DEIM Forum 2018 P4-6

進化戦略を併用したNeural Networkの重み最適化

清水　洸希†　　小宮山純平††　　豊田　正史††

† 東京大学大学院 情報理工学系研究科 〒 113–8654 東京都文京区本郷 7–3–1
†† 東京大学 生産技術研究所 〒 153–8505 東京都目黒区駒場 4–6–1
E-mail: †{shimizu,jkomiyama,toyoda}@tkl.iis.u-tokyo.ac.jp

あらまし　本論文は, Neural Network (以下 NN) の重みの最適化を, 進化戦略と勾配法とを組み合わせた手法によって行う手法を提案するものである. NN の最適化は一般的に, 確率的勾配降下法 (Stocastic Gradient Descent, 以下 SGD) に基づく誤差逆伝播法 (Back Propagation, 以下 BP) によって行われている. SGD は非常に単純でありながら, 高速かつ高性能な最適化を行うことができる一方で, BP の性質上, 深い NN では, 出力層から遠い層になるほど勾配が消失したり発散したりすることにより, 最適化が適切に行われないという報告がなされている. この問題を解決するために, さまざまなアプローチが取られているが, 本手法では, NN の一層目を進化計算の一種である進化戦略 (Evolution Strategy, 以下 ES) を用い, それ以降の層を SGD を用いた最適化を行った. 評価には MNIST および CIFAR-10 による分類問題を用い, 提案手法が SGD のみを用いた手法を上回ったことを示した.

キーワード　進化計算, 最適化, 機械学習, 深層学習

1. はじめに

計算グラフの一種である NN は, 近年において, 目覚ましい性能を発揮している. 特に, 画像認識や機械翻訳といった分野では, 人間に匹敵する [1], あるいは上回る [2] という結果が得られている. そのような功績を支えているのが, NN の最適化手法の SGD である.

SGD は, NN の損失関数の勾配を求め, その勾配が小さくなる方向へ NN の重みを更新するという非常に単純なアルゴリズムでありながら, 非常に良い性能を発揮している. また, BP より出力層に近い層から順番に最適化を行っていくことで, 高速化・省メモリ化にも成功している. しかし, SGD と BP はその性質上, 勾配消失問題や損失関数の制約 (微分可能性や連続性) の問題が存在する. 勾配消失問題とは, 深い (中間層の多い) NN において, 出力層から遠い層になるほど, 誤差の伝播が滞り, 勾配が消失あるいは発散することで最適化が適切に行われないというものである [3].

本研究ではこの勾配消失問題に着目し, ES による最適化を併用することでその解決を図る. ES とは, 進化計算と呼ばれるアルゴリズムの一種であり, SGD および BP と異なり, NN の重みを一様に最適化することができ, また, 損失関数の制約も受けない. 進化計算とは, 解の候補となる個体群を生成し, 目的関数による評価でよい性能を上げた個体を用いて新たな個体群を生成していくことで最適化を行うものであり, 並列性・解の頑強性・探索範囲に優れるという特徴を持つ [4]. ES は, 解の生成を正規分布に基づいて行い, その確率分布を通して目的関数の最適化を行うという特徴がある. 提案手法では, ES の重みを一様に最適化できるという点に着目し, NN の一層目を ES の一種である CMA-ES (Covariance Matrix Adaptation-ES) [5] によって最適化し, それ以降の層を SGD によって最適化を行う.

CMA-ES は, 確率分布に多変量正規分布を用いることで, 解の探索速度と性能の向上を図ったもので, 一般的な最適化問題においては通常の ES と比べて良い結果が得られている [6]. 一方で, 100 次元程度の最適化問題を対象として設計されたアルゴリズムであることから, 最低でも数千次元を超える近年の NN においてはあまり使用される例がない. 本研究では, GPU を用いた実装を行うことで, 集団 (目的関数の解候補の集合) の大きさを飛躍的に向上させた上で探索速度を維持し, 約 15000 次元の最適化問題にも CMA-ES を適用させることに成功した.

実験では, MNIST と呼ばれる手書き数字のデータセットおよび CIFAR-10 と呼ばれる乗り物と動物の画像データセットによる分類問題を使用し, CMA-ES と SGD との比較, 提案手法と SGD との比較の 2 つの実験を通して, CMA-ES の NN の最適化における有効性と, 提案手法の有効性を示す.

2. 関連研究

NN の重みの最適化を ES によって行った研究は数多くあるが, SGD と ES とを組み合わせた研究は非常に少ない. まず, ES のみを用いた研究を紹介し, 次に SGD と ES とを組み合わせた研究を紹介する.

進化計算による最適化が SGD と同等の性能を残せることを始めた示したものとして, Morse らの研究 [7] が挙げられる. Morse らは, 一般的な進化計算の手法が個体値の評価を全ての学習データ (full-batch) を用いて行うのに対して, mini-batch と呼ばれる少ないトレーニングデータに区切った評価を行うことで, 時系列データを用いた実験においては, 二乗誤差による評価で SGD をも上回る性能を挙げることに成功した. 一方で, 実験に用いた NN の大きさが中間層のユニット数で高々1000 程度と小さいことは著者らも論文中で触れており, 最低でも 7840 のユニット数を用いる本研究とは異なる.

大規模な NN において, ES を用いた最適化で SGD に並んだものとして, Zhang らの研究 [8] が挙げられる. Zhang らは, 正

Algorithm 1 CMA-ES

Initialize $m = 0$ and $C = Diag(1)$
while $gen \leqq MaxGen$ **do**
    Generate new population by multivariate normal distribution
    Evaluate population
    Update mean and covariance of distribution based on fitness
    $gen++$
end while

規分布で生成した重みの更新量として, 目的関数の評価値に応じた報酬をかけたものを用いることで, MNIST の分類問題に対して, SGD と同等の結果を残した. Zhang らが, ES により重みの更新量を生成し, また, ES のみを用いた場合でも SGD と同等の性能を残すことを目指したことに対して, 本研究は, CMA-ES により重みそのものを生成し, SGD と組み合わせることで性能の向上を図ったものであるという点において異なるものである.

SGD と ES を組み合わせたものとして, Magoulas らの研究 [9] が挙げられる. Magoulas らは, SGD によって最適化された重みの値を, Differential Evolution Strategy (DES) を用いて組み換えることで, SGD のみを用いた場合よりもよい結果を残した. Magoulas らの手法が, SGD による重みの値を DES によって調整しているのに対し, 本提案手法は CMA-ES による最適化が SGD の最適化から独立であるという点で異なるものである. また, 対象としている NN の大きさも 500 程度と小さい.

3. 提 案 手 法

提案手法は, SGD と CMA-ES を組み合わせたものであるため, まず CMA-ES について軽く説明を行う. その後, 提案手法のアルゴリズムについて述べる.

3.1 CMA-ES

CMA-ES にはいくつかの種類があるが, 提案手法では Information Geometric Optimization (IGO) [10] に基づく CMA-ES を用いた. IGO は, SGD が重みで定義される目的関数に自然勾配法を適用するのに対して, 重みの確率分布を決定するパラメタ θ で定義される目的関数に対して自然勾配法を適用する. 重み w は, 確率分布 P_{θ^t} によって生成され, CMA-ES の場合, 多変量正規分布 (Multivariate normal distribution) の平均 m および分散共分散行列 C が θ に相当する. m および C は重みとバイアスのセット x_i とその集団における順位によって決定されるパラメタ w によって更新される.

Algorithm 2 提案手法

Initialize all weight and bias
while $e \leqq Epoches$ **do**
    Select mini-batch from training data
    Generate new population (weight and bias of weight1) by multivariate nomal distribution
    Evaluate population by mini-batch
    Update mean and covariance of distribution based on fitness
    Set the best individual as weight1
    Update all weight by SGD
    $e++$
end while

$$m^{t+1} = m^t + \eta_m \sum_{i=1}^{N} \tilde{w}_i (x_i - m^t) \quad (1)$$

$$C^{t+1} = C^t + \eta_c \sum_{i=1}^{N} \tilde{w}_i \left(\frac{(x_i - m^t)(x_i - m^t)^T}{\sigma^t} - C^t \right) \quad (2)$$

$$p_\sigma = p_\sigma + C^{t-\frac{1}{2}} \frac{m^{t+1} - m^t}{\sigma_k} \quad (3)$$

$$\sigma^{t+1} = \sigma^t * \exp\left(\frac{\|p_\sigma\|}{E\|N(0, I)\|} - 1 \right) \quad (4)$$

$$\tilde{w}_i = \frac{1}{N} w\left(\frac{i - 1/2}{N} \right) \quad (5)$$

本研究では, 学習率に相当する η について目的関数の次元数 (CMA-ES によって最適化する重みの要素数) を d として, $\eta_m = 1.0$, $\eta_c = 1.0/(d^2 \sum_i w_i^2)$ とし, w については, 上位 4 割を 1, 下位 6 割を 0 とした.

3.2 SGD と CMA-ES の併用

SGD と CMA-ES の併用に当たっては, Algorithm2 に示したように, 1 層目の重みとバイアスを CMA-ES を用いて最適化し, その後 SGD を用いて全ての層の重みとバイアスを最適化するというステップを繰り返した. CMA-ES による最適化においては, Morse らが提案した mini-batch による手法を用いた.

4. 評価実験と結果

本節では, はじめに実験に用いたデータセットについて述べる. 次に, CMA-ES が NN の最適化において有効であることを示すために, 1 層の NN を用いて CMA-ES による最適化と SGD による最適化との比較を行う. 最後に, 10 層の NN を用いて, 提案手法と SGD のみを用いた手法の比較を行う. また, SGD の一種であり, SGD の学習率 (lr: learning rate) を最適化の度合いに応じて変化させる ADAM (Adaptive Moment Estimation) [11] との比較も行った.

4.1 データセット

評価に当たっては, 2 種類のデータセットによる分類問題を使用した. 1 つめは, MNIST と呼ばれる 0~9 の手書き数のデータセットを使用した. 55000 個のデータを 49500 個の学習データと 5500 個のテストデータに分けて最適化と評価を行った. 2 つめは, CIFAR-10 と呼ばれる 10 クラスの乗り物や動物の RGB

表 1　CMA-ES と SGD との比較 (MNIST)

手法	F1-measure	収束 Epoch
CMA-ES	0.918	32
ADAM	0.924	19
SGD (lr=0.001)	0.907	-
SGD (lr=0.01)	0.922	749

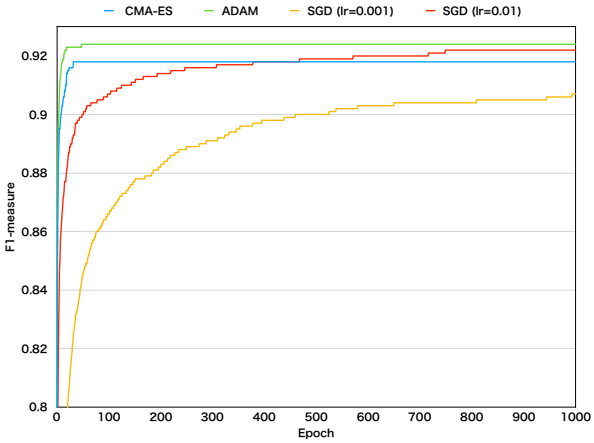

図 1　CMA-ES と SGD との比較 (MNIST)

画像のデータセットを使用した. 本実験では次式を用いて RGB 画像をグレイスケール画像に変換し, 50000 個のデータを 40000 個の学習データと 10000 個のテストデータに分けて最適化および評価を行った.

$$\text{GrayScale} = 0.299 * \text{R} + 0.587 * \text{G} + 0.114 * \text{B} \qquad (6)$$

4.2　CMA-ES と SGD との比較

4.2.1　MNIST

本実験では, CMA-ES の集団の大きさを 5000 とし, mini-batch の大きさは 256 とした. また, SGD については, 学習率を 0.001 および 0.01, バッチサイズを 256 とした. Epoch 数は共通に 1000 回とした. 使用する NN は入力層 784 次元, 出力層 10 次元の 2 層 NN とし, 出力層の活性化関数は softmax 関数, 損失関数には負の対数尤度を用いた.

実験の結果を表 1 および図 1 に示した. F 値と収束 Epoch は, 1000 回の Epoch 中における最大値とそのときの Epoch を表記した. 学習率が 0.001 の SGD については, 1000Epoch 中に収束しなかったため, 収束 Epoch を記さなかった. CMA-ES は SGD および ADAM には及ばなかったものの, ほぼ同等の性能を残した. Epoch を見ると, 最適化に用いたデータ数あたりの収束速度は ADAM が最も速く, 2 番目に CMA-ES となっており, 比較して SGD は収束がかなり遅くなった. 実際には, CMA-ES は 1Epoch 中に 5000 個の NN の最適化を行っているため, 実行時間あたりでの収束速度では ADAM と SGD が大きく上回った.

4.2.2　CIFAR-10

本実験では, CMA-ES の集団の大きさを 4000 とし, mini-batch の大きさは 256 とした. また, SGD については, 学習率を 0.001 および 0.01, バッチサイズを 64 とした. その他は, 4.2.1 と同様の設定を用いた.

表 2　CMA-ES と SGD との比較 (CIFAR-10)

手法	F1-measure	収束 Epoch
CMA-ES	0.317	33
ADAM	0.299	63
SGD (lr=0.001)	0.312	223
SGD (lr=0.01)	0.311	64

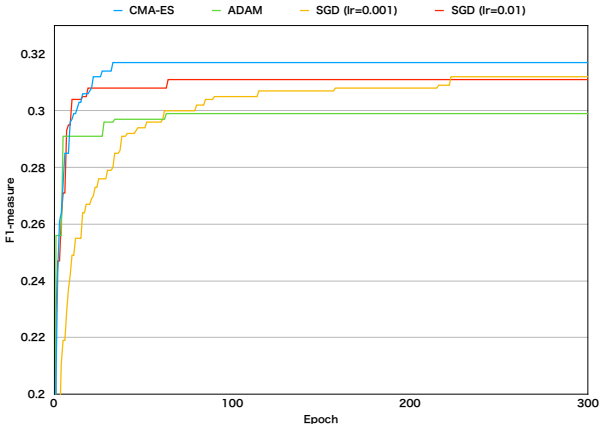

図 2　CMA-ES と SGD との比較 (CIFAR-10)

表 3　提案手法と SGD との比較 (MNIST)

手法	F1-measure	収束 Epoch
提案手法 (SGD)	0.982	10
提案手法 (ADAM)	0.980	18
ADAM	0.982	82
SGD (lr=0.001)	0.963	118
SGD (lr=0.01)	0.970	24

実験の結果を表 2 および図 2 に示した. 4.2.1 と同様に F 値と収束 Epoch を示した. この実験においては, CMA-ES が ADAM と SGD を上回る性能を発揮した. また, SGD が ADAM を大きく上回る結果となった. 収束 Epoch 数についても CMA-ES が最も速く, ついで ADAM, SGD となった.

4.3　提案手法と SGD の比較

4.3.1　MNIST

本実験では, CMA-ES の集団の大きさを 1000 とし, mini-batch の大きさは 256 とした. SGD については, いずれの場合も同じ条件とし, 学習率を 0.01, バッチサイズを 64 に設定した. Epoch 数は共通に 1000 とした. 使用する NN は, 10 層の NN で中間層のユニット数はいずれも 200 で活性化関数は relu 関数を用いた. その他は 4.2 と同様の設定を用いた.

実験の結果を表 3 および図 3 に示した. 提案手法は SGD と組み合わせたとき, SGD のみを用いたものを上回っており, ADAM のみを用いたものと同じ結果を得た. 一方で ADAM と組み合わせた場合は, ADAM のみを用いたものを下回る結果となった. Epoch 数を比べると提案手法は最も速く収束しているが, CMA-ES の 1 バッチにつき SGD により 1Epoch 分最適化を行っているため, 単純な比較はできない.

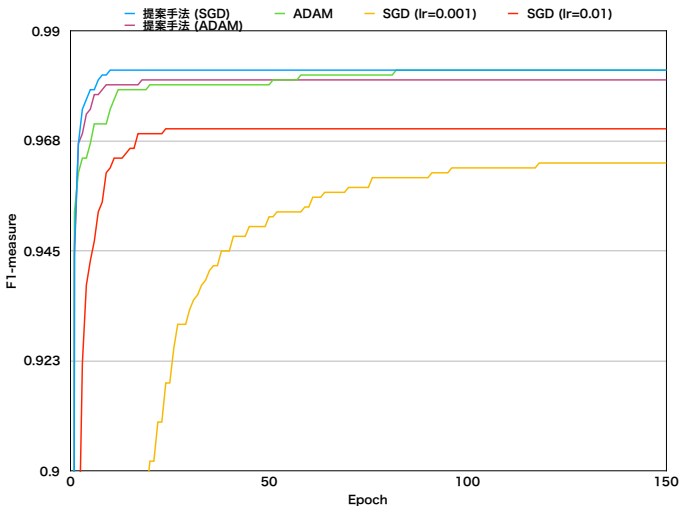

図 3　提案手法と SGD との比較 (MNIST)

表 4　提案手法と SGD との比較 (CIFAR-10)

手法	F1-measure	収束 Epoch
提案手法 (SGD)	0.327	8
ADAM	0.402	27
SGD (lr=0.001)	0.436	164
SGD (lr=0.01)	0.442	40

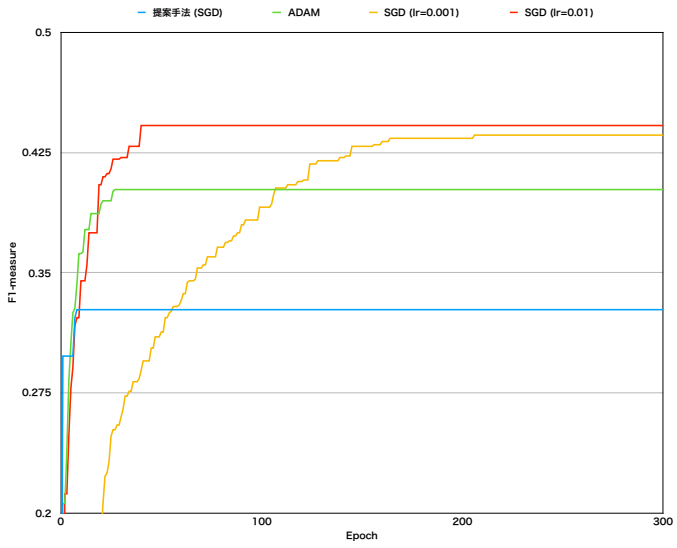

図 4　提案手法と SGD との比較 (CIFAR-10)

4.3.2　CIFAR-10

本実験では, CMA-ES の集団の大きさを 800 とし, mini-batch の大きさは 256 とした. その他は 4.3.1 と同様の設定を用いた.

実験の結果を表 4 および図 4 に示した. 提案手法は, 収束 Epoch は他の手法よりも少ないものの, F 値が他を大きく下回る結果となった. また, 一層のときと同様に SGD が ADAM を上回る結果となった.

4.4　考　　察

4.2.1 および 4.2.2 から, 一層の NN において, MNIST では ADAM が最もよく, ついで SGD, CMA-ES であったのが, CIFAR-10 では真逆に CMA-ES が最もよく, ついで SGD, ADAM となる結果が得られた. このことから, CIFAR-10 は MNIST と比較して, 局所解が多数存在し, CMA-ES は複数のネットワークを集団として保持しながら探索することでそれらの局所解に陥りにくいという仮説が考えられる. ADAM が SGD を下回ったのは, 局所解が多数存在したために, 比較的良い局所解の近傍で学習率が収束してしまったことによるものであると考えられる.

4.3.1 から, 10 層の NN において, 提案手法を SGD と組み合わせた場合は SGD を上回る結果が得られ, ADAM と組み合わせた場合は ADAM を下回る結果が得られた. このことから, SGD は, 多層化された NN においては, 出力層から遠い層の最適化が十分に行えず, CMA-ES と組み合わせたことでこれを解消できたと考えられる. 一方で, ADAM は最適化の度合いに応じて徐々に学習率を小さくする手法であることから, 学習率が ADAM の最適化の過程で小さくなってしまい, CMA-ES による重み行列の分布の変化に対応できなかったと考えられる.

4.3.2 については, 提案手法は良い結果を得られなかったが, 4.2.2 より CMA-ES 自体は SGD を上回っていることから, 最適化する対象の次元数に対して, 集団の大きさが十分に足りていなかったことが原因であると考えられる 4.2.2 において, 10240 次元の最適化に対して 4000 個の NN を用いたのに対して, 4.3.2 では 204800 次元の最適化に対して, メモリの制約上, 800 個の NN しか用いることができなかったため, 十分な最適化が行えなかったと推察される.

5.　お わ り に

本論文では, 多層 NN における SGD の勾配消失問題を解決するために, 1 層目を CMA-ES を用いて最適化を行う手法を提案した. 実験の結果, 提案手法が SGD のみを用いた手法を F 値による評価で上回り, CMA-ES が勾配消失問題を回避できる可能性を秘めていることがわかった. 一方で, 毎回のステップで数千個の NN を生成する必要があることから, SGD と比較して莫大な時間がかかってしまうという問題も生じた. また, CMA-ES が多数の NN の情報を持たなければならないことから, メモリの制約が生じ, 十万を超えるような超高次元の問題に対しては十分な最適化が行えないという結果も得られた.

今後の課題としては, (1) より多くのパラメタ (中間層) を CMA-ES で最適化できるようにする, (2) 今回のような画像データだけではなく言語などの時系列データでの実験を行う, (3)CMA-ES の次元を削減することで一度により多くの NN を評価できるようにする, (4)4.4 で挙げた仮説の検証, といったことが挙げられる.

謝　　辞

本研究は JSPS 科研費 16H02905, 17K12736 の助成を受けたものです.

文　　献

[1] Stanford Vision Lab. ImageNet, 2015.
[2] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva

Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. pp. 1–23, 2016.

[3] Sepp Hochreiter and Paolo Frasconi. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. *A Field Guide to Dynamical Recurrent Networks*, 2009.

[4] Hitoshi Iba. 進化計算と深層学習. *Ohmsha*, 2017.

[5] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. Vol. 102, No. 2006, pp. 75–102, 2016.

[6] Anne Auger, Steffen Finck, Nikolaus Hansen, Raymond Ros, Anne Auger, Steffen Finck, Nikolaus Hansen, Raymond Ros, and Comparison Tables. Comparison Tables : BBOB 2009 Function Testbed To cite this version : HAL Id : inria-00471251 Comparison Tables : BBOB 2009 Function Testbed technique. 2010.

[7] Gregory Morse and Kenneth O. Stanley. Simple Evolutionary Optimization Can Rival Stochastic Gradient Descent in Neural Networks. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*, No. Gecco, pp. 477–484, 2016.

[8] Xingwen Zhang, Jeff Clune, and Kenneth O. Stanley. On the Relationship Between the OpenAI Evolution Strategy and Stochastic Gradient Descent. 2017.

[9] G D Magoulas, V P Plagianakos, and M N Vrahatis. Hybrid methods using evolutionary algorithms for on-line training. *Ijcnn'01: International Joint Conference on Neural Networks, Vols 1-4, Proceedings*, Vol. 3, No. 1, pp. 2218–2223, 2001.

[10] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-Geometric Optimization Algorithms: A Unifying Picture via Invariance Principles. Vol. 18, pp. 1–65, 2011.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. dec 2014.