\mathbf{w}_k)\right)^{-1} \nabla E(\mathbf{w}_k). \tag{43}$$

さらに, ニュートン法における (3) の更新ベクトル $\mathbf{v}_{k+1}$ をスカラー値であるステップサイズ α_k を用いて (44) として示すことができる [74].

$$\mathbf{v}_{k+1} = -\alpha_k \left(\nabla^2 E(\mathbf{w}_k)\right)^{-1} \nabla E(\mathbf{w}_k). \tag{44}$$

ここで, $\boldsymbol{\eta}_k = \alpha_k \left(\nabla^2 E(\mathbf{w}_k)\right)^{-1}$ であり, ステップサイズ α_k は直線探索法 (Line Search method) である, Armijo 条件 (Armijo's Condition) または Wolf 条件 (Wolf's Condition) によって求まる [1, 2, 20–22, 26]. この手法は 1 次手法と比較して, 解の近くでは早い収束速度を持っていた. しかし, ヘッセ行列が正定値行列である保証がなく, 初期値によっては反復が不安定となる. また, ヘッセ行列を求める必要があるため, 計算コストが問題によっては莫大となる欠点があった [1, 2, 20–22].

2.4.2　Quasi-Newton method

準ニュートン法 (Quasi-Newton method, QN) は, GD における大域的収束性とニュートン法における局所的に早い収束性を併せ持ち, 非凸最適化問題に対する有効な手法として提案された [1, 2, 20, 21]. QN では, Newton 法におけるヘッセ行列を正定値対称行列 (Positive Definite Symmetric Matrix) である $\mathbf{H}_k^{\mathrm{QN}}$ で近似することにより収束性を保証すると共にヘッセ行列を求めるための 2 階微分を省いた, 超 1 次収束を持つアルゴリズムである [20, 21]. QN における学習率 $\boldsymbol{\eta}_k$ は, スカラー値のステップサイズ α_k および $\mathbf{H}_k^{\mathrm{QN}}$ を用いて,

$$\boldsymbol{\eta}_k = \alpha_k \mathbf{H}_k^{\mathrm{QN}}, \tag{45}$$

として定義される. QN の更新ベクトル $\mathbf{v}_{k+1}$ を (46) に示す.

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k^{\mathrm{QN}} \nabla E\left(\mathbf{w}_k\right), \tag{46}$$

ここで, $\mathbf{H}_k^{\mathrm{QN}}$ は, (47) に示すように $E(\mathbf{w})$ のヘッセ行列の逆近似行列を表す.

$$\left(\nabla^2 E(\mathbf{w}_k)\right)^{-1} \approx (\mathbf{B}_k^{\mathrm{QN}})^{-1} = \mathbf{H}_k^{\mathrm{QN}}, \tag{47}$$

QN は, $\mathbf{B}_k^{\mathrm{QN}}$ を (40) の $\nabla^2 E(\mathbf{w}_k)$ に代入することで導出される.

$$E(\mathbf{w}) \simeq \hat{E}(\mathbf{w}) = E(\mathbf{w}_k) + \nabla E(\mathbf{w}_k)^{\mathrm{T}} \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{B}_k^{\mathrm{QN}}\Delta\mathbf{w}, \tag{48}$$

ここで, (48) の最適解は $\partial\hat{E}(\mathbf{w})/\partial\mathbf{w} = 0$ とおくと, (49) となる.

$$\mathbf{B}_k^{\mathrm{QN}}\Delta\mathbf{w} = -\nabla E(\mathbf{w}_k). \tag{49}$$

(49) では $\mathbf{w}$ の更新量 $\Delta\mathbf{w}$ を $\mathbf{w}_k$ から $\mathbf{w}_{k+1}$ への移動方向と考えると, $\Delta\mathbf{w} = \mathbf{w}_{k+1} - \mathbf{w}_k$ と考えられる. QN では (49) の連立 1 次方程式の解として $\Delta\mathbf{w}$ を求める. これにより, ヘッセ行列の計算が不要となる. QN では, ヘッセ行列の情報を近似行列に取り込むため, (50) のセカント条件 (Secant Condition) が満たされるように, 正定値対称性行列である $(\mathbf{B}_k^{\mathrm{QN}})^{-1} = \mathbf{H}_k^{\mathrm{QN}}$ を更新する必要がある [20–22, 26].

$$\mathbf{y}_k = \mathbf{B}_{k+1}^{\mathrm{QN}}\mathbf{s}_k, \quad \text{または} \quad \mathbf{s}_k = \mathbf{H}_{k+1}^{\mathrm{QN}}\mathbf{y}_k. \tag{50}$$

ここで, $\mathbf{s}_k$ と $\mathbf{y}_k$ をそれぞれ (51) と (52) に示す.

$$\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k, \tag{51}$$

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k), \tag{52}$$

(50) のセカント条件を導出するため, $\nabla E(\mathbf{w}_k)$ を $\mathbf{w}_{k+1}$ の回りでテイラー展開すると, (53) が得られる.

$$\nabla E(\mathbf{w}_k) \simeq \nabla E(\mathbf{w}_{k+1}) + \nabla^2 E(\mathbf{w}_{k+1})(\mathbf{w}_k - \mathbf{w}_{k+1}) + ..., \tag{53}$$

ここで, (53) を 2 次の項までだけとし, $\nabla^2 E(\mathbf{w}_{k+1}) = \mathbf{B}_{k+1}^{\mathrm{QN}}$ とすると (54) が成立する.

$$\mathbf{B}_k^{\mathrm{QN}}(\mathbf{w}_{k+1} - \mathbf{w}_k) \simeq \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k). \tag{54}$$

ここで, 右辺と左辺の差分の項にそれぞれ $\mathbf{s}_k$ と $\mathbf{y}_k$ を代入すると, (50) のセカント条件が得られる. これまでに, $\mathbf{H}_k^{\mathrm{QN}}$ を更新する手法として, ランク 1 更新式に基づく SR1(Symmetric Rank-1) 公式とランク 2 更新式に基づく DFP(Davidon-Fletcher-Powell), BFGS(Broyden-Fletcher-Glodfarb-Shanno) 公式, そして Broyden 公式族 (Broyden Family) が提案されている [1, 2, 20, 21]. 近年, BFGS 公式がもっとも効果的であると考えられているため, 多くの QN を用いた

最適化では BFGS が使用されている [21]. BFGS 公式を (55) と (56) に示す.

$$\mathbf{B}_{k+1}^{\mathrm{QN}} = \mathbf{B}_{k}^{\mathrm{QN}} - \frac{\mathbf{B}_{k}^{\mathrm{QN}}\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_{k}^{\mathrm{QN}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_{k}^{\mathrm{QN}}\mathbf{s}_k} + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}}\mathbf{s}_k}. \tag{55}$$

$$\begin{aligned}\mathbf{H}_{k+1}^{\mathrm{QN}} &= \mathbf{H}_{k}^{\mathrm{QN}} - \frac{(\mathbf{H}_{k}^{\mathrm{QN}}\mathbf{y}_k)\mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k(\mathbf{H}_{k}^{\mathrm{QN}}\mathbf{y}_k)^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k} + \left(1 + \frac{\mathbf{y}_k^{\mathrm{T}}\mathbf{H}_{k}^{\mathrm{QN}}\mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)\left(\frac{\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right) \\ &= \left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{\mathrm{T}} \mathbf{H}_{k}^{\mathrm{QN}}\left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right) + \left(\frac{\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right).\end{aligned} \tag{56}$$

(55) の $\mathbf{B}_{k+1}^{\mathrm{QN}}$ 行列から (56) の $\mathbf{H}_{k+1}^{\mathrm{QN}}$ 行列への導出に関しては付録 A に示す. さらに, QN のアルゴリズムを Algorithm 8 に示す.

Algorithm 8 Quasi-Newton method (QN) -BFGS-

Require: ϵ, k_{max}
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$, $\mathbf{H}_1^{\mathrm{QN}} = \mathbf{I}$(unit matrix)
1: $k = 1$
2: Calculate $\nabla E(\mathbf{w}_1)$;
3: **while** ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
4:　Calculate stepsize α_k;
5:　Update $\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k^{\mathrm{QN}} \nabla E(\mathbf{w}_k)$;
6:　Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
7:　Calculate $\nabla E(\mathbf{w}_{k+1})$;
8:　Calculate $\mathbf{s}_k$ and $\mathbf{y}_k$ using (51) and (52);
9:　Update $\mathbf{H}_{k+1}^{\mathrm{QN}}$ using (56);
10:　$k = k + 1$;
11: **end while**
Return: $\mathbf{w}_k$

2.4.3　Nesterov's Accelerated Quasi-Newton method

慣性項付 2 次近似勾配モデルを用いた準ニュートン法, 別名ネステロフの加速準ニュートン法 (Nesterov's Accelerated Quasi-Newton method, NAQ) は, QN が誤差関数 $E(\mathbf{w})$ を $\mathbf{w}_k$ のまわりで 2 次近似するに対して, (57) に示すように, $E(\mathbf{w})$ を $\mathbf{w}_k + \mu_k\mathbf{v}_k$ のまわりで 2 次近似

する [32–34].

$$
\begin{aligned}
E\left(\mathbf{w}\right) \simeq \hat{E}(\mathbf{w}) = E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right) + \nabla E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right)^{\mathrm{T}} \Delta \mathbf{w} \\
+ \frac{1}{2} \Delta \mathbf{w}^{\mathrm{T}} \nabla^2 E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right) \Delta \mathbf{w}.
\end{aligned}
\tag{57}
$$

(57) では, $\Delta \mathbf{w} = \mathbf{w} - (\mathbf{w}_k + \mu_k \mathbf{v}_k)$ である. ここで, $\Delta \mathbf{w}$ に対する最適化の条件, $\partial \hat{E}(\mathbf{w}) / \partial \mathbf{w} = 0$ より, (58) を得る.

$$
\Delta \mathbf{w} = -\left(\nabla^2 E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right)\right)^{-1} \nabla E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right). \tag{58}
$$

ここで, $\mathbf{w}$ の更新量 $\Delta \mathbf{w}$ を $\mathbf{w}_k + \mu_k \mathbf{v}_k$ から $\mathbf{w}_{k+1}$ への移動方向と考えると, $\Delta \mathbf{w} = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k)$ と考えられる. 従って, NAQ の更新式を (59) の通りに示すことができる.

$$
\mathbf{w}_{k+1} = \left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right) - \left(\nabla^2 E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right)\right)^{-1} \nabla E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right). \tag{59}
$$

(59) は慣性項 $\mu_k \mathbf{v}_k$ を持つニュートン法と見なされる. ここで, ヘッセ行列の逆行列 $\nabla^2 E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ は, $\mathbf{H}_{k+1}^{\mathrm{NAQ}}$ 行列として近似され, その反復式は,

$$
\begin{aligned}
\mathbf{H}_{k+1}^{\mathrm{NAQ}} &= \mathbf{H}_k^{\mathrm{NAQ}} - \frac{(\mathbf{H}_k^{\mathrm{NAQ}} \mathbf{q}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k (\mathbf{H}_k^{\mathrm{NAQ}} \mathbf{q}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} + \left(1 + \frac{\mathbf{q}_k^{\mathrm{T}} \mathbf{H}_k^{\mathrm{NAQ}} \mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right)\left(\frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right) \\
&= \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{NAQ}} \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right) + \left(\frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right),
\end{aligned}
\tag{60}
$$

である. ここで, $\mathbf{p}_k$ と $\mathbf{q}_k$ はそれぞれ (61) と (62) より求められる.

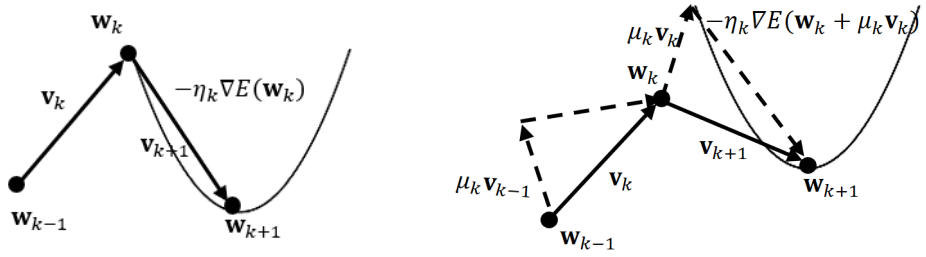
$$
\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k), \tag{61}
$$

$$
\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{62}
$$

(60) は, NAQ のセカント条件 $\mathbf{q}_k = (\mathbf{H}_{k+1}^{\mathrm{NAQ}})^{-1} \mathbf{p}_k$ およびランク 2 更新式から導出された [34]. さらに, (60) の更新行列 $\mathbf{H}_{k+1}^{\mathrm{NAQ}}$ は, $\mathbf{H}_k^{\mathrm{NAQ}}$ が正定値対称行列であれば, 正定値対称特性を保持することが証明されている [34]. NAQ における (3) の更新ベクトル $\mathbf{v}_{k+1}$ を (63) に示す.

$$
\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k^{\mathrm{NAQ}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{63}
$$

ここで, $0 < \mu_k < 1$ であり, 通常は 1 に近い値に設定される [14, 32–34]. また, $\mu_k = 0$ の場合, NAQ と QN は一致する. NAQ は, 慣性項 $\mu_k \mathbf{v}_k$ およびネステロフの加速勾配と呼ばれる $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ を用いて QN の収束速度を大幅に高速化した手法である. これは, 慣性項が



(a) QN　　(b) NAQ

図 21: QN と NAQ の重みの更新ベクトル表現

QN の加速において有効であることを示している [32–34]. QN と NAQ の重みの更新ベクトル表現を図 21 に示す [34].

NAQ のアルゴリズムを Algorithm 9 に示す. このアルゴリズムより, 学習ループにおいて $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ と $\nabla E(\mathbf{w}_{k+1})$ の 2 つの勾配計算が必要であることが確認できる. これらの勾配はそれぞれ Step.3 と 7 に示されている. 従って, NAQ は 1 反復において 2 回勾配を計算するため, QN と比較して 1 反復の計算時間が増加する. これは NAQ の欠点であると考えられる. しかし, NAQ は学習全体の反復回数が QN と比較して大幅に削減されているため, 欠点の影響があまり注目されていない手法である [32–34].

Algorithm 9 Nesterov's Accelerated Quasi-Newton method (NAQ)

Require: ϵ, k_{max}, μ_k

Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$, $\mathbf{H}_1^{\mathrm{NAQ}} = \mathbf{I}$(unit matrix), $\mathbf{v}_1 = 0$

1: $k = 1$
2: **while** ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
3: Calculate $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$;
4: Calculate stepsize α_k;
5: Update $\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k^{\mathrm{NAQ}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$;
6: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
7: Calculate $\nabla E(\mathbf{w}_{k+1})$;
8: Calculate $\mathbf{p}_k$ and $\mathbf{q}_k$ using (61) and (62);
9: Update $\mathbf{H}_{k+1}^{\mathrm{NAQ}}$ using (60);
10: $k = k + 1$;
11: **end while**

Return: $\mathbf{w}_k$

2.5　まとめ

本章では, 最適化問題と NN の概念, そして NN における勾配の導出に関して説明を行った. さらに, NN において最適化手法として用いられる 7 種類の 1 次手法, GD [1, 48, 49], CM [1,11,12,48,49], NAG [1,13,14,48,49], AdaGrad [1,9,16,48,49], RMSprop [1,9,17,48,49], AdaDelta [1,9,18,48,49] そして Adam [1,9,19,48,49] と 3 種類の 2 次 (超 1 次) 手法, Newton [20,21], QN [20,21] と NAQ [32–34] を紹介した. 1 次と 2 次手法のそれぞれのアルゴリズムの関連性を簡素にまとめた図を図 22 と 23 に示す.

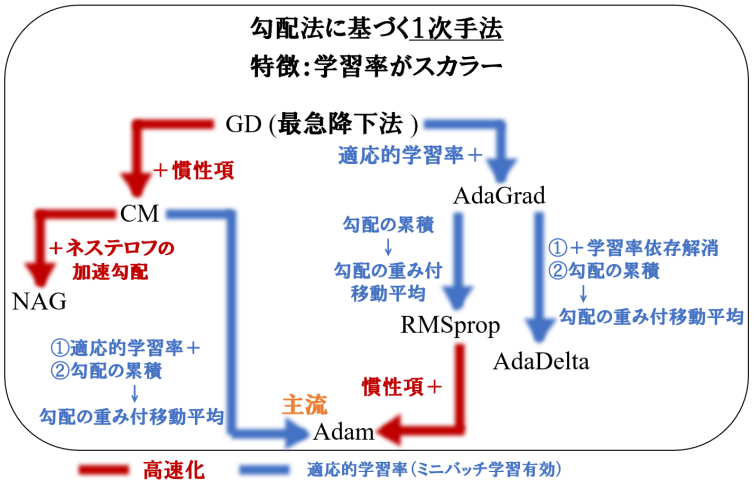


図 22: 1 次手法のアルゴリズムの関係

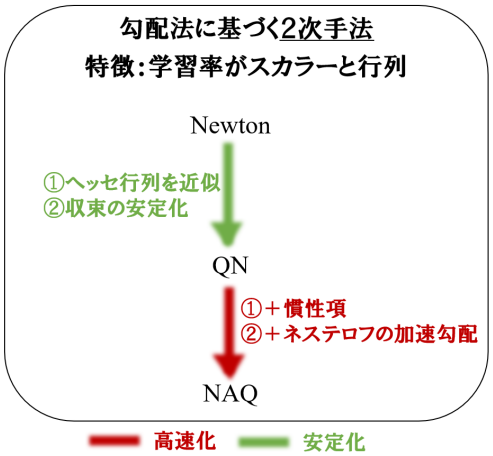


図 23: 2 次手法のアルゴリズムの関係

3　適応的ネステロフの加速準ニュートン法

本章では, 本研究の 1 つ目の課題であった, NAQ における慣性項のハイパーパラメータ問題に着目し, 解決手法として適応的ネステロフの加速準ニュートン法 (Adaptive Nesterov's Accelerated Quasi-Newton method, AdaNAQ) を提案する [35, 36]. 提案手法は, 学習初期の段階では, 慣性項を小さな値に設定し, 学習が安定した後, 慣性項を増加させることで学習を高速化させることを可能にする. 慣性項を適応的に変化させることにより, 学習が安定するため, 複数回の試行を行っても同精度の学習結果が得られる. 従って, AdaNAQ は従来手法の NAQ の欠点であったハイパーパラメータ問題および複数回の試行における重みの初期値に対する学習精度の依存を解消できる. 提案手法を非線形性の高い 2 つの関数近似問題と 2 つのマイクロ波回路モデリング問題に対する NN の学習に応用し, 従来手法と比較を行った上, その有効性を計算機実験により示す.

3.1　適応的慣性係数

最適化問題において, 一般的に慣性項のモーメント係数 μ_k の範囲は, $0 < \mu_k < 1$ である. NN の学習では, μ_k の非凸型性質のため, 通常では全ての反復において固定されるハイパーパラメータとして用いられる [13, 14, 32–34]. μ_k が 0 に近い場合, 慣性項の効果が制限されるため, 学習の収束速度が遅くなる. 一方, μ_k が 1 に近づくと, 学習速度は速くなる傾向があるが, 収束は不安定化する. 従って, ハイパーパラメータ μ_k は 1 に近い値を実験に基づいて設定することで, 高速な収束を得る. 近年, モーメント係数 μ_k を適応的に導出する 1 次手法が, 強凸性質を持つ最適化手法の分野で注目を集めている [13, 75]. 適応的慣性 (モーメント) 係数 μ_k は, 凸最適化問題に対する 1 次手法に基づく最適化手法として [13] で提案された. 提案された適応的慣性係数 μ_k(適応的 μ_k) を (64) と (65) に示す.

$$\mu_k = \frac{\theta_k(1-\theta_k)}{\theta_k^2 + \theta_{k+1}}, \tag{64}$$

$$\theta_{k+1}^2 = (1-\theta_{k+1})\theta_k^2 + \gamma\theta_{k+1}. \tag{65}$$

ここで, $\theta_1 = 1$ であり, $\gamma = \iota/L_p$ では L_p はリプシッツ (Lipschitz) 定数であり, ι は強い凸性 (Strong Convex Property) である [13]. γ の最適値を求めることが困難であるため, 通常, γ は十分に小さな値に設定される [76]. 本研究では, 従来の研究 [35, 36] に基づき $\gamma = 1.0 \times 10^{-5}$ と設定する. この最適化手法は, 強凸問題と $O(1/k^2)$ の収束率 [13] に対して, 従来の 1 次手

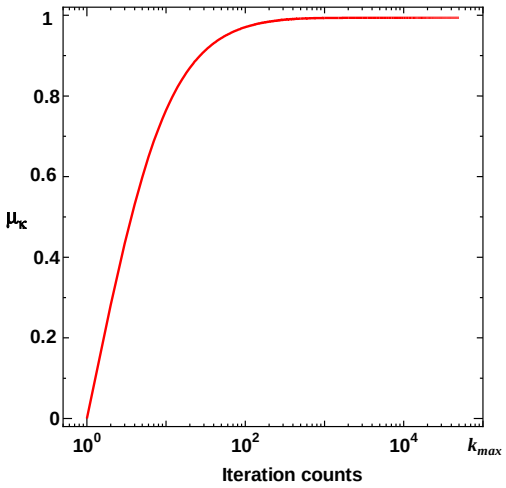


図 24: 反復回数に対する μ_k の変化

法よりも厳密な反復回数の下限をもたらした. 反復回数 k に対する μ_k の変化グラフを図 24 に示す. 図より, 反復回数の増加に伴って μ_k も徐々に増加することがわかる.

NAQ は, 学習全体の計算時間を QN と比較して, 大幅に高速化させることに成功した. 一方で, ハイパーパラメータであるモーメント係数 μ_k は反復全体で固定されていた. このため, 高速かつ安定した解を得るため, 慎重に選択する必要があった [32–34]. 具体的には, μ_k が 0 に近い値の場合, 解は小さな誤差を得るが, 収束に必要な反復回数は増加し, QN と同様または近い結果を得る. 一方で, μ_k が 1 に近い値に設定されていると, 学習時間は高速化するが, 学習の安定性は低下し, 誤差の減少が重み $\mathbf{w}$ の初期値に強く依存するようになる. 従って, 誤差は常に十分小さな値まで減少することができなくなる. よって, 学習の安定性に対しては, μ_k の値が小さな値に設定されるとき, ロバスト (Robust, 別名: 堅牢) 性が高く, 安定している. しかし, NAQ の高速な性質を維持するために, μ_k を増加させる必要がある. 従って, μ_k の値は多くの経験に基づいて設定する必要性があった [13, 14, 32–34]. 本研究では, この問題を解決するため, (64) と (65) の適応的 μ_k [13] を NAQ に導入した, 適応的ネステロフの加速準ニュートン法 (Adaptive Nesterov's Accelerated Quasi-Newton method, AdaNAQ) を提案する. 適応的モーメント係数 μ_k は, 学習中に, 徐々にその値を 0 から 1 に変化させることにより NAQ の高速な収束速度を維持できた. これにより学習の安定性を維持しつつ, 重み $\mathbf{w}$ の初期値に対するロバストな学習を可能とすることが期待される. AdaNAQ では前述の 2 つの要素が保たれるために, 適応的 μ_k の更新が非常に重要となる. もし, 学習初期の反復から適応的 μ_k が更新を行えば, 適応的 μ_k の値が早い段階で 1 に近づくため, 小さな誤差

を得ることが困難になる. 本研究では, この問題を克服するため, 実験に基づき, 初期の反復において, $\mu_1 = 0$ と設定し, 反復の増加につれて $\|\nabla E(\mathbf{w}_k)\| < 10^{-5}$ の条件を満たしたときより更新を行うように設定した. これは, NN の学習における最適解の周囲では勾配停滞, つまり, プラトーが発生する事実に基づいた理由を考慮し, 設定している [77]. 従って, 学習の初期の反復では慣性項による勾配のオーバーシュート (Overshoot) を避けるため, QN による更新を行う. その後, 条件が満たされたときより, 適応的 μ_k の更新が開始され, 慣性項による加速が始まる. これにより, NAQ の高速な収束性能を維持しつつ, 重み $\mathbf{w}$ の初期値に対するロバストな学習が行えると期待できる.

3.2　実験

提案手法 AdaNAQ の有効性を示すため, 2 つの関数近似問題と 2 つのマイクロ波回路モデリング問題, 合計 4 つのベンチマーク問題に対してシミュレーションを行った. 本研究で使用する強非線形問題の学習では, 先行研究 [5, 6, 32–34, 68, 69, 78, 79] に基づき, (4) の誤差関数として (15) の MSE を使用し, 各ニューロンは (16) のシグモイド活性化関数を持つ. シミュレーションでは, 任意のニューロン数を持つ中間層を含めた階層型 NN 用いた. 本研究で取り扱う強非線形問題の最適化では, 最適化アルゴリズムと同様にネットワーク構造も非常に重要となる. 一般的に, 複雑な問題に対する学習では, より大きなネットワークを使用することで, 高精度な学習解を得る. しかし, 大きなネットワークは計算コストの増加に繋がる. 従って, シミュレータによるモデリングコストの削減を理由に NN が応用されているマイクロ波回路のモデリング問題に対しては適していないと考えられる. 本研究では, マイクロ波回路のモデリングのように現実世界で応用されている問題に着目し, 小さなネットワークでも高精度な学習結果が得られるアルゴリズムの提案を試みる. 従って, ネットワークの構造は許容範囲の解が得られる最小規模のネットワークを実験的に決定した. シミュレーションでは, AdaNAQ の性能を GD, CM, NAG, optNAG [13](適応的慣性係数を用いた NAG), AdaGrad, AdaDelta, RMSprop, Adam, QN そして NAQ といった従来のアルゴリズムと比較し, 検討を行った. また, 重み $\mathbf{w}$ を $[-0.5, 0.5]$ の範囲の一様乱数で初期化し, 20 回の独立した実行を行った. AdaGrad, AdaDelta, RMSprop, Adam といった手法は, 主に確率的 (ミニバッチ) 手法として知られ, 応用されている. しかし, 本研究で対象とする問題では, バッチ手法が必要とされる [5]. 従って, 全てのアルゴリズムに対してバッチ手法を適用する [79]. AdaGrad, AdaDelta, RMSprop, Adam の各ハイパーパラメータはそれぞれ [16–19] で与えられた推奨値に設定した. CM, NAG および NAQ で使用する固定のモーメント係数 μ_k

は, 0.8, 0.85, 0.9, 0.95 そして 0.99 に設定した. 学習結果の表では, 誤差 $E(\mathbf{w})(\times 10^{-3})$ の中央 (Median), 平均 (Average(Ave.)), 最小 (Best), 最大 (Worst) 値に加え平均の計算時間 (*sec*) と収束までに必要な平均の反復回数 (k) を示す. 表では, 学習データ T_r に対する誤差は $E_{train}(\mathbf{w})$ として表記する. 1 つ目の関数近似問題およびマイクロ波回路モデリングの問題では, 学習データセット T_r から独立したテストデータセット T_e を用いて, NN の精度を評価し, 表にて $E_{test}(\mathbf{w})$ として示す. データセット T_r と T_e は $[-1.0, 1.0]$ の範囲で正規化されている. 全ての問題において, 学習の終了条件は $\epsilon = 1.0 \times 10^{-8}$ であり, 最大反復回数は $k_{\max} = 150,000$ とされている. 本実験では, GD, CM, NAG, optNAG, QN, NAQ および AdaNAQ のステップサイズ α_k(1 次手法では $\alpha_k = \eta_k$) を直線探索および Armijo 条件に基づいて更新する [21,34].

GD, CM および QN における Armijo の条件式:

$$E(\mathbf{w}_k + \alpha_k \mathbf{g}_k) \leq E(\mathbf{w}_k) + \chi \alpha_k \nabla E(\mathbf{w}_k)^{\mathrm{T}} \mathbf{g}_k. \tag{66}$$

NAG, NAQ と AdaNAQ における Armijo の条件式:

$$E(\mathbf{w}_k + \mu_k \mathbf{v}_k + \alpha_k \mathbf{g}_k) \leq E(\mathbf{w}_k + \mu_k \mathbf{v}_k) + \chi \alpha_k \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)^{\mathrm{T}} \mathbf{g}_k. \tag{67}$$

ここで, $0 < \chi < 1$ であり, 本研究では $\chi = 10^{-3}$ とした. $\mathbf{g}_k$ は探索方向ベクトルであり, GD, CM, NAG, QN そして NAQ と AdaNAQ(以降, (Ada)NAQ と表記する) ではそれぞれ $\nabla E(\mathbf{w}_k)$, $\nabla E(\mathbf{w}_k)$, $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$, $\mathbf{H}_k^{\mathrm{QN}} \nabla E(\mathbf{w}_k)$ そして $\mathbf{H}_k^{(\mathrm{Ada})\mathrm{NAQ}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ である. α_k の更新アルゴリズムを Algorithm 10 に示す.

Algorithm 10 Calculation of stepsize

Require: $\alpha_k = 1, i = 0, ls = 10$

```
for (i < ls) do
    if Armijo's condition (66) or (67) is satisfied then
        break;
    else
        Update α_k = (1/2)α_k;
    end if
    i = i + 1;
end for
```

Return: α_k

3.2.1　関数近似問題

提案手法の有効性を示す, 2 つの関数近似問題として, (68) に示す非線形関数近似問題を使用する [80].

$$f(x, a, b) = 1 + (x + 2x^2)\sin(ax^2 + b). \tag{68}$$

ここで, x と a の範囲はそれぞれ $x \in [-4, 4]$ と $a \in [-1, 1]$ である. この関数では, b の範囲を変えることにより, f_1 と f_2 の 2 種類の問題が考えられる. 最初に f_1 の b を $b = 0$ に設定する. これにより, f_1 は 2 つの入力は x と a を持ち, $|T_r|$ には 3,320 個の学習点, $|T_e|$ には 6,600 個のテストデータが含まれることとなる. 従って, (68) の出力が $f(x, a)$ であることを踏まえて, NN の構造は 55 個のニューロン数を持つ中間層 1 層を含む, 2-55-1(入力-中間ニューロン数-出力) とした. 中間ニューロン数は表 1 に示す実験結果より決定した. 表 1 では異なる中間層ニューロン数を持つネットワークに対して, QN, NAQ(μ_k = 0.85 と 0.9) および

表 1: f_1 問題に対する中間層ニューロン数の比較結果

Hidden Neurons	Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst
35	QN	-	5.05 / 5.10 / 1.99 / 8.66	4.82 / 4.87 / 1.79 / 8.37
	NAQ	0.8	5.37 / 5.45 / 1.54 / 9.89	5.15 / 5.22 / 1.43 / 9.60
		0.85	5.03 / 5.27 /1.67 / 13.83	4.70 / 5.06 / 1.59 / 13.40
		0.9	5.23 / 5.30 / 1.84 / 11.83	5.04 / 5.07 / 1.71 / 11.40
	AdaNAQ	-	4.71 / 4.90 / 1.53 / 8.33	4.47 / 4.70 / 1.45 / 8.03
45	QN	-	1.19 / 1.44 / 0.56 / 4.88	1.04 / 1.35 / 0.53 / 4.69
	NAQ	0.8	1.03 / 1.36 / 0.46 / 4.12	0.94 / 1.29 / 0.43 / 4.00
		0.85	1.18 / 1.61 / 0.70 / 4.28	1.10 / 1.53 / 0.66 / 4.02
		0.9	1.31 / 3.35 / 0.58 / 42.33	1.22 / 3.23 / 0.54 / 41.40
	AdaNAQ	-	0.93 / 1.51 / 0.60 / 4.42	0.84 / 1.44 / 0.57 / 4.28
55	QN	-	**0.58** / **0.67** / 0.32 / 2.59	**0.52** / **0.63** / 0.31 / 2.46
	NAQ	0.8	**0.55** / **0.58** / 0.36 / 1.01	**0.50** / **0.54** / 0.35 / 0.90
		0.85	**0.48** / **0.50** / 0.30 / 0.81	**0.44** / **0.47** / 0.28 / 0.73
		0.9	**0.43** / **0.52** / 0.28 / 1.22	**0.40** / **0.49** / 0.26 / 1.10
	AdaNAQ	-	**0.38** / **0.40** / 0.25 / 0.63	**0.36** / **0.38** / 0.24 / 0.60
65	QN	-	0.35 / 0.35 / 0.22 / 0.58	0.33 / 0.33 / 0.21 / 0.54
	NAQ	0.8	0.31 / 0.29 / 0.15 / 0.40	0.29 / 0.27 / 0.14 /0.37
		0.85	0.30 / 0.31 / 0.15 / 0.70	0.29 / 0.30 / 0.14 / 0.71
		0.9	0.22 / 3.35 / 0.82 / 65.63	0.21 / 124.0 / 0.17 / 2591.0
	AdaNAQ	-	0.24 / 0.25 / 0.13 / 0.45	0.23 / 0.38 / 0.13 / 0.42
75	QN	-	0.26 / 0.27 / 0.14 / 0.46	0.24 / 0.54 / 0.13 / 0.43
	NAQ	0.8	0.21 / 0.21 / 0.14 / 0.33	0.20 / 0.20 / 0.13 / 0.30
		0.85	0.19 / 0.20 / 0.12 / 0.36	0.18 / 0.18 / 0.12 / 0.34
		0.9	0.15 / 0.16 / 0.10 / 0.29	0.14 / 0.15 / 0.09 / 0.27
	AdaNAQ	-	0.13 / 0.14 / 0.09 / 0.32	0.12 / 0.14 / 0.08 / 0.32

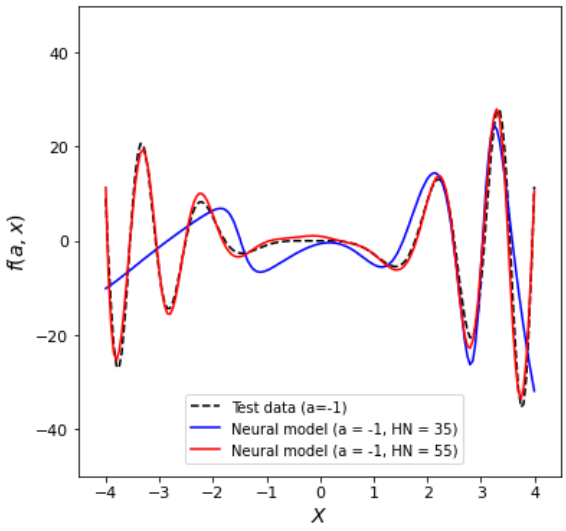


図 25: 関数 (68) に対する HN = 35 と 55 の NN モデルとテストモデルの比較

AdaNAQ の性能を示す. 本研究では, 可能な限り少ないニューロン数で学習を行うことにより, 強非線形関数を高速にモデル化し, より高い汎化能力を持つ NN を実現させることを目指す. 表 1 より, 中間ニューロン数が 55 個のとき, モデル化に適した十分小さな誤差 [78], $E_{train\&test}(\mathbf{w}) < 10^{-3}$ を平均と中央値の誤差で得られていることがわかる. 図 25 にテストデータと AdaNAQ によって学習された中間ニューロン数 35 個の NN モデルと中間ニューロン数 55 個の NN モデルの結果を示す. 図では, HN は中間ニューロン数を表し, HN=35 と 55 の NN モデルはそれぞれのネットワークにおけるもっとも最小なテスト誤差, 1.45×10^{-3} と 0.24×10^{-3} である. 図より, 55 個のニューロン数を持つ NN とテストデータが良好な一致を示していることがわかる. 本研究では, 経験に基づいて全てのシミュレーションにおけるネットワーク構造を決定した.

次に, f_1 の問題に対する他の最適化手法のパフォーマンス結果を表 2 に示す. 表より GD, CM, NAG, optNAG, AdaGrad, AdaDelta, RMSprop, Adam などの 1 次手法は, 2 次手法と比較して十分に小さい誤差を得ることができなかったことがわかる. 2 次手法では, QN は NAQ そして AdaNAQ と比較して, 反復回数が多いため, 収束までの計算時間が長くなる. NAQ では, 固定のモーメント係数 μ_k が増加するにつれて収束速度は増加する一方で, 平均の誤差が大きくなり安定性は低下する. さらに, NAQ の $\mu_k = 0.95$ と 0.99 のときの最大誤差がそれぞれ 42.30 と 42.40×10^{-3} であり, 1 次手法で得られた誤差の最小値とほぼ同様である. 加えて, μ_k の増加によって大きくなった誤差は, 反復回数の増加に影響している. 一方で, 提案手法の AdaNAQ は, 高速な収束速度を維持しながらも, 誤差の平均および中央値で常に十分小

表 2: f_1 問題に対する AdaNAQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (*sec*)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst
GD	-	42.30 / 42.28 / 42.10 / 42.30	410	150,000	41.30 / 41.30 / 41.20 / 41.40
CM	0.8	38.66 / 39.12 / 32.80 / 42.50	223	107,681	37.30 / 38.30 / 32.10 / 44.20
	0.85	37.84 / 37.59 / 32.80 / 42.50	264	117,589	37.10 / 36.70 / 31.90 / 41.50
	0.9	34.50 / 34.60 / 17.50 / 42.50	218	105,594	33.50 / 44.70 / 16.90 / 190.1
	0.95	42.30 / 35.20 / 8.96 / 42.50	107	55,779	41.40 / 34.70 / 8.55 / 41.50
	0.99	42.45 / 40.53 / 2.03 / 42.50	16	8,971	41.50 / 40.60 / 1.93 / 52.10
NAG	0.8	42.50 / 42.26 / 40.40 / 42.50	95	54,162	41.50 / 4605.1 / 39.40 / 95900.1
	0.85	42.50 / 42.48 / 42.30 / 42.50	42	24,013	41.50 / 22074.0 /41.40 / 190501.0
	0.9	42.50 / 42.49 / 42.30 / 42.50	5	3,248	41.50 / 13292.0 / 41.40 / 129001.0
	0.95	42.50 / 42.50 / 42.50 / 42.50	0.03	15	42.20 / 16964.0 / 41.50 / 120093.1
	0.99	42.50 / 42.50 / 42.50 / 42.50	0.03	14	41.50 / 4363.3 / 41.50 / 63300.0
OptNAG	-	33.00 / 34.41 /16.60 / 41.40	336	150,000	32.10 / 33.50 / 16.00 / 40.40
AdaGrad	-	39.60 / 39.49 / 36.10 / 42.10	181	150,000	38.70 / 38.50 / 35.10 / 41.10
AdaDelta	-	299.5 / 426.4 / 52.59 / 986.6	222	150,000	303.0 / 426.5 / 52.61 / 987.6
RMSprop	-	33.30 / 33.11 / 32.00 / 33.30	176	150,000	32.40 / 32.30 / 31.10 / 32.50
Adam	-	9.63 / 11.12 / 1.30 / 31.60	174	150,000	8.39 / 10.80 / 1.23 / 30.80
QN	-	0.58 / 0.67 / 0.32 / 2.59	172	135,424	0.52 / 0.63 / 0.31 / 2.46
NAQ	0.8	0.55 / 0.58 / 0.36 / 1.01	136	43,792	0.50 / 0.54 / 0.35 / 0.90
	0.85	0.48 / 0.50 / 0.30 / 0.81	127	40,649	0.44 / 0.47 / 0.28 / 0.73
	0.9	0.43 / 0.52 /0.28 / 1.22	108	34,708	0.40 / 0.49 / 0.26 / 1.10
	0.95	0.44 / 2.59 / 0.26 / 42.30	84	26,960	0.41 / 2.53 / 0.24 / 41.40
	0.99	0.56 / 9.25 / 0.15 / 42.40	65	21,437	0.45 / 9.01 / 0.14 / 41.50
AdaNAQ	-	**0.38 / 0.40 / 0.25 / 0.63**	**84**	**28,337**	**0.36 / 0.38 / 0.24 / 0.60**

さい値を得ている. つまり, AdaNAQ は収束が速く, 重み $\mathbf{w}$ の初期値によらず, 全ての試行においてロバストな学習を行うことができることがわかる.

2 つ目の関数近似問題として, (68) の b を $b \in [-0.5, 0.5]$ の範囲に設定し, f_2 とする. この問題では, 入力は x, a そして b の 3 つの要素であり, $|T_r|$ には 10,080 個の学習点が含まれる. f_2 の問題ではテストデータが存在しないため, この問題では学習誤差, 計算時間および反復回数の比較のみを行う. 学習に使用する NN の中間ニューロン数を f_1 と同様に 55 個とする. 従って, NN の構造は 3-55-1 となる. f_2 の実験結果を表 3 に示す. 表 3 では, 1 次手法は f_1 と同様に十分小さな誤差を得られなかったため, ここでは除外した. 表より, f_2 の問題に対する実験結果は f_1 と同様の指向があることがわかる. つまり, 大きな固定の μ_k の値を持つ NAQ の反復回数は少ないものの, 誤差の最大値と最小値が大幅に掛け離れ, 平均値が高くなっている. これは, アルゴリズムが不安定であることを意味する. 一方で, AdaNAQ では, 誤差の平均値と中央値が共に小さな値を得ることができている. 従って, 提案手法は重み $\mathbf{w}$ の初期値によらず, 収束速度の低下が少なく, ロバストであることがわかる. これより, 提案手法

AdaNAQ は, 非線形性の高い関数問題の学習に有効であり, 実用的であると結論付けられる.

表 3: f_2 問題に対する AdaNAQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (*sec*)	Iteration counts (k)
QN	-	0.77 / 1.03 / 0.44 / 2.59	849	117,312
NAQ	0.8	0.70 / 1.24 / 0.39 / 3.49	717	47,137
	0.85	0.63 / 2.59 / 0.33 / 38.42	556	36,565
	0.9	0.59 / 3.47 / 0.37 / 38.42	446	29,286
	0.95	0.64 / 5.89 / 0.27 / 38.55	355	23,260
	0.99	0.67 / 9.77 / 0.31 / 38.50	373	22,546
AdaNAQ	-	**0.49 / 0.56 / 0.36 / 1.96**	408	26,913

3.2.2　マイクロ波回路モデリング問題 1：microstrip Low-Pass Filter

次に, AdaNAQ の有効性を調べるため, マイクロ波回路のモデリング問題である LPF を検討する. 詳細を 2.4 節に記載したこの問題では, 入力は周波数 f と幅 D の 2 つであり, 出力は S パラメータの大きさ $|\mathbf{S}_{11}|$ と $|\mathbf{S}_{21}|$ の 2 つである. NN の構造は, 45 個のニューロン数を持つ中間層 1 層を含むと 2-45-2 となる. LPF の構造図と入力である周波数 f と幅 D そして出力の S パラメータの関係を示した図それぞれ 18 と 19 に示す. LPF の実験結果を表 4 に示す. 表より, これまでの実験と同様に, 1 次手法は小さな学習およびテスト誤差が得られないことがわかる. NAQ の収束速度は, 固定の μ_k が増加するに連れて増加する. しかし, $\mu_k = 0.95$ と 0.99 では, NAQ の学習は不安定になる. これは, 平均と中央および最小と最大の誤差の差からもわかる. 一方で, NAQ ($\mu_k = 0.9$) と AdaNAQ では, 中央, 平均, 最小そして最大の誤差, 全てにおいて非常に小さな値を得ることができている. このことから, AdaNAQ は, 複数回の試行において重み $\mathbf{w}$ の初期値によらず堅牢であるが, NAQ の堅牢性はハイパーパラメータであるモーメント係数 μ_k に強く依存すると結論付けられる.

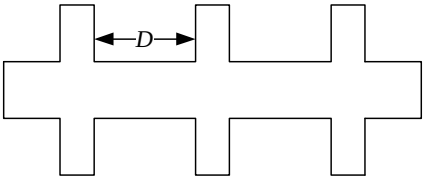


図 18: LPF の構造

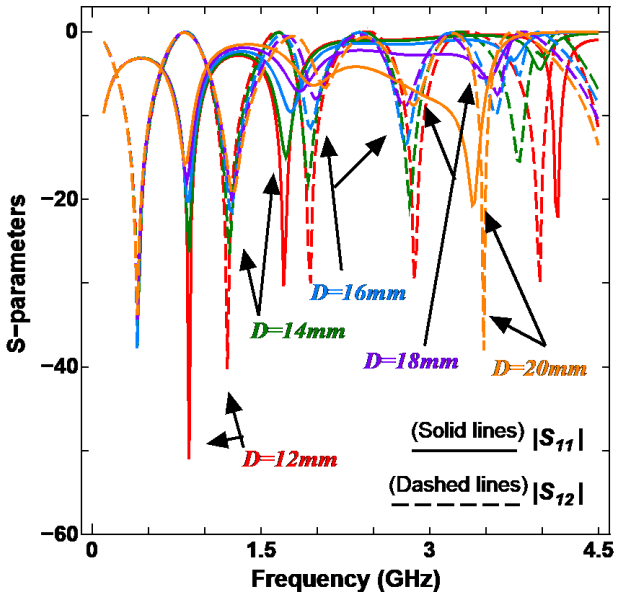


図 19: LPF の学習データ

このシミュレーションでは, モデリングの精度を測定するため, AdaMoQ によって学習された NN の最小テスト誤差である $E_{test}(\mathbf{w}) = 0.44 \times 10^{-3}$ のモデルを, テストデータ $D = 13, 15, 17, 19$mm と比較し, そのグラフを図 26〜29 に示す. 図 26〜29 より, NN のモデルとテストデータが良好な一致を示していることがわかり, これは高精度なモデリングができていることを示す.

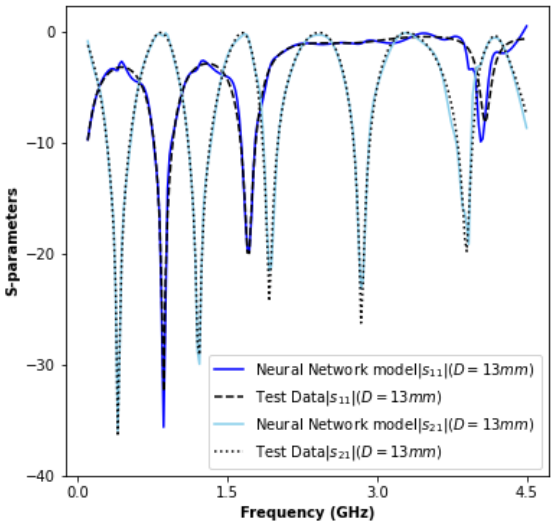


図 26: LPFD = 13 mm に対する AdaNAQ の NN モデルとテストデータの比較

表 4: LPF に対する AdaNAQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst
GD	-	24.94 / 24.99 / 24.80 / 25.30	218	150,000	21.48 / 21.51 / 21.38 / 21.70
CM	0.8	21.15 / 19.72 / 7.07 / 26.90	182	142,866	18.10 / 16.70 / 5.16 / 22.71
	0.85	18.84 / 18.28 / 7.75 / 29.21	147	121,464	17.17 / 28.00 / 5.68 / 205.0
	0.9	8.57 / 16.28 / 5.62 / 29.21	117	99,428	6.22 / 15.21 / 3.87 / 59.63
	0.95	28.90 / 23.87 / 3.37 / 29.21	57	50,916	25.46 / 21.84 / 2.18 / 47.80
	0.99	29.20 / 28.44 / 22.82 / 29.21	38	12,477	25.50 / 39.33 / 20.14 / 264.0
NAG	0.8	22.20 / 21.16 / 12.40 / 29.20	177	142,858	19.60 / 18.09 / 9.88 / 25.50
	0.85	22.15 / 20.40 / 11.50 / 29.20	166	135,716	19.55 / 17.60 / 9.02 / 26.00
	0.9	24.01 / 21.72 / 11.40 / 29.20	115	97,509	21.36 / 58.39 / 9.04 / 819.0
	0.95	29.17 / 26.11 / 10.60 / 29.21	64	35,743	25.46 / 24.31 / 8.59 / 52.86
	0.99	29.20 / 28.93 / 27.00 / 29.21	0.11	31	25.74 / 38.83 /25.40 / 174.9
optNAG	-	19.95 / 17.44 / 8.88 / 23.79	225	150,000	17.24 / 14.68 / 6.20 / 20.84
AdaGrad	-	24.82 / 24.81 / 24.27 / 25.09	136	150,000	21.43 / 21.39 / 21.08 / 21.52
AdaDelta	-	728.3 / 900.5 / 120.7 / 2266.4	134	150,000	729.2 / 902.7 / 119.3 / 2277.3
RMSprop	-	19.16 / 16.88 / 9.02 / 23.40	133	150,000	17.21 / 14.62 / 6.39 / 21.42
Adam	-	5.70 / 5.83 / 2.78 / 18.80	146	150,000	4.89 / 5.54 / 1.90 / 21.77
QN	-	0.73 / 0.80 / 0.59 / 1.49	127	102,629	0.64 / 1.65 / 0.44 / 11.3
NAQ	0.8	0.68 / 0.70 / 0.57 / 0.95	111	50,082	0.61 / 3.44 /0.42 / 50.00
	0.85	0.70 / 0.70 / 0.50 / 1.02	104	46,697	0.63 / 0.78 / 0.43 / 2.34
	0.9	**0.70 / 0.68 / 0.54 / 0.87**	**92**	**41,709**	0.70 / 0.91 /0.45 / 2.58
	0.95	0.81 / 24.69 / 0.44 / 395.0	110	50,117	1.39 / 122.0 / 0.51 / 2020.0
	0.99	16.62 / 53.11 / 0.54 / 446.0	31	14,878	20.05 / 139.0 / 0.54 / 1520.0
AdaNAQ	-	**0.64 / 0.65 / 0.53 / 0.96**	**67**	**31,677**	0.78 / 1.02 / 0.44 / 3.86

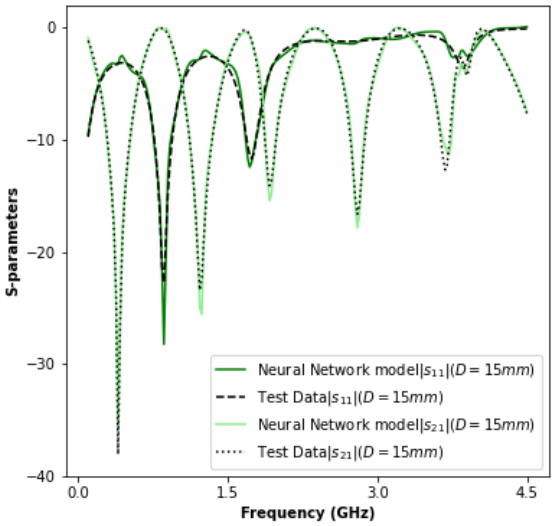


図 27: LPFD = 15 mm に対する AdaNAQ の NN モデルとテストデータの比較

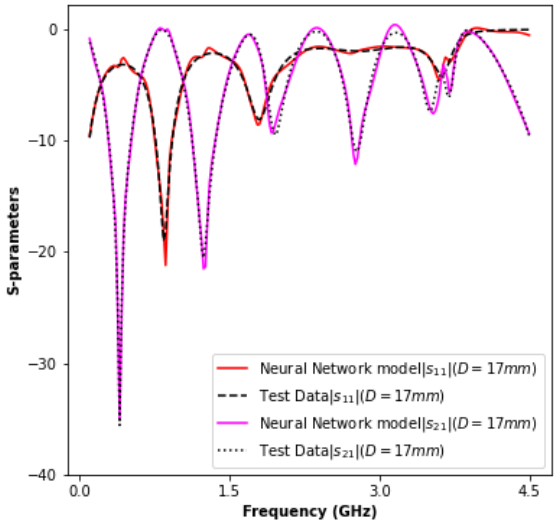


図 28: LPFD = 17 mm に対する AdaNAQ の NN モデルとテストデータの比較

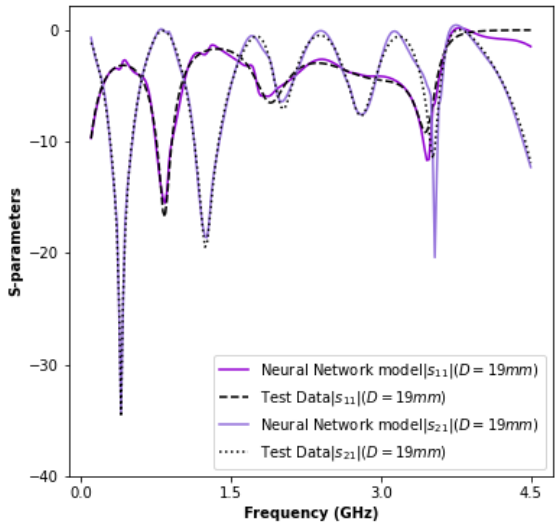


図 29: LPFD = 19 mm に対する AdaNAQ の NN モデルとテストデータの比較

3.2.3　マイクロ波回路モデリング問題 2：Microstrip Patch Antenna

最後に, 提案手法 AdaNAQ を図 30 に示す長方形型 Microstrip Patch Antenna (MPA) [81] の NN モデルの開発に応用する. MPA の NN モデルの入力は, 長さ L, 幅 W, 周波数 f およびマイクロストリップフィードの相対距離 (x/L) である R によって示される. 2 つの出力は, S パラメータ $|S_{11}|$ の実数部と虚数部に割り当てられる. 長さ L および幅 W の範囲は 2 mm 刻みの間隔で $10 \leq L, W \leq 30$ mm までのモデリングを考える. 周波数 f の範囲は 0.5 GHz 刻み

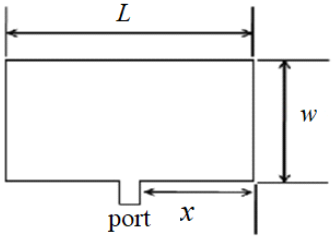


図 30: MPA の構造

の間隔で $0.5 \leq f \leq 5.0$ GHz までであり, R の範囲は 10 刻みの間隔で $10 \leq \mathrm{R} \leq 85$ とした. MPA 問題は 10,890 個のサンプルデータを持つ. 本研究のシミュレーションでは, $|T_r|$ と $|T_e|$ に対してそれぞれ 10,000 および 890 個のサンプルを全サンプルデータからランダムに分別した. MPA のモデリングにおいて, 学習データと入力の次元が大きいため, LPF よりも複雑な最適化問題となる. 従って, MPA では, ニューロン数 1,000 個以上を持つ中間層 1 層のネットワークでは小さな学習誤差を得ることができなかった. よって, NN の構造は 4-50-150-2 とした. つまり, NN のネットワークを拡大させ, 50 と 150 個のニューロン数を持つ中間層 2 層のネットワークで学習を行った. この実験では, AdaNAQ の性能を調べるため, Adam, QN および NAQ ($\mu_k = 0.8, 0.85, 0.9$, そして 0.95) との比較を行い, 最大反復回数は $k_{max} = 50,000$ とした. MPA モデリング問題の実験結果を表 5 に示す. 表より, Adam, QN, および全ての固定の μ_k における NAQ では十分に小さな学習とテスト誤差が得られていないことがわかる. 一方, AdaNAQ のみもっとも少ない反復回数と計算時間で, 学習と検証誤差を実用的な小さな値まで減少させることができている. さらに, AdaNAQ の平均と中央値誤差も十分に小さいことが確認できる. 従って, 提案手法が MPA のような大規模かつ複雑な問題に対しても有効であることが確認できた.

表 5: MPA に対する AdaNAQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median /Ave. / Best / Worst	Time (sec)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst
Adam	-	5.29 / 5.29 / 1.70 / 10.47	8,757	50,000	6.84 / 6.84 / 2.55 / 13.06
QN	-	1.91 / 1.91 / 1.75 / 2.09	7,416	27,856	3.29 / 3.31 / 2.91 /4.21
NAQ	0.8	3.57 / 69.10 / 1.44 / 313.7	5,846	11,993	5.05 / 96.70 / 2.70 / 631.2
	0.85	5.05 / 90.22 / 1.88 / 314.0	6,913	13,340	7.06 / 109.1 / 3.32 / 527.0
	0.9	5.69 / 58.86 / 1.91 / 310.0	4,215	9,560	7.72 / 77.68 / 3.06 / 535.3
	0.95	12.50 / 89.20 / 1.81 / 300.7	6,641	13,755	14.40 / 109.1 / 2.70 / 543.6
	0.99	166.5 / 160.5 / 7.10 / 284.0	2,066	4,657	176.2 / 176.2 / 9.05 / 550.1
AdaNAQ	-	**0.73 / 0.73 / 0.59 / 0.91**	**3,965**	**9,190**	**1.75 / 1.75 / 1.25 / 2.34**

3.3　まとめ

本章では, 本研究の 1 つ目の研究課題であった, NAQ [32–34] のハイパーパラメータ問題と学習の安定性に着目し, 解決手法として 適応的ネステロフの加速準ニュートン法 (Adaptive Nesterov's Accelerated Quasi-Newton method, AdaNAQ) を提案した. NAQ では, 慣性係数がハイパーパラメータであったため, 最適な値に設定する必要があった. このため, 多くの経験と時間を必要とした. さらに, 複数回の試行における学習精度は NN の重み $\mathbf{w}$ の初期値に強く依存し, 安定した学習が行えなかった. 本研究で提案した AdaNAQ では, 凸最適化問題に対する 1 次手法の NAG による学習において提案された適応的慣性係数を NAQ に導入した. これにより, NAQ の慣性係数は学習初期の段階では 0 に近い値で学習を行い, 学習が安定した後, 1 に近い値を得ることで, 学習を高速化させる. 適応的慣性係数は学習中における NAQ を安定化させるため, 複数回の試行を行っても, 学習は近い解に収束する. 従って, 提案手法 AdaNAQ は, NAQ におけるハイパーパラメータを除去し, ロバストで効果的な収束特性を得ることで, NAQ の問題点を克服することに成功した. 提案手法の有効性は 2 つの非線形問題と 2 つの強非線形問題に対する NN の学習に応用し, 計算機実験を行い, 従来アルゴリズムとの比較を行った上, 示した. 本研究では, 低コストで高精度な学習アルゴリズムの提案を目指すため, 使用する全てのネットワークが, 高精度な解を得る最小な NN 構造に設定した. その結果, AdaNAQ はマイクロ波回路のモデリングに実用的かつ効果的であることが確認された. 従って, 提案手法はマイクロ波回路のような強非線形問題の正確な NN モデルを提供するために有効であると結論付けられる.

4　慣性付準ニュートン法

本章では, 本研究の 2 つ目の課題であった, NAQ の計算時間の増加に着目し, 解決手法として NAQ で用いられているネステロフの加速勾配を近似した新たな準ニュートン法に基づく学習アルゴリズムを提案する. これまでに, QN の改良手法がいくつも提案されている. これまでの改良手法の中で, 慣性項による高速化が行われた手法として, マルチステップ準ニュートン法 (Multi-Step Quasi-Newton method, MSQN) [27–29] と NAQ [32–34] が挙げられる. MSQN は過去の学習ステップのパラメータと勾配を曲率情報の更新にのみ用いる手法である. 一方で, NAQ は慣性項とネステロフの加速勾配を曲率情報の更新に使用すると共に, 更新ベクトルにも用いることで, QN の反復回数と総計算時間を大幅に削減した. 結果として, NAQ は, MSQN 以上に QN の高速化に成功した. しかし, NAQ では 1 反復において 2 つの勾配, 通常勾配とネステロフの加速勾配を計算する必要があり, これは 1 反復の計算時間を増加させる欠点に繋がった. 本研究では，誤差関数を 2 次関数と見なすことで NAQ に用いられているネステロフの加速勾配を通常勾配の線形和として近似する. 具体的に, 2 次手法では誤差関数を 2 次のテイラー展開として近似しているため, 誤差関数と 2 次のティラー展開で得られた 2 次関数との間に親和性を見出すことができると考える. また, NN の誤差関数は滑らかな関数であるため, 反復点のまわりでは 2 次関数と見なすことが可能であると考える. 従って, 反復点における誤差関数を 2 次関数と見なした場合, その誤差は小さく, 2 次関数と近似した誤差関数の勾配ベクトルは線形関数と見なすことができる. よって, ネステロフの加速勾配を通常勾配の重み付の線形和として近似できる. 以上の考え方に基づき, ここでは 1 反復における勾配計算を 1 回に抑え, 計算時間の高速化を目指した, 慣性付準ニュートン法 (Momentum Quasi-Newton method, MoQ) を提案する [37–39]. MoQ は, NAQ の高い学習精度を維持しつつも, 1 回の反復に必要な計算時間を短縮することが期待される. さらに, 本章では, MoQ のハイパーパラメータである慣性係数問題を解決するため, 適応的慣性係数を導入し, 適応的慣性付準ニュートン法 (Adaptive Momentum Quasi-Newton method, AdaMoQ) として提案する. 提案手法を非線形性の高い 2 つの関数近似問題, 2 つのマイクロ波回路モデリング問題そして 2 つの分類問題に対する NN の学習に応用し, 従来手法と比較を行った上, その有効性を計算機実験により示す.

4.1 Multi-Step Quasi-Newton method

マルチステップ準ニュートン法 (Multi-Step Quasi-Newton method, MSQN) は, QN のヘッセ行列 $\mathbf{H}_k^{\mathrm{QN}}$ の更新において, 過去のステップの $\mathbf{w}_m$ と $\nabla E\left(\mathbf{w}_m\right), (m = k-1, k-2, \ldots)$ を用いて, セカント条件を拡張させることにより, QN を高速化した手法である [27–29]. MSQN における (3) の更新ベクトル $\mathbf{v}_{k+1}$ を (69) に示す.

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k^{\mathrm{MSQN}} \nabla E\left(\mathbf{w}_k\right), \tag{69}$$

ここで, ヘッセ行列 $\mathbf{H}_k^{\mathrm{MSQN}}$ は (70) と (71) の $\hat{\mathbf{s}}_k$ と $\hat{\mathbf{y}}_k$ を用いて, (72) で更新される.

$$\hat{\mathbf{s}}_k = \mathbf{w}_{k+1} - (1+\nu_k)\mathbf{w}_k + \nu_k \mathbf{w}_{k-1}, \tag{70}$$

$$\hat{\mathbf{y}}_k = \nabla E(\mathbf{w}_{k+1}) - (1+\nu_k)\nabla E(\mathbf{w}_k) + \nu_k \nabla E(\mathbf{w}_{k-1}), \tag{71}$$

$$\mathbf{H}_{k+1}^{\mathrm{MSQN}} = \left(\mathbf{I} - \frac{\hat{\mathbf{y}}_k \hat{\mathbf{s}}_k^{\mathrm{T}}}{\hat{\mathbf{s}}_k^{\mathrm{T}} \hat{\mathbf{y}}_k}\right)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{MSQN}} \left(\mathbf{I} - \frac{\hat{\mathbf{y}}_k \hat{\mathbf{s}}_k^{\mathrm{T}}}{\hat{\mathbf{s}}_k^{\mathrm{T}} \hat{\mathbf{y}}_k}\right) + \frac{\hat{\mathbf{s}}_k \hat{\mathbf{s}}_k^{\mathrm{T}}}{\hat{\mathbf{s}}_k^{\mathrm{T}} \hat{\mathbf{y}}_k}. \tag{72}$$

ここで ν_k は, (73) より更新される加速項である [29].

$$\nu_k = \frac{\|\hat{\mathbf{s}}_{k-1}\|^2}{2\|\hat{\mathbf{s}}_{k-2}\|^2 + \|\hat{\mathbf{s}}_{k-1}\|\|\hat{\mathbf{s}}_{k-2}\|}. \tag{73}$$

一般的に, MSQN の過去の記憶量 m は任意に設定することができる. 本研究では, 最も効率的とされる記憶量 $m = 2$ の場合を考える [28, 29]. MSQN のアルゴリズムを Algorithm 11 に示す.

Algorithm 11 Multi-Step Quasi-Newton method (MSQN)

Require: ϵ, k_{max}
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$, $\mathbf{H}_1^{\mathrm{MSQN}} = \mathbf{I}$(unit matrix)
1: $k = 1$
2: Calculate $\nabla E(\mathbf{w}_1)$;
3: **while** ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
4: Calculate stepsize α_k;
5: Update $\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k^{\mathrm{MSQN}} \nabla E(\mathbf{w}_k)$;
6: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
7: Calculate $\nabla E(\mathbf{w}_{k+1})$;
8: Update ν_k using (73);
9: Calculate $\hat{\mathbf{s}}_k$ and $\hat{\mathbf{y}}_k$ using (70) and (71);
10: Update $\mathbf{H}_{k+1}^{\mathrm{MSQN}}$ using (72);
11: $k = k + 1$;
12: **end while**
Return: $\mathbf{w}_k$

4.2 Momentum Quasi-Newton method

本研究では, NAQ における 1 反復の計算時間を短縮するため, その勾配に着目した. NAQ では, $\mathbf{H}_k^{\mathrm{NAQ}}$ 行列の更新において, $\mathbf{w}_k$ での勾配 $\nabla E(\mathbf{w}_k)$(以降, 通常勾配) と $\mathbf{w}_k$ から慣性項 $\mu_k \mathbf{v}_k$ だけ移動した点での勾配 $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$, ネステロフの加速勾配を用いている. 一方, QN では通常勾配のみが用いられている. 従って, これは NAQ の 1 反復の計算時間を QN と比較して, 増加させる原因となる. 本研究では, この問題を解決した準ニュートン法に基づく新しい学習アルゴリズム, 慣性付準ニュートン法 (Momentum Quasi-Newton method, MoQ) を提案する. MoQ は, ネステロフの加速勾配ベクトルを現反復の重み $\mathbf{w}_k$ とその 1 反復前の重み $\mathbf{w}_{k-1}$ における通常勾配ベクトルの重み付きの線形和として近似することで, 1 反復で用いられる勾配計算回数を 1 度に抑えることを目指した. 提案手法 MoQ は, 誤差関数を $\mathbf{w}_k + \mu_k \mathbf{v}_k$ の近傍で 2 次関数と見なすことで, (4) に示す誤差関数 E ($\mathbf{w}$) を近似して実現した手法である. この仮定は, 2 次手法の設計に一般的に用いられる (57) の 2 次のテイラー展開 $\hat{E}(\mathbf{w})$ に相似している. ここで, $\hat{E}(\mathbf{w})$ は 2 次関数であるが, $E(\mathbf{w})$ は高次の非線形関数である. しかし, この近似は 2 次近似に基づく手法では有効である. つまり, 1 反復内で $E(\mathbf{w})$ はティラー展開を用いて 2 次関数に近似されており, その反復での勾配を導出する際にも 2 次関数と近似することを考えた場合, その誤差は小さいと考えられる. これより $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ の勾配ベク

トルは線形として近似できる. 従って, (74) を得る.

$$\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \simeq \nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{v}_k). \tag{74}$$

さらに, $\mathbf{v}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$ であるため, (74) は次のように書き換えることができる.

$$\begin{aligned} \nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{v}_k) &= \nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{w}_k - \mathbf{w}_{k-1}) \\ &= (1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}). \end{aligned} \tag{75}$$

(74) と (75) よりネステロフの加速勾配は通常勾配の重み付き線形和, つまりモーメント係数 μ_k を用いた $\nabla E(\mathbf{w}_k)$ と $\nabla E(\mathbf{w}_{k-1})$ の外挿として近似できることがわかる. 本研究では (75) で示したモーメント係数 μ_k を用いた勾配ベクトル $\mu_k \nabla E(\mathbf{v}_k) = \mu_k\{\nabla E(\mathbf{w}_k) - \nabla E(\mathbf{w}_{k-1})\}$ を, 慣性勾配 (Momentum Gradient, MoG) と呼ぶ. 従って, MoQ は 2 つの慣性項 $\mu_k \mathbf{v}_k$ および $\mu_k \nabla E(\mathbf{v}_k)$ を用いて QN を加速する手法と見なすことができる. MoQ における (3) の更新ベクトル $\mathbf{v}_{k+1}$ を (76) に示す.

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k^{\mathrm{MoQ}} \{(1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}. \tag{76}$$

(76) において, $\mathbf{H}_k^{\mathrm{MoQ}}$ 行列は (77) に従って更新される.

$$\begin{aligned} \mathbf{H}_{k+1}^{\mathrm{MoQ}} &= \mathbf{H}_k^{\mathrm{MoQ}} - \frac{(\mathbf{H}_k^{\mathrm{MoQ}}\hat{\mathbf{q}}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k(\mathbf{H}_k^{\mathrm{MoQ}}\hat{\mathbf{q}}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} + \left(1 + \frac{\hat{\mathbf{q}}_k^{\mathrm{T}}\mathbf{H}_k^{\mathrm{MoQ}}\hat{\mathbf{q}}_k}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right)\left(\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right) \\ &= \left(\mathbf{I} - \frac{\hat{\mathbf{q}}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{MoQ}} \left(\mathbf{I} - \frac{\hat{\mathbf{q}}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right) + \left(\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right). \end{aligned} \tag{77}$$

ここで, MoQ のセカント条件 $\hat{\mathbf{q}}_k = (\mathbf{H}_k^{\mathrm{MoQ}})^{-1}\mathbf{p}_k$ は満たされ, $\mathbf{p}_k$ と $\hat{\mathbf{q}}_k$ はそれぞれ (78) と (79) で示される.

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k) = \mathbf{w}_{k+1} - \{(1+\mu_k)\mathbf{w}_k - \mu_k \mathbf{w}_{k-1}\}, \tag{78}$$

$$\hat{\mathbf{q}}_k = \nabla E(\mathbf{w}_{k+1}) - \{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{w}_{k-1})\}. \tag{79}$$

MoQ のアルゴリズムを Algorithm 12 に示す. アルゴリズムでは, 学習ループにおいて計算される必要がある勾配は, Step.7 の $\nabla E(\mathbf{w}_{k+1})$ のみである. つまり, ネステロフの加速勾配の計算を省き, NAQ の問題点であった 1 反復における 2 回の勾配計算を 1 回に抑えることができたことを示している.

Algorithm 12 Momentum Quasi-Newton method (MoQ)

Require: ϵ, k_{max}, μ_k
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$, $\mathbf{H}_1^{\mathrm{MoQ}} = \mathbf{I}$(unit matrix), $\mathbf{v}_1 = 0$
$k = 1$
Calculate $\nabla E(\mathbf{w}_k)$;
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
  Calculate stepsize α_k;
  Update $\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k^{\mathrm{MoQ}} \{(1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}$;
  Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
  Calculate $\nabla E(\mathbf{w}_{k+1})$;
  Calculate $\mathbf{p}_k$ and $\hat{\mathbf{q}}_k$ using (61) and (62);
  Update $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$ using (77);
  $k = k + 1$;
end while
Return: $\mathbf{w}_k$

次に, MoQ の重みの更新ベクトルの表現を図 31 に示す. この図では, 青いベクトルは NAQ の更新, 赤いベクトルは MoQ の更新そしてグレーのベクトルは過去の更新ベクトルを示す. ここで, 誤差関数 $E(\mathbf{w})$ が 2 次関数である場合, つまり, (74) において等号が成立するとき, (80) の関係性も成立する.

$$
\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k + \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k^{\mathrm{NAQ}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \\
&= \mathbf{w}_k + \mu_k \mathbf{w}_k - \mu_k \mathbf{w}_{k-1} - \alpha_k \mathbf{H}_k^{\mathrm{MoQ}} \{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{w}_{k-1}) \\
&= (1+\mu_k)\mathbf{w}_k - \alpha_k (1+\mu_k)\mathbf{H}_k^{\mathrm{MoQ}} \nabla E(\mathbf{w}_k) - \left(\mu_k \mathbf{w}_{k-1} - \alpha_k \mu_k \mathbf{H}_k^{\mathrm{MoQ}} \nabla E(\mathbf{w}_{k-1})\right) \\
&= (1+\mu_k)\left(\mathbf{w}_k - \alpha_k \mathbf{H}_k^{\mathrm{MoQ}} \nabla E(\mathbf{w}_k)\right) - \mu_k \left(\mathbf{w}_{k-1} - \alpha_k \mathbf{H}_k^{\mathrm{MoQ}} \nabla E(\mathbf{w}_{k-1})\right).
\end{aligned}
\tag{80}
$$

ここで, 誤差関数 $E(\mathbf{w})$ の曲率情報の放物線は, $\mathbf{H}_k^{\mathrm{NAQ}}$ と $\mathbf{H}_k^{\mathrm{MoQ}}$ が等しいため同じであると考えられる. 従って, 図 31 では, MoQ の更新ベクトルは NAQ のと同様であるように示されている.

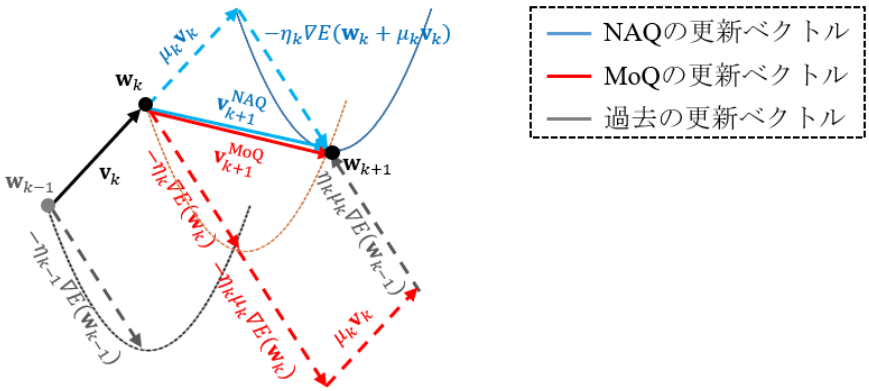


図 31: MoQ の重みの更新ベクトル表現

4.2.1　MoQ の収束特性

本節では提案手法 MoQ の収束特性について考察する. (77) に従って更新される $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$ は $\mathbf{H}_k^{\mathrm{MoQ}}$ が正定値対称行列である仮定の下で, 正定値性が保たれる. つまり, 1 反復前の行列 $\mathbf{H}_k^{\mathrm{MoQ}}$ が正定値ならば, 以下が成立する:

(*a*): (77) の $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$ は MoQ のセカント条件, $\hat{\mathbf{q}}_k = \left(\mathbf{H}_{k+1}^{\mathrm{MoQ}}\right)^{-1}\mathbf{p}_k$ を満たす.

(*b*): $\mathbf{H}_k^{\mathrm{MoQ}}$ が対称ならば, $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$ も対称である.

(*c*): $\mathbf{H}_k^{\mathrm{MoQ}}$ が正定値ならば, $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$ も正定値である.

〈(*a*) の証明〉: (77) およびセカント条件 $\hat{\mathbf{q}}_k = \left(\mathbf{H}_{k+1}^{\mathrm{MoQ}}\right)^{-1}\mathbf{p}_k$ より:

$$\left(\mathbf{H}_{k+1}^{\mathrm{MoQ}}\right)^{-1}\mathbf{p}_k = \left(\left(\mathbf{I}-\frac{\mathbf{p}_k\hat{\mathbf{q}}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right)^{\mathrm{T}}\mathbf{H}_k^{\mathrm{MoQ}}\left(\mathbf{I}-\frac{\hat{\mathbf{q}}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right)+\left(\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right)\right)^{-1}\mathbf{p}_k \tag{81}$$

ここで (81) に Sherman-Morrison-Woodbury 公式 [21](付録 A, (A.4)) を適用することにより,

$$=\left(\mathbf{B}_k^{\mathrm{MoQ}}+\frac{\hat{\mathbf{q}}_k\hat{\mathbf{q}}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}-\frac{\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k}\right)\mathbf{p}_k \tag{82}$$

ここで $\mathbf{B}_k^{\mathrm{MoQ}} = \left(\mathbf{H}_k^{\mathrm{MoQ}}\right)^{-1}$ とした.

$$\left(\mathbf{H}_{k+1}^{\mathrm{MoQ}}\right)^{-1}\mathbf{p}_k = \mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k+\left(\frac{\hat{\mathbf{q}}_k\hat{\mathbf{q}}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}\right)\mathbf{p}_k-\left(\frac{\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k}\right)\mathbf{p}_k = \hat{\mathbf{q}}_k.$$

□

〈(*b*) の証明〉: これは (77) より明らかである.

□

〈(*c*) の証明〉：まず, (78) と (79) の $\mathbf{p}_k$ と $\hat{\mathbf{q}}_k$ に対して,

$$\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k > 0 \tag{83}$$

が成立することを示す. ここで, 厳密な直線探索によってステップサイズ α_k を求めた場合,

$$\frac{dE\left(\mathbf{w}_{k+1}\right)}{d\alpha_k} = -\nabla E(\mathbf{w}_{k+1})^{\mathrm{T}}\mathbf{H}_k^{\mathrm{MoQ}}\{\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k)\}) = 0, \tag{84}$$

となる. 従って,

$$\begin{aligned}\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k &= \{\nabla E(\mathbf{w}_{k+1}) - (\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k))\}^{\mathrm{T}}\{\mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k\mathbf{v}_k)\} \\ &= \{\nabla E(\mathbf{w}_{k+1}) - (\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k))\}^{\mathrm{T}}\{-\alpha_k\mathbf{H}_k^{\mathrm{MoQ}}\left(\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k)\right)\},\end{aligned} \tag{85}$$

ここで, $\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k) \neq 0$ であるため,

$$\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k = \alpha_k\left(\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k)\right)^{\mathrm{T}}\mathbf{H}_k^{\mathrm{MoQ}}\left(\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{v}_k)\right) > 0, \tag{86}$$

が成立する. ここで,

$$\nabla E\left(\mathbf{w}_k\right) + \mu_k\nabla E\left(\mathbf{v}_k\right) = \left(1 + \mu_k\right)\nabla E\left(\mathbf{w}_k\right) - \mu_k\nabla E\left(\mathbf{w}_{k-1}\right), \tag{87}$$

$$\mathbf{w}_k + \mu_k\mathbf{v}_k = \left(1 + \mu_k\right)\mathbf{w}_k - \mu_k\mathbf{w}_{k-1}. \tag{88}$$

である. 次に, $\mathbf{B}_{k+1}^{\mathrm{MoQ}}$ の正定値性, つまり, 任意のベクトル $\mathbf{r} \neq \mathbf{0}$ に対して,

$$\mathbf{r}^{\mathrm{T}}\mathbf{B}_{k+1}^{\mathrm{MoQ}}\mathbf{r} > 0, \tag{89}$$

であることを示す. (89) では, 簡略化のために, $\left(\mathbf{H}_{k+1}^{\mathrm{MoQ}}\right)^{-1}$ を $\mathbf{B}_{k+1}^{\mathrm{MoQ}}$ として表す. ここで, $\mathbf{B}_{k+1}^{\mathrm{MoQ}}$ が正定値行列ならば, その逆行列 $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$ も正定値行列であることは明らかである [23, 25]. $\mathbf{B}_k^{\mathrm{MoQ}}$ は正定値行列であるので, 正則行列 $\mathbf{C}$ を用いて, $\mathbf{B}_k^{\mathrm{MoQ}} = \mathbf{C}\mathbf{C}^{\mathrm{T}}$ と分解できる. そこで,

$\mathbf{t} = \mathbf{C}^{\mathrm{T}}\mathbf{r}$ および $\mathbf{u} = \mathbf{C}^{\mathrm{T}}\mathbf{p}_k$ とおくと, (89) は

$$\begin{aligned}
\mathbf{r}^{\mathrm{T}}\mathbf{B}_{k+1}^{\mathrm{MoQ}}\mathbf{r} &= \mathbf{r}^{\mathrm{T}}\left(\mathbf{B}_k^{\mathrm{MoQ}} + \frac{\hat{\mathbf{q}}_k\hat{\mathbf{q}}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} - \frac{\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k}\right)\mathbf{r} \\
&= \mathbf{r}^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{r} + \frac{\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k\hat{\mathbf{q}}_k^{\mathrm{T}}\mathbf{r}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} - \frac{\mathbf{r}^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{r}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{B}_k^{\mathrm{MoQ}}\mathbf{p}_k} \\
&= \mathbf{r}^{\mathrm{T}}\mathbf{C}\mathbf{C}^{\mathrm{T}}\mathbf{r} + \frac{\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k\hat{\mathbf{q}}_k^{\mathrm{T}}\mathbf{r}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} - \frac{\mathbf{r}^{\mathrm{T}}\mathbf{C}\mathbf{C}^{\mathrm{T}}\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\mathbf{C}\mathbf{C}^{\mathrm{T}}\mathbf{r}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{C}\mathbf{C}^{\mathrm{T}}\mathbf{p}_k} \\
&= \mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{(\mathbf{r}\hat{\mathbf{q}}_k)^2}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} - \frac{\left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2}{\mathbf{u}^{\mathrm{T}}\mathbf{u}} \\
&= \frac{\left(\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right)\left(\mathbf{u}^{\mathrm{T}}\mathbf{u}\right) - \left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2\right)}{\mathbf{u}^{\mathrm{T}}\mathbf{u}} + \frac{\left(\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k\right)^2}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k},
\end{aligned} \tag{90}$$

と変形できる. (86) より, (90) の第 2 項に対して,

$$\frac{\left(\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k\right)^2}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \geq 0, \tag{91}$$

が成立する. 次に, (90) の第 1 項に関して, Cauchy-Schwarz 公式 [21] より, 任意の 0 でないベクトル $\mathbf{t}$ と $\mathbf{u}$ に対して,

$$\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right)\left(\mathbf{u}^{\mathrm{T}}\mathbf{u}\right) \geq \left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2, \tag{92}$$

が成立する. (91) および (92) より, (90) に対して,

$$\frac{\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right)\left(\mathbf{u}^{\mathrm{T}}\mathbf{u}\right) - \left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2}{\mathbf{u}^{\mathrm{T}}\mathbf{u}} + \frac{\left(\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k\right)^2}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \geq 0, \tag{93}$$

が成立する. ここで, (93) の等号が成立するには

$$\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right)\left(\mathbf{u}^{\mathrm{T}}\mathbf{u}\right) - \left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2 = 0, \tag{94}$$

かつ

$$\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k = 0, \tag{95}$$

が成立する時である. ここで, (94) が成立するには, 任意の値 $\gamma\,(\neq 0)$ を用いて, $\mathbf{t} = \gamma\mathbf{u}$ が成立した時である. この時, $\mathbf{t} = \gamma\mathbf{u}$ より, $\mathbf{r} = \gamma\mathbf{p}_k$ となる. これを (95) に代入すると,

$$\mathbf{r}^{\mathrm{T}}\hat{\mathbf{q}}_k = \gamma\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k = 0, \tag{96}$$

となる. (96) は $\gamma \neq 0$ および (86) に矛盾する. つまり, (90) の等号が成立しないことを意味する. 従って,

$$\mathbf{r}^{\mathrm{T}}\mathbf{B}_{k+1}^{\mathrm{MoQ}}\mathbf{r} > 0. \tag{97}$$

が成立する.

□

以上より, 提案手法 MoQ は NAQ [33, 34] および QN [21] と同様な収束性を持つことがわかる. つまり, (76) を用いて, 更新を行う MoQ では, 常に誤差関数 $E(\mathbf{w})$ が減少する方向に更新させることが可能な勾配法であることがわかる.

さらに, 本研究では MoQ における数値の安定性とグローバル収束を保持するために, (98) に示すグローバル収束項 [24, 82] を (79) の $\hat{\mathbf{q}}_k$ に導入した. 本研究では, シミュレーションにおける公平なアルゴリズム比較を行うため, QN, MSQN, NAQ および AdaNAQ に対しても MoQ と同様にグローバル収束項を導入している [24, 82].

$$\hat{\mathbf{q}}_k = \nabla E(\mathbf{w}_{k+1}) - (1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{w}_{k-1}) + \xi_k \mathbf{p}_k = \delta_k + \xi_k \mathbf{p}_k, \tag{98}$$

ここで, ξ_k は MoQ の収束を保持する係数であり, (99) と (100) で求める.

$$\xi_k = \omega \parallel (1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}) \parallel + \max\left\{ \frac{-\delta_k^{\mathrm{T}} \mathbf{p}_k}{\parallel \mathbf{p}_k \parallel^2}, 0 \right\}, \tag{99}$$

$$\begin{cases} \omega = 2 & \text{if} \quad \parallel (1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}) \parallel^2 > 10^{-2}, \\ \omega = 100 & \text{if} \quad \parallel (1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}) \parallel^2 < 10^{-2}. \end{cases} \tag{100}$$

4.2.2　計算コスト

2 次手法の計算コストとメモリ量を表 6 に示す. 表では, 勾配評価コストは nd として表し, n は学習サンプル数, d はパラメータの次元である. ヘッセ行列の逆近似行列の更新コストを d^2 とする. 表の全てのアルゴリズムでは, ステップサイズは直線探索法により決定されるため, 探索条件を満たすまで ζ 回関数評価を行う. NAQ は QN と MoQ が各反復で勾配を 1 度計算するのに比べ, 勾配を 2 度計算している. 従って, NAQ は追加の計算コスト nd を持つ. 一方, QN と MoQ は各反復あたりの勾配計算回数が 1 回であるため, 同じ計算コストを持つと考えられる.

保存されるメモリ量に関しては, QN, NAQ および MoQ では, 過去のヘッセ行列を保持す

るため, d^2 のメモリが必要である. さらに, QN と NAQ では $\mathbf{s}_k$, $\mathbf{y}_k$, $\mathbf{p}_k$ そして $\mathbf{q}_k$ を計算する必要があるため, $2(n+1)d$ のメモリ量が追加で必要となる. 一方, MoQ における $\hat{\mathbf{q}}_k$ の計算では, ネステロフの加速勾配の近似を行っているため, 過去の勾配を 1 つ多く記憶する必要がある. 従って, メモリ量は $(3n+2)d$ となる.

表 6: 2 次手法の計算コストとメモリ量の比較

Algorithm	Computational Cost	Storage
QN	$nd + d^2 + \zeta nd$	$d^2 + 2(n+1)d$
NAQ	$2nd + d^2 + \zeta nd$	$d^2 + 2(n+1)d$
MoQ	$nd + d^2 + \zeta nd$	$d^2 + (3n+2)d$

ここで, 本研究で使用する最小のパラメータ次元 $d = 31$ を持つ問題から最大のパラメータ次元 $d = 7,960$ を持つ問題まで, 次元 d の値を少しずつ増加させ, 次元 d に対する計算コストとメモリ量の関係性を図 32 と 33 に示す. グラフを見やすくするため, 図 32 と 33 の y– 軸と x– 軸の表記はそれぞれ対数と線形としたと共に, 全アルゴリズムの計算コストおよびメモリ量で共通している項を除外した. 図 32 では, QN と MoQ の計算コストが同じであるため, グラフが重なっている. また NAQ では, QN と MoQ と比較して勾配計算回数が多いため, コストも増加している. 図 33 では, QN と NAQ に必要なメモリ量が同じであるため, グラフが重なっている一方で, MoQ では記憶する必要がある勾配が 1 つ増えるため, メモリ量も増加している.

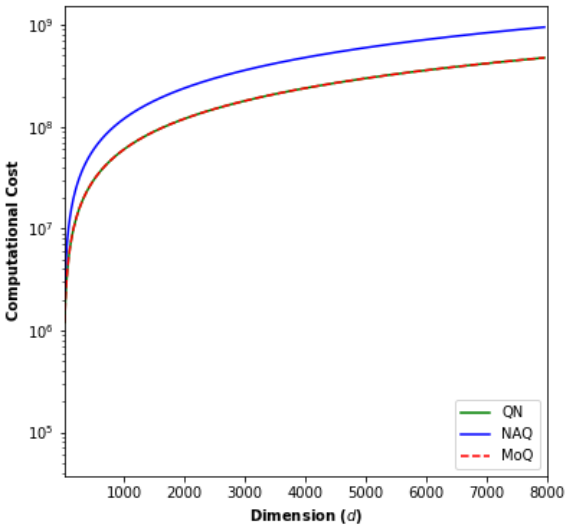


図 32: 2 次手法の計算コストの比較

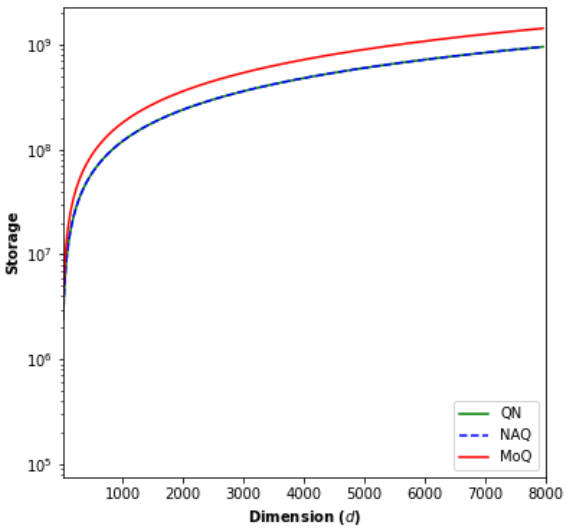


図 33: 2 次手法のメモリ量の比較

4.2.3　適応的慣性係数

本研究において, MoQ のロバスト性を向上させるため, 3.1 節に示した (64) と (65) の適応的モーメント係数 μ_k を MoQ に導入し, 適応的慣性付準ニュートン法 (Adaptive Momentum Quasi-Newton method, AdaMoQ) として提案する [40].

$$\mu_k = \frac{\theta_k(1-\theta_k)}{\theta_k^2 + \theta_{k+1}}, \tag{64}$$

$$\theta_{k+1}^2 = (1-\theta_{k+1})\theta_k^2 + \gamma\theta_{k+1}. \tag{65}$$

ここで, $\theta_1 = 1$ であり, $\gamma = 1.0 \times 10^{-5}$ と設定する. AdaMoQ では, NAQ と同様にハイパーパラメータ μ_k が含まれている. 従って, 適応的慣性係数 μ_k を導入することで, 学習内で徐々にその値を 0 から 1 に変化させる. これにより, AdaMoQ は NAQ の高速な性質を維持しつつ, AdaNAQ と同様に学習の安定性を向上させ, 重み $\mathbf{w}$ の初期値に対するロバストな学習を可能とする.

4.3　実験

本研究では提案手法 MoQ と AdaMoQ(以降, (Ada)MoQ) の有効性を示すため, 2 つの関数近似問題, 2 つのマイクロ波回路モデリング問題そして 2 つの分類問題に対するシミュレーションを行った. シミュレーションでは, 任意のニューロン数を持つ中間層を含めた階層型 NN を用いた. 本研究の実験において提案手法 MoQ と AdaMoQ は, 1 次手法の GD, CM, NAG, 適応的モーメント係数を用いた NAG(OptNAG), AdaGrad, RMSprop, AdaDelta, Adam および 2 次手法の QN, MSQN, NAQ そして AdaNAQ の従来手法と比較され, その効果に対して議論される. AdaGrad, RMSprop, AdaDelta および Adam における各ハイパーパラメータは, それぞれの論文で推奨値とされている値に設定した [16–19]. 本研究では強非線形関数の NN によるモデリングを対象としているため, 全てのシミュレーションでバッチ学習を行う [79]. シミュレーションに用いる全ての例題において, 10 回の異なる重み $\mathbf{w}$ の初期値に対して学習を行い, $\mathbf{w}$ は $[-0.5, 0.5]$ 内の一様乱数で初期化される. CM, NAG, NAQ および MoQ に用いられる固定のモーメント係数は $0.8 \leq \mu_k \leq 0.95$ の 0.05 刻みとした [14, 34]. 学習された各 NN の結果は, $E(\mathbf{w})$ の中央, 平均, 最小および最大値に加え平均の計算時間 (sec) および収束までに必要な平均の反復回数 (k) によって各表で示される. T_r の誤差は, $E_{train}(\mathbf{w})$ で示される. テスト (検証) 誤差は $E_{test}(\mathbf{w})$ として示され, この誤差は T_r の学習データセットに依存しないテストデータセット T_e を用いて, (4) で計算される. 誤差関数および活性化関数に関しては, 先行研究 [5, 6, 32–34, 68, 69, 78, 79] に基づき, 2 つの関数近似問題と 2 つのマイクロ波回路のモデリング問題では誤差関数を (15) の MSE と (16) のシグモイド関数とした. また, 2 つの画像分類問題では, (19) の CE と (20) のソフトマックス関数に設定した. 入力の各要素および T_r と T_e は, 実験において $[-1.0, 1.0]$ の範囲で正規化されている. 本研究の実験では, 各アルゴリズムにおける収束率 (%) も考察されている. 収束率は, 学習における最大反復回数 k_{max} 内で収束し, 勾配がオーバーシュートせずに解を得た割合を示す. 学習終了条件は, 1 つ目の関数近似問題と 2 つの分類問題では $\epsilon = 10^{-6}$, そして 2 つ目の関数近似問題および 2 つのマイクロ波回路のモデリング問題では $\epsilon = 10^{-8}$ とした. 最大反復回数 $k_{max} = 150,000$ とし, ステップサイズ α_k は直線探索と Armijo 基準を用いて最適化を行う [20, 21, 34]. MSQN および (Ada)MoQ に対する Armijo の条件式を (66) と (101) に示す.

MSQN における Armijo の条件式:

$$E(\mathbf{w}_k + \alpha_k \mathbf{g}_k) \leq E(\mathbf{w}_k) + \chi \alpha_k \nabla E(\mathbf{w}_k)^{\mathrm{T}} \mathbf{g}_k. \tag{66}$$

(Ada)MoQ における Armijo の条件式:

$$E(\mathbf{w}_k + \mu_k \mathbf{v}_k + \alpha_k \mathbf{g}_k) \leq E(\mathbf{w}_k + \mu_k \mathbf{v}_k) + \chi \alpha_k \{(1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}^{\mathrm{T}} \mathbf{g}_k. \tag{101}$$

ここで, $0 < \chi < 1$ であり, 本研究では $\chi = 10^{-3}$ とした. $\mathbf{g}_k$ は探索方向ベクトルであり, MSQN そして (Ada)MoQ はそれぞれ $\mathbf{H}_k^{\mathrm{MSQN}} \nabla E(\mathbf{w}_k)$ と $\mathbf{H}_k^{\mathrm{(Ada)MoQ}} \{(1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}$ である.

4.3.1　関数近似問題 1

本研究では, (Ada)MoQ の有効性を調べるため, 3.2.1 節の (68) に示す関数近似問題のモデリングを行った [68, 80].

$$f(x, a, b) = 1 + (x + 2x^2)\sin(ax^2 + b). \tag{68}$$

この関数では $b = 0$ とし, 入力 x と a のそれぞれの範囲は $x \in [-4, 4)$ および $a \in [-1, 1)$ とした. $|T_r|$ と $|T_e|$ はそれぞれ 3,320 個の学習データポイントと 6,600 個のテストデータポイントを含む. この問題における NN の構造は 2-55-1 である. また, 経験に基づき AdaGrad, RMSprop, AdaDelta および Adam の学習率をそれぞれ $\eta_k = 0.01, 0.1, 1.0$ および 0.1 とした. シミュレーションの結果を表 7 に示す. 表 7 より, 本実験で対象としている問題が, BP, CM, (opt)NAG, AdaGrad, RMSprop および AdaDelta などの 1 次手法にとって, 最大反復回数 k_{max} 内で十分に小さな学習誤差が得られないことがわかる. 一方, Adam においては, 学習誤差は減少するが, 反復回数が多いため準ニュートン法に基づく全ての 2 次手法よりも計算時間を必要とする. 2 次手法の比較において, 慣性項を用いた手法である (Ada)NAQ と (Ada)MoQ は, QN と MSQN と比較して, 少ない反復回数で高速な学習を可能にしている. つまり, 慣性項が準ニュートン法に基づくアルゴリズムにおいて学習の収束を高速化させる有効な手法であることがわかる. 提案手法 MoQ と NAQ の比較の観点では, MoQ の学習誤差, 反復回数およびテスト誤差は NAQ と類似している. 一方, 計算時間においては, MoQ は NAQ よりも高速に収束している. つまり, (75) は $\mathbf{w}_k + \mu_k \mathbf{v}_k$ における勾配の適切な近似であると結論付けられる. 従って, MoQ は NAQ における勾配計算回数の削減と学習の高速化に成功したと言える. 一方, 学習の収束率において, 大きな固定値の $\mu_k = 0.9, 0.95$ を持つ MoQ は不安定である. つまり, 勾配が誤差を減少できないため, $\mathbf{w}$ の初期値に対する収束率が 100% 未満となる. ただし, この問題は適応的 μ_k を用いた AdaMoQ において改善され, μ_k の値が増加

表 7: 関数近似問題 (68) に対する MoQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Convergence Rate(%)
GD	-	42.31 / 42.28 / 42.23 / 42.31	410	150,000	41.34 / 41.33 / 41.26 / 41.35	-
CM	0.8	38.07 / 39.04 / 32.80 / 42.35	234	120,430	37.16 / 38.11 / 31.97 / 41.38	40
	0.85	33.78 / 36.85 / 32.71 / 42.49	200	97,401	32.89 / 91.71 / 31.88 / 599.12	40
	0.9	38.11 / 36.60 / 24.28 / 42.49	201	100,179	37.20 / 35.72 / 23.52 / 41.52	40
	0.95	42.34 / 39.50 / 27.74 / 42.49	63	34,585	41.44 / 40.48 / 26.95 / 56.96	80
NAG	0.8	42.50 / 42.27 / 40.44 / 42.49	93	52,311	41.53 / 17337.0/ 39.46 / 102448.7	70
	0.85	42.49 / 42.49 / 42.47 / 42.49	0.04	18	41.75 / 16252.9 / 41.52 / 90521.6	100
	0.9	42.48 / 42.48 / 42.45 / 42.49	0.04	18	331.4 / 12374.2 / 41.52 / 82422.4	100
	0.95	42.49 / 42.48 / 42.45 / 42.49	0.03	15	111.4 / 4107.3 / 41.52 / 35249.4	100
OptNAG	-	32.85/ 34.92 / 32.76 / 41.40	372	150,000	32.02 / 34.04 / 31.93 / 40.43	-
AdaGrad	-	39.72 / 39.42 / 35.05 / 42.21	251	150,000	38.74 / 38.44 / 34.06 / 41.24	-
AdaDelta	-	76.50 / 72.12 / 42.38 / 92.93	250	150,000	70.27 / 71.16 / 41.42 / 91.98	-
RMSprop	-	33.24 / 33.12 / 32.24 / 33.34	250	150,000	32.40 / 32.28 / 31.41 / 32.49	-
Adam	-	3.92 / 6.83 / 1.80 / 25.17	253	150,000	3.67 / 6.57 / 1.72 / 24.49	-
QN	-	0.453 / 0.483 / 0.319 / 0.830	257	94,212	0.401 / 0.454 / 0.303 / 0.765	100
MSQN	-	0.620 / 0.662 / 0.408 / 1.21	214	78,489	0.593 / 0.618 / 0.385 / 1.10	100
NAQ	0.8	0.567 / 0.861 / 0.336 / 3.85	192	44,348	0.534 / 0.822 / 0.317 / 3.70	100
	0.85	0.432 / 0.547 / 0.362 / 1.17	165	38,171	0.407 / 0.517 / 0.341 / 1.10	100
	0.9	0.393 / 0.382 / 0.223 / 0.500	132	30,427	0.372 / 0.361 / 0.213 / 0.473	100
	0.95	0.521 / 4.26 / 0.291 / 37.85	**102**	**23,546**	0.494 / 4.14 / 0.274 /36.95	100
AdaNAQ	-	0.442 / 0.459 / 0.268 / 0.679	120	27,701	0.421 / 0.434 / 0.254 / 0.611	**100**
MoQ	0.8	0.434 / 0.469 / 0.308 / 0.726	118	43,168	0.414 / 0.447 / 0.293 / 0.722	100
	0.85	0.516 / 3.01 / 0.327 / 26.06	119	43,640	0.407 / 2.94 / 0.308 / 25.55	100
	0.9	0.421 / 0.525 / 0.349 / 0.995	**75**	**27,516**	0.399 / 0.489 / 0.327 / 0.902	90
	0.95	0.471 / 0.730 / 0.328 / 2.66	80	29,204	0.433 / 0.639 / 0.313 / 2.55	80
AdaMoQ	-	0.460 / 0.457 / 0.220 / 0.842	**74**	**27,046**	0.435 / 0.428 / 0.206 / 0.773	**100**

することによって, MoQ は高速な収束速度を失うことがなくなった. 加えて, AdaMoQ でも AdaNAQ の慣性項による効果が維持されつつ, その反復時間の削減に成功していることが表からわかる.

この実験では, モデリングの精度を測定するため, AdaMoQ によって学習されたニューラルモデルの出力をテストデータと比較し, それを図 34 に示した. 入力 a は $a = -1$ とし, ニューラルモデルはテスト誤差がもっとも小さな値, $E_{test}(\mathbf{w}) = 0.220 \times 10^{-3}$ とした. 図 34 より, AdaMoQ のニューラルモデルがテストモデルを高精度にモデリングできていることが確認できる.

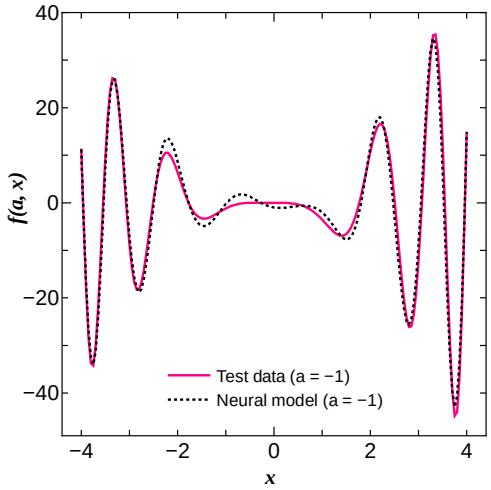


図 34: 関数 (68) に対する AdaMoQ の NN モデルとテストモデルの比較

4.3.2　関数近似問題 2: Levy Function

次に, (102) に示す Levy 関数 ($\mathrm{R}^d \to \mathrm{R}^1$) を用いてシミュレーションを行う. Levy 関数は最適化問題のベンチマーク問題として用いられる多峰性関数である [83].

$$\begin{aligned} f(x_1 \dots x_d) = \frac{\pi}{d}\Bigg\{ &\sum_{l=1}^{d-1}[(x_l - 1)^2(1 + 10\sin^2(\pi x_{l+1}))] \\ &+10\sin^2(\pi x_1) + (x_d - 1)^2\Bigg\}, x_l \in [-4, 4], \forall l. \end{aligned} \tag{102}$$

ここで, d は入力の次元を示す. 図 35 に (102) の Levy 関数, $d = 2$ 次元の関数を示す. この図より, Levy 関数が非常に非線形な入出力特性を持つことがわかる. 本研究では, 問題の非線形性を更に増加させるため, 入力ベクトル x の次元を $d = 5$ に設定した. 入力 $\mathbf{x}_i, i \in 1, \dots, 5$ を $[-4, 4]$ の間の一様乱数で学習データとして生成し, その学習データ数は $|T_r| = 5,000$ とした. この問題では, 出力は $f(x_1 \dots x_d)$ であり, 50 個のニューロン数を持つ 1 層の中間層を含む NN を用いて学習を行った. 従って, NN の構造は 5-50-1 となる. Levy 関数のシミュレーション結果を表 8 に示す. 今後, シミュレーションの比較において前節で GD, CM, (Opt)NAG, AdaGrad, RMSprop および AdaDelta が学習誤差を減少できなかったため, 比較の対象外とする. 表 8 より, Adam は学習誤差を減少させているが, 最大反復回数 k_{max} までに学習が続くため, 計算時間を長く必要とすることがわかる. このため, Adam の収束率は “-” として示した. 2 次手法の比較において, QN, MSQN, NAQ および MoQ は十分に誤差が減少している. ここで, NAQ ($\mu_k = 0.95$) と MoQ ($\mu_k = 0.9$ と 0.95) の誤差の平均値および最大値は他の

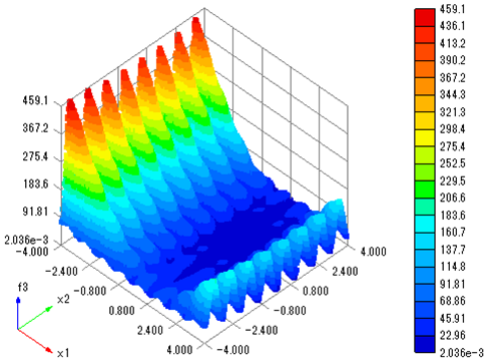


図 35: Levy 関数 $f(x_1, x_2)$

アルゴリズムと比較して大きくなっている. 一方で, 最小値は同等の誤差を得ている. つまり, MoQ は固定の μ_k の 2 つの値で学習が不安定であるため, MoQ の固定の μ_k は NAQ と比較して慎重に選択する必要がある. これに対して AdaNAQ と AdaMoQ は安定して, 中央, 平均, 最小と最大の誤差を減少できている点から, 適応的 μ_k は学習を安定化させ, 複数の試行において重み $\mathbf{w}$ の初期値に対するロバスト性を向上できることが分かる. 計算時間の高速化に関しては, 前節と同様の結果が得られたことから, (Ada)MoQ は有効な学習アルゴリズムであることを示すことができた.

表 8: Levy 関数に対する MoQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Iteration counts (k)	Convergence Rate(%)
Adam	-	0.075 / 0.075 / 0.049 / 0.099	698	150,000	-
QN	-	0.004 / 0.004 / 0.002 / 0.009	90	14,983	100
MSQN	-	0.099 / 0.099 / 0.099 / 0.099	43	8,360	100
NAQ	0.8	0.005 / 0.005 / 0.003 / 0.012	63	5,787	100
	0.85	0.004 / 0.005 / 0.003 / 0.011	59	5,301	100
	0.9	0.005 / 0.006 /0.003 / 0.018	56	5,024	100
	0.95	0.004 / 3.07 / 0.002 / 30.72	53	4,691	100
AdaNAQ	-	0.007 / 0.006 / 0.003 / 0.010	48	4,748	100
MoQ	0.8	0.007 / 0.086 / 0.003 / 0.807	**36**	5,902	100
	0.85	0.004 / 0.015 / 0.003 / 0.113	**34**	5,454	100
	0.9	0.007 / 1.82 / 0.003 / 18.09	**29**	**4,748**	100
	0.95	0.679 / 5.15 / 0.004 / 51.38	**29**	**4,637**	100
AdaMoQ	-	0.006 / 0.007 / 0.004 / 0.010	**33**	5,739	100

4.3.3　マイクロ波回路モデリング問題１: microstrip Low-Pass Filter

次に, 提案手法 (Ada)MoQ を 2.4 節および 3.2.2 節で用いた microstrip Low-Pass Filter (LFP) [34, 69–71] の NN モデルの開発に応用する. この問題では, 入力層のニューロン数は周波数 f と幅 D の 2 つであり, 出力層のニューロン数は S パラメータの大きさ $|\mathbf{S}_{11}|$ と $|\mathbf{S}_{21}|$ の 2 つである. 従って, NN の構造は, 45 個のニューロン数を持つ中間層 1 層を含めて, 2-45-2 となる. さらに, このシミュレーションでは, モーメント係数 μ_k の影響を詳細に調べるため, 固定の μ_k の範囲を $\mu_k = 0.7$ および 0.75 まで広げ, 比較を行った. LPF のシミュレーション結果を表 9 に示す. 前の実験と同様, 1 次手法は最大反復回数 k_{max} 内で, 十分に小さな誤差を取得できなかったため, 比較の対象外とした. 表 9 より, この問題でも QN に基づく学習アルゴリズムでは慣性項が高速化に強く影響していることがわかる. ただし, μ_k が固定である NAQ と MoQ では, μ_k の増加に伴い学習速度も高速化するが, 重み $\mathbf{w}$ の初期値に対する不安定さも増加する. MoQ は, NAQ と同等の最小誤差を得られるが, MoQ は誤差を減少できずに収束してしまう結果を得ることがある. このため, MoQ における収束率は μ_k の増加に伴い低下する. この傾向は非線形性が非常に高い (強非線形) 問題におけるモーメント法ではよく見られる傾向であり, 慣性項の影響でモーメント係数が大きい場合, 特に, 反復の初期において勾配がオーバーシュートしてしまうことに起因する. この実験では, オーバーシュートした勾配の学習を停止させるため, $\|\nabla E(\mathbf{w}_k)\| > 10^3$ となったら学習を終了させている. 一方, 適応的 μ_k を用いた AdaNAQ と AdaMoQ では, 学習の不安定さに対する問題を克服し, 学習の収束率を 100% まで向上させることができた. さらに, AdaMoQ は適応的慣性係数を用いた手法の中で, 最も高速なアルゴリズムであることが表よりわかる. 結果, 提案手法の AdaMoQ は, 計算速度を犠牲にすることなく重み $\mathbf{w}$ の初期値に対してロバストな学習を行うことができるアルゴリズムであることを示した. 従って, 誤差関数を 2 次関数と仮定し, ネステロフの加速勾配を通常勾配の重み付きの線形和に近似することで NAQ の性能を維持しつつ, さらに, 学習の収束速度も高速化されることが結論付けられる. この結果は, 図 36 および図 37 よりも明白にわかる. ここで, 図 36 は QN, MSQN, AdaNAQ そして AdaMoQ によって学習された NN の最小の学習誤差における誤差と反復回数の変化を示すグラフである. 図 36 より, AdaMoQ が AdaNAQ の性能を維持できていることが見受けられる. 一方, 学習誤差と時間の変化グラフを示す図 37 では, 提案手法 AdaMoQ が最も高速なアルゴリズムであることがわかる.

このシミュレーションでは, モデリングの精度を測定するため, AdaMoQ によって学習された NN の最小テスト誤差である $E_{test}(\mathbf{w}) = 0.472 \times 10^{-3}$ のモデルを, テストデータ

D = 13, 15, 17, 19 mm と比較し, そのグラフを図 38〜41 に示す. 図 38〜41 より, NN のモデルとテストデータが良好な一致を示していることがわかり, これは高精度なモデリングができていることを示す.

表 9: LPF に対する MoQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Convergence Rate(%)
Adam	-	6.84 / 6.84 / 5.66 / 7.97	115	150,000	5.03 / 4.98 / 3.85 / 5.86	-
QN	-	0.772 / 0.763 / 0.627 / 0.911	120	106,287	0.640 / 0.901 / 0.534 / 2.01	90
MSQN	-	0.708 / 0.746 / 0.624 / 0.969	110	96,938	0.555 / 0.900 / 0.430 / 2.65	100
NAQ	0.7	0.720 / 0.727 / 0.607 / 0.904	126	66,133	0.819 / 1.87 / 0.466 / 8.94	100
	0.75	0.677 / 0.698 / 0.608 / 0.856	113	59,100	0.759 / 0.743 / 0.406 / 1.07	100
	0.8	0.792 /0.791 / 0.570 / 1.08	89	46,861	0.962 / 8.02 / 0.435 / 65.32	100
	0.85	0.729 / 0.742 / 0.526 / 0.994	95	49,823	0.922 / 1.24 / 0.406 / 3.18	100
	0.9	0.684 / 0.670 / 0.499 / 0.908	84	44,149	0.683 / 1.10 / 0.442 / 3.60	100
	0.95	0.661 / 4.09 / 0.571 / 22.30	43	22,338	1.20 / 3.79 / 0.525 /19.60	80
AdaNAQ	-	**0.596 / 0.615 / 0.486 / 0.739**	64	33,715	0.652 / 0.667 / 0.464 / 1.12	**100**
MoQ	0.7	0.659 / 0.660 / 0.599 / 0.710	81	72,090	0.749 / 0.965 / 0.566 /2.18	80
	0.75	0.753 / 0.740 / 0.626 / 0.826	43	38,350	0.797 / 0.819 / 0.539 /1.31	70
	0.8	0.677 / 0.649 / 0.569 / 0.727	38	34,361	0.703 / 0.819 / 0.482 / 1.60	50
	0.85	0.677 / 1.63 / 0.635 / 5.17	23	20,742	0.988 / 7.35 /0.586 / 33.30	50
	0.9	0.660 / 0.667 / 0.641 / 0.705	16	14,262	0.647 / 0.636/ 0.488 / 0.761	40
	0.95	0.567 / 0.604 / 0.545 / 0.735	14	12,847	0.798 / 0.713 / 0.578 / 0.870	40
AdaMoQ	-	**0.675 / 0.615 / 0.472 / 0.751**	**31**	**27,627**	0.798 / 1.14 / 0.591 / 2.15	**100**

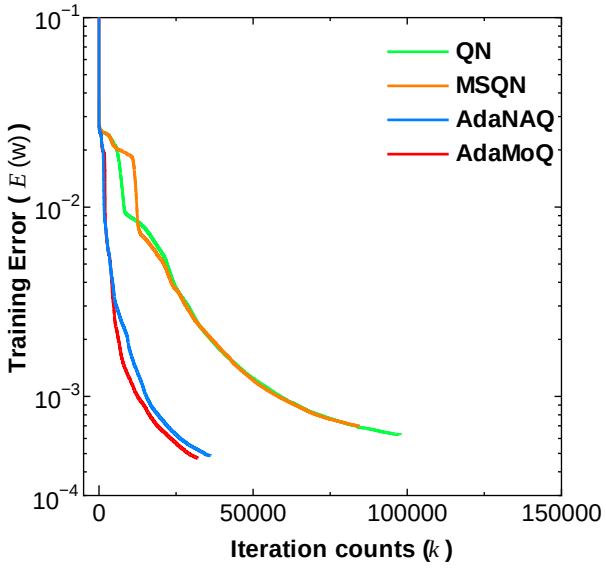


図 36: LPF の学習誤差と反復回数グラフ

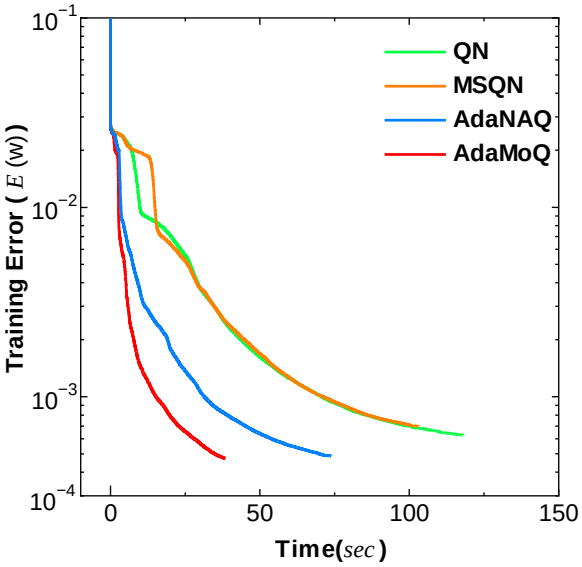


図 37: LPF の学習誤差と時間グラフ

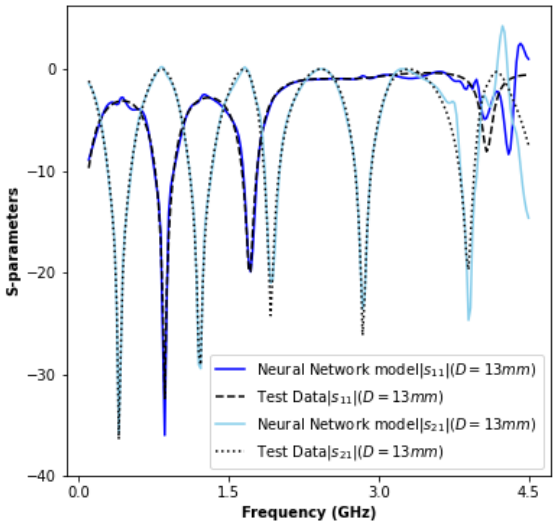


図 38: LPFD = 13 mm に対する AdaMoQ の NN モデルとテストデータの比較

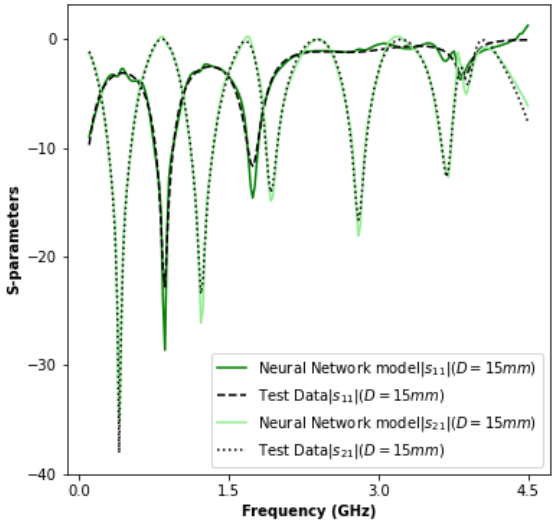


図 39: LPFD = 15 mm に対する AdaMoQ の NN モデルとテストデータの比較

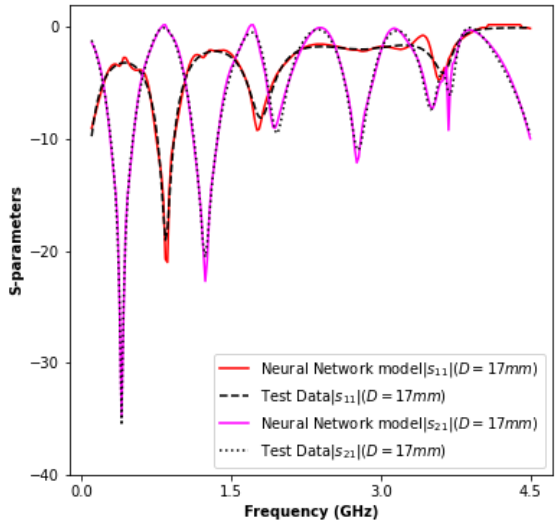


図 40: LPFD = 17 mm に対する AdaMoQ の NN モデルとテストデータの比較

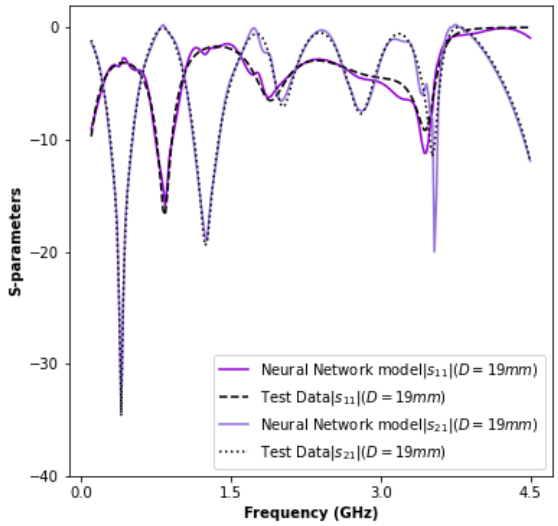


図 41: LPFD = 19 mm に対する AdaMoQ の NN モデルとテストデータの比較

4.3.4　マイクロ波回路モデリング問題 2: Microstrip Patch Antenna

2 つ目のマイクロ波回路モデリング問題として, 3.2.3 節にて示した図 30 の長方形型 Microstrip Patch Antenna(MPA) [81] を考える. 3.2.3 節で紹介したように, この問題の入力は, 長さ L, 幅 W, 周波数 f およびマイクロストリップフィードの相対距離 (x/L) である R によって示される. 2 つの出力は, S パラメータ $|S_{11}|$ の実数部と虚数部に割り当てられる. NN の構造は 4-50-150-2 であり, $|T_r|$ と $|T_e|$ に対してそれぞれ 10,000 と 890 個のサンプルを全サンプルデータからランダムに分別した. MPA にはロバスト性の高い学習アルゴリズムが必要とされる. 従って, この実験では, ロバスト性が高い AdaNAQ と AdaMoQ のみの学習を行い, 比較を行った. MPA モデリング問題の結果を表 10 に示す. 表より, 両アルゴリズムが高い収束率で小さなテストと学習誤差を得られていることが示されている. さらに, 反復回数においても両手法, 同様である. これは, AdaNAQ と AdaMoQ によって学習された NN の最小の学習誤差における誤差と反復回数の変化グラフを示す図 42 よりわかる. 一方, 計算時間においては AdaMoQ が高速であり, これは誤差と学習全体の時間の変化グラフを示す図 43 より明白である. 従って, 提案手法 AdaMoQ は MPA のような大規模かつ強非線形問題に対して有効であると結論付けられる. さらに, 本研究の目的であった NAQ の高速化は, これまでの実験と同様に, NAQ の性能を維持しながら成功している.

表 10: MPA に対する AdaMoQ のシミュレーション結果

Algorithm	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Convergence Rate(%)
AdaNAQ	0.429 / 0.436 / 0.383 / 0.545	8,604	18,760	1.66 / 1.72 / 1.44 / 1.97	100
AdaMoQ	0.425 / 0.440 / 0.377 / 0.531	**4,919**	**18,573**	1.73 / 1.70 / 1.36 / 2.44	100

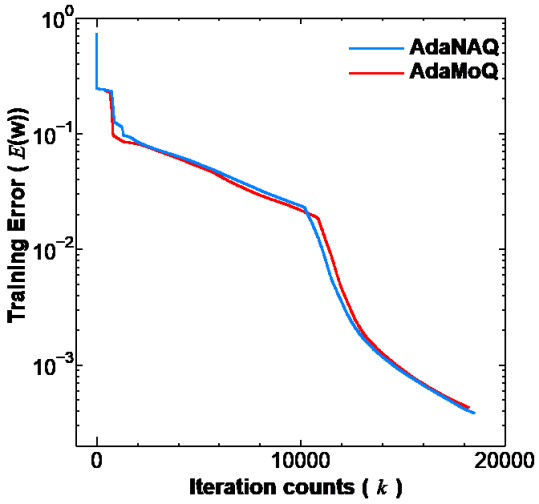


図 42: MPA の学習誤差と反復回数グラフ

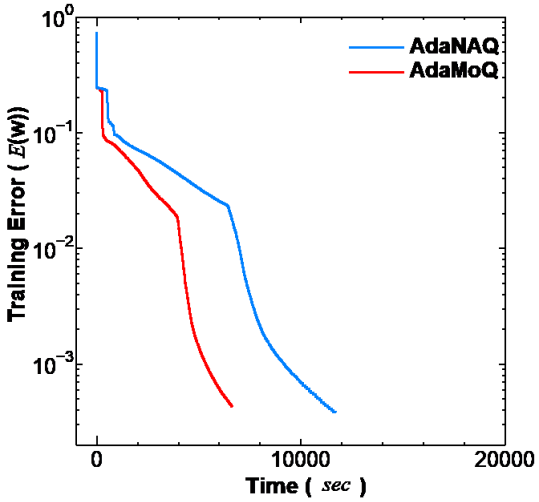


図 43: MPA の学習誤差と時間グラフ

4.3.5　分類問題 1: 8×8 MNIST handwritten digit dataset

これまでの実験では, 1 次手法が最大反復回数以内に収束できない, 強非線形問題を検討した. しかし, この実験で用いる分類問題では, 1 次手法, 特に Adam が有効であり, その有効性は様々な文献で示されている [1, 9, 19, 48, 49]. 本実験では, 提案手法 MoQ の評価を行うための最初の分類問題として, 図 44 にデータのサンプルを示す 8×8 ピクセルの手書き数字文字 MNIST のデータセット [84–86] を考える. このデータセットは, 28×28 ピクセルの手書き数字文字 MNIST [1, 7, 87] のデータセットを縮小したものである. 8×8 MNIST は 1,797 個のサンプルデータを持つ. 本実験では, ランダムに全データサンプルの 75% (1,347 サンプル) を $|T_r|$ そして 25% (450 サンプル) を $|T_e|$ として分割した [85]. NN の入力と出力はそれぞれ画像データのピクセル数 (次元数) 64 とクラス数 10 である. 本実験ではヘッセ行列の計算が必要である MSQN, QN, NAQ および MoQ を比較アルゴリズムとして利用するため, 大規模なネットワークを用いた学習は困難である. 従って, 10 個のニューロン数を持つ中間層 1 層からなるもっともシンプルなネットワーク構造, つまり, 64-10-10 を使用する. 提案手法の性能を比較するアルゴリズムとして, 1 次手法の中でもっとも有効なアルゴリズムである Adam と 2 次手法を用いた. さらに, NAQ と MoQ のモーメント係数は適応的モーメント係数 μ_k に設定した. 一般的にこの問題はミニバッチ (確率的) 学習法に使用されているが, 本研究の提案手法はバッチ学習に基づいているため, 全てのアルゴリズムにバッチ学習を適応した. Adam のハイパーパラメータに関しては [19] の推奨値とした. シミュレーション結果を表 11 に示す. 表より, 強非線形問題の学習では最大反復回数以内に収束できなかった Adam の収束率は 100% であることがわかる. さらに, Adam の学習精度の中央値が全てのアルゴリズムの中でもっとも高い 100% である. Adam のテスト誤差に関しては, 他の手法と同様であったが, バッチ学習では, 2 次手法と比較して, 多くの反復回数と学

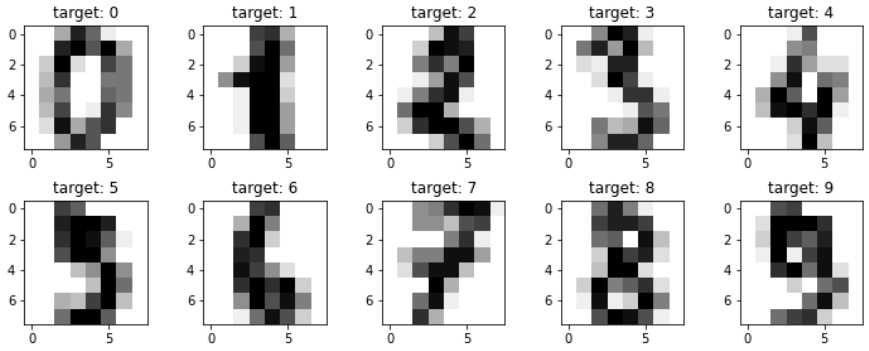


図 44: 8×8 MNIST 手書き数字文字のデータセットサンプル

表 11: 8 × 8 MNIST に対する AdaMoQ のシミュレーション結果

Algorithm	μ_k	$Accuracy_{train}(\%)$ Median / Ave. / Best / Worst	Time (*sec*)	Iteration counts (k)	$Accuracy_{test}(\%)$ Median / Ave. / Best / Worst	Convergence rate (%)
Adam	-	100 / 99.97 / 100 / 99.92	14	5,481	95.88 / 95.80 / 96.44 / 95.11	100
QN	-	99.92 / 99.90 / 99.92 / 99.77	4.03	1,165	96.22 / 96.13/ 96.88 / 95.55	100
MSQN	-	99.92 / 99.91 / 99.92 / 99.77	5.13	1,211	96.11 / 96.15 / 97.11 / 95.55	100
AdaNAQ	-	99.70 / 99.70 / 99.85 / 99.48	0.46	73	95.22 / 95.13 / 95.77 / 93.77	100
AdaMoQ	-	99.70 / 99.72 / 99.85 / 99.62	**0.27**	72	95.33 / 95.11 / 96.00 / 93.33	100

習時間を必要とした. 2 次手法の比較では, AdaNAQ と AdaMoQ が QN と MSQN に近い学習とテスト精度を得ることができている. しかし, QN と MSQN は AdaNAQ と AdaMoQ に比べて, より多くの反復回数と学習時間を必要とした. また, 提案手法 AdaMoQ は全てのアルゴリズムの中でもっとも高速に収束することがわかる. これらの考察は, 学習誤差と反復回数, 学習誤差と時間, テスト精度と反復回数そしてテスト精度と時間のグラフである図 45 〜48 からも明らかである. 本実験では, グラフを見やすくするため, 図 45 と 46 の y– 軸と x– 軸の表記はそれぞれ対数と線形とした. 一方で, 図 47 と 48 では, y– 軸と x– 軸はそれぞれ線形と対数として設定した. 図 45 と 47 より, AdaNAQ と AdaMoQ は反復回数の早い段階で収束し, その性能はほぼ同様であることがわかる. 図 46 と 48 から, 提案手法 AdaMoQ は AdaNAQ を時間軸に対して高速化していることがわかる. 従って, MoQ における (75) の近似が関数モデリングで使用した (15) の MSE 誤差と同様に (19) の CE に対しても有効であることが結論付けられる.

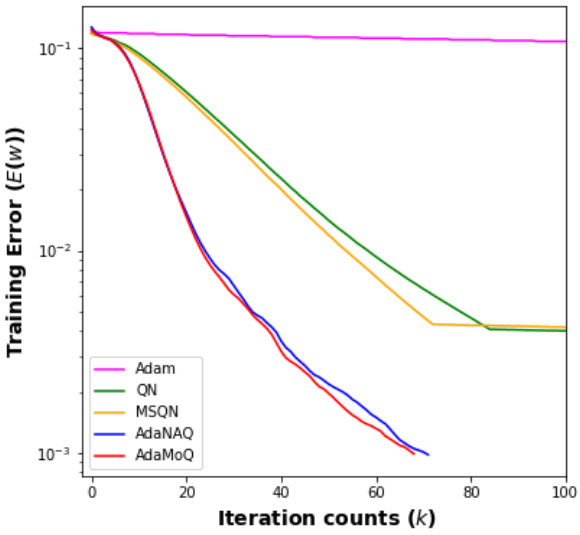


図 45: 8 × 8 MNIST の学習誤差と反復回数グラフ

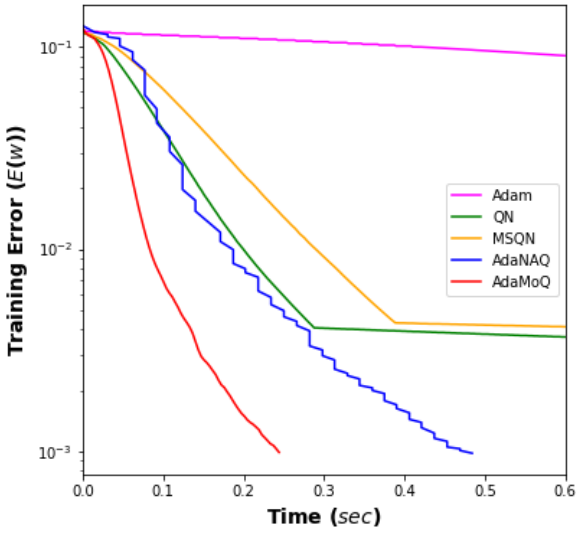


図 46: 8 × 8 MNIST の学習誤差と時間グラフ

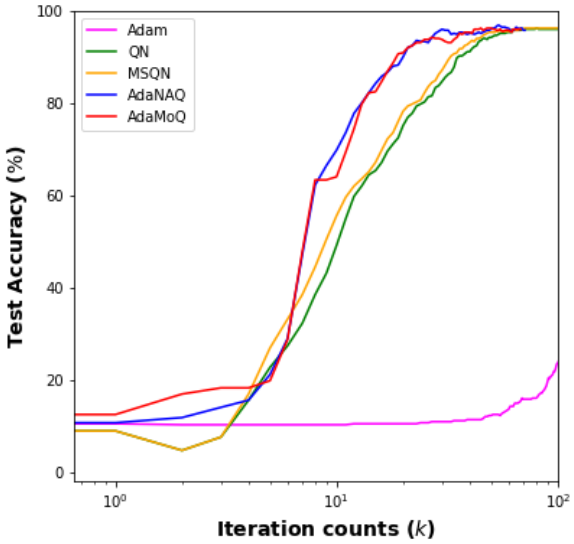


図 47: 8 × 8 MNIST のテスト精度と反復回数グラフ

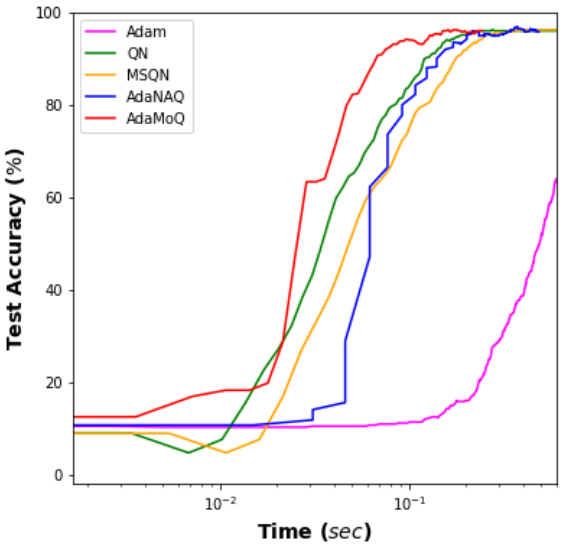


図 48: 8×8 MNIST のテスト精度と時間グラフ

4.3.6　分類問題 2: Three-Spirals dataset

最後の分類問題として, 図 49 に示すように螺旋特性を持つデータポイントが 3 つのクラス (青, 緑そして赤) に分類される Three-Spirals 問題である [88]. この問題は, NN の学習において比較的複雑な分類ベンチマーク問題として知られている. ネットワークの入力は x と y の座標であり, 出力はクラスの数 $L = 3$ である. 各クラスは 350 個のデータを持ち, $|T_r| = 1,050$ である. NN の構造は 2-10-3 とした. シミュレーション結果を表 12 に示す. 表より, 全ての

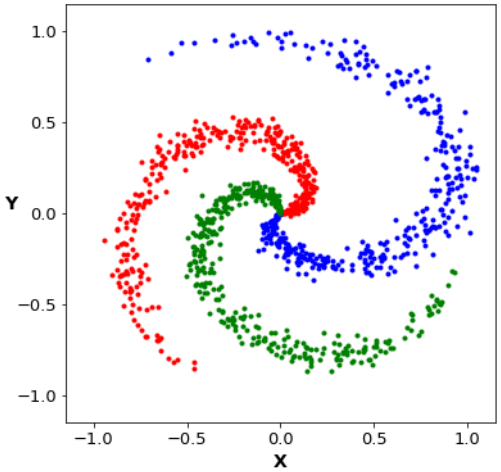


図 49: Three-Spirals のデータセット構造

表 12: Three-Spirals に対する AdaMoQ のシミュレーション結果

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (*sec*)	Iteration counts (k)	$\text{Accuracy}_{train}(\%)$ Median / Ave. / Best / Worst	Convergence rate (%)
Adam	-	0.853 / 0.853 / 0.827 / 0.890	16	55,910	99.80 / 99.80 / 99.80 / 99.80	100
QN	-	0.827 / 0.827 / 0.813 / 0.842	4.03	8,830	99.80 / 99.80 / 99.80 / 99.80	100
MSQN	-	0.829 / 0.832 / 0.815 / 0.852	3.90	8,584	99.80 / 99.80 / 99.80 / 99.80	100
AdaNAQ	-	0.645 / 0.639 / 0.596 / 0.666	0.60	778	99.80 / 99.80 / 99.80 / 99.80	100
AdaMoQ	-	0.620 / 0.632 / 0.596 / 0.712	**0.305**	**742**	99.80 / 99.80 / 99.80 / 99.80	100

アルゴリズムがほぼ同じ学習誤差を得ることができ, 100% 収束していることがわかる. さらに, 精度も同じである. 分類問題 1 の 8×8 MNIST と同様に, Adam は準ニュートン法に基づくアルゴリズムと比較して, より多くの反復回数と時間を必要とする. 2 次手法の比較では, AdaMoQ がもっとも速く, もっとも少ない反復回数で収束している. この結果は, 学習誤差と反復回数, 学習誤差と時間, 学習精度と反復回数そして学習精度と時間のグラフを示した図 50〜53 からも明らかである. なお, 図 50 と 51 の y– 軸と x– 軸は前の問題と同様にそれぞれ対数と線形表示であり, 図 52 と 53 では線形と対数表示である. 図 50 と 52 から, AdaMoQ は AdaNAQ とほぼ同様の性能で収束していることがわかる. また, 図 51 と 53 よりは, AdaMoQ が AdaNAQ を時間軸で高速化していることがわかる.

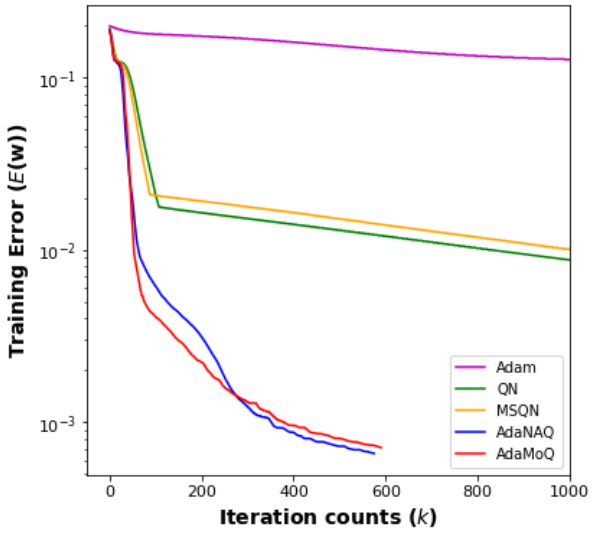


図 50: Three-Spirals の誤差と反復回数グラフ

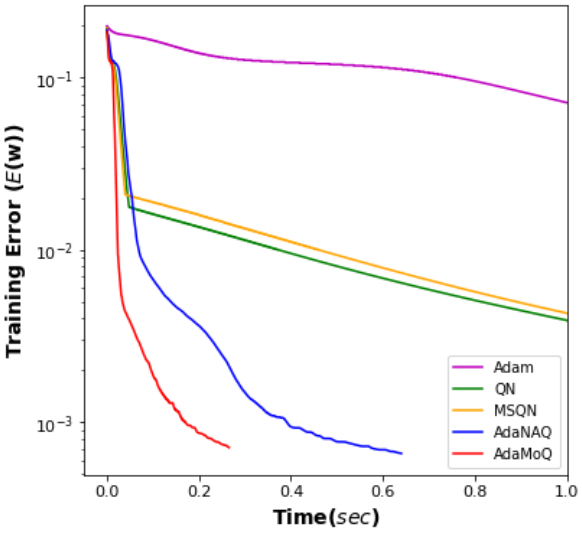


図 51: Three-Spirals の誤差と時間グラフ

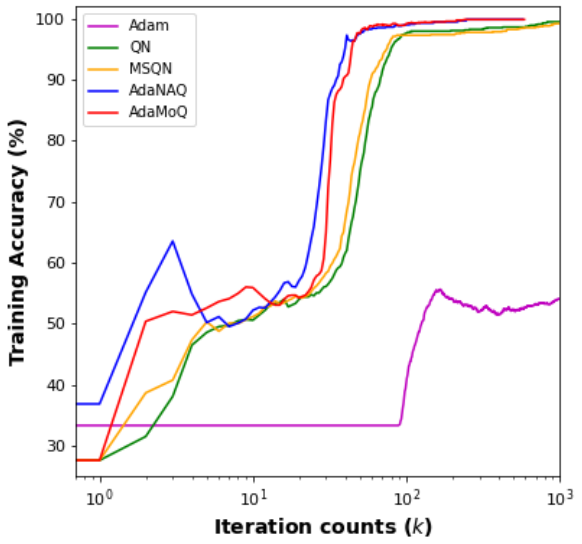


図 52: Three-Spirals の精度と反復回数グラフ

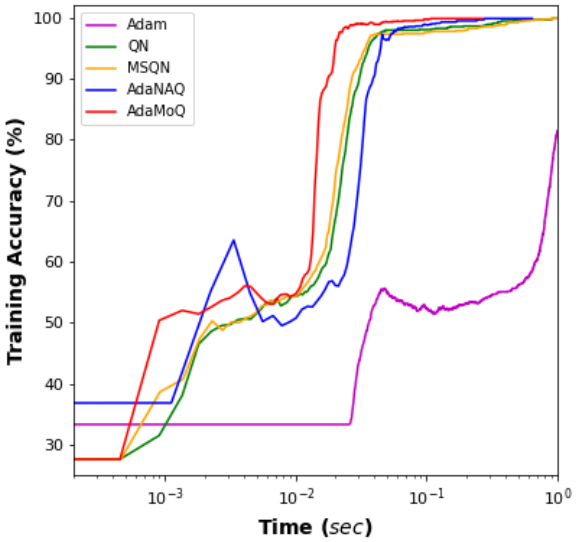


図 53: Three-Spirals の精度と時間グラフ

このシミュレーションでは, 学習精度を測定するため, Adam と AdaMoQ によって学習された NN におけるテストデータの分類を調べた. テスト結果を図 54 と 55 に示す. これらの図から, 学習済み NN はいずれもオーバーフィットすることなく, 2 次元平面テストデータに対して良好に分類できることがわかる.

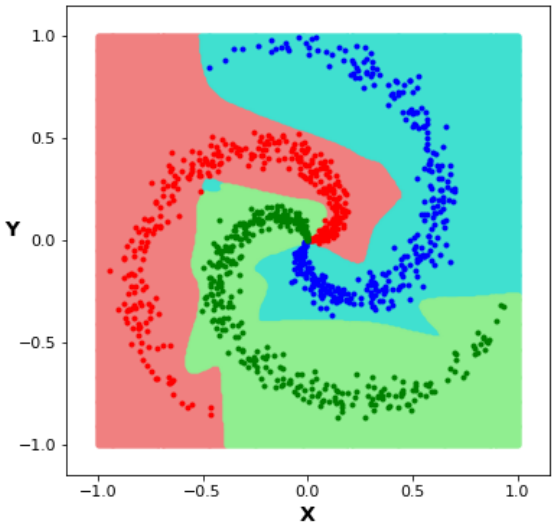


図 54: Three-Spirals の Adam による検証結果

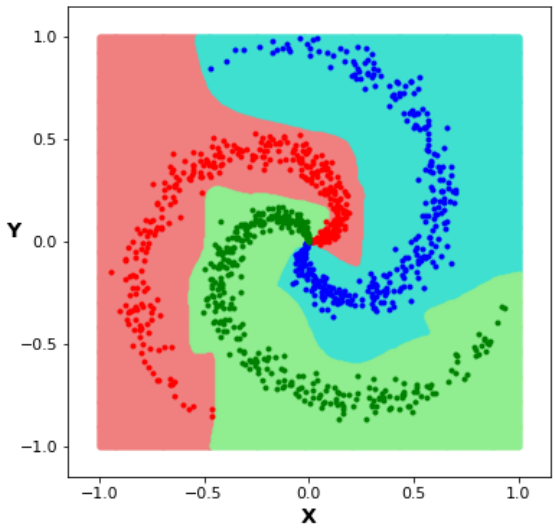


図 55: Three-Spirals の AdaMoQ による検証結果

4.4　まとめ

本章では, 本研究の 2 つ目の課題 1 つであった NAQ [32–34] の問題点, 計算時間の増加に着目し, その解決策として慣性項を用いた新しい準ニュートン法, 慣性付準ニュートン法 (Momentum Quasi-Newton method, MoQ) を提案した. 提案手法は, NAQ で使用されているネステロフの加速勾配を誤差関数が 2 次関数と仮定し, 通常勾配の重み付の線形和として近似した. 提案手法の性能を強非線形問題である 2 つの関数近似問題と 2 つのマイクロ波回路モデリング問題に応用することで示した. 提案手法 MoQ は NAQ の近似手法であるにも関わらず, NAQ の性能を維持しながら, 高速化を実現した. さらに, MoQ に適応的慣性係数 μ_k を応用した手法, 適応的慣性付準ニュートン法 (Adaptive Momentum Quasi-Newton method, AdaMoQ) では, 重み $\mathbf{w}$ の初期値に対してロバストで安定した学習を実現できた. 本章では, 1 次手法が得意とする 2 つの分類問題に対しても提案手法の性能を調べた. 実験結果より, バッチ学習を行う NN において提案手法がもっとも効率的なアルゴリズムであると結論付けられる. しかし, 提案手法 AdaNAQ と (Ada)MoQ では, 重み $\mathbf{w}$ の更新にヘッセ行列の逆近似行列が必要となる. さらに, 本研究の提案手法はバッチ学習に基づいている. 従って, AdaNAQ および (Ada)MoQ の計算コストは高く, 大規模な問題の学習には適さない問題点がある. 従って, 計算コストを削減する改良を必要とする.

5　慣性付記憶制限準ニュートン法

本章では, 本研究の 3 つ目の課題である, NAQ および MoQ の計算コストに着目し, 解決策を提案する. 最初に提案する手法は NAQ および MoQ に記憶制限手法 (Limited-Memory strategy) を導入した, 記憶制限ネステロフの加速準ニュートン法 (Limited-Memory Nesterov's Accelerated Quasi-Newton method, LNAQ) [41–43] と記憶制限慣性付準ニュートン法 (Limited-Memory Momentum Quasi-Newton method, LMoQ) [44] である. 2 次手法ではヘッセ行列が使用されているため, 大規模なデータセットおよび NN のバッチ学習においては, 重み $\mathbf{w}$ の更新に必要な計算コストは $O(d^2)$ と高価になる. 従って, 使用するメモリ量や計算コストの点で学習が困難となる. この問題点を克服するため, 準ニュートン法に対して記憶制限手法が提案された [20–22, 26]. この手法はヘッセ行列の次元数を任意の値に制限し, 重み $\mathbf{w}$ の更新を行う手法である. 従って, 2 次手法の強非線形問題に対する有効性をある程度維持しながらも, 計算コストを大幅に削減することが可能となる. 本研究では, 記憶制限手法を NAQ と MoQ に導入することで, NAQ と MoQ の計算コストを削減しつつも, 従来手法の記憶制限準ニュートン法 (Limited-Memory Quasi-Newton method, LQN) の高速化を目指す. 提案手法を 2 つの関数近似問題と 1 つのマイクロ波回路のモデリング問題に対する NN の学習に応用し, 従来手法と比較することで, その有効性を示す.

5.1　Limited-Memory Quasi-Newton method

記憶制限準ニュートン法 (Limited-memory Quasi-Newton method, LQN) は QN におけるメモリの使用量を削減する目的で導出された手法である [21, 22, 26]. 具体的には, QN における (56) の $\mathbf{H}_{k+1}^{\mathrm{QN}}$ 行列に必要なメモリを大幅に削減すると共に, 計算量の削減を行うことで, 次元の大きな問題に対応する学習を可能にした手法である. LQN における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{LQN}}$ を用いて (103) に示す.

$$\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k^{\mathrm{LQN}}, \tag{103}$$

$$\mathbf{g}_k^{\mathrm{LQN}} = -\mathbf{H}_k^{\mathrm{LQN}} \nabla E(\mathbf{w}_k). \tag{104}$$

ここで, $\mathbf{H}_k^{\mathrm{LQN}}$ 行列の更新式は以下のように展開される.

$$\mathbf{H}_{k+1}^{\mathrm{LQN}} = \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}\right)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{LQN}} \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}\right) + \left(\frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}\right), \tag{105}$$

$$= \mathbf{G}_k^{\mathrm{T}} \mathbf{H}_k^{\mathrm{LQN}} \mathbf{G}_k + \frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}. \tag{106}$$

ここで, $\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k$ と $\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k)$ であり, $\mathbf{G}_k$ は

$$\mathbf{G}_k = \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}\right), \tag{107}$$

であるため, $\mathbf{H}_k^{\mathrm{LQN}}$ の更新式は (108) として表される.

$$\mathbf{H}_k^{\mathrm{LQN}} = \mathbf{G}_{k-1}^{\mathrm{T}} \mathbf{H}_{k-1}^{\mathrm{LQN}} \mathbf{G}_{k-1} + \frac{\mathbf{s}_{k-1} \mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}} \mathbf{y}_{k-1}}. \tag{108}$$

ここで (108) を (105) に代入すると,

$$\mathbf{H}_{k+1}^{\mathrm{LQN}} = \mathbf{G}_k^{\mathrm{T}} \mathbf{G}_{k-1}^{\mathrm{T}} \mathbf{H}_{k-1}^{\mathrm{LQN}} \mathbf{G}_{k-1} \mathbf{G}_k \mathbf{G}_k^{\mathrm{T}} \frac{\mathbf{s}_{k-1} \mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}} \mathbf{y}_{k-1}} \mathbf{G}_k + \frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}, \tag{109}$$

を得る. これを $k = 1$ まで繰り返すことで, $\mathbf{H}_{k+1}^{\mathrm{LQN}}$ は以下の様に変形される.

$$\begin{aligned} \mathbf{H}_{k+1}^{\mathrm{LQN}} \simeq\ & (\mathbf{G}_1 \dots \mathbf{G}_{k-1} \mathbf{G}_k)^{\mathrm{T}} \mathbf{H}_1^{\mathrm{LQN}} (\mathbf{G}_1 \dots \mathbf{G}_{k-1} \mathbf{G}_k) \\ & + (\mathbf{G}_2 \dots \mathbf{G}_{k-1} \mathbf{G}_k)^{\mathrm{T}} \frac{\mathbf{s}_1 \mathbf{s}_1^{\mathrm{T}}}{\mathbf{s}_1^{\mathrm{T}} \mathbf{y}_1} (\mathbf{G}_2 \dots \mathbf{G}_{k-1} \mathbf{G}_k) + \dots \\ & + (\mathbf{G}_{k-1} \mathbf{G}_k)^{\mathrm{T}} \frac{\mathbf{s}_{k-2} \mathbf{s}_{k-2}^{\mathrm{T}}}{\mathbf{s}_{k-2}^{\mathrm{T}} \mathbf{y}_{k-2}} (\mathbf{G}_{k-1} \mathbf{G}_k) + \\ & \mathbf{G}_k^{\mathrm{T}} \frac{\mathbf{s}_{k-1} \mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}} \mathbf{y}_{k-1}} \mathbf{G}_k + \frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}. \end{aligned} \tag{110}$$

ただし, $\mathbf{H}_1^{\mathrm{LQN}}$ は正定値対称な初期行列である. このように式変形すると, QN では k 回の反復すべてを記憶しているとみなすことができる. ここで, LQN では, $\mathbf{H}_{k+1}^{\mathrm{LQN}}$ の記憶量を m 反

復前までとすることで導出できる. (110) を m 反復前までの記憶に制限すると,

$$\begin{aligned}\mathbf{H}_{k+1}^{\mathrm{LQN}} \simeq (\mathbf{G}_{k-m+1}\ldots\mathbf{G}_{k-1}\mathbf{G}_k)^{\mathrm{T}}\mathbf{H}_{k^1}^{\mathrm{LQN}}(\mathbf{G}_{k-m+1}\ldots\mathbf{G}_{k-1}\mathbf{G}_k)\\ +(\mathbf{G}_{k-m+2}\ldots\mathbf{G}_{k-1}\mathbf{G}_k)^{\mathrm{T}}\frac{\mathbf{s}_{k-m+1}\mathbf{s}_{k-m+1}^{\mathrm{T}}}{\mathbf{y}_{k-m+1}^{\mathrm{T}}\mathbf{s}_{k-m+1}}(\mathbf{G}_2\ldots\mathbf{G}_{k-m+2}\mathbf{G}_k)\\ +\ldots+(\mathbf{G}_{k-1}\mathbf{G}_k)^{\mathrm{T}}\frac{\mathbf{s}_{k-2}\mathbf{s}_{k-2}^{\mathrm{T}}}{\mathbf{y}_{k-2}^{\mathrm{T}}\mathbf{s}_{k-2}}(\mathbf{G}_{k-1}\mathbf{G}_k)+\\ \mathbf{G}_k^{\mathrm{T}}\frac{\mathbf{s}_{k-1}\mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{s}_{k-1}}\mathbf{G}_k+\frac{\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}}\mathbf{s}_k}.\end{aligned} \tag{111}$$

となる. ただし, $\mathbf{H}_{k^1}^{\mathrm{LQN}}$ は正定値対称な対角行列とする. このとき, 行列 $\mathbf{G}_k$ は対角行列とベクトルで構成されていることを考慮すると, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{LQN}} = -\mathbf{H}_k^{\mathrm{LQN}}\nabla E(\mathbf{w}_k)$ は行列を用いる必要はなくなり, ベクトルの内積のみの計算で求められる. 従って, $d\times d$ 次元の $\mathbf{H}_k^{\mathrm{LQN}}$ 行列を記憶する必要が無くなり, $2\times m$ 本の $\mathbf{w}$ と同じ次元 (d) のベクトルを記憶するだけとなる. LQN のアルゴリズムを Algorithm 14 に示す. また, 記憶制限手法の探索方向ベクトルを求めるためのアルゴリズム, Two-Loop recursion を Algorithm 13 に示す [21, 22, 26].

Algorithm 13 Limited-Memory Quasi-Newton method (LQN)

Require: ϵ, k_{max}
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$
1: $k = 1$
2: Calculate $\nabla E(\mathbf{w}_1)$;
3: **while** ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
4: Calculate direction vector of $\mathbf{g}_k^{\mathrm{LQN}}$ using Algorithm 14;
5: Calculate stepsize α_k using Armijo's condition;
6: Update $\mathbf{v}_{k+1} = \alpha_k\mathbf{g}_k^{\mathrm{LQN}}$;
7: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
8: Calculate $\nabla E(\mathbf{w}_{k+1})$;
9: $k = k + 1$;
10: **end while**
Return: $\mathbf{w}_k$

Algorithm 14 Direction Vector of LQN (Two-Loop recursion)

Require: m

Calculate $\mathbf{g}_k^{\mathrm{LQN}} = -\nabla E(\mathbf{w}_k)$;
Calculate $\mathbf{s}_k$ and $\mathbf{y}_k$;
for $(i = k, k-1, ..., k - min(k, m-1))$ **do**
    $\phi_i = \mathbf{s}_i^{\mathrm{T}} \mathbf{g}_k^{\mathrm{LQN}} / \mathbf{s}_i^{\mathrm{T}} \mathbf{y}_i$;
    $\mathbf{g}_k^{\mathrm{LQN}} = \mathbf{g}_k^{\mathrm{LQN}} - \phi_i \mathbf{y}_i$;
end for
if $(k > 1)$ **then**
    $\mathbf{g}_k^{\mathrm{LQN}} = (\mathbf{y}_k \mathbf{s}_k{}^{\mathrm{T}} / \mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k) \mathbf{g}_k^{\mathrm{LQN}}$;
end if
for $(i = k - min(k, (m-1)), ..., k-1, k)$ **do**
    $\tau = \mathbf{y}_i^{\mathrm{T}} \mathbf{g}_k^{\mathrm{LQN}} / \mathbf{s}_i^{\mathrm{T}} \mathbf{y}_i$;
    $\mathbf{g}_k^{\mathrm{LQN}} = \mathbf{g}_k^{\mathrm{LQN}} - (\phi_i - \tau) \mathbf{s}_i$;
end for

Return: $\mathbf{g}_k^{\mathrm{LQN}}$

5.2　慣性項による記憶制限準ニュートン法の高速化

本節では, LQN の高速化および NAQ と MoQ の計算コストを削減するため, NAQ と MoQ に記憶制限手法を導入した, 記憶制限 NAQ と記憶制限 MoQ を提案する.

5.2.1　Limited-Memory Nesterov's Accelerated Quasi-Newton method

ここでは, QN に対して提案された記憶制限手法を NAQ に応用した, 記憶制限ネステロフの加速準ニュートン法 (Limited-Memory Nesterov's Accelerated Quasi-Newton method, LNAQ) を提案する. 提案手法は, NAQ の高速な学習の収束速度を保ちながら, 行列の計算量を削減することが可能となる. LNAQ は, LQN と同様な導出過程を持つ. 従って, LQN と同様に (112) に対して, m 反復前までの記憶に制限する.

$$\mathbf{H}_{k+1}^{\mathrm{LNAQ}} = \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{LNAQ}} \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right) + \left(\frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right), \tag{112}$$

$$= \hat{\mathbf{G}}_k^{\mathrm{T}} \mathbf{H}_k^{\mathrm{LNAQ}} \hat{\mathbf{G}}_k + \frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}, \tag{113}$$

$$\hat{\mathbf{G}}_k = \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}\right), \tag{114}$$

とおくと, m 反復までの $\mathbf{H}_{k+1}^{\mathrm{LNAQ}}$ の更新式

$$\begin{aligned}\mathbf{H}_{k+1}^{\mathrm{LNAQ}} \simeq\ & (\hat{\mathbf{G}}_{k-m+1}\ldots\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})^{\mathrm{T}}\mathbf{H}_{k^1}^{\mathrm{LNAQ}}(\hat{\mathbf{G}}_{k-m+1}\ldots\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})\\ &+(\hat{\mathbf{G}}_{k-m+2}\ldots\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})^{\mathrm{T}}\frac{\mathbf{p}_{k-m+1}\mathbf{p}_{k-m+1}^{\mathrm{T}}}{\mathbf{p}_{k-m+1}^{\mathrm{T}}\mathbf{q}_{k-m+1}}(\hat{\mathbf{G}}_{2}\ldots\hat{\mathbf{G}}_{k-m+2}\hat{\mathbf{G}}_{k})\\ &+\ldots+(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})^{\mathrm{T}}\frac{\mathbf{p}_{k-2}\mathbf{p}_{k-2}^{\mathrm{T}}}{\mathbf{p}_{k-2}^{\mathrm{T}}\mathbf{q}_{k-2}}(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})+\\ &\hat{\mathbf{G}}_{k}^{\mathrm{T}}\frac{\mathbf{p}_{k-1}\mathbf{p}_{k-1}^{\mathrm{T}}}{\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{q}_{k-1}}\hat{\mathbf{G}}_{k}+\frac{\mathbf{p}_{k}\mathbf{p}_{k}^{\mathrm{T}}}{\mathbf{p}_{k}^{\mathrm{T}}\mathbf{q}_{k}},\end{aligned}\tag{115}$$

を得る. ここで, $\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k\mathbf{v}_k)$ と $\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k)$ である. LNAQ における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は探索方向ベクトル $\mathbf{g}_k^{\mathrm{LNAQ}}$ を用いて, (116) に示す.

$$\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \alpha_k\mathbf{g}_k^{\mathrm{LNAQ}},\tag{116}$$

$$\mathbf{g}_k^{\mathrm{LNAQ}} = -\mathbf{H}_k^{\mathrm{LNAQ}}\nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k).\tag{117}$$

LNAQ とその探索方向ベクトル $\mathbf{g}_k^{\mathrm{LNAQ}}$ のアルゴリズムをそれぞれ Algorithm 15 と 16 に示す. Algorithm 15 より, LNAQ も NAQ と同様に学習ループ内で 2 回の勾配計算, Step.3 と 8 のネステロフの加速勾配 $\nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k)$ と通常勾配 $\nabla E(\mathbf{w}_{k+1})$ が必要であることがわかる. これは, LQN と比較して, 各反復の計算時間の増加に繋がるが, 慣性項とネステロフの加速勾配の影響により LQN の高速化が期待できる [32–34].

Algorithm 15 Limited-Memory Nesterov's Accelerated Quasi-Newton method (LNAQ)

Require: ϵ, k_{max}, μ_k
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d, \mathbf{v}_1 = 0$
1: $k = 1$
2: **while** ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
3: Calculate $\nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k)$;
4: Calculate Direction Vector of $\mathbf{g}_k^{\mathrm{LNAQ}}$ using Algorithm 16;
5: Calculate stepsize α_k using Armijo's condition;
6: Update $\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \alpha_k\mathbf{g}_k^{\mathrm{LNAQ}}$;
7: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
8: Calculate $\nabla E(\mathbf{w}_{k+1})$;
9: $k = k + 1$;
10: **end while**
Return: $\mathbf{w}_k$

Algorithm 16 Direction Vector of LNAQ (Two-Loop recursion)

Require: m

Calculate $\mathbf{g}_k^{\mathrm{LNAQ}} = -\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$;
Calculate $\mathbf{p}_k$ and $\mathbf{q}_k$;
for $(i = k, k-1, ..., k - min(k, m-1))$ **do**
    $\phi_i = \mathbf{p}_i^{\mathrm{T}} \mathbf{g}_k^{\mathrm{LNAQ}} / \mathbf{p}_i^{\mathrm{T}} \mathbf{q}_i$;
    $\mathbf{g}_k^{\mathrm{LNAQ}} = \mathbf{g}_k^{\mathrm{LNAQ}} - \phi_i \mathbf{q}_i$;
end for
if $(k > 1)$ **then**
    $\mathbf{g}_k^{\mathrm{LNAQ}} = (\mathbf{q}_k \mathbf{p}_k{}^{\mathrm{T}} / \mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k) \mathbf{g}_k^{\mathrm{LNAQ}}$;
end if
for $(i = k - min(k, (m-1)), ..., k-1, k)$ **do**
    $\tau = \mathbf{q}_i^{\mathrm{T}} \mathbf{g}_k^{\mathrm{LNAQ}} / \mathbf{p}_i^{\mathrm{T}} \mathbf{q}_i$;
    $\mathbf{g}_k^{\mathrm{LNAQ}} = \mathbf{g}_k^{\mathrm{LNAQ}} - (\phi_i - \tau) \mathbf{p}_i$;
end for

Return: $\mathbf{g}_k^{\mathrm{LNAQ}}$

5.2.2 Limited-Memory Momentum Quasi-Newton method

次に, LNAQ の欠点であった 1 反復の計算時間の増加に着目し, その解決策として記憶制限手法を MoQ に導入した慣性付記憶制限準ニュートン法 (Limited-Memory Momentum Quasi-Newton method, LMoQ) を提案する. LMoQ も LNAQ および LQN と同様な導出過程を持つ.

$$\mathbf{H}_{k+1}^{\mathrm{LMoQ}} = \left(\mathbf{I} - \frac{\hat{\mathbf{q}}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{q}}_k}\right)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{LMoQ}} \left(\mathbf{I} - \frac{\hat{\mathbf{q}}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{q}}_k}\right) + \left(\frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{q}}_k}\right), \tag{118}$$

$$= \hat{\mathbf{G}}_k^{\mathrm{T}} \mathbf{H}_k^{\mathrm{LMoQ}} \hat{\mathbf{G}}_k + \frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{q}}_k}, \tag{119}$$

$$\hat{\mathbf{G}}_k = \left(\mathbf{I} - \frac{\hat{\mathbf{q}}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{q}}_k}\right), \tag{120}$$

とおくと, m 反復までに記憶を制限した $\mathbf{H}_{k+1}^{\mathrm{LMoQ}}$ の更新式は

$$\begin{aligned}\mathbf{H}_{k+1}^{\mathrm{LMoQ}} \simeq\ & (\hat{\mathbf{G}}_{k-m+1}\ldots\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})^{\mathrm{T}}\mathbf{H}_{k^1}^{\mathrm{LMoQ}}(\hat{\mathbf{G}}_{k-m+1}\ldots\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})\\ &+(\hat{\mathbf{G}}_{k-m+2}\ldots\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})^{\mathrm{T}}\frac{\mathbf{p}_{k-m+1}\mathbf{p}_{k-m+1}^{\mathrm{T}}}{\mathbf{p}_{k-m+1}^{\mathrm{T}}\hat{\mathbf{q}}_{k-m+1}}(\hat{\mathbf{G}}_{2}\ldots\hat{\mathbf{G}}_{k-m+2}\hat{\mathbf{G}}_{k})\\ &+\ldots+(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})^{\mathrm{T}}\frac{\mathbf{p}_{k-2}\mathbf{p}_{k-2}^{\mathrm{T}}}{\mathbf{p}_{k-2}^{\mathrm{T}}\hat{\mathbf{q}}_{k-2}}(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_{k})+\\ &\hat{\mathbf{G}}_{k}^{\mathrm{T}}\frac{\mathbf{p}_{k-1}\mathbf{p}_{k-1}^{\mathrm{T}}}{\mathbf{p}_{k-1}^{\mathrm{T}}\hat{\mathbf{q}}_{k-1}}\hat{\mathbf{G}}_{k}+\frac{\mathbf{p}_{k}\mathbf{p}_{k}^{\mathrm{T}}}{\mathbf{p}_{k}^{\mathrm{T}}\hat{\mathbf{p}}_{k}}.\end{aligned}\tag{121}$$

となる. ここで, $\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k\mathbf{v}_k)$ と $\hat{\mathbf{q}}_k = \nabla E(\mathbf{w}_{k+1}) - \{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{w}_{k-1})\}$ である. LMoQ における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は探索方向ベクトル $\mathbf{g}_k^{\mathrm{LMoQ}}$ を用いて, (122) に示す.

$$\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \alpha_k\mathbf{g}_k^{\mathrm{LMoQ}},\tag{122}$$

$$\mathbf{g}_k^{\mathrm{LMoQ}} = -\mathbf{H}_k^{\mathrm{LNAQ}}\{(1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k\nabla E(\mathbf{w}_{k-1})\}.\tag{123}$$

本研究では, グローバル収束を保持するため, 全ての記憶制限手法に対して, 4.2.1 節で紹介した (98) のグローバル収束項を導入する. さらに, LNAQ および LMoQ の学習の安定性を向上させるため, 3 章で提案した適応的慣性係数 μ_k を導入する. LMoQ とその探索方向ベクトル $\mathbf{g}_k^{\mathrm{LMoQ}}$ のアルゴリズムをそれぞれ Algorithm 17 と 18 に示す.

Algorithm 17 Limited-Memory Momentum Quasi-Newton method (LMoQ)

Require: ϵ, k_{max}, μ_k
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d, \mathbf{v}_1 = 0$
$k = 1$
Calculate $\nabla E(\mathbf{w}_1)$;
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
　Update adaptive μ_k using (64) and (65);
　Calculate Direction Vector of $\mathbf{g}_k^{\mathrm{LMoQ}}$ using Algorithm 18;
　Calculate stepsize α_k using Armijo's condition;
　Update $\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \alpha_k\mathbf{g}_k^{\mathrm{LMoQ}}$;
　Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
　Calculate $\nabla E(\mathbf{w}_{k+1})$;
　$k = k + 1$;
end while
Return: $\mathbf{w}_k$

Algorithm 18 Direction Vector of LMoQ (Two-Loop recursion)

Require: m

Calculate $\mathbf{g}_k^{\mathrm{LMoQ}} = -\{(1+\mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}$;
Calculate $\mathbf{p}_k$ and $\hat{\mathbf{q}}_k$;
for $(i = k, k-1, ..., k - min(k, m-1))$ **do**
    $\phi_i = \mathbf{p}_i^{\mathrm{T}} \mathbf{g}_k^{\mathrm{LMoQ}} / \mathbf{p}_i^{\mathrm{T}} \hat{\mathbf{q}}_i$;
    $\mathbf{g}_k^{\mathrm{LMoQ}} = \mathbf{g}_k^{\mathrm{LMoQ}} - \phi_i \hat{\mathbf{q}}_i$;
end for
if $(k > 1)$ **then**
    $\mathbf{g}_k^{\mathrm{LMoQ}} = (\hat{\mathbf{q}}_k \mathbf{p}_k{}^{\mathrm{T}} / \mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{q}}_k) \mathbf{g}_k^{\mathrm{LMoQ}}$;
end if
for $(i = k - min(k, (m-1)), ..., k-1, k)$ **do**
    $\tau = \hat{\mathbf{q}}_i^{\mathrm{T}} \mathbf{g}_k^{\mathrm{LMoQ}} / \mathbf{p}_i^{\mathrm{T}} \hat{\mathbf{q}}_i$;
    $\mathbf{g}_k^{\mathrm{LMoQ}} = \mathbf{g}_k^{\mathrm{LMoQ}} - (\phi_i - \tau)\mathbf{p}_i$;
end for

Return: $\mathbf{g}_k^{\mathrm{LMoQ}}$

5.3　計算コスト

本節では, 2 次手法において記憶制限手法を用いない場合 (通常) と用いた場合の計算コストとメモリ量の比較を行う. それぞれの計算コストとメモリ量を表 13 に示す. 表では, 勾配評価コストは nd とし, n は学習サンプル数, d はパラメータの次元である. ヘッセ行列の逆近似行列の更新コストを d^2 とする. 一方で, 記憶制限手法では, 任意のメモリ量 m までヘッセ行列のコストを削減できるため, $4md + 2d$ とした. 表の全てのアルゴリズムでは, ステッ

表 13: 記憶制限手法の計算コストとメモリ量の比較

	Algorithm	Computational Cost	Storage
Normal	QN	$nd + d^2 + \zeta nd$	$d^2 + 2(n+1)d$
	NAQ	$2nd + d^2 + \zeta nd$	$d^2 + 2(n+1)d$
	MoQ	$nd + d^2 + \zeta nd$	$d^2 + (3n+2)d$
Limited-Memory	LQN	$nd + 4md + 2d + \zeta nd$	$2md + 2(n+1)d$
	LNAQ	$2nd + 4md + 2d + \zeta nd$	$2md + 2(n+1)d$
	LMoQ	$nd + 4md + 2d + \zeta nd$	$2md + (3n+2)d$

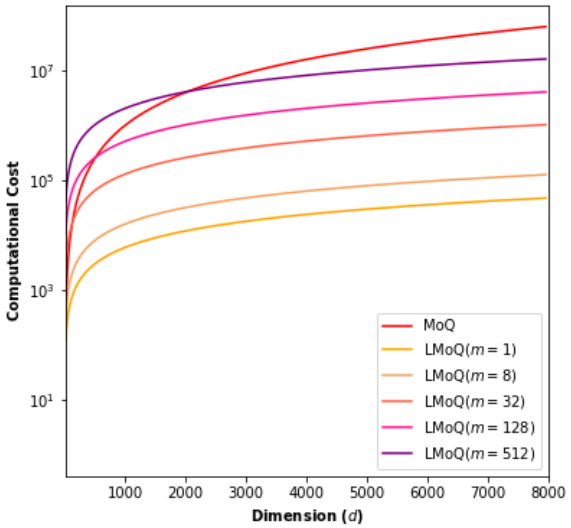


図 56: 2 次手法と記憶制限手法の計算コストの比較

プサイズは直線探索法により決定されるため, 探索条件を満たすまで ζ 回関数評価を行う. 通常と記憶制限手法の双方において, NAQ は QN と MoQ が各反復で勾配を 1 回計算するのに比べ, 勾配を 2 回計算している. 従って, NAQ は追加の計算コスト nd を持つ. さらに, QN と MoQ は各反復あたりの勾配計算回数が 1 回であるため, 同じ計算コストを持つと考えられる. 一方で, 保存されるメモリ量に関しては, 通常の QN, NAQ および MoQ では, 過去の

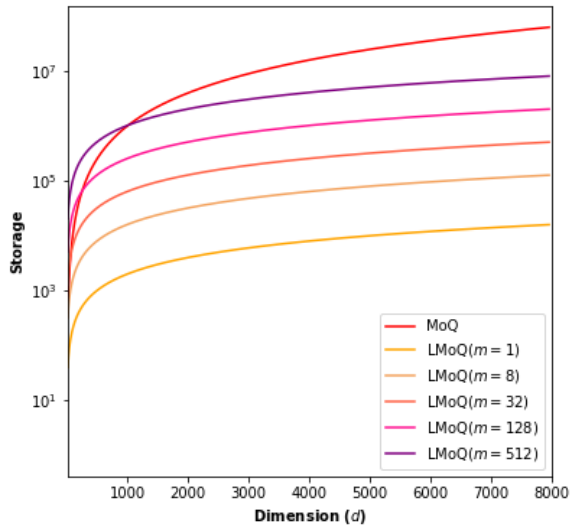


図 57: 2 次手法と記憶制限手法のメモリ量の比較

ヘッセ行列を保持するため, d^2 のメモリが必要である. しかし記憶制限手法では, 任意の記憶量 m まで削減するため, ヘッセ行列の保持に必要なメモリ量は $2md$ となる. 一方で, QN, NAQ, LQN そして LNAQ では $\mathbf{s}_k$, $\mathbf{y}_k$, $\mathbf{p}_k$ そして $\mathbf{q}_k$ を計算する必要があるため, $2(n+1)d$ のメモリ量が必要である. ただし, MoQ と LMoQ における $\hat{\mathbf{q}}_k$ の計算では, ネステロフの加速勾配の近似を行っているため, 過去の勾配を 1 つ多く記憶する必要がある. 従って, メモリ量は $(3n+2)d$ となる.

ここで, 本研究で使用する最小のパラメータ次元 $d = 31$ を持つ問題から最大のパラメータ次元 $d = 7,960$ を持つ問題まで, 次元 d の値を少しずつ増加させ, 次元 d に対する MoQ と LMoQ ($m = 1, 8, 32, 128, 512$) の計算コストとメモリ量の関係性を図 56 と 57 に示す. グラフを見やすくするため, 図 56 と 57 の y– 軸と x– 軸の表記はそれぞれ対数と線形としたと共に, 全アルゴリズムの計算コストおよびメモリ量で共通している項を除外した. 図 56 と 57 より, 記憶量 m の増加に伴い, LMoQ の計算コストとメモリ量も増加していることがわかる. さらに, 次元 d の増加と共に MoQ が LMoQ よりも計算コストとメモリ量を必要とする. 従って, 記憶制限手法は通常の 2 次手法の計算コストとメモリ量を削減できていると結論付けられる.

5.4　実験

本研究では, 提案手法 LNAQ と LMoQ の有効性を示すため, 2 つの関数近似問題と 1 つのマイクロ波回路モデリング問題に対するシミュレーションを行った. シミュレーションでは, 任意のニューロン数を持つ中間層を含めた階層型 NN を用いた. 各ニューロンの活性化関数は (16) のシグモイド関数とし, 誤差関数は (15) の MSE に設定した. LNAQ および LMoQ の性能を, GD, CM, NAG, optNAG, AdaGrad, AdaDelta, RMSprop, Adam, QN, AdaNAQ, AdaMoQ および LQN の従来手法と比較し, 示す. シミュレーションに用いる全ての例題において, 10 回の異なる重み $\mathbf{w}$ の初期値に対して学習を行い, $\mathbf{w}$ は $[-0.5, 0.5]$ 内の一様乱数で初期化される. 入力の各要素および T_r と T_e は, 実験において $[-1.0, 1.0]$ の範囲で正規化されている. CM と NAG に用いられる固定のモーメント係数は $0.8 \leq \mu_k \leq 0.95$ の 0.05 刻みとした [14, 34]. また, LNAQ と LMoQ における慣性項のモーメント係数には適応的慣性係数を用いた. 記憶制限手法の記憶量 m は d 次元以下の 2 の累乗に設定した. AdaGrad, RMSprop, AdaDelta および Adam における各ハイパーパラメータは, それぞれの論文で推奨値とされている値に設定した [16–19]. さらに, 本研究では強非線形関数の NN によるモデリングを対象としているため, 全てのシミュレーションでバッチ学習を行う [79]. 学習された各 NN の結果は,

$E(\mathbf{w})$ の中央, 平均, 最小および最大値に加え平均の計算時間 (*sec*), 平均の反復毎の計算時間 (*sec*) そして平均の反復回数 (*k*) によって各表で示されている. ステップサイズ α_k は直線探索と Armijo の条件を用いて最適化を行う [20, 21, 34].

5.4.1　関数近似問題 1

最初に, LNAQ と LMoQ の有効性を調べるため, (68) に示す関数近似問題のモデリングを行った [68, 80].

$$f(x, a, b) = 1 + (x + 2x^2)\sin(ax^2 + b). \tag{68}$$

この問題では, $a = 1, b = 0$ とし, 入力 x の範囲は $x \in [-4, 4)$ とした. 従って, (68) を (124) のように表すことができる.

$$f(x) = 1 + (x + 2x^2)\sin(-x^2), \qquad |x| \leq 4, \tag{124}$$

ここで, 出力は $f(x)$ であり, 学習サンプル数は $|T_r| = 400$, テストサンプル数 $|T_r| = 10,000$ である. 本実験における NN の構造は, 10 個のニューロン数を持つ中間層 1 層を含む 1-10-1 である. この実験では, 記憶制限手法とそのメモリ量別のパフォーマンスを調べるため, LQN, LNAQ および LMoQ のみの比較を行う. さらに, この問題では, パラメータの次元は $d = 31$ である. 従って, 記憶量 $m = 1, 2, 4, 8$ そして 16 とする. 終了条件は $\epsilon = 10^{-6}$ そして $k_{max} = 150,000$ とし, さらに, 各アルゴリズムの学習が収束できるように, $E(\mathbf{w}) \leq 10^{-3}$ 以下になったら学習を強制的に終了させた. この問題のシミュレーション結果を表 14 に示す. 表では, $E_{train,test}(\mathbf{w})$ の中央と平均値に加え, 平均の計算時間 (*sec*), 1 反復の計算時間 (*sec*)($\times 10^{-3}$) そして平均の反復回数 (*k*) が示されている. 本実験では全てのアルゴリズムに対して, 学習終了までに十分に小さな誤差を得た試行回数の割合を収束率 (%) として示した. 表より, 全ての記憶制限手法において, メモリ量 m の値が小さいと, 収束までに多くの反復回数を必要とするが, 1 反復における計算時間が短いことがわかる. 一方で, メモリ量 m の値が増加すると, 収束に必要な反復回数は減少するが, 1 反復の計算時間が増加する. LQN と慣性項を用いた記憶制限手法の比較では, 全てのメモリ量 m の値に対して, LNAQ と LMoQ は LQN の高速化に成功している. さらに, 通常の NAQ と MoQ の関係性は記憶制限手法を導入しても保たれることを各反復の計算時間からわかる. 一方で, メモリ量 m が小さいとき, 学習の安定性は低下していることがわかる. なお, LQN では, メモリ量 m の値が増加しても収束率は向上しないのに対して, 適応的慣性係数を用いた LNAQ と LMoQ では収束率

が向上していることがわかる. 従って, 2 次手法の高速化に対して効果的であった慣性項は, 記憶制限手法に対しても有効であると結論付けられる.

表 14: 関数 (124) に対する記憶量 m 毎のシミュレーション結果

Algorithm	m	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave.	Iteration counts (k)	Time (sec)	Per iteration time(sec)($\times 10^{-3}$)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave.	Convergence rate (%)
LQN	1	0.99 / 0.99	81,577	10.20	0.125	1.00 / 1.00	80
	2	0.99 / 0.99	39,260	5.39	0.137	0.99 / 0.99	70
	4	0.99 / 0.99	32,741	5.08	0.155	1.00 / 1.00	70
	8	1.00 / 1.00	29,342	5.63	0.191	1.00 / 1.00	60
	16	0.99 / 0.99	26,138	6.75	0.258	0.98 / 0.98	60
LNAQ	1	1.00 / 1.00	**30,704**	6.49	0.211	1.00 / 1.00	80
	2	0.99 / 1.03	**27,702**	5.99	0.211	1.00 / 1.03	60
	4	1.00 / 0.99	**11,388**	2.69	0.236	1.00 / 1.00	80
	8	1.00 / 0.99	**8,932**	2.43	0.272	1.00 / 1.00	100
	16	0.99 / 0.99	**5,525**	1.91	0.345	1.00 / 1.00	90
LMoQ	1	1.00 / 1.03	32,163	**4.44**	0.138	1.00 / 1.02	80
	2	1.00 / 1.00	**25,644**	**3.74**	0.145	1.00 / 1.00	70
	4	1.00 / 0.99	13,716	**2.25**	0.164	1.00 / 1.00	80
	8	1.00 / 1.00	8,972	**1.79**	0.199	1.00 / 1.00	90
	16	0.99 / 0.99	**5,222**	**1.41**	0.270	1.00 / 1.00	100

5.4.2　関数近似問題 2: Levy Function

次に, 4.3.2 節にて紹介した (102) に示す Levy 関数 ($\mathrm{R}^d \to \mathrm{R}^1$) を用いてシミュレーションを行う.

$$
\begin{aligned}
f(x_1 \dots x_d) = \frac{\pi}{d}\Bigg\{ &\sum_{l=1}^{d-1}[(x_l - 1)^2(1 + 10\sin^2(\pi x_{l+1}))] \\
&+10\sin^2(\pi x_1) + (x_d - 1)^2\Bigg\}, x_l \in [-4, 4], \forall l.
\end{aligned}
\tag{102}
$$

この問題では入力の次元 $d = 5$ である. 入力 $\mathbf{x}_d, d \in 1, \dots, 5$ を $[-4, 4]$ の間の一様乱数で学習データとして生成し, その学習データ数は $|T_r| = 5,000$ とした. この問題では, 出力は $f(x_1 \dots x_d)$ であり, 50 個のニューロン数を持つ 1 層の中間層を含む NN を用いて学習を行った. 従って, NN の構造は 5-50-1 であり, 重み $\mathbf{w}$ の次元は $d = 351$ となる. 従って, 記憶量 $m = 1, 2, 4, 8, 16, 32, 64, 128$ そして 256 とする. この問題では 20,000 反復以内に十分小さな誤差を得ることができるため, $k_{max} = 20,000$ とし, 終了条件は $\epsilon = 10^{-8}$ に設定した. 記憶量 m 毎の LQN, LNAQ そして LMoQ と他の従来手法の比較結果を図 58 と 59 そして表 15 に

示す. 図 58 と 59 では, 図を見やすくするため, GD, OptNAG, AdaGrad, AdaDelta, RMSprop, Adam, LQN($m = 8$), LNAQ($m = 8$) そして LMoQ($m = 8$) のみを比較対象とし, それぞれ 10 回のシミュレーションの中でもっとも小さい誤差を得た解のみを表示する. 表より, 最大反復回数 $k_{max} = 20,000$ 内で 1 次手法は誤差の平均および中央値で十分に小さな値を得られていないことがわかる. また, LQN ($m = 1$ と 2), LNAQ ($m = 1$) そして LMoQ ($m = 1$ と 4) でも曲率情報の影響が少ないため, 1 次手法と同様に誤差の平均および中央値で十分に小さな値を得られていない. しかし, 記憶量 m の増加に伴い, 曲率情報の影響が強くなるため誤差の平均および中央値も小さくなる. しかし, 記憶量 $m = 256$ までは最大反復回数以内に収束できないことが表よりわかる. 従って, LQN, LNAQ そして LMoQ が QN, NAQ そして MoQ のように最大反復回数以内に収束するためにはより大きなメモリ量 m を必要とする. 一方で, LQN, LNAQ および LMoQ の比較では, 通常の 2 次手法と同様に慣性項は記憶制限手法の高速化に強く影響している. そのため, LQN がより多くの反復で得られる誤差を LNAQ と LMoQ は速い段階で得ている. これは, 図 58 からも明白である. 一方で, 時間の比較では, LMoQ は勾配計算回数の観点で, LNAQ を高速化している. 図 59 より, LMoQ が少ない計算時間でより小さな誤差が得られていることがわかる. この実験でも前の実験と同様に, 慣性項を用いた記憶制限手法の LNAQ および LMoQ は LQN を高速化できたと結論付けられる. さらに, 記憶制限手法は 1 次手法と 2 次手法の狭間にあるアルゴリズムであるため, 2 次手法ほどの解の精度は得られないが, 1 次手法よりは高い精度の解を得られていることが表 15 よりわかる.

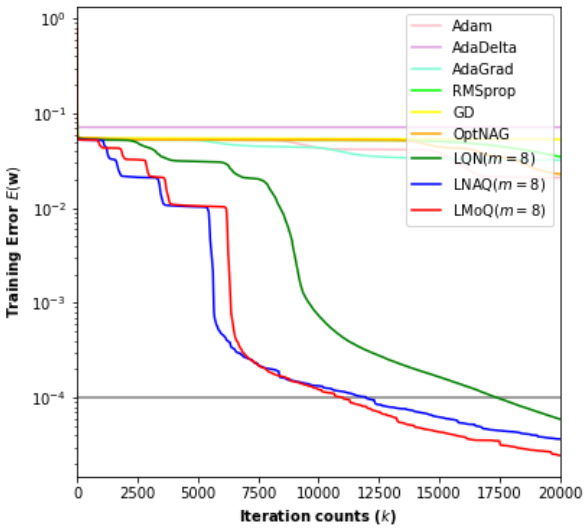


図 58: 記憶制限手法に対する Levy 関数の学習誤差と反復回数グラフ

表 15: Levy 関数に対する記憶量 m 毎のシミュレーション結果

Algorithm	μ_k	m	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Per iteration (sec)($\times 10^{-3}$)	Iteration counts (k)
GD	-	-	53.35 / 53.37 / 53.03/ 53.75	121	0.0060	20,000
CM	0.8	-	52.28 /51.67 / 41.21 / 54.83	87	5.4	16,022
	0.85	-	52.24 / 46.82 / 2.58 / 54.82	96	5.3	18,011
	0.9	-	39.19 / 34.70 / 0.260 / 54.83	84	5.2	16,035
	0.95	-	53.86 / 37.61 / 0.167 / 54.83	39	5.0	7,897
NAG	0.8	-	54.83 /54.19 / 48.46 / 54.83	50	5.0	10,011
	0.85	-	54.82 / 52.56 / 41.58 / 54.83	66	4.6	14,008
	0.9	-	54.83 / 54.83 / 54.82 / 54.83	9	4.6	2,017
	0.95	-	54.83 / 54.83 / 54.82 / 54.83	0.1	5.4	20
OptNAG	-	-	44.48 / 41.81 / 22.92 / 52.31	122	6.0	20,000
AdaGrad	-	-	34.44 / 37.59 / 32.06 / 51.58	74	3.7	20,000
AdaDelta	-	-	64.49 / 75.34 / 54.65 / 141.1	74	3.7	20,000
RMSprop	-	-	43.95 / 45.63 / 34.90 / 52.34	74	3.7	20,000
Adam	-	-	21.47 / 25.60 / 20.95 / 41.08	74	3.7	20,000
QN	-	-	0.004 / 0.004 / 0.002 / 0.009	90	6.0	14,983
AdaNAQ	-	-	0.007 / 0.006 / 0.003 / 0.010	48	10.1	4,748
AdaMoQ	-	-	0.006 / 0.007 / 0.004 / 0.010	33	5.7	5,739
LQN	-	1	32.54 / 35.06 / 26.39 / 42.88	100	5.0	20,000
	-	2	9.24 / 7.28 / 0.263 / 20.87	99	4.9	20,000
	-	4	0.172 / 1.21 / 0.095 / 10.31	100	5.0	20,000
	-	8	0.113 / 0.888 / 0.059 / 7.88	100	4.9	20,000
	-	16	0.092 / 0.088 / 0.042 / 0.145	105	5.2	20,000
	-	32	0.063 / 0.064 / 0.029 / 0.095	104	5.2	20,000
	-	64	0.034 / 0.043 / 0.026 / 0.086	111	5.5	20,000
	-	128	0.032 / 0.032 / 0.016 / 0.055	126	6.3	20,000
	-	256	0.025/ 0.030 / 0.016 / 0.070	154	7.7	20,000
LNAQ	-	1	10.87 / 12.79 / 0.179 / 31.83	174	8.6	20,000
	-	2	0.260 / 3.29 / 0.187 / 10.52	172	8.6	20,000
	-	4	0.115 / 0.123 / 0.092 / 0.203	177	8.8	20,000
	-	8	0.048 / 0.052 / 0.036 / 0.078	173	8.6	20,000
	-	16	0.012 / 0.012 / 0.0072 / 0.018	175	8.7	20,000
	-	32	0.0052 / 0.0091 / 0.0036 / 0.024	179	8.9	20,000
	-	64	0.014/ 0.015 / 0.0047 / 0.040	187	9.3	20,000
	-	128	0.013 / 0.015 / 0.0035 / 0.039	202	10.1	20,000
	-	256	0.036 / 0.048 / 0.0043 / 0.150	242	12.0	20,000
LMoQ	-	1	5.01 / 5.44 / 0.160 / 11.18	98	4.9	20,000
	-	2	0.237 / 3.22 / 0.198 / 10.37	97	4.8	20,000
	-	4	1.16 / 1.15 / 0.107 / 10.27	97	4.8	20,000
	-	8	0.039 / 0.046 / 0.024 / 0.100	98	4.9	20,000
	-	16	0.015/ 0.015 / 0.011 / 0.019	100	5.0	20,000
	-	32	0.0088 / 0.011 / 0.0038 / 0.021	104	5.1	20,000
	-	64	0.0086 / 0.011 / 0.0039 / 0.026	120	6.0	20,000
	-	128	0.017 / 0.029 / 0.0043 / 0.103	126	6.3	20,000
	-	256	0.023 / 0.033 / 0.0093 / 0.078	157	7.8	20,000

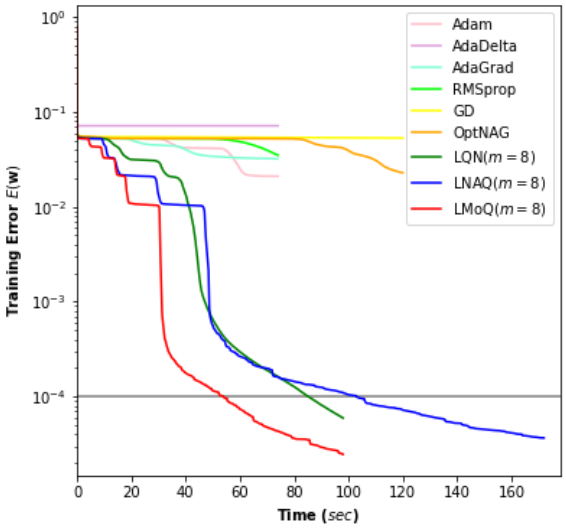


図 59: 記憶制限手法に対する Levy 関数の学習誤差と時間グラフ

5.4.3　マイクロ波回路モデリング問題：microstrip Low-Pass Filter

最後に, 提案手法 LNAQ と LMoQ を 2.4 節および 3.2.2 節で紹介した microstrip Low-Pass Filter (LFP) [34, 69–71] の NN モデルの開発に応用する. この問題では, 入力層のニューロン数は周波数 f と幅 D の 2 つであり, 出力層のニューロン数は S パラメータの大きさ $|\mathbf{S}_{11}|$ と $|\mathbf{S}_{21}|$ の 2 つである. NN の構造は, 45 個のニューロン数を持つ中間層 1 層を含めて, 2-45-2 となり, パラメータの次元は $d = 227$ となる. 従って, 記憶量 $m = 1, 2, 4, 8, 16, 32, 64$ そして 128 とする. 本実験では, マイクロ波回路のモデリング問題における記憶制限手法の効果を調査するため, モデリングに有効であった 2 次手法の QN, AdaNAQ そして AdaMoQ と記憶制限手法との比較を行う. 終了条件は $\epsilon = 10^{-8}$ に設定した. さらに, 4.3.3 節において, AdaNAQ と AdaMoQ は $k = 50,000$ 回未満の反復で収束したため, $k_{max} = 50,000$ とした. LPF のシミュレーション結果を表 16 に示す. 表より, 記憶制限手法では, 記憶量 m の増加に伴い, 学習誤差およびテスト誤差が小さくなっている. 特に, LNAQ($m = 128$) の最小のテスト誤差が AdaMoQ の最小値と同等であることがわかる. しかし, QN, LQN, LNAQ および LMoQ では, 最大反復回数以内で十分に小さい学習誤差を得られず, 収束しない. 従って, AdaNAQ と AdaMoQ と比較して, より多くの反復回数を必要とする. 2 次手法と記憶制限手法の時間の比較では, 記憶制限手法は 1 反復毎の計算時間が増加し, 全体の学習時間も増加している. 記憶制限手法における時間の増加の原因として, ステップサイズ α_k を更新する際, Armijo の条件を満たすため, 直線探索が多く行われることが挙げられる. LQN と LNAQ および LMoQ

の比較では, これまでの実験と同様に慣性項が高速化に影響している. しかし, 記憶制限手法は慣性項を導入しても, マイクロ波回路のモデリング問題のように入出の関係に強非線形特性を持つ問題に対しては, 通常の 2 次手法と比較して学習精度が低下することが結論付けられる.

表 16: LPF に対する記憶量 m 毎のシミュレーション結果

Algorithm	m	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst	Time (sec)	Per iteration (sec)($\times 10^{-3}$)	Iteration counts (k)	$E_{test}(\mathbf{w})(\times 10^{-3})$ Median / Ave. / Best / Worst
QN	-	1.44 / 1.48 / 1.26 / 1.87	56	1.12	50,000	0.90 / 0.91 / 0.69 / 1.45
AdaNAQ	-	0.59 / 0.61 / 0.48 / 0.73	64	1.89	33,715	0.65 / 0.66 / 0.46 / 1.12
AdaMoQ	-	0.67 / 0.61 / 0.47 / 0.75	31	1.12	27,627	0.79 / 1.14 / 0.59 / 2.15
LQN	1	21.95 / 22.34 / 21.01/ 23.97	154	3.08	50,000	18.88 / 19.28 / 17.91 / 20.91
	2	8.00 / 9.11 / 7.64 / 18.59	164	3.28	50,000	5.75 / 6.91 / 5.45 / 16.59
	4	8.02 / 9.78 / 7.48 / 18.12	173	3.46	50,000	5.84 / 7.76 / 5.48 / 16.55
	8	7.61 / 7.57 / 6.97 / 8.82	182	3.64	50,000	5.64 / 5.66 / 4.84 / 7.09
	16	7.13 / 7.13 / 6.26 / 7.75	199	3.98	50,000	5.23 / 5.26 / 4.64 / 5.85
	32	6.91 / 6.99 / 6.25 / 7.64	226	4.52	50,000	5.16 / 5.18 / 4.28 / 5.94
	64	6.82 / 7.21 / 5.24 / 13.24	285	5.70	50,000	5.14 / 7.88 / 3.49 / 34.18
	128	5.64 / 5.78 / 5.05 / 6.80	405	8.10	50,000	3.99 / 4.29 / 3.59 / 6.18
LNAQ	1	20.75 / 20.59 / 9.33 / 24.21	338	6.76	50,000	17.92 / 17.69 / 6.73 / 20.96
	2	19.77 / 16.26 / 8.76 / 20.36	337	6.74	50,000	17.14 / 13.65 / 6.19 / 17.73
	4	8.07 / 9.04 / 7.01 / 19.27	342	6.84	50,000	5.77 / 6.77 / 4.73 / 16.83
	8	5.69 / 5.79 / 4.08 / 7.07	349	6.98	50,000	4.43 / 4.31 / 2.51 / 5.99
	16	2.98 / 3.57 / 2.52 / 5.94	365	7.30	50,000	2.28 / 2.67 / 1.50 / 4.62
	32	1.71 / 1.70 / 1.34 / 2.52	377	7.54	50,000	1.07 / 1.28 / 0.70 / 3.06
	64	1.42 / 1.42 / 1.19 / 1.63	456	9.12	50,000	0.83 / 1.07 / 0.72 / 2.20
	128	1.23 / 1.32 / 1.11 / 1.74	578	11.5	50,000	0.72 / 0.82 / 0.58 / 1.19
LMoQ	1	20.23 / 20.49 / 15.24 / 24.73	175	3.50	50,000	17.28 / 17.59 / 12.26 / 21.40
	2	19.84 / 17.82 / 9.02 / 20.69	177	3.54	50,000	17.21 / 15.20 / 6.34 / 18.13
	4	8.34 / 9.12 / 6.84 / 19.26	181	3.62	50,000	6.08 / 6.90 / 4.68 / 16.96
	8	6.81 / 6.47 / 5.12 / 7.17	188	3.76	50,000	5.20 / 4.93 / 3.51 / 6.12
	16	3.02 / 3.20 / 2.39 / 4.45	203	4.06	50,000	2.00 / 2.39 / 1.40 / 4.88
	32	1.50 / 171 / 1.37 / 2.80	227	4.54	50,000	0.98 / 1.13 / 0.72 / 1.78
	64	1.39 / 1.44 / 1.10 / 2.23	292	5.84	50,000	0.80 / 0.82 / 0.57 / 1.32
	128	1.42 / 1.50 / 1.17 / 2.13	411	8.22	50,000	0.97 / 1.08 / 0.65 / 2.07

5.5　まとめ

本章では, 本研究の 3 つ目の課題であった NAQ [32–34] の問題点, 計算コストに着目し, その削減のために記憶制限手法を導入した記憶制限ネステロフの加速準ニュートン法 (Limited-Memory Nesterov's Accelerated Quasi-Newton method, LNAQ) および慣性付記憶制限準ニュートン法 (Limited-Memory Momentum Quasi-Newton method, LMoQ) を提案した. 提案した LNAQ と LMoQ の有効性を, 2 つの関数近似問題と 1 つのマイクロ波回路のモデリング問題に対する NN の学習に応用し, シミュレーションを行った. シミュレーションより, 記憶制限手法を用いた 2 次手法の比較においては, メモリ量 m が小さいときは収束までに多くの反復回数を必要とするが, 1 反復の計算時間が少ないことがわかった. また, メモリ量 m の増加に連れて学習は高速化するが, 1 反復の計算時間が増加してしまう. 一方で, 慣性項を用いた記憶制限手法は, LQN [20–22, 26] の高速化に成功した. 2 つ目の関数近似問題における提案手法と他のアルゴリズム, GD [1, 48, 49], CM [1, 11, 12, 48, 49], NAG [1, 13, 14, 48, 49], OptNAG [13], AdaGrad [1, 9, 16, 48, 49], AdaDelta [1, 9, 18, 48, 49], RMSprop [1, 9, 17, 48, 49], Adam [1, 9, 19, 48, 49], QN [20, 21], NAQ [32–34], MoQ, LQN の比較では, 設定した最大反復回数以内で 1 次手法は, 十分小さな学習誤差を得ることができなかった. 記憶制限手法では, 記憶量 m が小さい値に設定されているとき, アルゴリズムのパフォーマンスは 1 次手法に近づくため, 学習時間は少ないが小さな誤差を得ることは困難である. 一方で, 記憶量が増加するに連れて, 曲率情報であるヘッセ行列の影響が大きくなる. 従って, より小さい誤差を得るが計算コストの増加により計算時間も増加する. しかし, 通常の 2 次手法との比較では, 解の精度は低いため, 強非線形特性が高いマイクロ波回路のモデリング問題のような問題の学習は困難である. 一方で, 記憶制限手法に対する慣性項の影響は, これまでのように LQN の高速化に繋がった. 従って, 記憶制限手法における慣性項は有効であると結論付けられる.

6　慣性付メモリレス準ニュートン法

本章では, 再度 NAQ の 3 つ目の課題点, 計算コストの高さに着目した. これまでに, 2 次手法の計算コスト削減するため, 様々な研究が行われてきた. その一例に, 5 章で紹介した記憶制限 QN(LQN) が挙げられる [20–22, 26]. 記憶制限手法では, 確かに計算コストは通常の 2 次手法と比較して削減されるが, 1 次手法と比べて, 制限されたメモリ量 m 分高くなる. 一方で, LQN ($m = 1$) と同様の計算コストを持つ応用研究の一つとして, メモリレス手法 (MemoryLess strategy) を用いた QN, メモリレス準ニュートン法 (MemoryLess Quasi-Newton method, MLQN) が挙げられる [30, 89–93]. MLQN は過去のヘッセ行列を保持せず, ベクトルの内積のみでパラメータを更新する. 従って, その計算コストはほぼ 1 次手法と同様であると考えられる. さらに, MLQN と同様な計算コストを持つ LQN ($m = 1$) は厳密な直線探索を行ったとき, パラメータの更新に Hestenes-Stiefel 公式 (HS) を用いた非線形共役勾配法 (Nonlinear Conjugate Gradient) と密接な関係がある [21, 22, 26]. LQN と非線形共役勾配法の関係性を付録 B に示す. 本研究では, 2 種類の慣性項を用いた準ニュートン法, NAQ と MoQ に対してメモリレス手法を導入し, メモリレスネステロフの加速準ニュートン法 (MemoryLess Nesterov's Accelerated Quasi-Newton method, MLNAQ) そしてメモリレス慣性付準ニュートン法 (MemoryLess Momentum Quasi-Newton method, MLMoQ) を提案する [45–47]. 提案手法は関数近似問題と 3 つの分類問題に対する NN の学習に応用し, 従来手法と比較した上, その有効性を示す.

6.1　Conjugate Gradient method

共役勾配法 (Conjugate Gradient method, CG) は反復的に降下する共役方向 (Conjugate Direction) によって, ヘッセ行列の逆行列の計算を効率的に回避した手法である [1, 21, 22, 26, 93, 94]. 共役勾配法では, 狭義凸 2 次関数の最小化問題を解くための共役勾配法を線形共役勾配法 (Linear Conjugate Gradient method), そして非線形関数の最小化問題を解くための共役勾配法を非線形共役勾配法 (Nonlinear Conjugate Gradient method) と呼び分けられている. 本研究では, 非線形問題の最小化に着目しているため, 非線形 CG(以降, CG) にのみ着目する. CG における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は探索方向ベクトル $\mathbf{g}_k^{\mathrm{CG}}$ を用いて (125) に示す.

$$\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k^{\mathrm{CG}}, \tag{125}$$

$$\mathbf{g}_k^{\mathrm{CG}} = -\nabla E(\mathbf{w}_k) + \beta_k \mathbf{g}_{k-1}^{\mathrm{CG}}. \tag{126}$$

ステップサイズ α_k は直線探索と Armijo の条件 [21, 22, 26] によって決定される. パラメータ β_k は Fletcher-Reeves 公式 (FR), Polak-Ribiere 公式 (PR), Polak-Ribiere plus 公式 (PR+), Hestenes-Stiefel 公式 (HS) そして Dai-Yuan 公式 (DY) から求めることができる. それぞれの公式を順番に以下に示す.

$$\beta_k^{\mathrm{FR}} = \frac{\|\nabla E(\mathbf{w}_k)\|^2}{\|\nabla E(\mathbf{w}_{k-1})\|^2}, \tag{127}$$

$$\beta_k^{\mathrm{PR}} = \frac{\nabla E(\mathbf{w}_k)^{\mathrm{T}}\mathbf{y}_{k-1}}{\|\nabla E(\mathbf{w}_{k-1})\|^2}, \tag{128}$$

$$\beta_k^{\mathrm{PR+}} = max\left\{\frac{\nabla E(\mathbf{w}_k)^{\mathrm{T}}\mathbf{y}_{k-1}}{\|\nabla E(\mathbf{w}_{k-1})\|^2}, 0\right\}, \tag{129}$$

$$\beta_k^{\mathrm{HS}} = \frac{\nabla E(\mathbf{w}_k)^{\mathrm{T}}\mathbf{y}_{k-1}}{(\mathbf{g}_{k-1}^{\mathrm{CG}})^{\mathrm{T}}\mathbf{y}_{k-1}}, \tag{130}$$

$$\beta_k^{\mathrm{DY}} = \frac{\|\nabla E(\mathbf{w}_k)\|^2}{(\mathbf{g}_{k-1}^{\mathrm{CG}})^{\mathrm{T}}\mathbf{y}_{k-1}}, \tag{131}$$

ここで, $\mathbf{y}_{k-1} = \nabla E(\mathbf{w}_k) - \nabla E(\mathbf{w}_{k-1})$ である [26, 93]. CG のアルゴリズムを Algorithm 19 に示す.

Algorithm 19 Conjugate Gradient method (CG)

Require: ϵ, k_{max}
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$

$k = 1$
Calculate $\nabla E(\mathbf{w}_1)$;
Update $\mathbf{g}_1^{\mathrm{CG}} = -\nabla E(\mathbf{w}_1)$;
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
  Calculate β_k using (127), (128), (129), (130) or (131);
  Update $\mathbf{g}_k^{\mathrm{CG}} = -\nabla E(\mathbf{w}_k) + \beta_k \mathbf{g}_{k-1}^{\mathrm{CG}}$;
  Calculate stepsize α_k using Armijo's condition;
  Update $\mathbf{v}_{k+1} = -\alpha_k \mathbf{g}_k^{\mathrm{CG}}$;
  Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
  Calculate $\nabla E(\mathbf{w}_{k+1})$;
  Calculate $\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k)$;
  $k = k + 1$;
end while

Return: $\mathbf{w}_k$

6.2 MemoryLess Quasi-Newton method

メモリレス準ニュートン法 (MemoryLess Quasi-Newton method, MLQN) は QN の計算コストを大幅に削減し, 2 次手法の収束特性を維持したまま, 1 次手法と同程度の計算コストを実現するために提案された手法である [30, 89–93]. MLQN における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLQN}}$ を用いて (132) に示す.

$$\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k^{\mathrm{MLQN}}, \tag{132}$$

$$\mathbf{g}_k^{\mathrm{MLQN}} = -\mathbf{H}_k^{\mathrm{QN}} \nabla E(\mathbf{w}_k). \tag{133}$$

ステップサイズ α_k は直線探索と Armijo の条件 [21] によって決定される. $\mathbf{H}_k^{\mathrm{MLQN}}$ はヘッセ行列の逆近似行列を表し, 以下の BFGS 公式により各反復で更新される [21],

$$\mathbf{H}_{k+1}^{\mathrm{QN}} = \mathbf{H}_k^{\mathrm{QN}} - \left(\frac{(\mathbf{H}_k^{\mathrm{QN}} \mathbf{y}_k)\mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k (\mathbf{H}_k^{\mathrm{QN}} \mathbf{y}_k)^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k} \right) + \left(1 + \frac{\mathbf{y}_k^{\mathrm{T}} \mathbf{H}_k^{\mathrm{QN}} \mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k} \right) \left(\frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k} \right). \tag{56}$$

QN における更新では, 行列 $\mathbf{H}_k^{\mathrm{QN}}$ を格納するためのメモリが必要とされる. MLQN [30,89–93] では, $\mathbf{H}_{k+1}^{\mathrm{QN}}$ 行列の更新は (56) で行うが, その一つ前の反復における $\mathbf{H}_k^{\mathrm{QN}}$ はスケーリングされた単位行列 $\tau_k \mathbf{I}$（$\tau_k > 0$）に置き換える. 従って, MLQN の探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLQN}}$ は以下のように導出される.

$$\mathbf{g}_k^{\mathrm{MLQN}} = -\left((\tau_k \mathbf{I}) - \left(\frac{\tau_k (\mathbf{y}_k \mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k \mathbf{y}_k^{\mathrm{T}})}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k} \right) + \left(1 + \frac{\tau_k \mathbf{y}_k^{\mathrm{T}} \mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k} \right) \left(\frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k} \right) \right) \cdot \nabla E(\mathbf{w}_k) \tag{134}$$

$$= -\tau_k (\nabla E(\mathbf{w}_k) - (\beta_k \mathbf{y}_k + \psi_k \mathbf{s}_k)) - (1 + \tau_k \omega_k) \beta_k \mathbf{s}_k, \tag{135}$$

ここで,

$$\begin{aligned} \mathbf{s}_k &= \mathbf{w}_{k+1} - \mathbf{w}_k, \\ \mathbf{y}_k &= \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k), \\ \beta_k &= \frac{\mathbf{s}_k^{\mathrm{T}} \nabla E(\mathbf{w}_k)}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}, \\ \psi_k &= \frac{\mathbf{y}_k^{\mathrm{T}} \nabla E(\mathbf{w}_k)}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}, \\ \omega_k &= \frac{\mathbf{y}_k^{\mathrm{T}} \mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}, \end{aligned} \tag{136}$$

であり, セルフスケーリング係数は

$$\tau_k = \frac{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}{\mathbf{y}_k^{\mathrm{T}}\mathbf{y}_k}, \tag{137}$$

である [90]. (134) より, $\mathbf{g}_k^{\mathrm{MLQN}}$ は行列を保存せずにベクトルの内積だけで計算されている, つまり, メモリが大幅に削減可能な手法であることがわかる. 従って, 計算量は 1 次手法と同程度に削減されていると言える. MLQN のアルゴリズムを Algorithm 20 に示す.

Algorithm 20 MemoryLess Quasi-Newton method (MLQN)

Require: ϵ, k_{max}
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$
1: $k = 1$
2: Calculate $\nabla E(\mathbf{w}_1)$;
3: **while** $(\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max})$ **do**
4: Update $\mathbf{g}_k^{\mathrm{MLQN}}$ using (134);
5: Calculate stepsize α_k using Armijo's condition;
6: Update $\mathbf{v}_{k+1} = -\alpha_k \mathbf{g}_k^{\mathrm{MLQN}}$;
7: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
8: Calculate $\nabla E(\mathbf{w}_{k+1})$;
9: Calculate $\mathbf{s}_k$ and $\mathbf{y}_k$;
10: $k = k + 1$;
11: **end while**
Return: $\mathbf{w}_k$

6.3　慣性項によるメモリレス手法の高速化

本節では, MLQN に対する慣性項の有効性を示すと共に, NAQ と MoQ の計算コストを LNAQ と LMoQ よりもさらに削減するため, NAQ と MoQ にメモリレス手法を導入し, メモリレス NAQ とメモリレス MoQ として提案する.

6.3.1　MemoryLess Nesterov's Accelerated Quasi-Newton method

メモリレスネステロフの加速準ニュートン法 (MemoryLess Nesterov's Accelerated Quasi-Newton method, MLNAQ) における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLNAQ}}$ を用いて (138) に示す.

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k + \eta_k \mathbf{g}_k^{\mathrm{MLNAQ}}, \tag{138}$$

$$\mathbf{g}_k^{\mathrm{MLNAQ}} = -\mathbf{H}_k^{\mathrm{NAQ}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{139}$$

ここで, 探索ベクトル $\mathbf{g}_k^{\mathrm{MLNAQ}}$ を説明する前に NAQ の探索ベクトル $\mathbf{g}_k^{\mathrm{NAQ}}$ について考える. (139) では, $\mu_k \mathbf{v}_k$ は慣性項であり, モーメント係数 μ_k は適応的に求まる. 行列 $\mathbf{H}_{k+1}^{\mathrm{NAQ}}$ は以下のように更新される.

$$\mathbf{H}_{k+1}^{\mathrm{NAQ}} = \mathbf{H}_k^{\mathrm{NAQ}} - \left(\frac{(\mathbf{H}_k^{\mathrm{NAQ}} \mathbf{q}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k (\mathbf{H}_k^{\mathrm{NAQ}} \mathbf{q}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} \right) + \left(1 + \frac{\mathbf{q}_k^{\mathrm{T}} \mathbf{H}_k^{\mathrm{NAQ}} \mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} \right) \left(\frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} \right), \tag{60}$$

ここで, $\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k)$ と $\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ である. NAQ は QN と同様に $\mathbf{H}_k^{\mathrm{NAQ}}$ の更新に過去の行列情報が必要である. 従って, QN と同様に計算量を削減するため, メモリレス手法を NAQ に導入した MLNAQ を提案する. MLNAQ の $\mathbf{g}_k^{\mathrm{MLNAQ}}$ は (140) から求まる.

$$\mathbf{g}_k^{\mathrm{MLNAQ}} = -\left((\tau_k \mathbf{I}) - \left(\frac{\tau_k (\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k \mathbf{q}_k^{\mathrm{T}})}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} \right) + \left(1 + \frac{\tau_k \mathbf{q}_k^{\mathrm{T}} \mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} \right) \left(\frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k} \right) \right) \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \tag{140}$$

$$= -\tau_k (\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) - (\beta_k \mathbf{q}_k + \psi_k \mathbf{p}_k)) - (1 + \tau_k \omega_k) \beta_k \mathbf{p}_k, \tag{141}$$

ここで, それぞれのスカラーは

$$\begin{aligned} \beta_k &= \frac{\mathbf{p}_k^{\mathrm{T}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}, \\ \psi_k &= \frac{\mathbf{q}_k^{\mathrm{T}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}, \\ \omega_k &= \frac{\mathbf{q}_k^{\mathrm{T}} \mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}, \\ \tau_k &= \frac{\mathbf{p}_k^{\mathrm{T}} \mathbf{q}_k}{\mathbf{q}_k^{\mathrm{T}} \mathbf{q}_k}, \end{aligned} \tag{142}$$

の内積から求まる. (140) より, MLNAQ の探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLNAQ}}$ では, 行列を保存せずにベクトルの内積だけで求められていることがわかる. MLNAQ のアルゴリズムを Algorithm 21 に示す. アルゴリズムより, MLNAQ は Step.4 と 9 に示すように, 各反復においてネステロフの加速勾配 $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ と通常勾配 $\nabla E(\mathbf{w}_k)$ の 2 種類を計算しなければならないことがわかる.

Algorithm 21 MemoryLess Nesterov's Accelerated Quasi-Newton method (MLNAQ)

Require: ϵ, k_{max}, μ_k
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$and $\mathbf{v}_1 = \mathbf{0}$
1: $k = 1$
2: **while** $(\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max})$ **do**
3: Culculate μ_k using (64) and (65);
4: Calculate $\nabla E\left(\mathbf{w}_k + \mu_k \mathbf{v}_k\right)$;
5: Update $\mathbf{g}_k^{\mathrm{MLNAQ}}$ using (140);
6: Calculate stepsize α_k using Armijo's condition;
7: Update $\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k + \alpha_k \mathbf{g}_k^{\mathrm{MLNAQ}}$;
8: Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
9: Calculate $\nabla E(\mathbf{w}_{k+1})$;
10: Calculate $\mathbf{p}_k$ and $\mathbf{q}_k$;
11: $k = k + 1$;
12: **end while**
Return: $\mathbf{w}_k$

6.3.2 MemoryLess Momentum Quasi-Newton method

本研究では, NAQ の問題点を克服するため, MoQ を提案した. MoQ では誤差関数を 2 次関数とみなし, ネステロフの加速勾配 $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ を (75) に示すように通常勾配の重み付の線形和として近似する.

$$\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \simeq \nabla E(\mathbf{w}_k) + \mu_k \nabla E(\mathbf{v}_k) = (1 + \mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}), \tag{75}$$

MoQ にメモリレス手法を導入した, メモリレス慣性付準ニュートン法 (MemoryLess Momentum Quasi-Newton method, MLMoQ) における (3) の更新ベクトル $\mathbf{v}_{k+1}$ は, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLMoQ}}$ を用いて (143) に示す.

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k + \eta_k \mathbf{g}_k^{\mathrm{MLMoQ}}, \tag{143}$$

$$\mathbf{g}_k^{\mathrm{MLMoQ}} = -\mathbf{H}_k^{\mathrm{MoQ}}\{(1 + \mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}. \tag{144}$$

ここで, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLMoQ}}$ を説明する前に MoQ の探索方向ベクトル $\mathbf{g}_k^{\mathrm{MoQ}}$ について考える. (143) では, $\mu_k \mathbf{v}_k$ は慣性項であり, モーメント係数 μ_k は適応的に求まる. 行列 $\mathbf{H}_{k+1}^{\mathrm{MoQ}}$

は以下のように更新される.

$$\mathbf{H}_{k+1}^{\mathrm{MoQ}} = \mathbf{H}_{k}^{\mathrm{MoQ}} - \left(\frac{(\mathbf{H}_{k}^{\mathrm{MoQ}}\hat{\mathbf{q}}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k(\mathbf{H}_{k}^{\mathrm{MoQ}}\hat{\mathbf{q}}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \right) + \left(1 + \frac{\hat{\mathbf{q}}_k^{\mathrm{T}}\mathbf{H}_{k}^{\mathrm{MoQ}}\hat{\mathbf{q}}_k}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \right) \left(\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \right). \tag{77}$$

ここで, $\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k\mathbf{v}_k)$ と $\hat{\mathbf{q}}_k = \nabla E(\mathbf{w}_{k+1}) - \{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{w}_{k-1})\}$ である. MoQ も NAQ と同様にメモリレス手法を導入すると, 探索方向ベクトル $\mathbf{g}_k^{\mathrm{MLMoQ}}$ は

$$\mathbf{g}_k^{\mathrm{MLMoQ}} = -\left((\tau_k\mathbf{I}) - \left(\frac{\tau_k(\hat{\mathbf{q}}_k\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k\hat{\mathbf{q}}_k^{\mathrm{T}})}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \right) + \left(1 + \frac{\tau_k\hat{\mathbf{q}}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \right) \left(\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k} \right) \right) \{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{w}_{k-1})\} \tag{145}$$

$$= -\tau_k(\{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{w}_{k-1})\} - (\beta_k\hat{\mathbf{q}}_k + \psi_k\mathbf{p}_k)) - (1+\tau_k\omega_k)\beta_k\mathbf{p}_k, \tag{146}$$

となる. ここで, それぞれのスカラーは

$$\begin{aligned}
\beta_k &= \frac{\mathbf{p}_k^{\mathrm{T}}\{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{w}_{k-1})\}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}, \\
\psi_k &= \frac{\hat{\mathbf{q}}_k^{\mathrm{T}}\{(1+\mu_k)\nabla E(\mathbf{w}_k) + \mu_k\nabla E(\mathbf{w}_{k-1})\}}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}, \\
\omega_k &= \frac{\hat{\mathbf{q}}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}, \\
\tau_k &= \frac{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{q}}_k}{\hat{\mathbf{q}}_k^{\mathrm{T}}\hat{\mathbf{q}}_k},
\end{aligned} \tag{147}$$

の内積から求まる. MLMoQ のアルゴリズムを Algorithm 22 に示す. アルゴリズムより, MLMoQ は Step.9 でのみ勾配を計算していることがわかる.

Algorithm 22 MemoryLess Momentum Quasi-Newton method (MLMoQ)

Require: ϵ, k_{max}
Initialize: $\mathbf{w}_1 \in \mathbb{R}^d$and $\mathbf{v}_1 = \mathbf{0}$
$k = 1$
Calculate $\nabla E(\mathbf{w}_1)$;
while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**
　Culculate μ_k using (64) and (65);
　Update $\mathbf{g}_k^{\mathrm{MLMoQ}}$ using (145);
　Calculate stepsize α_k using Armijo's condition;
　Update $\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k + \alpha_k \mathbf{g}_k^{\mathrm{MLMoQ}}$;
　Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$;
　Calculate $\nabla E(\mathbf{w}_{k+1})$;
　Calculate $\mathbf{p}_k$ and $\hat{\mathbf{q}}_k$;
　$k = k + 1$;
end while
Return: $\mathbf{w}_k$

6.4　計算コスト

本節では, 2 次手法の通常, 記憶制限手法そしてメモリレス手法を用いた場合の計算コストとメモリ量の比較を行う. 計算コストとメモリ量を表 17 に示す. 表では, 勾配評価コストは nd として表し, n は学習サンプル数, d はパラメータの次元である. ヘッセ行列の逆近似行列の更新コストを d^2 とする. 記憶制限手法では, 任意のメモリ量 m までヘッセ行列コストを削減できるため, $4md + 2d$ とした. 一方で, メモリレス手法ではヘッセ行列と勾配の内積により, 探索ベクトルを更新するためそのコストは $4d$ となる. 表の全てのアルゴリズムでは, ステップサイズは直線探索法により決定されるため, 探索条件を満たすまで ζ 回関数評価を行う. メモリレス手法も通常と記憶制限手法と同様に, NAQ は QN と MoQ よりも勾配計算回数が 1 回多くあるため, 勾配計算コストは $2nd$ となる. なお, QN と MoQ は同じ勾配計算コストを持つ. 一方で, 保存されるメモリ量に関しては, メモリレス手法では過去のヘッセ行列を保持する必要がないため, $\mathbf{s}_k$, $\mathbf{y}_k$, $\mathbf{p}_k$, $\mathbf{q}_k$ そして $\hat{\mathbf{q}}_k$ のみが記憶される. 従って, メモリ量は $2md$ である. また, MLMoQ では, MoQ と LMoQ と同様にネステロフの加速勾配の近似を行っているため, 過去の勾配を 1 つ多く記憶する必要がある. 従って, メモリ量は $(3n + 2)d$ となる.

ここで, 本研究で使用する最小のパラメータ次元 $d = 31$ を持つ問題から最大のパラメータ次元 $d = 7,960$ を持つ問題まで, 次元 d の値を少しずつ増加させ, 次元 d に対する MoQ と

LMoQ ($m = 1, 32, 512$) と MLMoQ の計算コストとメモリ量の関係性を図 60 と 61 に示す. グラフを見やすくするため, 図 60 と 61 の y– 軸と x– 軸の表記はそれぞれ対数と線形としたと共に, 全アルゴリズムの計算コストおよびメモリ量で共通している項を除外した. 図 60 と 61 より, 次元 d の増加に伴い MoQ は指数関数的に増加している. 一方で, 定数倍に増加する記憶制限手法では, 記憶量 m の増加に伴い, LMoQ の計算コストとメモリ量が MoQ に近づいていることがわかる. 一方で, MLMoQ はもっとも低い計算コストとメモリ量を持っている. また LMoQ ($m = 1$) は MLMoQ に近い計算コストを持つことがわかる. 従って, メモリレス手法は記憶制限手法と通常の 2 次手法と比較して, 計算コストとメモリ量を削減できていると結論付けられる.

表 17: メモリレス手法の計算コストとメモリ量の比較

	Algorithm	Computational Cost	Storage
Normal	QN	$nd + d^2 + \zeta nd$	$d^2 + 2(n+1)d$
	NAQ	$2nd + d^2 + \zeta nd$	$d^2 + 2(n+1)d$
	MoQ	$nd + d^2 + \zeta nd$	$d^2 + (3n+2)d$
Limited-Memory	LQN	$nd + 4md + 2d + \zeta nd$	$2md + 2(n+1)d$
	LNAQ	$2nd + 4md + 2d + \zeta nd$	$2md + 2(n+1)d$
	LMoQ	$nd + 4md + 2d + \zeta nd$	$2md + (3n+2)d$
MemoryLess	MLQN	$nd + 4d + \zeta nd$	$2(n+1)d$
	MLNAQ	$2nd + 4d + \zeta nd$	$2(n+1)d$
	MLMoQ	$nd + 4d + \zeta nd$	$(3n+2)d$

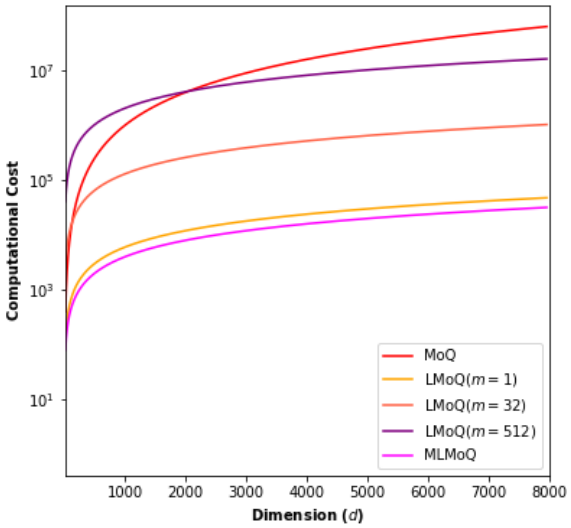


図 60: 2 次と手法, 記憶制限手法そしてメモリレス手法の計算コストの比較

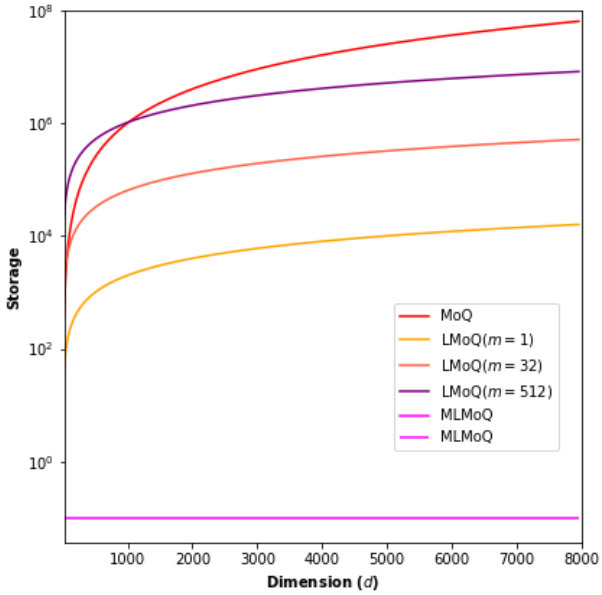


図 61: 2 次と手法, 記憶制限手法そしてメモリレス手法のメモリ量の比較

6.5　実験

本研究では, 提案手法 MLNAQ と MLMoQ の有効性を示すため, 関数近似問題と 3 つの分類問題に対するシミュレーションを行った. 本実験では MLNAQ と MLMoQ を従来手法の Adam [1, 19], CG [30], LQN ($m = 1, 2, 16$) [21] そして MLQN [31] と比較しながら, 提案手法の性能について議論を行う. なお, 提案手法はバッチ手法に基づいているため, 全てのアルゴリズムに対してバッチ学習を適応する. Adam におけるパラメータは全て [19] の推奨値とした. CG のパラメータ β_k に関しては, HS 公式に基づいて更新を行う. CG, LQN, MLQN, MLNAQ および MLMoQ のステップサイズ α_k は直線探索と Armijo の条件に基づき更新される. 公平な比較のため, MLQN, MLNAQ および MLMoQ の $\mathbf{y}_k$, $\mathbf{q}_k$ そして $\hat{\mathbf{q}}_k$ に対して, グローバル収束項 [24, 39] を導入した. MLNAQ および MLMoQ では学習の安定性を向上させるため, 適応的モーメント係数 μ_k を使用した. 各シミュレーションでは 10 回の異なる重み $\mathbf{w}$ の初期値に対して学習を行い, $\mathbf{w}$ は $[-0.5, 0.5]$ 内の一様乱数で初期化した. さらに, 学習データ $|T_r|$ とテストデータ $|T_e|$ は $[-1.0, 1.0]$ の範囲で正規化されている. 終了条件は全ての問題に対して $\epsilon = 10^{-6}$ and $k_{max} = 150,000$ とした.

6.5.1　関数近似問題

最初に, MLNAQ と MLMoQ の有効性を調べるため, 5.4.1 節で紹介した (124) に示す関数近似問題のモデリングを行った [68, 80].

$$f(x) = 1 + (x + 2x^2)\sin(-x^2), \qquad |x| \leq 4, \tag{124}$$

この問題では, 入力 x の範囲は $x \in [-4, 4)$ とした. また, 学習サンプル数は $|T_r| = 400$ である. 実験に用いる NN の構造は 1-10-1 であり, 誤差関数 $E(\mathbf{w})$ は (15) の MSE に設定した. 各ニューロンの活性化関数として (16) のシグモイド関数を用いた. シミュレーション結果を表 18 に示す. 表では, $E(\mathbf{w})$ の中央と平均値に加え, 平均の計算時間 (sec), 各反復毎の計算時間 (sec)($\times 10^{-3}$) そして平均の反復回数 (k) が示されている. 全てのアルゴリズムに対して, 学習終了までに十分に小さな誤差を得た試行回数の割合を収束率 (%) として示した. 本実験では, Adam, CG, LQN ($m = 1, 2, 16$), MLQN, MLNAQ そして MLMoQ に加え, LNAQ ($m = 1, 2, 16$) と LMoQ ($m = 1, 2, 16$) も比較の対象とした. 表より, 全てのアルゴリズムが同程度の学習誤差で収束していることがわかる. この問題では, 学習アルゴリズムに対して問題の非線形特性が比較的に強く, 学習データ $|T_r|$ も少ないため, LMoQ ($m = 16$) 以外の全て

のアルゴリズムが収束率 100% を達成できなかった. Adam, CG, LQN ($m = 1$) そして MLQN では, 学習が収束するまでに多くの反復回数を必要とする. 一方, LQN, LNAQ と LMoQ では, メモリサイズ m が大きくなるにしたがって, 曲率情報の影響が大きくなるため, 学習性能が向上する. そのため, 反復回数が減少し, 収束が早まる. しかし, メモリ量に比例して計算量も増加するため, 1 反復にかかる時間が長くなることがわかる. 従って, LQN ($m = 16$), LNAQ ($m = 16$) および LMoQ ($m = 16$) では, 反復回数は少ないが, 1 反復あたりの計算時間は最大となる. メモリレス手法の比較では, MLNAQ と MLMoQ は MLQN よりも少ない反復回数で収束していることがわかる. これは, MLQN における慣性項が有効であることを意味する. さらに, MLMoQ の 1 反復あたりの時間は MLQN とほぼ同じである, 一方で, MLNAQ は勾配計算回数の欠点により多くの計算時間を必要とする. 結果として, MLMoQ はこの実験で用いたメモリレス手法のアルゴリズムの中でもっとも高速であると言える. 従って, 慣性項はメモリレス手法に対しても有効であると結論付けられる.

この実験では, モデリングの精度を測定するため, MLMoQ で学習した NN モデルの出力とテストデータを比較し, 図 62 に示した. 図より, MLMoQ の NN モデルとテストモデルが良好な一致を示していることが確認できる.

表 18: 関数近似問題 (68) に対する MLMoQ のシミュレーション結果

Algorithm	m	Iteration counts (k)	Time (sec)	Per iteration time(sec)($\times 10^{-3}$)	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave.	Convergence rate (%)
Adam	-	90,908	7.75	0.085	0.999 / 1.028	80
CG	-	52,636	6.00	0.113	0.996 / 0.995	60
LQN	1	81,577	10.20	0.125	0.999 / 0.999	80
	2	39,260	5.39	0.137	0.999 / 0.999	70
	16	26,138	6.75	0.258	0.999 / 0.999	60
LNAQ	1	30,704	6.49	0.211	1.00 / 1.00	80
	2	27,702	5.99	0.211	0.999 / 1.03	60
	16	5,525	1.91	0.345	0.999 / 0.999	90
LMoQ	1	32,163	4.44	0.138	1.00 / 1.03	80
	2	25,644	3.74	0.145	1.00 / 1.00	70
	16	5,222	1.41	0.270	0.999 / 0.998	100
MLQN	-	80,729	9.79	0.121	0.999 / 0.999	80
MLNAQ	-	33,493	6.85	0.205	0.999 / 0.999	80
MLMoQ	-	29,834	**3.65**	0.122	0.999 / 0.999	70

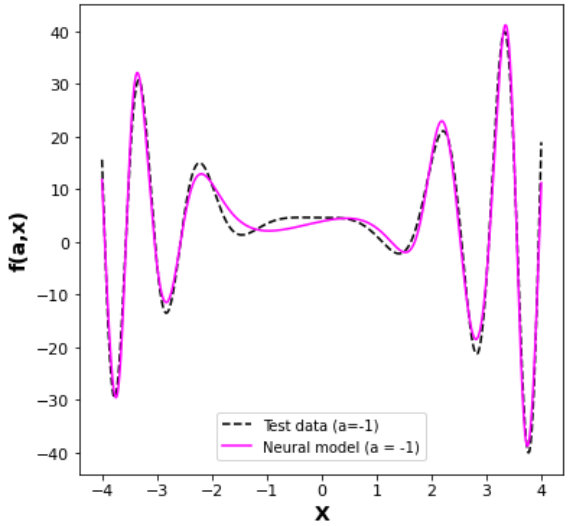


図 62: 関数 (124) に対する MLMoQ の NN モデルとテストデータの比較

6.5.2　分類問題 1: Three-Spirals detaset

次に, 4.3.6 節にて紹介した Three-Spirals の分類問題 [88] に対して, MLNAQ と MLMoQ の有効性を調べる. Three-Spirals の学習サンプルの構造を図 49 に示す. この問題では, 学習データ数は $|T_r| = 1,050$ であり, NN の構造は 2-10-3 とした. 分類問題では誤差関数および出力層の活性化関数をそれぞれ (19) の CE と (20) のソフトマックス関数とする. なお, 中間層の活性化関数として (16) のシグモイド関数を使用する. シミュレーション結果を表 19 に示す. 表では, $E(\mathbf{w})$ の中央と平均値に加え, 平均の計算時間 (sec), 1 反復の計算時間 (sec)($\times 10^{-3}$) そして平均の反復回数 (k) が示されている. この実験では, 慣性項を用いたメモリレス手法の比較に着目するため, LNAQ および LMoQ を除外する. 表より, 比較で使用し

表 19: Three-Spirals に対する MLMoQ のシミュレーション結果

Algorithm	Iteration counts (k)	Time (sec)	Per iteration time(sec)($\times 10^{-3}$)	$E_{train}(\mathbf{w})(\times 10^{-3})$ Median / Ave.	Convergence rate (%)
Adam	55,038	14.64	0.266	0.840 / 0.838	100
CG	150,000	57.21	0.381	1.378 / 1.381	100
LQN(1)	9,918	3.98	0.401	0.826 / 0.825	100
LQN(2)	9,290	3.80	0.409	0.826 / 0.827	100
LQN(16)	8,461	4.43	0.523	0.819 / 0.821	100
MLQN	9,820	3.85	0.392	0.824 / 0.826	100
MLNAQ	700	0.46	0.657	0.672 / 0.663	100
MLMoQ	590	**0.23**	0.389	0.643 / 0.649	100

た全てのアルゴリズムが同様の学習誤差に収束し, 収束率も 100% であることがわかる. 反復回数の比較では, QN に基づくアルゴリズムは Adam や CG よりも早く収束している. また, MLNAQ と MLMoQ は LQN と MLQN に比べ, 極めて少ない反復回数で解を得ることができる. MLNAQ と MLMoQ の比較では, MLMoQ が MLNAQ より高速である. またこれらの結果は学習精度と時間のグラフを示した, 図 63 からもわかる. 従って, 提案手法 MLMoQ は比較に用いた全てのアルゴリズムの中でもっとも高速であると結論付けられる.

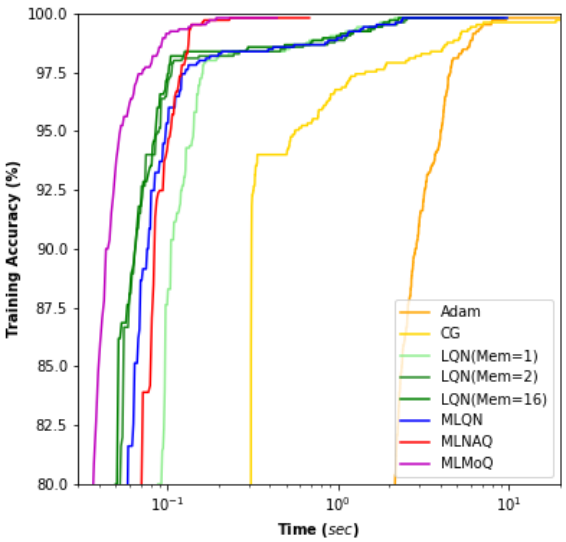


図 63: Three-Spirals の学習精度と時間グラフ

この実験でも, 学習精度を測定するため, MLMoQ で学習した NN モデルの出力と 2 次元平面テストデータを比較し, 図 64 に示した. 図より, MLMoQ のニューラルモデルはオーバーフィットすることなく, 2 次元平面を良好に分類できることが確認できる.

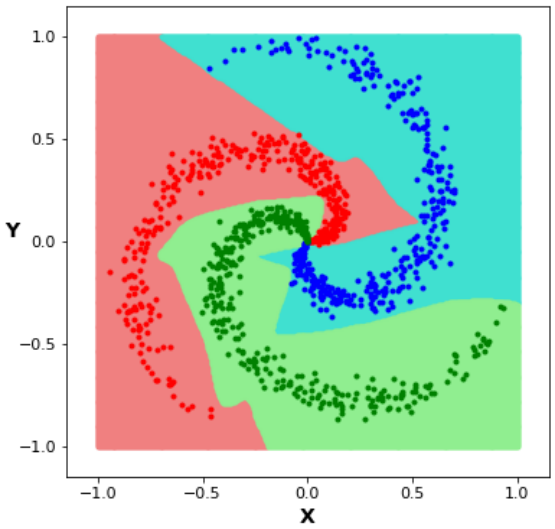


図 64: Three-Spirals の MLMoQ による検証結果

6.5.3　分類問題 2: 8×8 MNIST handwritten digit dataset

次に, 4.3.5 節で紹介した 8×8 ピクセルの手書き数字文字 MNIST に対して, 提案手法の有効性を調べる. 8×8 ピクセル MNIST の学習サンプルを図 44 に示す. 8×8 MNIST は 1,797 個のサンプルデータを持ち, 本実験では, ランダムに全データサンプルの 75% (1,347 サンプル) を $|T_r|$ そして 25% (450 サンプル) を $|T_e|$ として分割した [85]. NN の構造は 64-10-10 である. 各アルゴリズムのもっとも良い結果を学習誤差と反復回数およびテスト精度と時間のグラフで表し, それぞれを図 65 と 66 に示す. 図 65 では, 見やすさのため, x– 軸と y– 軸をそれぞれ線形と対数表示にした. また, 図 66 では, x– 軸と y– 軸をそれぞれ対数と線形表示に設定した. 図 65 より, MLNAQ と MLMoQ がほぼ同精度であることがわかる. この問題では, LQN と MLQN もほぼ同精度である. 一方で, Adam および CG では収束までにより多くの反復回数を必要とすることがわかる. 従って, QN に基づく手法が Adam と CG と比べて, 学習の速い段階で小さな誤差を得て収束できると言える. QN に基づく手法の比較では, MLNAQ と MLMoQ は LQN と MLQN に比べて, 少ない反復回数で小さな誤差を得て収束していることが確認できる. 一方で, 学習時間の比較を示す図 66 では, MLMoQ が他のアルゴリズムよりも速く収束することがわかる. 従って, 提案手法 MLMoQ はこの問題に対しても有効であると結論付けられる.

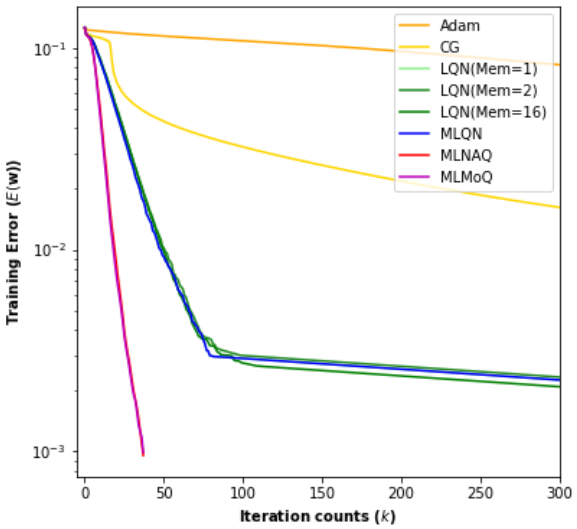


図 65: 8 × 8 MNIST の学習誤差と反復回数グラフ

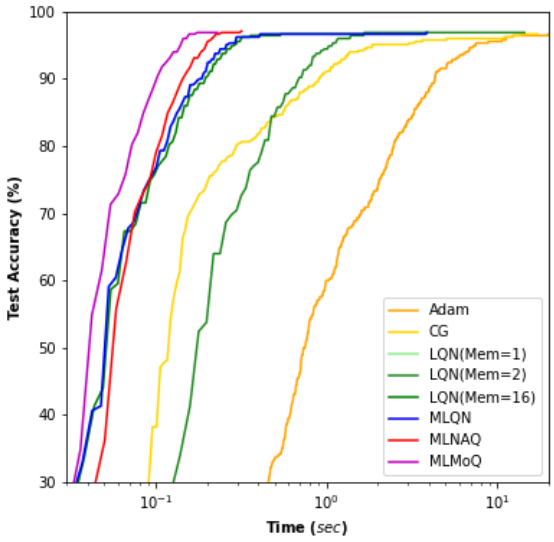


図 66: 8 × 8 MNIST の検証精度と時間グラフ

6.5.4　分類問題 3: 28×28 MNIST handwritten digit dataset

最後に, 標準的な MNIST, 28×28 ピクセルの手書き数字文字 MNIST データセット [1,87] に対して, 提案手法の性能を Tensorflow [95] を用いて評価を行う. この実験では, Tensorflow ver.2.6 上で MLQN と MLMoQ を実装し, Google Colaboratory [96] 環境でシミュレーションを行った. Tensorflow を用いた MLMoQ 最適化手法の実装は, [97] にて公開している. 28×28 MNIST の学習とテストデータセットはそれぞれ $|T_r| = 60,000$ と $|T_e| = 10,000$ である [1,87]. 28×28 MNIST のデータサンプルの一部を図 67 に示す. NN の入力と出力はそれぞれ画像データのピクセル (次元) 数 784 とクラス数 10 であり, NN の構造は 784-10-10 である. Google Colaboratory では実行する度に, 割り当てられるリソースが変更するため, 正確な計算時間の比較は困難である. 従って, 反復回数と学習誤差およびテスト精度での比較を行う. 比較アルゴリズムは Adam, MLQN そして MLMoQ とする. Adam のハイパーパラメータは Tensorflow のデフォルト値に設定した. なお, この実験でも, 全てのアルゴリズムに対してバッチ学習法を適応する. MLQN と MLMoQ のステップサイズ α_k は固定とし, その値を $\alpha_k = 1.0$ と設定した. MLMoQ におけるモーメント係数には適応的慣性係数を用いた. 各アルゴリズムのもっとも良い結果を, 学習誤差と反復回数およびテスト精度と反復回数のグラフに表し, それぞれを図 68 と 69 に示した. これらの図より, どのアルゴリズムも 2,000 反復以内で同程度の誤差と精度に達していることがわかる. しかし, その中でも提案手法 MLMoQ は, 他のアルゴリズムよりも速い段階で低い誤差および高い精度を得ることができる. 従って, 提案する MLMoQ は Tensorflow における 28×28 MNIST 問題に対しても高速に収束する特性を持つことが結論付けられる.

図 67: 28×28 MNIST のデータサンプルの一部

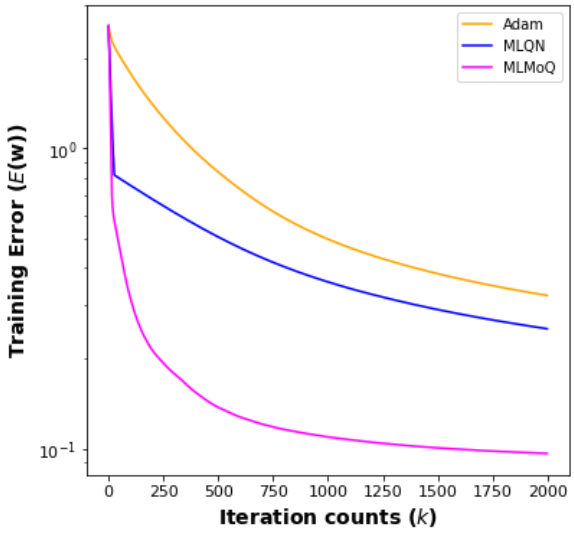


図 68: 28×28 MNIST の学習誤差と反復回数グラフ

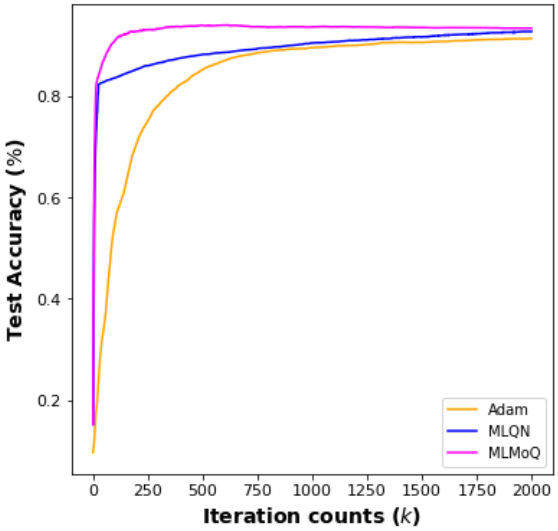


図 69: 28×28 MNIST のテスト精度と反復回数グラフ

6.6　まとめ

本章では, 本研究の 3 つ目の課題であった NAQ [32–34] の計算コストに着目し, 2 次手法の NAQ と MoQ が 1 次手法と同程度な計算コストを得られるようにメモリレス手法を導入した. 提案手法はメモリレスネステロフの加速準ニュートン法 (MemoryLess Nesterov's Accelerated Quasi-Newton method, MLNAQ) とメモリレス慣性付準ニュートン法 (MemoryLess Momentum Quasi-Newton method, MLMoQ) とした. MLNAQ と MLMoQ は通常の NAQ と MoQ と同様に慣性項の影響によって MLQN [30, 89–93] を高速化した. 提案手法の有効性は関数近似問題および 3 つの分類問題に対する学習に応用し, シミュレーションにより示した. メモリレスを用いた提案手法, 特に MLMoQ は 1 次手法と比較して曲率情報の影響を受けているため, シミュレーションとして用いた問題に対しては有効であった. しかし, メモリ量が大きい記憶制限手法と比較して, 曲率情報の影響が弱いため, マイクロ波回路のモデリング問題などのような強非線形問題の学習に対しては学習性能が低下し, 通常の 2 次手法の高い学習精度を得ることは困難となる. 従って, 強非線形問題の学習を可能とするため, 今後さらなる改良を必要とする.

7 結論

本研究では, 2 次手法における慣性項の影響を調査し, その体系化を目指した. このため, 2 次手法の準ニュートン法 (Quasi-Newton method, QN) [1, 20–22] を慣性項とネステロフの加速勾配を用いて大幅に高速化したネステロフの加速準ニュートン法 (Nesterov's Accelerated Quasi-Newton method, NAQ) [32–34] に着目した. NAQ ではいくつかの問題点が存在した. 本研究では, 2 次手法における慣性項の影響を調査するため, NAQ の問題点を克服した新たなアルゴリズムを 4 つ提案した.

まず 1 つ目として 3 章において適応的慣性係数を用いた NAQ, 適応的ネステロフの加速準ニュートン法 (Adaptive Nesterov's Accelerated Quasi-Newton method, AdaNAQ) を提案した. 提案手法 AdaNAQ は NAQ のハイパーパラメータである慣性項のモーメント係数に着目し, その問題点を克服した手法である. 具体的に, NAQ における慣性係数はハイパーパラメータであるため, 学習では全ての反復で固定されていた [34]. これは, 学習を重みの初期値に対して不安定化させ, 複数回の試行において同じ学習精度を得られない原因であった. 従って, モーメント係数の最適値を決めるには多くの実験, 経験と時間を必要とした. この問題を解決するため, 本研究では, これまで凸最適化問題の最適化に対して 1 次手法のネステロフの加速勾配法 (Nesterov's Accelerated Gradient method, NAG) [1, 9, 13–15] で使用されていた, 適応的慣性係数 [13] を NAQ に導入し, AdaNAQ として提案した. さらに, AdaNAQ をニューラルネットワーク (NN) の学習に応用することで, NN における適応的慣性係数の影響に関しても調べた. AdaNAQ の有効性は 2 つの関数近似問題と 2 つのマイクロ波回路のモデリングに対するシミュレーションにより 3.2 節にて示した. シミュレーション結果より, 提案手法は NAQ のハイパーパラメータ問題を解決しただけでなく, NN の学習における収束の安定性と重みの初期値に対する学習の堅牢性 (ロバスト性) も向上した.

2 つ目に 4 章にて慣性付準ニュートン法 (Momentum Quasi-Newton method, MoQ) を提案した. MoQ は, NAQ の各反復における学習時間の増加問題に着目し, その解決策として提案したアルゴリズムである. 具体的に, NAQ は慣性項とネステロフの加速勾配により QN の全体の学習時間および反復回数を大幅に高速化した. しかし, NAQ では, ヘッセ行列の逆近似行列 (**H** 行列) を求めるため, 2 つの勾配, $\mathbf{w}_k$ の点における勾配 (通常勾配) と $\mathbf{w}_k + \mu_k \mathbf{v}_k$ の点における勾配 (ネステロフの加速勾配) の計算を必要とした. 従って, 通常勾配の計算のみを必要とする QN と比較して, NAQ は各反復における計算時間が増加した. 本研究では, この問題を解決するため, 誤差関数が 2 次関数であると見なし, ネステロフの加速勾配を通常勾配の重み付の線形和として近似し, 慣性付勾配 (Momentum Gradient, MoG) とした. 本研究

では, 2 次手法では誤差関数を 2 次のテイラー展開として近似しているため, 誤差関数と 2 次のティラー展開で得られた 2 次関数との間に親和性を見出すことができると考える. さらに, NN の誤差関数は滑らかな関数であるため, 反復点のまわりでは 2 次関数と見なすことが可能であると考える. この 2 つの考え方により, 誤差関数を 2 次関数と見なした場合, その誤差は小さく, 2 次関数と近似した誤差関数の勾配ベクトルは線形関数と見なすことができる. 従って, ネステロフの加速勾配を通常勾配の重み付の線形和として近似することができる. 以上の考え方に基づき, 本研究では, MoG と慣性項を QN に導入し, MoQ とした. MoQ では, **H** 行列の更新が現在と過去 2 反復の勾配により更新できるため, 各反復における計算時間は QN と同様である. MoQ の有効性は 2 つの関数近似問題, 2 つのマイクロ波回路のモデリング問題そして 2 つの分類問題に対する NN の学習に応用し, シミュレーション結果を 4.3 節に示した. 関数近似問題では, 提案手法 MoQ は NAQ の学習性能を維持しながら, 学習時間を高速化した. しかし, 非線形性がより高いマイクロ波回路のモデリング問題では, 固定の慣性係数の値により, 収束率が異なり, ロバストな学習が困難となった. そこで, 適応的慣性係数を MoQ に導入することにより, AdaNAQ の学習性能が維持されながら, 収束の安定性と重みの初期値に対する学習のロバスト性が向上した. 本研究では, 適応的慣性係数を導入した MoQ を適応的慣性付準ニュートン法 (Adaptive Momentum Quasi-Newton method, AdaMoQ) とした. また, AdaMoQ は分類問題でも同様に高い学習精度と高速なパフォーマンスを示した. しかし, MoQ では NAQ と同様に **H** 行列の更新が必要であるため, 計算コストは高く, さらに, MoQ では, 過去の勾配を 1 つ余分に保存する必要があるため, QN そして NAQ と比べてメモリ量も高くなる問題点があった.

3 つ目に 5 章にて記憶制限ネステロフの加速準ニュートン法 (Limited-Memory Nesterov's Accelerated Quasi-Newton method, LNAQ) と慣性付記憶制限準ニュートン法 (Limited-Memory Momentum Quasi-Newton method, LMoQ) を提案した. 提案手法は, NAQ および MoQ の計算コストに着目し, 計算コストを削減するための一つ目のアプローチである. 記憶制限手法 [20–22, 26] は QN に対して提案された応用研究の一つであり, **H** 行列の次元 d を任意記憶量 m に制限する手法である. 記憶制限準ニュートン法 (Limited-Memory Quasi-Newton method, LQN) [20–22, 26] では, $m = 1$ の場合, 計算量が 1 次手法に近いコストとなる. ただし, m の値が小さい場合, 曲率情報である **H** 行列の影響が弱くなるため, 2 次手法と比較して性能が低下する. また, m が増加するに連れて, 曲率情報の影響が強くなるため, 最適解が 2 次手法の解に近づが, 1 反復の計算コストも増加する. このため, 記憶制限手法の計算コストおよび立ち位置は 1 次手法と 2 次手法の狭間に存在するアルゴリズムとなる. 本研究では, NAQ と MoQ に記憶制限手法を導入することにより, 計算コストの削減および LQN の

高速化を目指した. なお, LNAQ と LMoQ の慣性係数には適応的慣性係数を用いた. 提案手法の有効性は, 2 つの関数近似問題と 1 つのマイクロ波回路のモデリング問題に対する NN の学習に応用し, シミュレーション結果を 5.4 節に示した. シミュレーション結果より, 記憶制限手法では, 曲率情報であるヘッセ行列の影響を制限しているため, 非線形問題では最大反復回数以内で 2 次手法と同様な誤差が得られないことがわかった. しかし, 1 次手法との比較では, 速い段階で小さな誤差を得られている. LQN と慣性項を用いた記憶制限手法の比較では, LNAQ と LMoQ は比較に使用した全ての記憶量において LQN を高速化した. また, NAQ と MoQ の関係性は LNAQ と LMoQ でも保たれた. 従って, 記憶制限手法では, マイクロ波回路のモデリング問題のような非線形性が強い問題の学習は困難であるが, 慣性項は 2 次手法と同様に記憶制限手法でも有効であると結論付けられる.

最後に, 6 章にてメモリレスネステロフの加速準ニュートン法 (MemoryLess Nesterov's Accelerated Quasi-Newton method, MLNAQ) とメモリレス慣性付準ニュートン法 (MemoryLess Momentum Quasi-Newton method, MLMoQ) を提案した. この手法は, NAQ と MoQ の計算コストを削減するための 2 つ目のアプローチである. メモリレス手法も記憶制限手法と同様に QN の計算コストを削減するために提案された [30, 89–93]. メモリレス準ニュートン法 (MemoryLess Quasi-Newton method, MLQN) [30, 89–93] では, 過去の行列情報をスケーリングされた単位行列とし, $\mathbf{H}$ 行列をベクトルの内積だけで求める手法である. 従って, その計算コストは 1 次手法とほぼ同様である. 本研究では, MLQN の高速化および NAQ と MoQ の計算コストを削減するため, メモリレス手法を NAQ と MoQ に導入し, MLNAQ と MLMoQ とした. 提案手法の有効性は, 関数近似問題と 3 つの分類問題に対する NN の学習に応用し, シミュレーション結果を 6.5 節に示した. シミュレーション結果より, 提案手法は MLQN を高速化したことがわかる. また, NAQ と MoQ の関係性はメモリレス手法を導入していても保たれた. メモリレス手法と記憶制限手法との比較では, メモリレス手法の反復回数は増加するものの, 各反復における計算時間が記憶制限手法よりも少ないことがわかる. MLNAQ および MLMoQ そして記憶量 m が小さい値の LNAQ および LMoQ では計算コストの削減に成功したが, 提案した 2 つのアルゴリズムで, 曲率情報の影響が強く失われるため, 非線形性が高いマイクロ波回路のモデリング問題では学習が困難となる問題点に繋がった.

上記のように, 本研究では 2 次手法における慣性項の影響を調査するため, 2 次手法の QN に慣性項を導入した. 慣性項は 1 次手法の学習速度に対して良いパフォーマンスを見せたように, 2 次手法においても高速化の影響を及ぼし, 2 次手法のパフォーマンスを向上させた. 従って, 慣性項は 2 次手法においても非常に有効であると結論付けられる. しかし, 記憶制限手法およびメモリレス手法では, 慣性項を用いても通常の 2 次手法と同程度の学習精度で

強非線形問題の最適化が困難であった. これらを踏まえると, 将来, 計算機の性能がさらに向上し, **H** 行列の計算コストが無視できる環境になれば, 慣性項を用いた 2 次手法は強非線形問題の最適化に対して非常に有効なアルゴリズムであると言える. しかし, 現代の計算機において大規模な強非線形問題を高精度かつ高速に処理するため, 慣性項を用いた記憶制限手法およびメモリレス手法を改良させ, 体系化する必要がある.

本研究に対する今後の展望として, アルゴリズムの観点と誤差関数の定義の観点からの改良が考えられる.

まず 1 つ目にアルゴリズムの観点からの改良を考える. 具体的には [98] を手がかりに適応的慣性項を用いた MoQ の収束性の証明に関する検討を行う. さらに, 様々な最適化問題に対する適応的慣性係数を用いた MoQ の有効性を調査する. 次に, 大規模な強非線形問題の学習を目指すため, 提案手法と確率的勾配法 [1] または確率分散縮小勾配法 [99] との融合を検討する. これにより, 大規模な強非線形問題の学習を可能とするミニバッチ学習に基づく新たな学習アルゴリズムの提案を目指す.

そして, 2 つ目に誤差関数の定義の観点からの改良を考える. 強非線形データでは, 一見ノイズに捉えられるデータでも, 実際にはシステムのモデルデータとしては非常に重要な要素になりうることがある. 従って, これを学習可能とする誤差関数の検討を行う必要がある.

以上のようにこれまでの提案手法に対する改良を行うことで, 本研究の目標とする洗練された新たな慣性項を用いた 2 次手法の提案を達成させられると期待できる.

付録

A. B 行列から H 行列の導出法

付録 A では, 2.4.2 節に示した (55) の $\mathbf{B}_{k+1}^{\mathrm{QN}}$ 行列から (56) の $\mathbf{H}_{k+1}^{\mathrm{QN}}$ 行列への導出について説明する.

BFGS 公式を用いた QN では, ヘッセ行列 $\nabla^2 E(\mathbf{w}_k)$ を $\mathbf{B}_k$ 行列で近似することができる. また, ヘッセ行列の逆行列 $\nabla^2 E(\mathbf{w}_k)^{-1}$ を $\mathbf{H}_k$ 行列で近似することも可能である. 従って, 行列 $\mathbf{B}_k$ と $\mathbf{H}_k$ の関係は

$$\mathbf{H}_k = \mathbf{B}_k^{-1}, \tag{A.1}$$

となる. $\mathbf{B}_k$ 行列の逆行列を計算することで $\mathbf{H}_k$ 行列を導出することができる. $\mathbf{H}_k$ 行列の導出を以下に示す [22, 26].

最初に $\mathbf{B}_k$ 行列の更新式を (A.2) に示す.

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} \mathbf{B}_k}{\mathbf{s}_k^{\mathrm{T}} \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}, \tag{A.2}$$

ここでは, $\mathbf{s}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$ そして $\mathbf{y}_k = \nabla E(\mathbf{w}_k) - \nabla E(\mathbf{w}_{k-1})$ である. 次に, (A.3) の逆行列の補助定理であるシャーマン・モリソン・ウッドベリー公式 (Sherman-Morrison-Woodbury Formula, SMW) [100, 101] を考える.

$$(\mathbf{R} + \mathbf{UJC})^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{U}\left(\mathbf{J}^{-1} + \mathbf{C}\mathbf{R}^{-1}\mathbf{U}\right)^{-1}\mathbf{C}\mathbf{R}^{-1}. \tag{A.3}$$

(A.3) では, $\mathbf{R}$, $\mathbf{U}$, $\mathbf{J}$ そして $\mathbf{C}$ は行列積が定義できる適切なサイズである. ここで, ベクトルと行列の積に対する SMW を (A.4) に示す [102].

$$\left(\mathbf{R} + \frac{\mathbf{u}\mathbf{j}^{\mathrm{T}}}{c}\right)^{-1} = \mathbf{R}^{-1} - \frac{\mathbf{R}^{-1}\mathbf{u}\mathbf{j}^{\mathrm{T}}\mathbf{R}^{-1}}{c + \mathbf{j}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{u}}. \tag{A.4}$$

ここで, $\mathbf{R}$ を逆行列を持つ $d \times d$ 次元の行列, $\mathbf{u}$ と $\mathbf{j}$ を d 次元のベクトル, c をスカラーとす

る. $\mathbf{B}_k$ から $\mathbf{H}_k$ を導出するため, SMW のベクトルと行列の積の公式を $\mathbf{B}_{k+1}^{-1}$ に適応する.

$$\begin{aligned}\mathbf{B}_{k+1}^{-1} &= \left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1} + \frac{\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1} \frac{\mathbf{B}_k\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\mathbf{s}_k} \left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1}}{1 - \frac{1}{\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\mathbf{s}_k}\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1}\mathbf{B}_k\mathbf{s}_k} \\ &= \left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1} + \frac{\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1} \mathbf{B}_k\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k \left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\mathbf{s}_k - \mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1}\mathbf{B}_k\mathbf{s}_k}.\end{aligned} \tag{A.5}$$

ここで, (A.5) の右辺の第 1 項に対して再び, (A.4) の SMW 公式を適応すると,

$$\begin{aligned}\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1} &= \mathbf{B}_k^{-1} - \frac{\mathbf{B}_k^{-1}\frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\mathbf{B}_k^{-1}}{1 + \frac{1}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k} \\ &= \mathbf{B}_k^{-1} - \frac{\mathbf{B}_k^{-1}\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k},\end{aligned} \tag{A.6}$$

が得られる. (A.6) を踏まえて, (A.6) の第 2 項を鑑みると分子と分母をそれぞれ (A.7) と (A.8) に式変形できる.

$$\begin{aligned}\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1}\mathbf{B}_k\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1} &= \left(\mathbf{I} - \frac{\mathbf{B}_k^{-1}\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k}\right)\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}\left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}\mathbf{B}^{-1}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k}\right) \\ &= \mathbf{s}_k\mathbf{s}_k^{\mathrm{T}} - \frac{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k}\left(\mathbf{B}_k^{-1}\mathbf{y}_k\mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k\mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\right) + \left(\frac{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k}\right)^2\mathbf{B}_k^{-1}\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1},\end{aligned} \tag{A.7}$$

$$\mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\mathbf{s}_k - \mathbf{s}_k^{\mathrm{T}}\mathbf{B}_k\left(\mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{-1}\mathbf{B}_k\mathbf{s}_k = \frac{\left(\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k\right)^2}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}\mathbf{B}_k^{-1}\mathbf{y}_k}. \tag{A.8}$$

従って, (A.6), (A.7) そして (A.8) を (A.5) にそれぞれ代入し, $\mathbf{B}_k$ を $\mathbf{H}_k$ とすると, $\mathbf{H}_k$ の更新式 (A.9) が得られる.

$$\begin{aligned}\mathbf{H}_{k+1} &= \mathbf{H}_k - \frac{(\mathbf{H}_k\mathbf{y}_k)\mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k(\mathbf{H}_k\mathbf{y}_k)^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k} + \left(1 + \frac{\mathbf{y}_k^{\mathrm{T}}\mathbf{H}_k\mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)\frac{\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k} \\ &= \left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)^{\mathrm{T}}\mathbf{H}_k\left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right) + \left(\frac{\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right).\end{aligned} \tag{A.9}$$

B. 記憶制限準ニュートン法と共役勾配法の関係性

付録 B では, 6.1 節に示した記憶制限準ニュートン法 (LQN) と共役勾配法 (CG) の関係性を考える. 記憶制限手法では記憶量 $m = 1$ のとき, $\mathbf{H}_k^1 = \mathbf{I}$ と置き, 正確な直線探索を実行することで CG に帰着する [22, 26]. 具体的に, (B.1) を踏まえて, $m = 1$ のときの記憶制限手法の $\mathbf{H}_k$ を (B.2) に示す.

$$\mathbf{G}_{k-1} = \mathbf{I} - \frac{\mathbf{s}_{k-1}\mathbf{y}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}, \tag{B.1}$$

$$\begin{aligned}\mathbf{H}_k &= \mathbf{G}_{k-1}^{\mathrm{T}}\mathbf{G}_{k-1} + \frac{\mathbf{s}_{k-1}\mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}} \\ &= \left(\mathbf{I} - \frac{\mathbf{s}_{k-1}\mathbf{y}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\right)\cdot\left(\mathbf{I} - \frac{\mathbf{y}_{k-1}\mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\right) + \frac{\mathbf{s}_{k-1}\mathbf{s}_{k-1}^{\mathrm{T}}}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}.\end{aligned} \tag{B.2}$$

ここで, $\mathbf{s}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$ そして $\mathbf{y}_k = \nabla E(\mathbf{w}_k) - \nabla E(\mathbf{w}_{k-1})$ である. 従って探索方向ベクトル $\mathbf{g}_k$ は,

$$\mathbf{g}_k = -\mathbf{H}_k\nabla E(\mathbf{w}_k) \tag{B.3}$$

$$= -\nabla E(\mathbf{w}_k) + \frac{\mathbf{s}_{k-1}^{\mathrm{T}}\nabla E(\mathbf{w}_k)}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\mathbf{y}_{k-1} + \frac{1}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\left(\mathbf{y}_{k-1}^{\mathrm{T}}\nabla E(\mathbf{w}_k) - \mathbf{s}_{k-1}^{\mathrm{T}}\nabla E(\mathbf{w}_k) - \frac{\left(\mathbf{s}_{k-1}^{\mathrm{T}}\nabla E(\mathbf{w}_k)\right)\left(\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}\right)}{\mathbf{s}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\right), \tag{B.4}$$

となる. ここで, 正確な直線探索を実行すれば, 点 $\mathbf{w}_k$ が $\mathbf{g}_{k-1}$ 方向での誤差関数の最小点になる. 従って, $\mathbf{g}_{k-1}^{\mathrm{T}}\nabla E(\mathbf{w}_k) = 0$ が成立する. ここで, $\mathbf{s}_{k-1} = \alpha_{k-1}\mathbf{g}_{k-1}$ であると考慮すると, 探索方向ベクトルは,

$$\mathbf{g}_k = -\nabla E(\mathbf{w}_k) + \frac{\mathbf{y}_{k-1}^{\mathrm{T}}\nabla E(\mathbf{w}_k)}{\mathbf{g}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\mathbf{g}_{k-1}, \tag{B.5}$$

で与えられる. (B.5) は Hestenes-Stiefel(HS) 公式 [93] を用いた CG の探索方向ベクトルと同様である. 従って, 記憶制限手法では, $m = 1$ の場合は CG, $m = d$ の場合は BFGS 公式を用いた QN に対応する. よって, LQN は CG と QN の中間的な手法であると考えられる.

参考文献

[1] I. Goodfellow, Y. Bengio, and A. Courville, "*Deep learning (adaptive computation and machine learning series)*", MIT Press, November 2016.

[2] S. Hayakin, "*Neural networks and learning machines*", Pearson, 3^{rd} *edition*, November 2008.

[3] H. Kabir, L. Zhang, M. Yu, P. H. Aaen, J. Wood, and Q. J. Zhang, "Smart modeling of microwave devices", *IEEE Microwave Magazine*, vol. 11, no. 3, pp. 105–118, May 2010. doi:10.1109/MMM.2010.936079.

[4] Q. J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, "Artificial neural networks for RF and microwave design-from theory to practice", *IEEE transactions on microwave theory and techniques*, vol. 51, no. 4 pp. 1339–1350, April 2003. doi:10.1109/TMTT.2003.809179.

[5] H. Ninomiya, S. Wan, H. Kabir, Z. Zhang, and Q. J. Zhang, "Robust training of microwave neural network models using combined global/local optimization techniques", *IEEE MTT-S International Microwave Symposium (IMS) Digest*, pp. 995–998, September 2008. doi:10.1109/MWSYM.2008.4633002.

[6] H. Ninomiya, "A hybrid global/local optimization technique for robust training and its application to microwave neural network models", *Journal of Signal Processing*, vol. 14, no. 3, pp. 213–222, May 2010.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998. doi:10.1109/5.726791.

[8] A. Kao, and S. R. Poteet, (Eds.), "*Natural language processing and text mining*", Springer Science+Business Media, 2007. doi:10.1007/978-1-84628-754-1.

[9] S. Ruder, "An overview of gradient descent optimization algorithm", arXiv preprint arXiv:1609.04747, September 2016. doi:10.48550/arXiv.1609.04747.

[10] B. T. Polyak, "*Introduction to Optimization*", Optimization Software, May 1987.

[11] S. K. Zavriev, and F. V. Kostyuk, "Heavy-ball method in nonconvex optimization problems", *Computational Mathematics and Modeling* vol. 4, no. 4, pp. 336–341, October 1993. doi:10.1007/BF01128757.

[12] D. Bertsimas, and J. N. Tsitsiklis, "*Introduction to linear optimization (athena scientific series in optimization and neural computation)*", Athena Scientific, vol. 6, pp. 479–530, February 1997.

[13] Y. Nesterov, "*Introductory Lectures on Convex Optimization: A Basic Course*", Springer Science+Business Media New York, vol. 87, December 2003. doi:10.1007/978-1-4419-8853-9.

[14] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning", *Proc. International conference on machine learning (ICML)*, vol. 28, pp. III−1139–III−1147, June 2013. doi:10.5555/3042817.3043064.

[15] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$", *Doklady an ussr*, vol. 269, no.3, pp. 543–547, 1983.

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *The Journal of Machine Learning Research*, vol. 12, no. 7, pp. 2121–2159, July 2011. doi:10.5555/1953048.2021068.

[17] T. Tieleman, and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[18] M. D. Zeiler, "ADADELTA: An adaptive learning rate method", arXiv print arXiv:1212.5701, December 2012. doi:10.48550/arXiv.1212.5701.

[19] D. P. Kingma, and J. Ba, “Adam: A Method for Stochastic Optimization”, *Proc. 3rd international conference for learning representations (ICLR)*, pp. 1–13, May 2015. doi:10.48550/arXiv.1412.6980.

[20] W. Forst, and D. Hoffmann, “*Optimization - Theory and Practice*”, Springer Science+Business Media, July 2010. doi:10.1007/978-0-387-78977-4.

[21] J. Nocedal, and S. J. Wright, “*Numerical Optimization Second Edition*”, Springer New York, 2^{nd} *edition*, July 2006. doi:10.1007/978-0-387-40065-5.

[22] 矢部 博, 八巻 直一, “非線形計画法”, 朝倉書店, 1999 年 6 月.

[23] 茨木俊秀, 福島雅夫, “最適化の手法”, 情報数学講座 14, 共立出版株式会社, 1993 年 7 月.

[24] D. H. Li, and M. Fukushima, “A modified BFGS method and its global convergence in nonconvex minimization”, *Journal of Computational and Applied Mathematics*, vol. 129, no. 1–2, pp. 15–35, April 2001. doi:10.1016/S0377-0427(00)00540-9.

[25] 福島雅夫, “新版 数理計画入門”, 朝倉書店, 2011 年 2 月.

[26] 矢部 博, “工学基礎最適化とその応用”, 数理工学社, 2006 年 3 月.

[27] J. A. Ford, and I. A. R. Moghrabi, “Alternative parameter choices for multi-step quasi-Newton methods”, *Optimization Methods and Software*, vol. 2, no. 3–4, pp. 357–370, April 1993. doi:10.1080/10556789308805550.

[28] I. A. R. Moghrabi, “Curvature-based quasi-Newton methods for optimization”, *IJPAM*, vol. 119, no. 1, pp. 131–143, June 2018. doi: 10.12732/ijpam.v119i1.11.

[29] J. A. Ford, and S. J. Yull, “A New Multi-Step Quasi-Newton Method for Unconstrained Optimization”, University of Essex, 2007. https://pdfs.semanticscholar.org/0725/f253ec6845bd1d85d8c4f1988ae592ffa5ae.pdf (Last accessed : Augest 12, 2022).

[30] D. F. Shanno, “On the convergence of a new conjugate gradient algorithm,” *SIAM Journal on Numerical Analysis*, vol. 15, no. 6, pp. 1247–1257, December 1978. doi:10.1137/0715085.

[31] C. Kou, and Y. H. Dai, “A Modified Self-Scaling Memoryless Broyden-Fletcher-Goldfarb-Shanno Method for Unconstrained Optimization”, *Journal of Optimization Theory and Applications*, vol. 165, no. 1, pp. 209–224, April 2015. doi:10.1007/s10957-014-0528-4.

[32] 二宮洋, “ネステロフの加速準ニュートン法による学習アルゴリズムの提案”, 信学技法, *IEICE NLP* 2015–141, pp. 87–92, 2016 年 1 月.

[33] H. Ninomiya, “A novel quasi-Newton-Based Training using Nesterov’s Accelerated Gradient for Neural Network”, *Proc. International Conference on Artificial Neural Networks (ICANN)*, Part II, LNCS 9887, pp. 540 ,September 2016.

[34] H. Ninomiya, “A novel quasi-Newton optimization for neural network training incorporating Nesterov’s accelerated gradient”, *NOLTA*, vol. E8–N, no. 4, pp. 289–301, October 2017. doi:10.1587/nolta.8.289.

[35] S. Mahboubi, and H. Ninomiya, “A Novel quasi-Newton with Momentum Training for Microwave Circuit Models using Neural Networks”, *Proc. IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 629–632, December 2018. doi: 10.1109/ICECS.2018.8617883.

[36] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, “A Robust quasi-Newton Training with Adaptive Momentum for Microwave Circuit Models in Neural Networks”, *Journal of Signal Processing*, vol. 24, no. 1, pp. 11–17, January 2020. doi:10.2299/jsp.24.11.

[37] マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ二, 二宮 洋, 浅井秀樹, “準ニュートン法における慣性項の影響に関する研究”, 信学技報, vol. 118, no. 498, *IEICE NLP* 2018–137, pp. 69–74, 2019 年 3 月.

[38] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, “Momentum acceleration of quasi-Newton Training for Neural Networks”, *Proc. The Pacific Rim International Conferences on Artificial Intelligence (PRICAI)*, pp. 268–281, August 2019. doi:10.1007/978-3-030-29911-8_21.

[39] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, “Momentum Acceleration of quasi-Newton based Optimization Technique for Neural Network Training”, *IEICE NOLTA Journal*, vol. E12–N, no. 3, pp. 554–574, July 2021. doi:10.1587/nolta.12.554.

[40] マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ二, 二宮 洋, 浅井秀樹, “適応的慣性項を用いた準ニュートン学習法に関する研究”, *IEICE* 総合大会, N–1, 2020 年 3 月.

[41] マハブービ・シャヘラザード, 二宮 洋, “慣性付 2 次近似勾配モデルを用いた記憶制限準ニュートン法の有 効性に関する研究”, 信学技報, *IEICE NLP*2017–32, pp. 23–28, 2017 年 7 月.

[42] S. Mahboubi, and H. Ninomiya, “A novel training algorithm based on limited-memory quasi-Newton method with Nesterov’s accelerated gradient for neural networks”, *Proc. IARIA FUTURE COMPUTING*, pp. 1–3, February 2018.

[43] S. Mahboubi, and H. Ninomiya, “A Novel Training Algorithm based on Limited-Memory quasi-Newton Method with Nesterov’s Accelerated Gradient in Neural Networks and its Application to Highly-Nonlinear Modeling of Microwave Circuit”, *IARIA International Journal on Advances in Software*, vol. 11, no. 3&4, pp. 323–334, November 2018.

[44] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, T. Kamio, and H. Asai, “A modified limited memory Nesterov’s accelerated quasi-Newton”, 電子情報通信学会 *NOLTA* ソサエティー大会, no. 37, 2021 年 6 月.

[45] マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ, 二宮洋, 浅井秀樹, “慣性付メモリーレス準ニュートン学習法に関する研究”, 電子情報通信学会 *NOLTA* ソサエティー大会, no.13, 2021 年 6 月.

[46] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, “On the study of Memory-Less quasi-Newton Method with Momentum Term for Neural Network Training”, *Proc. IEICE NOLTA Nonlinear Science Workshop*, NLSW–11, December 2021.

[47] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, “On the study of Memory-Less quasi-Newton Method with Momentum Term for Neural Network Training”, *IEICE NOLTA Journal*, vol. 13, no. 2, pp. 271–276, April 2022. doi:10.1587/nolta.13.271.

[48] 岡谷貴之, “深層学習”, 講談社, 2015 年 4 月.

[49] 滝雅人, “これならわかる深層学習入門”, 講談社, 2017 年 10 月.

[50] 天谷 賢治, “工学のための最適化手法入門”, 数理工学社, 2008 年 5 月.

[51] W. S. McCulloch, and W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bulletin of mathematical biophysics*, vol. 5, pp. 115–133, December 1943. doi:10.1007/BF02478259.

[52] F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, *Psychological Review*, vol. 65, no. 6, pp. 386–408, November 1958. doi:10.1037/h0042519.

[53] B. Widrow, and M. E. Hoff, “Adaptive switching circuits”, *Stanford University Stanford Electronics Labs*, *IRE WESCON Convention Record*, vol. 4, pp. 96–104, 1960.

[54] F. Rosenblatt, “*Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*”, Cornell Aeronautical Lab Inc Buffalo NY, March 1961. doi:10.1007/978-3-642-70911-1_20.

[55] M. Minsky, and Seymour Papert, “*Perceptrons: An Introduction to Computational Geometry*”, MIT Press, June 1969.

[56] S. Amari, “Dreaming of mathematical neuroscience for half a century”, *Neural Networks*, vol. 37, pp. 48–51, June 1997. doi:10.1109/PGEC.1967.264666.

[57] D. E. Rumelhart, and J. L. McClelland, “Learning Internal Representations by Error Propagation”, *MIT Press*, pp. 318–362, January 1986. doi:10.1016/B978-1-4832-1446-7.50035-2.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, *Nature*, vol. 323, no. 6088, pp. 533–536, October 1986. doi:10.1038/323533a0.

[59] G. Cybenko, “Approximation by superpositions of a sigmoidal function”, *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, December 1989. doi:10.1007/BF02551274.

[60] S. J. Russell, and P. Norvig, “*Artificial intelligence a modern approach*”, Pearson Education, 3^{rd} *edition*, December 2009.

[61] D. Marr, “A theory of cerebellar cortex”, *Journal of Physiology*, vol. 202, no. 2, pp. 437–470, Jun 1969. doi:10.1113/jphysiol.1969.sp008820.

[62] J. S. Albus, “Theory of Cerebellar Function”, *Mathematical Biosciences*, vol. 10, no. 1/2, pp. 25–61 , February 1971. doi:10.1016/0025-5564(71)90051-4.

[63] M. Ito, M. Sakurai, and P. Tongroach, “Climbing fibre induced depression of both mossy fibre responsiveness and glutamate sensitivity of cerebellar Purkinje cells”, *Journal of Physiology*, vol. 324, pp. 113–134, March 1982. doi:10.1113/jphysiol.1982.sp014103.

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ”ImageNet classification with deep convolutional neural networks”, *Proc. NeurIPS*, vol. 25, pp. 1097–1105, December 2012. doi:10.1145/3065386.

[65] Computational Science Education Project, “*Mathematical optimization*”, Virtual book, 1995. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://aiinfinance.com/GenMathOpt.pdf (Last accessed : January 21, 2023).

[66] M. J. Kochenderfer, and T. A. Wheeler, “*Algorithms for Optimization*”, MIT Press, March 2019.

[67] M. Riedmiller, and H. Braun, “Rprop-A fast adaptive learning algorithm”, *ISCIS VII*, 1992.

[68] 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, December 2010.

[69] H. Ninomiya, “Microwave neural network models using improved online quasi-Newton training algorithm”, *Journal of Signal Processing*, vol. 15, no. 6, pp. 483–488, November 2011.

[70] H. Sharma, and Q. J. Zhang, “Transient electromagnetic modeling using recurrent neural network”, *IEEE MTT-S International Microwave Symposium Digest*, pp. 1597–1600, June 2005. doi:10.1109/MWSYM.2005.1517009.

[71] W. J. R. Hoefer, and P. P. M. So, “*The MEFiSTo-2D Theory*”, Victoria, BC, Canada: Faustus Scientific Corporation, April 2001.

[72] D. Kumar, “Design of planar filters using fractal geometry and EBG structures”, University of Delhi, 2012.

[73] *Sonnet*, Full-Wave 3D Planar Electromagnetic Field Solver Software for High Frequency EM Simulation, Sonnet Software, Inc. http://www.sonnetsoftware.com (Last accessed : Augest 18, 2022).

[74] A. M. Buchanan, and A. W. Fitzgibbon, “Damped newton algorithms for matrix factorization with missing data”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 316–322, July 2005. doi:10.1109/CVPR.2005.118.

[75] A. Beck, and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”, *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, March 2009. doi:10.1137/080716542.

[76] B. O’Donoghue, and E. Candes, “Adaptive restart for accelerated gradient schemes”, *Foundations of Computational Mathematics (FoCM)*, vol. 15, no. 3, pp. 715–732, June 2015. doi:10.1007/s10208-013-9150-3.

[77] K. Fukumizu, and S. I. Amari, "Local minima and plateaus in hierarchical structures of multilayer perceptrons", *Neural Networks*, vol. 13, no. 3, pp. 317–327, May 2000. doi:10.1016/S0893-6080(00)00009-5.

[78] H. Ninomiya, "Distributed robust training of multilayer neural networks using normalized risk-averting error", *Proc. IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind and Brain (CCMB)*, pp. 134–140, December 2014. doi:10.1109/CCMB.2014.7020706.

[79] H. Ninomiya, "Dynamic sample size selection based quasi-Newton training for highly nonlinear function approximation using multilayer neural networks", *Proc. IEEE & INNS International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, August 2013. doi:10.1109/IJCNN.2013.6706976.

[80] N. Benoudjit, C. Archambeau, A. Lendasse, J. Leel, and M. Verleysen, "Width optimization of the Gaussian kernels in radial basis function networks", *Proc. European Symposium on Artificial Neural Networks (ESANN)*, pp. 425–432, April 2002.

[81] Y. Cao, G. Wang, and Q. J. Zhang, "A New Training Approach for Parametric Modeling of Microwave Passive Components Using Combined Neural Networks and Transfer Functions", *IEEE Transactions on Microwave Theory and Techniques*, vol. 57, no. 11, pp. 2727–2742, October 2009. doi:10.1109/TMTT.2009.2032476.

[82] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, and H. Asai, "Implementation of a Modified Nesterov's Accelerated Quasi-Newton Method on Tensorowflow", *Proc. IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1147–1154, December 2018. doi:10.1109/ICMLA.2018.00185.

[83] D. Gao, N. Ruan, and W. Xing, editors, "*Advances in Global Optimization*", Springer Proceedings in Mathematics & Statistics, November 2014. doi:10.1007/978-3-319-08377-3.

[84] E. Alpaydin, and C. Kaynak, "Optical recognition of handwritten digits data set", *UCI Machine Learning Repository*, vol. 64, no. 5620, July 1998.

[85] D.P. Kroese, Z. Botev, T. Taimre, and R. Vaisman, "*Data Science and Machine Learning: Mathematical and Statistical Methods*", Chapman and Hall/CRC, November 2019.

[86] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, and H. Asai, "A stochastic quasi-newton method with Nesterov's accelerated gradient", *Proc. Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, Springer, Cham, Part I, pp. 743–760, September 2019. doi:10.1007/978-3-030-46150-8_43.

[87] Y. LeCun, C. Cortes, and C. J. C. Burges, "THE MNIST DATABASE of handwritten digits", http://yann.lecun.com/exdb/mnist (Last accessed : Augest 18, 2022).

[88] J. Bassey, X. Li, and L. Qian, "An Experimental Study of Multi-Layer Multi-Valued Neural Network", *Proc. IEEE International Conference on Data Intelligence and Security (ICDIS)*, pp. 233–236, June 2019. doi: 10.1109/ICDIS.2019.00043.

[89] M. S. Apostolopoulou, D. G. Sotiropoulos, I. E. Livieris, and P. Pintelas, "A memoryless BFGS neural network training algorithm", *IEEE International Conference on Industrial Informatics*, pp. 216–221, June 2009. doi:10.1109/INDIN.2009.5195806.

[90] C. X. Kou, and Y. H. Dai, " A Modified Self-Scaling Memoryless Broyden–Fletcher–Goldfarb–Shanno Method for Unconstrained Optimization", *Journal of Optimization Theory and Applications*, vol. 165, pp. 209–224, March 2014. doi:10.1007/s10957-014-0528-4.

[91] S. Nakayama, "A hybrid method of three-term conjugate gradient method and memoryless quasi-Newton method for unconstrained optimization", *Sut Journal of Mathematical*, vol. 54, no. 1, pp. 79–98, June 2018. doi:10.55937/sut/1547386572.

[92] 成島 康史, 中山 舜民, 矢部 博, “無制約最適化問題に対するメモリーレス準ニュートン法について”, 応用数理, vol. 29, no. 4, pp. 8–17, December 2019. doi:10.11540/bjsiam.29.4_8.

[93] S. Nakayama, “Memoryless quasi-Newton methods for large-scale unconstrained optimization”, 東京理科大学学術リポジトリ, June 2019. doi:10.20604/00002308.

[94] M. R. Hestenes, and E. Stiefel, “Methods of conjugate gradients for solving linear systems,” *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, December 1952. doi:10.6028/jres.049.044.

[95] Tensorflow, https://www.tensorflow.org. (Last accessed : Augest 18, 2022)

[96] Google Colablatory, https://colab.research.google.com. (Last accessed : Augest 18, 2022)

[97] https://github.com/ninomiyalab/Memory_Less_Momentum_Quasi_Newton. (Last accessed : Augest 18, 2022)

[98] H. F. Walker, and P. Ni, “Anderson acceleration for fixed-point iterations”, *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, May 2011. doi:10.1137/10078356X.

[99] R. Johnson, and T. Zhang, “Accelerating Stochastic Gradient Descent using Predictive Variance Reduction”, *Advances in Neural Information Processing Systems*, vol. 1, no. 3, pp. 315–323, December 2013. doi: 10.5555/2999611.2999647.

[100] J. Sherman, and W. J. Morrison, “Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix (abstract)”, *Annals of Mathematical Statistics*, vol. 20, no. 4, pp. 621, December 1949. doi:10.1214/aoms/1177729959.

[101] J. Sherman, and W. J. Morrison, “Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix”, *Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, March 1950. doi:10.1214/aoms/1177729893.

[102] M. S. Bartlett, “An Inverse Matrix Adjustment Arising in Discriminant Analysis”, *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 107–111, March 1951. doi:10.1214/aoms/1177729698.

研究業績

博士論文に関連する学術論文

[1] S. Mahboubi, R. Yamatomi, S. Indrapriyadarsini, H. Ninomiya and H. Asai, “On the study of Memory-Less quasi-Newton Method with Momentum Term for Neural Network Training.”, *IEICE NOLTA Journal*, vol. 13, no. 2, pp. 271–276, April 2022. [6 章]

[2] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya and H. Asai, “Momentum Acceleration of quasi-Newton based Optimization Technique for Neural Network Training.”, *IEICE NOLTA Journal*, vol. E12–N, no. 3, pp. 554–574, July 2021. [4 章]

[3] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya and H. Asai, “A Robust quasi-Newton Training with Adaptive Momentum for Microwave Circuit Models in Neural Networks.”, *Journal of Signal Processing*, vol.24, no. 1, pp. 11–17, January 2020. [3 章]

上記以外の研究業績

• 学術論文誌（査読あり）

筆頭著者

[4] S. Mahboubi and H. Ninomiya, “A Novel Training Algorithm based on Limited-Memory quasi-Newton Method with Nesterov’s Accelerated Gradient in Neural Networks and its Application to Highly-Nonlinear Modeling of Microwave Circuit.”, *IARIA International Journal on Advances in Software*, vol. 11, no. 3&4, pp. 323–334, November 2018. [5 章]

その他

[5] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya and H. Asai, “Accelerating Symmetric Rank-1 Quasi-Newton Method with Nesterov’s Gradient for Training Neural Networks.”, *MDPI Algorithms journal*, vol. 15, no. 1:6, pp. 1–16, January 2022.

[6] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya and H. Asai, “A Nesterov’s Accelerated quasi-Newton method for Global Routing using Deep Reinforcement Learning.”, *IEICE NOLTA Journal*, vol. E12–N, no. 3, pp. 323–335, January 2022.

[7] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya and H. Asai, “aSNAQ: An adaptive stochastic Nesterov’s accelerated quasi-Newton method for training RNNs.”, *IEICE NOLTA Journal*, vol. 11, no. 4, pp. 409–421, January 2020.

• 国際学会プロシーディング（査読あり）

筆頭著者

[8] S. Mahboubi, R. Yamatomi, S. Indrapriyadarsini, H. Ninomiya and H. Asai, "On the study of Memory-Less quasi-Newton Method with Momentum Term for Neural Network Training.", *Proc. IEICE NOLTA Nonlinear Science Workshop*, NLSW–11, Online, December 2021. [6 章]

[9] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya and H. Asai, "Momentum acceleration of quasi-Newton Training for Neural Networks.", *Proc. The Pacific Rim International Conferences on Artificial Intelligence (PRICAI)*, vol. 11671, pp. 268–281, Yanuca Island, Cuvu, Fiji, August 2019. [4 章]

[10] S. Mahboubi and H. Ninomiya, "A Novel quasi-Newton with Momentum Training for Microwave Circuit Models using Neural Networks.", *Proc. IEEE International Conference on Electronics Circuits and Systems (ICECS)*, pp. 629–632, Bordeaux, Paris, December 2018. [3 章]

[11] S. Mahboubi and H. Ninomiya, "A novel training algorithm based on limited-memory quasi-Newton method with Nesterov's accelerated gradient for neural networks.", *Proc. The International Academy, Research and Industry Association (IARIA), FUTURE COMPUTING*, pp. 1–3, Barcelona, Spain, February 2018. [5 章]

その他

[12] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, T. Kamio and H. Asai, "A Stochastic Momentum Accelerated Quasi-Newton Method for Neural Networks.", *Proc.* 36^{th} *Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence, Student Abstract*, pp. 12973–12974, Online, February 2022.

[13] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, T. Kamio and H. Asai, "A Nesterov's Accelerated quasi-Newton method for Global Routing using Deep Reinforcement Learning.", *Proc. NOLTA Symposium*, pp. 251–254, Online, November 2020.

[14] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, T. Kamio and H. Asai, "A Neural Network Approach to Analog Circuit Design Optimization using Nesterov's Accelerated Quasi-Newton Method.", *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.1, Online, October 2020.

[15] M. D. sudeera H., S. Mahboubi and H. Ninomiya, "Acceleration Technique of Two-Phase Quasi-Newton method with Momentum for Optimization Problem.", *Proc. The International Academy, Research and Industry Association (IARIA), eKNOW*, Online, March 2020.

[16] S. Yasuda, S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya and H. Asai, "A Stochastic Variance Reduced Nesterov's Accelerated Quasi-Newton Method.", *Proc. IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1874–1879, Boca Raton, Florida, USA, December 2019.

[17] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya and H. Asai, "An Adaptive Stochastic Nesterov's Accelerated Quasi Newton Method for Training RNNs.", *Proc. NOLTA Symposium*, pp. 208–211, Kuala Lumpur, Malaysia, December 2019.

[18] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya and H. Asai, "A stochastic quasi-newton method with Nesterov's accelerated gradient.", *Proc. ECML-PKDD, Springer*,vol. 11906, pp. 743-760, Wurzburg, Germany, September 2019. https://ecmlpkdd2019.org/downloads/paper/594.pdf (Last accessed Augest 12, 2022).

[19] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya and H. Asai, “Implementation of a Modified Nesterov’s Accelerated Quasi-Newton Method on Tensorowflow.”, *Proc. IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1147–1154, Orlando, Florida, USA, December 2018. [4 章]

• 国内学会

筆頭著者

[20] マハブービ・シャヘラザード, 二宮 洋, “慣性項による確率的重み差分伝播法の高速化に関する研究”, *IEICE* 信学技報 非線形問題研究会, *NLP*2022–9, pp. 40–45, 大阪, 2022 年 6 月.

[21] マハブービ・シャヘラザード, 山富龍, 二宮 洋, “重み差分伝播法を用いた確率的勾配学習に関する研究”, *IEICE* 信学技報 非線形問題研究会, *NLP*2021–88, pp. 61–66, オンライン, 2022 年 1 月.

[22] マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ, 二宮洋, 浅井秀樹, “慣性付メモリーレス準ニュートン学習法に関する研究”, 電子情報通信学会 *NOLTA* ソサエティー大会, No.13, オンライン, 2021 年 6 月. [6 章]

[23] マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ二, 二宮 洋, 浅井秀樹, “適応的慣性項を用いた準ニュートン学習法に関する研究”, *IEICE* 総合大会, N–1–4, オンライン, 2020 年 3 月. [4 章]

[24] マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ二, 二宮 洋, 浅井秀樹, “準ニュートン法における慣性項の影響に関する研究”, 信学技報, Vol.118, No.498, *IEICE NLP*2018–137, pp. 69–74, 福井, 2019 年 3 月. [4 章]

[25] マハブービ・シャヘラザード, 二宮 洋, “慣性付 2 次近似勾配モデルを用いた記憶制限準ニュートン法の有 効性に関する研究”, *IEICE NLP*2017–32, pp. 23–28, 宮古島, 2017 年 7 月. [5 章]

[26] マハブービ・シャヘラザード, 二宮 洋, “ネステロフの加速勾配を用いた Quickprop 学習法の高速化”, *IEICE* 総合大会, N–1–23, 名古屋, 2017 年 3 月.

その他

[27] 山富龍, マハブービ・シャヘラザード, 二宮 洋, “2 次情報を用いた二重適応縮約型学習アルゴリズムにおけるネステロフの加速勾配の効果に関する研究”, 電子情報通信学会 *NOLTA* ソサエティー大会, NLS–30, 大阪, 2022 年 6 月.

[28] 山富龍, マハブービ・シャヘラザード, 二宮 洋, “Hutchinson 手法に基づく行列対角化を用いた 2 次近似勾配学習法に関する研究”, *IEICE* 信学技報 非線形問題研究会, *NLP*2021–89, pp. 67–70, オンライン, 2022 年 1 月.

[29] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, T. Kamio and H. Asai, “A modified limited memory Nesterov ’ s accelerated quasi-Newton.”, 電子情報通信学会 *NOLTA* ソサエティー大会, No.37, オンライン, 2021 年 6 月. [5 章]

[30] 鮫嶋優太, マハブービ・シャヘラザード, 二宮 洋, “NADIAN：ネステロフの加速ダイナミックニュートン学習法に関する研究”, 電子情報通信学会 *NOLTA* ソサエティー大会, No.12, オンライン, 2021 年 6 月.

[31] 田中和真, マハブービ・シャヘラザード, 二宮 洋, “慣性付き記憶制限準ニュートン法を用い深層強化学習の TensorFlow への実装”, *IEICE* 総合大会, N–1–13, オンライン, 2020 年 3 月.

[32] 田中和真, マハブービ・シャヘラザード, 二宮 洋, “慣性項を用いた記憶制限準ニュートン法による深層強化学習”, *IEICE* 総合大会, N–1–11, オンライン, 2020 年 3 月.

[33] センディルクマール・インドラプリヤダルシ二, マハブービ・シェヘラザード, 二宮洋, 浅井秀樹, “Neural Network を用いた Analog 回路設計”, *IEICE* 総合大会, N–1–7, オンライン, 2020 年 3 月.

[34] 安田壮太, マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ二, 二宮洋, 浅井秀樹, “確率的分散低減を用いたネステロフの加速準ニュートン法”, *IEICE* 総合大会, N–1–5, オンライン, 2020 年 3 月.

[35] 藤瀬和洋, センディルクマール・インドラプリヤダルシ二, マハブービ・シェヘラザード, 二宮洋, 浅井秀樹, “様々な最適化法における Cascade-correlation の性能調査”, *IEEJ* 電子回路研究会, ECT–19–093, 東京, 2019 年 12 月.

[36] 北川信, センディルクマール・インドラプリヤダルシ二, マハブービ・シェヘラザード, 二宮洋, 浅井秀樹, “Knowledge Based Neural Network の概念を用いた学習モデルの効率について”, *IEEJ* 電子回路研究会, ECT–19–086, 東京, 2019 年 12 月.

[37] 佐藤佑哉, マハブービ・シャヘラザード, センディルクマール・インドラプリヤダルシ二, 二宮洋, 浅井秀樹, “ステロフの加速勾配を利用した Barzilai-Borwein 法の高速化”, *IEICE* 総合大会, N–1–17, 東京, 2019 年 3 月.

謝辞

この研究を遂行するにあたり, 終始暖かく見守り, 支え, 丁寧かつ熱心な指導を賜りました指導教員の二宮洋教授に心より深く感謝を申し上げます.

本論文を提出するにあたり, 副査をお願いし, ご助言を賜りました湘南工科大学の渡辺重佳名誉教授, 中上川友樹教授そして佐々木智志准教授に厚く御礼を申し上げます.

共同研究において, お世話になりました静岡大学の浅井秀樹名誉教授そして広島市立大学の神尾武司講師に, 感謝の意を表します.

共同研究において静岡大学のインドラプリヤダルシ二さんは多くの活動で良き友であり, 積極的なサポートをしてくださいました. 心よりお礼申し上げます. また, 湘南工科大学の山富龍さんも本研究の共同研究者として積極的なサポートをしてくださいました. ありがとうございます.

博士課程 3 年間をご支援を賜りました財団法人本庄国際奨学財団ならびに修士課程 2 年間と学部 2 年間の合計 4 年間をご支援を賜りました渡邊來様と脇吉昭様をはじめとする国際ロータリー第 2780 地区茅ヶ崎湘南ロータリークラブの皆様および国際ロータリー第 2780 地区の皆様に心より感謝を申し上げます.

最後に, 大学生活においてお世話になりました皆様, そして私を育て, 人生を支え, サポートをしてくれた家族に心より感謝を申し上げます.