B-005

自然勾配近似法を用いた大規模並列深層学習における ハイパーパラメータ最適化

長沼 大樹[†1,a)]　岩瀬 駿[†1,b)]　郭 林昇[†1,c)]　中田 光[†1,d)]　横田 理央[†1,e)]

概要：深層学習では極めて冗長な数のパラメータを持つ DNN を膨大な学習データを用いて学習することで他の機械学習手法を圧倒する高い性能を発揮している．一方で，DNN の学習には膨大な計算時間がかかるため，大規模並列化によって学習時間を短縮するのが喫緊の課題である．深層学習で従来用いられてきた確率的勾配降下法 (SGD) では，大規模並列化に伴うバッチサイズの増加により汎化性能が劣化することが報告されている．Martens らによる自然勾配法 [3] を近似した手法である K-FAC[4] では，バッチサイズの増加が汎化性能に影響しないことが示されている．大規模並列学習において，SGD と K-FAC を用いた場合の学習速度や収束速度をそれぞれの最高性能で比較するため，依存するハイパーパラメータの最適化を行い比較検討を行なった．

Naganuma Hiroki[†1,a)]　Iwase Shun[†1,b)]　Kaku Linsho[†1,c)]　Nakata Hikaru[†1,d)]　Yokota Rio[†1,e)]

1. はじめに

深層学習は深層ニューラルネットワーク (以下 DNN) は生物の神経回路を模した計算モデルを用いた機械学習の一手法であり他の機械学習手法と比べて非常に高い学習性能を達成している．一方で，深層学習に用いられるパラメータなどのデータ量の指数関数的増加による学習の長時間化が問題となっており，大規模並列化によって学習時間を短縮するのが喫緊の課題である．

深層学習をはじめとする機械学習の分野においては，特定の学習データセットに偏った性能ではなく，未知のデータに対する汎化性能の向上が最も重要視される．繰り返し行われる学習一回あたりの入力データ量 (バッチサイズ) を大きくし学習時間を短縮する試みは多く行われてきたが，深層学習で従来用いられてきた一次の最適化アルゴリズムである確率的勾配降下法 (以下 SGD) では汎化性能が劣化することが報告されている [1].

本研究では DNN のパラメータ空間 W の集合をリーマン多様体とみなし，勾配計算を行う自然勾配法 (以下 NGD)[3] に着目する．NGD は，勾配計算に用いられるフィッシャー情報行列の逆行列の計算に膨大な時間がかかるため深層学習に用いられてこなかったが，近年，この逆行列を近似的に求める様々な手法が複数提案されている．Martens らが提案したクロネッカー因子分解を用いた近似手法 K-FAC[4] を用いた DNN の学習においては，バッチサイズの増加に対し汎化性能が大きく劣化しないことが実験結果として報告されているが，これらの研究では比較対象を含めてハイパーパラメータの最適化が行われているのか不明瞭である．

そこで本研究では，深層学習における最適化アルゴリズムとしての K-FAC を適切に評価検証するため，深層学習に使われる主流な最適化アルゴリズムである SGD および Adam と K-FAC における学習のハイパーパラメータの最適化を行ない，最適なハイパーパラメータでのバッチサイズへの依存関係を調査した．

†1 現在，東京工業大学
Presently with Tokyo Institute of Technology
a) hiroki11x@rio.gsic.titech.ac.jp
b) shun@rio.gsic.titech.ac.jp
c) kaku.l.aa@rio.gsic.titech.ac.jp
d) nakata.h.ac@rio.gsic.titech.ac.jp
e) rioyokota@gsic.titech.ac.jp

2. 深層学習における最適化アルゴリズム

2.1 確率的勾配降下法 (SGD; Stochastic Gradient Descent)

DNN では入力 $x \in \mathbb{R}^d$ に対し出力 $y = f(x) \in \mathbb{R}^{d'}$ を計算する．DNN の学習においては出力 y と期待される出力 $z \in \mathbb{R}^{d'}$ との不一致度を測る損失関数 $L(y, z)$ を定義し，その期待値を目的関数とする最小化問題を解くために反復的にニューラルネットワークのパラメータ W を更新する．損失関数に関しては，多クラス分類を問題とする場合には交差エントロピーが広く用いられている．W をユークリッド空間上の点と仮定すると，勾配 $\nabla L = \frac{\partial L}{\partial W}$ は L が増加する方向を向いている．したがって反復数 t でのパラメータ $w^{(t)}$ を勾配の逆方向，すなわち $-\nabla L$ 方向へ更新すれば L を減少させることができる．したがってパラメータの更新式は

$$w^{(t+1)} = w^{(t)} - \epsilon \nabla L$$

と表される．ϵ は学習率と呼ばれ，更新の幅を決める係数である．学習率はバッチサイズなどを考慮して決め，反復数などに応じて途中変更を加えるパラメータである．SGD ではミニバッチを用いて ∇L を計算し，反復的にパラメータを更新する．∇L の計算方法としては，出力 y と期待される出力 z の誤差を出力層から入力層にかけて伝播させることで ∇L を計算する誤差逆伝播法 [2] が主流である．

2.2 Momentum SGD

Momentum SDG は，SGD に慣性項 (Momentum) を付与した手法であり，以下の式のように重みの更新を行う．

$$m^{(t)} = \alpha m^{(t-1)} - \epsilon \nabla L \tag{1}$$

$$w^{(t+1)} = w^{(t)} + m^{(t)} \tag{2}$$

このときの α は慣性項のパラメータであり，前回の更新量に α 倍して加算することでパラメータの更新をより慣性的なものにするという狙いがある．

2.3 Adam; Adaptive moment estimation

Adam は学習率を自動で調整する手法であり，ハイパーパラメータが未知の場合でも，SGD やモーメンタム法と比較してより良い局所解を得られる傾向がある．

2.4 自然勾配法 (NGD; Narutal Gradient Descent)

SGD ではパラメータ W をユークリッド空間上の点と仮定し勾配を計算した．これに対し NGD ではパラメータ W の集合をリーマン計量がフィッシャー情報行列 F で定まるリーマン多様体と見なし勾配を計算する．フィッシャー情報行列は正定値対称行列であるため逆行列が存在し，NGD のパラメータの更新式は

$$w^{(t+1)} = w^{(t)} - \epsilon F^{-1} \nabla L \tag{3}$$

で表される．SGD を用いたニューラルネットワークの学習では損失関数 L の期待値を最小化するパラメータを探索するが，鞍点に陥り学習が停滞することがある．NGD は SGD と比べて鞍点に陥りにくいことや反復数に対して学習が速いこと，大きなミニバッチで学習を行っても精度が落ちにくいという特徴がある．一方，昨今のニューラルネットワークではパラメータ数が増加しており，F 及び F^{-1} を正確に計算することはメモリ消費量や計算量の観点から困難である．そこで F^{-1} をスパース行列近似 [5]，低ランク近似 [6]，クロネッカー因子分解を用いた近似 [4]，などを用いて計算する手法が考案されている．

2.5 K-FAC

K-FAC(Kronecker-Factored Approximate Curvature) はクロネッカー因子分解を用いた近似手法として Martens らにより考案された [4]．K-FAC では 2 段階の近似を行うことでメモリ消費量と計算量を抑えている．

2.5.1 近似理論

行列 M に対し $[M]_{i,j}$ は M の (i,j) 成分を，ベクトル v に対し $[v]_i$ は v の i 番目の成分を指すとする．vec を $M \in \mathbb{R}^{m \times n}$ を $v \in \mathbb{R}^{mn}$ に $[v]_{i+n(j-1)} = [M]_{i,j}$ を満たしながら変換するベクトル化関数と定義する．ニューラルネットワークの全層でのパラメータを並べたベクトル θ を

$$\theta = \left[\mathrm{vec}(W_1)^T\ \mathrm{vec}(W_2)^T \cdots \mathrm{vec}(W_l)^T\right]^T \tag{4}$$

とする．$p(y|x,\theta)$ を入力 x に対する出力 y の確率密度関数とし，行列 u に対し

$$Du = -\frac{d \log p(y|x,\theta)}{du} \tag{5}$$

を定義すると，フィッシャー情報行列の定義から

$$F = E\left[\frac{d \log p(y|x,\theta)}{d\theta} \frac{d \log p(y|x,\theta)}{d\theta}^T\right] \tag{6}$$

$$= E\left[D\theta D\theta^T\right] \tag{7}$$

である．$1 \le i,j \le l$ として $F_{i,j} \in \mathbb{R}^{d_{i-1}d_i \times d_{j-1}d_j}$ を

$$F_{i,j} = E[\mathrm{vec}(DW_i)\mathrm{vec}(DW_j)^T] \tag{8}$$

$g_i = DW_i\overline{a_{i-1}}$ とすると，誤差逆伝播法の過程より $DW_i = g_i\overline{a_{i-1}}^T$ である．クロネッカー積 $\otimes$ を用いると，$\mathrm{vec}(uv^T) = v \otimes u$ であることから式 8 は

$$F_{i,j} = E\left[\mathrm{vec}(DW_i)\mathrm{vec}(DW_j)^T\right]$$

$$= E\left[\overline{a}_{i-1}\overline{a}_{j-1}^T \otimes g_i g_j^T\right]$$

ここで $F_{i,j}$ に対して 1 段階目の近似を行い

$$F_{i,j} = E\left[\overline{a}_{i-1}\overline{a}_{j-1}^T \otimes g_i g_j^T\right]$$

$$\approx E\left[\overline{a}_{i-1}\overline{a}_{j-1}^T\right] \otimes E\left[g_i g_j^T\right]$$

$$= \overline{A}_{i-1,j-1} \otimes G_{i,j} \equiv \tilde{F}_{i,j}$$

を得る．ただし $\overline{A}_{i,j} = E\left[\overline{a}_i\overline{a}_j^T\right] \in \mathbb{R}^{d_i \times d_j}, G_{i,j} = E\left[g_i g_j^T\right] \in \mathbb{R}^{d_i \times d_j}$ であり，クロネッカー因子と呼ぶ．

2 段階目の近似として $\tilde{F}$ の非対角ブロックを零行列とする．これにより $\tilde{F}$ は

$$\breve{F} = \mathrm{diag}(\tilde{F}_{1,1}\ \tilde{F}_{2,2}\ \cdots\ \tilde{F}_{l,l}) \tag{9}$$

と近似される．ここでクロネッカー積の性質より

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{10}$$

が成り立つため，

$$\breve{F}^{-1} = \mathrm{diag}(\tilde{F}_{1,1}^{-1}\ \tilde{F}_{2,2}^{-1}\ \cdots\ \tilde{F}_{l,l}^{-1}) \tag{11}$$

$$= \mathrm{diag}(\overline{A}_{0,0}^{-1} \otimes G_{1,1}^{-1}\ \cdots\ \overline{A}_{l-1,l-1}^{-1} \otimes G_{l,l}^{-1}) \tag{12}$$

が従い，クロネッカー因子 $A_{i,i}^{-1}, G_{i,i}^{-1}$ を計算することで $\breve{F}^{-1}$ を計算できる．

3. 関連研究

DNN の大規模並列化によるラージバッチの学習における問題点や，汎化性能の劣化に関する研究が報告されている．

N Keskar[9] らは，分類タスクを問題設定としたラージバッチとスモールバッチによる SGD の学習の比較実験を行い，ラージバッチによる SGD の学習がスモールバッチの場合と比較し，局所最適解の一つである Sharp Minimum に収束し汎化性能を劣化させていることを報告している．スモールバッチの場合は Flat な解を探索していく傾向にあるのに対し，ラージバッチの場合は Sharp な解を探索していく傾向にあり，最適化の振る舞いに違いがあることを主張している．一方，E Hoffer[10] らは，ラージバッチに

よる学習を行った場合，同エポック数で比較すると学習回数が減少するため，十分な学習が行われない点を指摘している．学習率の倍率，学習のスケジューリング，バッチ正規化をバッチサイズに合わせて修正することで，スモールバッチによる学習と同等以上の学習が可能であることを報告している．

いずれも，SGD に関するもので，ユークリッド空間における一次の最適化に関する議論であり，リーマン空間である DNN のパラメータ空間の歪みを考慮する二次の最適化アルゴリズムである NDG を用いた場合の議論は行われていない．

4. ハイパーパラメータの最適化

DNN の学習においては，学習率やそのスケジューリング，バッチサイズや重み減衰などのハイパーパラメータが，その性能に大きく影響するが，ハイパーパラメータの最適化には多くの計算資源を必要とするため，深層学習における主流な最適化手法ではない自然勾配法やその近似手法である K-FAC においては行われてこなかった．

ハイパーパラメータ最適化の手法に関してはグリッドサーチやランダムサーチなどの手法が提案されているが，より効率よく探索を行うために，本研究では，Tree-structured Parzen Estimator(TPE)[8] によってハイパーパラメータ最適化を行う．TPE は，実験的に選択したハイパーパラメータとそのハイパーパラメータの評価 (DNN のテストデータに対する精度など) の組を蓄積し，蓄積されたデータ集合 $\mathcal{H}$ から次に探索するべきハイパーパラメータを提案する手法である．具体的には，あるパラメータ $\gamma \in (0,1)$ を用い，データ集合 $\mathcal{H}$ 中，上位 γ の割合の評価を出すハイパーパラメータ群によるカーネル密度推定関数を $l(x)$，下位 $1-\gamma$ の割合の性能を出すハイパーパラメータ群によるカーネル密度推定関数を $g(x)$ とした時に，$\frac{g(x)}{l(x)}$ を最も小さくする x を次のハイパーパラメータとして提案する．最適化においては，TPE によって提案されたハイパーパラメータについて実験と結果の蓄積を繰り返し，最も良い評価を得られるハイパーパラメータを探索する．カーネル密度推定関数 $l(x)$,$g(x)$ 構成時にデータ集合 $\mathcal{H}$ から確率的にデータを抽出した部分集合を用い，直近に提案したハイパーパラメータを避けることで，同じデータ集合から異なる探索点を提案することで、並列探索が可能である．

5. 実験

実験には，構造が単純な DNN の一つである LeNet5[12] を用い，CIFAR-10[*1]を学習データとしてを用いた．この学習データは、Data Augmentation と等の前処理を行なっている。K-FAC の実装には分散深層学習ライブラリ ChainerMN[*2]v1.3.0 および，NVIDIA が開発したノード間の通信ライブラリである NCCL2[*3]をともに用いた．ハイパーパラメータの最適化のため，データの蓄積に MongoDB[*4]を利用した HyperOpt[11] を用い，図 1 の通り複数の計算機からインターネットを経由して最適化を行う．

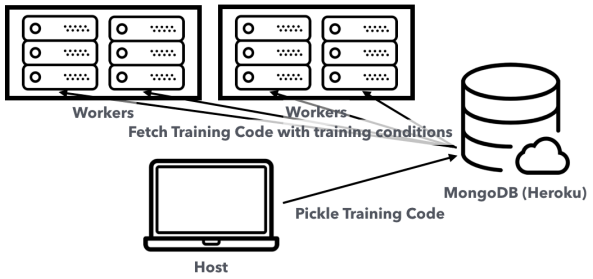

図 1 ハイパーパラメータの最適化実験

実験 1:ハイパーパラメータの最適化

九州大学情報基盤研究開発センターのスーパーコンピュータシステム ITO[*5]および，東京工業大学学術国際情報センターの TSUBAME3[*6]上においてワーカーを起動し，ワーカーは MongoDB からジョブを受け取り，指定の条件下での学習を行う．試行結果から 4 節で紹介した TPE アルゴリズムを用いて，次に行うべきハイパーパラメータ探索の試行条件を決定する．結果を MongoDB に常に蓄積し，汎化精度が最大となるハイパーパラメータの探索を行った．

実験における各最適化アルゴリズム (SGD，Adam，K-FAC) における特定のバッチサイズとその際の汎化精度が最大となる最適なハイパーパラメータの結果を表 1 に示す．

実験 2:ミニバッチサイズ変更

大きいサイズのミニバッチでの汎化精度の収束傾向を調査するため，ミニバッチサイズを 128 から 2048 まで 2 倍ずつ変化させた際の精度の収束を比較した (図 2)．横軸は反復数，縦軸は精度を表している．図からわかる通り，K-FAC では過学習が起きていることが汎化精度が低い原因だと考えられる．

SGD を始め全ての最適化アルゴリズムにおいて，バッチサイズが大きい場合，汎化誤差の収束が速いが，最適なハイパーパラメータを用いていても最終的な汎化誤差はバッチサイズが小さい場合に比べて劣化することが明らかとなった．一方，Adam は SGD に比べ，学習率の初期値や学習スケジューリングを自動的に決定してくれる反面，Flat な解に収束しづらいという問題が指摘されており，最適なハイパーパラメータで比較した結果，今回の実験に用いたデータセットや DNN モデルの規模では SGD が優位である結果となった．

[*1] https://www.cs.toronto.edu/ kriz/cifar.html

[*2] https://github.com/chainer/chainermn
[*3] https://developer.nvidia.com/nccl
[*4] https://www.mongodb.com/
[*5] https://www.cc.kyushu-u.ac.jp/scp/system/ITO/
[*6] http://www.gsic.titech.ac.jp/tsubame3

表 1　バッチサイズによる各種最適化アルゴリズムにおけるハイパーパラメータ

Momentum SGD									
batch size	val-acc	weight decay	lr	lr-decay	lr-decay-epoch	momentum			
128	0.853837	9.45528E-4	7.02133E-3	0.380818	70	0.851752			
512	0.849977	8.59781E-4	1.22424E-2	0.317370	60	0.865659			
2048	0.834328	7.13765E-4	1.13121E-2	0.689114	50	0.945714			
Adam									
batch size	val-acc	weight decay	α	α-decay	α-decay-epoch	β_1	β_2	eta	eps
128	0.818829	0	1.33886E-3	0.397648	60	0.872596	0.999	1.0	1.0E-8
512	0.819347	0	1.22052E-3	0.261723	100	0.874545	0.999	1.0	1.0E-8
2048	0.795179	0	1.26235E-3	0.183551	100	0.869701	0.999	1.0	1.0E-8
K-FAC									
batch size	val-acc	weight decay	lr	lr-decay	lr-decay-epoch	momentum	cov-ema-decay	damping	
128	0.741891	5.0E-5	0.001	0.5	20	0.9	0.99	0.001	
512	0.713202	8.32268E-4	9.42950E-3	0.446998	60	0.641641	0.876205	3.76662E-3	
2048	0.649841	5.0E-5	0.005	0.5	20	0.9	0.99	0.001	

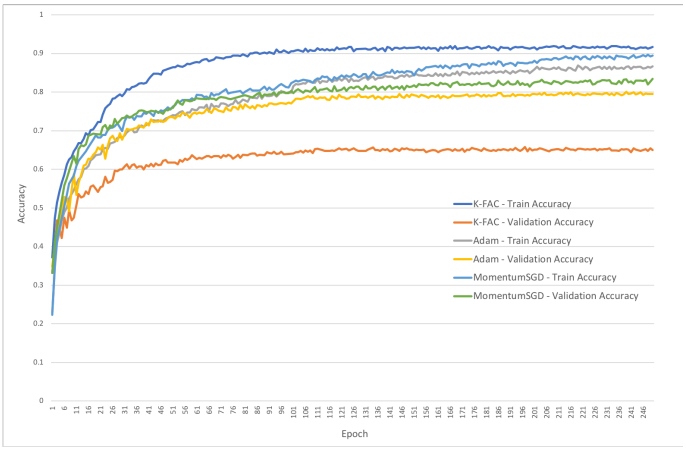

図 2　各最適化アルゴリズムの精度比較 (バッチサイズ=2048)

6.　おわりに

本研究では簡易的な DNN モデルの大きなバッチサイズの学習において，最適なハイパーパラメータの元，汎化精度を評価基準として K-FAC および従来手法の比較検証を行った．簡易的な DNN モデルにおける各種最適化アルゴリズムの特性と，K-FAC において不明瞭であった学習時の最適なハイパーパラメータを提案することができた．一方でさらなる K-FAC を用いた大規模分散学習の調査として以下のような課題がある．

6.1　今後の課題

より多くのパラメータ W をもつ DNN での実験

ResNet[7] などの今回実験に用いた DNN より多くのパラメータ W をもつ DNN に対して，K-FAC を用いて大規模分散学習した場合の精度と収束性を調査する．

より大きな学習データセットでの実験

汎化精度の収束の難しさは学習データのクラス分類数に依存する．本研究では 10 クラス分類の CIFAR-10 を用いたが，ILSVRC12[*7] で用いられた 1000 クラス分類用のデータセットなどより多くのクラス分類を持つ学習データを用いた場合の精度と収束性を調査する必要がある．

K-FAC の収束条件の理論的解明

本研究では実験的に簡易的なモデルでの K-FAC の汎化精度とそのハイパーパラメータについて示したが，その理論的な収束条件を明らかにすることで，さらに大規模並列化に向いた最適化手法の提案が期待される．

謝辞　本研究は，JST CREST JY280144 の支援を受けたものである．

[*7] http://www.image-net.org/challenges/LSVRC

参考文献

[1] Smith S, Kindermans P, et al: *Don't Decay the Learning Rate, Increase the Batch Size*, ICLR, 2018.
[2] Rumelhart D, Hinton G, Ronald J: *Learning representations by back-propagating errors*, Nature323, 1986.
[3] Shunichi A: *Natural Gradient Works Efficiently in Learning*, Neural Computation, 1998.
[4] Martens J, Grosse R: *Optimizing Neural Networks with Kronecker-factored Approximate Curvature*, ICML, 2015.
[5] Grosse R, Salakhudinov R: *Scaling up Natural Gradient by Sparsely Factorizing the Inverse Fisher Matrix*, PMLR, 2015.
[6] Roux N, Manzagol P, Bengio Y: *Topmoumoute online natural gradient algorithm*, NIPS, 2007.
[7] Wu S, Zhong S, Liu Y: *Deep Residual Learning for Image Recognition* Multimedia Tools and Applications, 2017.
[8] Bergstra J, Bardenet R, et al: *Algorithms for Hyper-Parameter Optimization*, NIPS, 2011.
[9] Keskar N, Mudigere D, et al: *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*, ICLR, 2017.
[10] Hoffer E, Hubara I, Soudry D: *Train longer, generalize better: closing the generalization gap in large batch training of neural networks* NIPS, 2017.
[11] Bergstra J, Yamins D, Cox D: *Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms*, SCIPY, 2013.
[12] Lecun Y, Haffner P, et al: *Object Recognition with Gradient-Based Learning*, Shape, Contour and Grouping in Computer Vision, 1999.