1U-07

深層学習における勾配の前処理法に関する検討

石川　智貴 †　　横田　理央 ‡
東京工業大学情報理工学院 †　　東京工業大学　学術情報国際センター ‡

1　はじめに

曲率行列（ヘッセ行列, フィッシャー情報行列,2 次モーメント）による勾配の前処理は深層学習の幅広いタスクにおいて重要な技術となっている．深層学習の最適化を高速化する二次最適化 [1] を始め，継続学習，枝刈り，ベイズ推論など様々な領域において勾配の前処理が使われている．深層学習の二次最適化では曲率行列を近似した上で勾配の前処理を行う．その際の近似アルゴリズムとして様々な種類のものが提案されているが，その特性の違いについてはほとんど調べられていない．そこで，本研究ではメモリ消費量や計算量などの計算的側面と学習の収束性の 2 つの観点からこれらのアルゴリズムを比較する．全体の学習時間は 1step あたりの計算時間と収束にかかるステップ数の積によって決まるため，この 2 つの特性の両方を調べることがとても重要である．

2　二次最適化

深層学習モデル $f(\boldsymbol{x};\boldsymbol{\theta})$ の予測分布を $p_{\boldsymbol{\theta}}(\boldsymbol{t}|\boldsymbol{x})$ とし，入力値 $\boldsymbol{x}$ が従う確率分布を $q(\boldsymbol{x})$ とする．あるサンプル $(\boldsymbol{x},\boldsymbol{t})$ に関する負の対数尤度関数を $\ell(\boldsymbol{x},\boldsymbol{t};\boldsymbol{\theta}) := -\log p_{\boldsymbol{\theta}}(\boldsymbol{t}|\boldsymbol{x})$ とする．この時，ミニバッチに関する経験損失は $\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{t})\in\mathcal{B}} \ell(\boldsymbol{x},\boldsymbol{t};\boldsymbol{\theta})$ と表される．

経験損失のヘッセ行列は

$$\boldsymbol{H} := \nabla^2\mathcal{L} = \left\langle \nabla^2\ell(\boldsymbol{x},\boldsymbol{t};\boldsymbol{\theta}) \right\rangle \tag{1}$$

で定義される．非凸最適化においては，鞍点を避けるために，ヘッセ行列の固有値を絶対値で入れ替えた $\boldsymbol{H}_{|\lambda|}$ がよく使われる．PSGD[5] では，この $\boldsymbol{H}_{|\lambda|}$ を曲率行列として用いている．

フィッシャー情報行列は対数尤度関数の勾配の共分散行列

$$\boldsymbol{F} := \mathbb{E}_{q(\boldsymbol{x})}\left[\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{t}'|\boldsymbol{x})}\left[\nabla \log p_{\boldsymbol{\theta}}\left(\boldsymbol{t}' \mid \boldsymbol{x}\right) \nabla \log p_{\boldsymbol{\theta}}\left(\boldsymbol{t}' \mid \boldsymbol{x}\right)^{\top}\right]\right] \tag{2}$$

で定義される．なお実際には，$\boldsymbol{F}$ の計算における $\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{t}'|\boldsymbol{x})}[\cdot]$ の期待値を Monte-Carlo 近似により推定した $\boldsymbol{F}_{n\mathrm{mc}}$ を用いることが多い．KFAC[2] は，この $\boldsymbol{F}_{n\mathrm{mc}}$ を曲率行列として用いている．

また，フィッシャー情報行列の期待値計算をモデルの予測分布ではなく，学習データに関する経験分布を用いて計算した行列

$$\boldsymbol{F}_{\mathrm{emp}} := \sum_{(\boldsymbol{x},\boldsymbol{t})\in\mathcal{B}} \nabla \log p_{\boldsymbol{\theta}}\left(\boldsymbol{t}' \mid \boldsymbol{x}\right) \nabla \log p_{\boldsymbol{\theta}}\left(\boldsymbol{t}' \mid \boldsymbol{x}\right)^{\top} \tag{3}$$

を経験フィッシャー情報行列という．SMW-NG[6] や SENG[3] は経験フィッシャー情報行列を曲率行列として用いている．なお Adam や Shampoo などの適応的勾配降下法では経験フィッシャー情報行列をミニバッチを用いてオンライン推定した行列

$$\hat{\boldsymbol{F}}_{\mathrm{emp}}^{\mathrm{batch}}(T) := \sum_{t=1}^{T} \alpha_t \boldsymbol{g}_t \boldsymbol{g}_t^{\top} \quad (0 \le \alpha_t \le 1)$$

を用いている．

深層学習においては計算時間やメモリコストの制約から曲率行列を直接計算することができないため曲率行列を低ランク近似する必要がある．SENG，SMW-NG では曲率行列をより低ランクのグラム行列で近似している．PSGD，KFAC，Shampoo では曲率行列を 2 つの行列のクロネッカー積で近似している．

なお，二次最適化では曲率行列の逆行列を計算する必要がある．逆行列計算では Cholesky 分解 (KFAC)，固有値分解 (Shampoo)，SMW 公式 (SENG,SMW-NG) などが使われている．また SKFAC[7] では，バッチサイズの大きさに応じて Cholesky 分解と SMW 公式を使い分けている．

また曲率行列は学習において大きく変化しないことがある．そのため，曲率行列とその逆行列の計算は毎回行わずに，一度計算した行列を再利用することで学習を高速化することができる．

3　実験

3.1　スループットおよびメモリ消費量の比較

KFAC，SKFAC，PSGD，Shampoo，SENG，SMW-NG について，スループットおよびメモリ消費量の比較を行った.[*1] 今回対象とするモデルは MLP[*2]，CNN[*3]，Resnet18，ViT-T である．実験は NVIDIA A100 GPU 上で行った．図 1 に各最適化手法のスループットおよび最大メモリ消費の SGD との比率を示す.

SMW 公式を利用している手法（SENG，SMW-NG，SKFAC）はバッチサイズ $|\mathcal{B}|$ が小さい時はスループットが高く，メモリ消費も小さい．しかし，SENG と SMW-NG は SMW 公式による計算の計算コストがバッチサイズ $|\mathcal{B}|$ に比例して大きくなる．これにより，バッチサイズ $|\mathcal{B}|$ が大きいときは SENG や SMW-NG の計算コストやメモリ消費が大きくなっている．なお ViT において表れるトークン数はバッチサイズに対応する．このトークン数は一般にバッチサイズに比べ大きい．そのため SENG や SMW-NG は ViT ではあまり現実的な手法となっていない．

一方，Shampoo はバッチサイズが小さい時はスループットが低く，メモリ消費の SGD 比も大きい．これは Shampoo の固有値分解の計算時間およびメモリ消費量が大きいことに由来する．しかし，Shampoo の勾配の前処理計算にはバッチサイズに比例する項が含まれないため，バッチサイズ $|\mathcal{B}|$ を大きくすることでスループットおよびメモリ消費量を大きく改善している．

KFAC や PSGD においても，計算量およびメモリ消費の主要項がバッチサイズ $|\mathcal{B}|$ に比例しない項となっている場合が多い．そのため，バッチサイズ $|\mathcal{B}|$ を大きくすることでスループットおよびメモリ消費量を改善することができる．

また曲率行列の計算の頻度削減をすることで二次最適化のスループットが SGD に近づいている．これにより，二次最適化手法ごとのスループットの差が小さくなっている．

3.2　学習における収束性の比較

SGD, AdamW, AdaHessian, KFAC, FOOF, PSGD, Shampoo, SENG について学習の比較実験を行った．タスクは 3 層の MLP（MNIST），Resnet18(CIFAR10) と ViT-T(CIFAR10) である．ViT-T は ImageNet で事前学習されたモデルの finetuning である．ハイパーパラメータ（学習率，ダンピング，指数移動平均減衰率，

A Study on Gradient Preconditioning Methods in Deep Learning
† Satoki Ishikawa, Tokyo Institute of technology
‡ Rio Yokota, Tokyo Institute of technology

[*1] 実験では ASDL(https://github.com/kazukiosawa/asdl) を使用している．
[*2] 3 層の多層パーセプトロンを用いた．中間層の幅は 2048 とした．
[*3] 2 層の Conv2d 層と 2 層の Linear 層からなる 4 層の CNN を用いた．

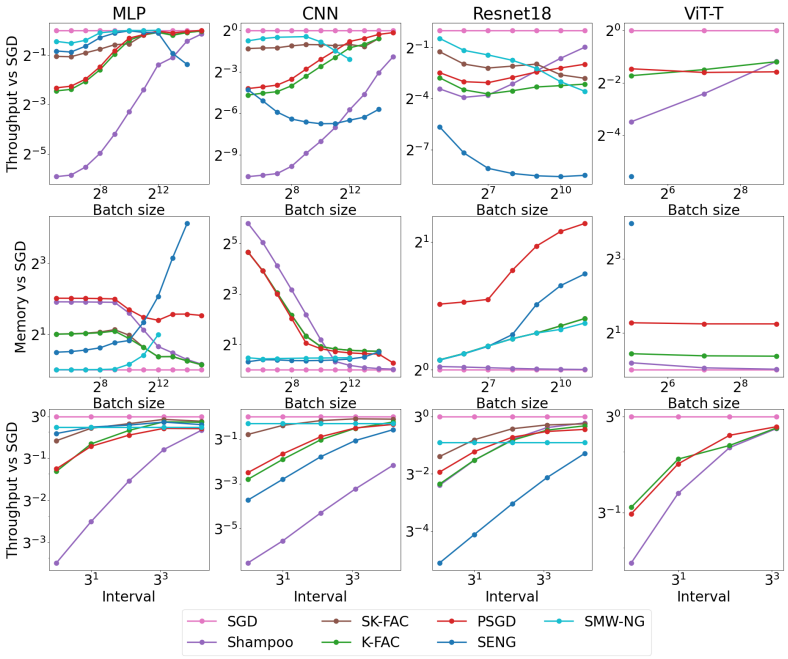

図 1　スループットおよびメモリ消費の SGD 比 (最下段はバッチサイズが 128 の時の結果)

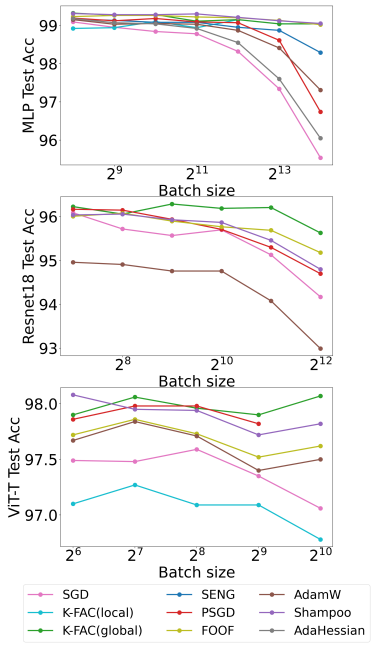

図 2　テスト正解率の比較

勾配クリッピングのノルム，曲率行列の更新頻度）はグリッド探索により十分に探索を行った．データ拡張も用いている．

バッチサイズを変化させた時のテスト正解率を調べた結果が図 2 である．二次最適化手法はバッチサイズの増大に伴う正解率の低下を SGD に比べて抑えることができている．

なお PSGD は MLP においてバッチサイズが 8192 以上になると急激に正解率が減少している．これは PSGD が反復的に曲率行列を計算していることに由来している．バッチサイズが大きい場合，同じ epoch 数で比較するとパラメータ更新の反復数が少なくなるため，曲率行列の質を十分に高めることができていない．

KFAC では曲率行列の計算に指数移動平均を適用する KFAC(local) と適用しない KFAC(global) を区別している．指数移動平均を利用することで過去のミニバッチの情報を考慮して大域的に曲率行列を推定することができる．バッチサイズが小さい場合，KFAC(local) は KFAC(global)，Shampoo，PSGD などの大域的に曲率行列を計算する方法に比べ正解率が低い．このことから，バッチサイズが小さい時は大域的に曲率行列を計算することが大切だと示唆される．なお KFAC にさらに近似を加えた FOOF[8] は今回の例では KFAC に比べ正解率が低い．

ViT-T の finetuning における二次最適化の有効性はほとんど調査されてこなかったが，ViT-T においても二次最適化は有効であることがわかった．また，大域的な曲率行列の推定は ViT-T の finetuning においてとりわけ重要な可能性がある．

4　おわりに

本研究では，二次最適化手法ごとの特性の違いについて調査した．これによりバッチサイズやモデルに応じて使うべき最適化手法が変わってくることが示された．KFAC，Shampoo，PSGD はバッチサイズを大きくすることで計算時間と収束性の両方の観点から，早い学習が実現できることがわかった．また二次最適化は ViT の学習においても有効であるが，計算コストの観点から ViT に適さない手法 (SENG,SMW-NG) があることもわかった．

謝辞

この成果は、国立研究開発法人新エネルギー・産業技術総合開発機構 (NEDO) の助成事業 (JPNP20006) の結果得られたものである．また，本研究を進めるにあたり，大沢和樹博士（ETH Zurich）に有益な助言をいただいた．ここに深謝の意を表す．

参考文献

[1] Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, et al. Large Scale Distributed Second-Order Optimization Using Kronecker-Factored Approximate Curvature for Deep Convolutional Neural Networks, IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR), Jun. 2019.

[2] James Martens, Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In Proceedings of International Conference on Machine Learning. 2015.

[3] Minghan Yang, Dong Xu, Zaiwen Wen, et al. Sketch-Based Empirical Natural Gradient Methods for Deep Learning. Journal of Scientific Computing, September 2022.

[4] Vineet Gupta, Tomer Koren, Yoram Singer. Shampoo: Preconditioned Stochastic Tensor Optimization. In Proceedings of International Conference on Machine Learning (ICML), March 2018.

[5] Xi-Lin Li. Preconditioned Stochastic Gradient Descent. IEEE Transactions on Neural Networks and Learning Systems, 2018.

[6] Yi Ren, Donald Goldfarb. Efficient Subsampled GaussNewton and Natural Gradient Methods for Training Neural Networks. 2019.

[7] Zedong Tang, Fenlong Jiang, Maoguo Gong. SKFAC: Training Neural Networks With Faster Kronecker-Factored Approximate Curvature. IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[8] Frederik Benzing. Gradient Descent on Neurons and its Link to Approximate Second-Order Optimization,2022