近似行列分解と分散深層学習

東京工業大学 学術国際情報センター 横田理央[1]
Rio Yokota
Global Scientific Information and Computing Center
Tokyo Institute of Technology

1　深層ニューラルネットの巨大化

最近の深層ニューラルネットワーク (DNN) の傾向を見ると、個々のタスクに特化した小規模なモデルを皆が冗長に学習するのではなく、大規模なモデルを用いて様々なタスクを一元的かつ継続的に学習する方向に向かっている (図 1)。しかも、このような巨大なモデルの学習においては、膨大な計算資源を有する一部の企業や研究機関のみが桁違いの性能を発揮しており、持てる者と持たざる者の格差が明確になってきている。また、Neural Architecture Search (NAS) により生成される巨大な DNN も、単一のタスクに用いていたのでは宝の持ち腐れであり、マルチタスクの継続学習に用いて初めてその真価を発揮する。さらに、エッジデバイスで推論に用いられる小型の DNN も、このような大型の DNN の蒸留、剪定、量子化によって生成されるものが最も高精度になる傾向があるため、超高精度・超大型の DNN を継続的に学習する技術は AI 分野における国際競争力の源泉となる。

巨大な DNN の継続学習は夢のある話ではあるが、言うほど簡単ではない。モデルが巨大になれば分散並列処理が必須になり、データ並列処理では収束性が悪化し、モデル並列処理では通信がボトルネックとなる。また、一つの DNN モデルに異なるタスクを継続的に学習させようとした場合、過去に学習した内容の致命的な忘却・干渉が起きる。これらに加えて、深層学習自体の根本的な問題として、ハイパーパラメータのチューニングに膨大な時間がかかることも忘れてはならない。つまり、巨大な DNN の継続学習には以下の 3 つの大きな課題が残っている。

1. 分散並列深層学習の収束性の向上
2. 継続学習における致命的忘却の抑制
3. ハイパーパラメータの予測

これら 3 つの課題を解決する上で鍵となるのが、巨大な行列を高速に解くために必要な線形代数分野と高性能計算分野の先端技術である。

2　巨大行列の近似解法

高性能計算分野で盛んに研究されてきた巨大な密行列の行列分解に関する種々の近似解法は、上記の 3 つの問題を解決できる方法でありながら深層学習分野でこれまで全く注目されていない。図 2 に示すように、クロネッカー因子分解 [8] を用いれば行列分解の演算量を $\mathcal{O}(N^3)$ から $\mathcal{O}(N^{3/2})$ に、H 行列 [5] を用いればこれをさらに $\mathcal{O}(N \log^2 N)$ に低減できる。深層学習分野では、これらの技術が十分に活用できていないために、

1. 収束性を向上させることができる二次最適化で用いられるヘッセ行列やフィッシャー行列

[1] rioyokota@gsic.titech.ac.jp

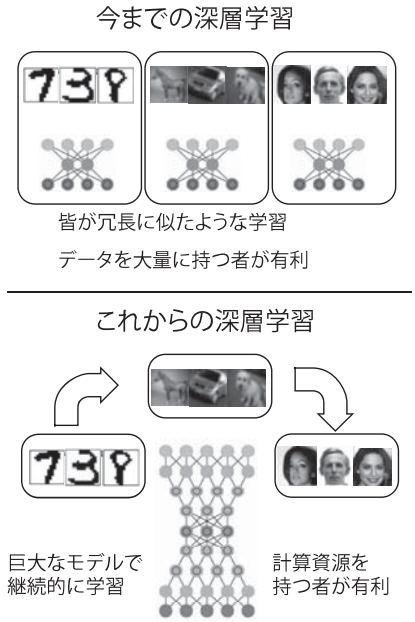

図 1: 深層学習におけるパラダイムシフト

2. 継続学習における致命的忘却を抑制するのに用いられるフィッシャー行列
3. 学習率やバッチサイズなどのハイパーパラメータの予測に用いられるヘッセ行列

などの計算が、本当は無視できる程度のオーバーヘッドで行える可能性があることがあまり知られていない。このため、既存研究ではこれらの巨大な密行列に対して、対角近似や低ランク近似などの非常に粗い近似を行っている。これに対して、クロネッカー因子分解や H 行列などを用いると高い精度を維持したまま計算量を $\mathcal{O}(N^3)$ から $\mathcal{O}(N^{3/2})$ や $\mathcal{O}(N\log^2 N)$ に低減できる。

3 巨大行列のクロネッカー分解

入力データとラベルを $(\mathbf{x}, \mathbf{t})$ とし、損失関数

$$\mathcal{L}(\boldsymbol{\theta}) = E\left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{t}|\boldsymbol{x})\right] \tag{1}$$

を定義する。ただし、$\boldsymbol{\theta}$ はニューラルネットの重み $\mathbf{w}$ とバイアス b からなるパラメータのベクトル、$E[\cdot]$ はミニバッチに関する期待値、$p_{\boldsymbol{\theta}}(\boldsymbol{t}|\boldsymbol{x})$ はそのニューラルネットが出力する確率分布である。このとき、η を学習率としたときの確率的勾配降下法は

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta\nabla\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right) \tag{2}$$

によってパラメータを更新する。これに対して、自然勾配法 [2] ではフィッシャー行列 $\mathbf{F}$ による前処理を行うことで

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta(\boldsymbol{F} + \lambda I)^{-1}\nabla\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right) \tag{3}$$

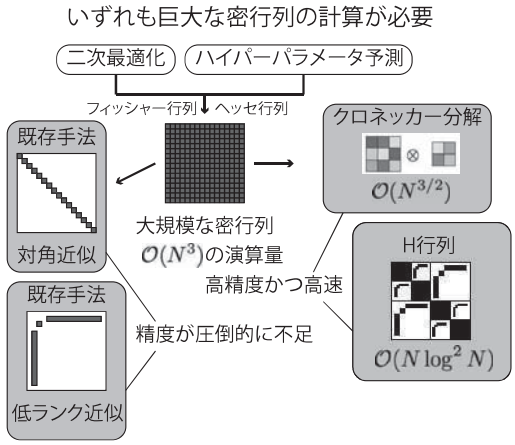

図 2: 深層学習に現れる巨大行列とその高速解法

によってパラメータを更新する。ただし、λ はフィッシャー行列を正則化するためのダンピング係数で、フィッシャー行列は

$$\boldsymbol{F} := \mathbb{E}_{\boldsymbol{x}\sim q}\left[\mathbb{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}}\left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})^{\mathrm{T}}\right]\right] \tag{4}$$

のように定義される。ただし、$\mathbb{E}_{\boldsymbol{x}\sim q}[\cdot]$ は真の分布 q からサンプリングされたデータ $\mathbf{x}$ に関する期待値、$\mathbb{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}}[\cdot]$ は分布 $p_{\boldsymbol{\theta}}$ からサンプリングされた出力 $\mathbf{y}$ に関する期待値を表す。ここで、$\mathbb{E}_{\boldsymbol{x}\sim q}[\cdot]$ をミニバッチごとの期待値で近似し、$\mathbb{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}}[\cdot]$ はクラスに関するモンテカルロサンプリングを行うと、フィッシャー行列の ℓ 層目の対角ブロックは

$$\hat{\boldsymbol{F}}_\ell := E\left[\nabla_{\boldsymbol{\theta}_\ell} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}_\ell} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})^{\mathrm{T}}\right] \tag{5}$$

のように近似できる。ただし、$\nabla_{\boldsymbol{\theta}_\ell}$ は ℓ 層目のパラメータでの微分である。これを用いて、ℓ 層目のパラメータの更新は

$$\boldsymbol{\theta}_\ell^{(t+1)} \leftarrow \boldsymbol{\theta}_\ell^{(t)} - \eta\left(\hat{\boldsymbol{F}}_\ell^{(t)} + \lambda \boldsymbol{I}\right)^{-1} \nabla_{\boldsymbol{\theta}_\ell}\mathcal{L}^{(t)} \tag{6}$$

のように書ける。また、ℓ 層目の入力を $\mathbf{a}_{\ell-1} \in \mathbb{R}^{d_{\ell-1}}$、出力を $\mathbf{s}_\ell \in \mathbb{R}^{d_\ell}$ とするとフィッシャー行列のクロネッカー因子はそれぞれ

$$\begin{aligned}\mathbf{G}_\ell &:= E\left[\nabla_{s_\ell} \log p_\theta(\boldsymbol{y}|\boldsymbol{x})\nabla_{s_\ell} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})^{\mathrm{T}}\right] \\ \mathbf{A}_{\ell-1} &:= E\left[\boldsymbol{a}_{\ell-1}\boldsymbol{a}_{\ell-1}^{\mathrm{T}}\right]\end{aligned}$$

のように定義され、フィッシャー行列はこれらを用いて

$$\hat{\boldsymbol{F}}_\ell = \boldsymbol{G}_\ell \otimes \boldsymbol{A}_{\ell-1} \tag{7}$$

のようにクロネッカー分解できる。フィッシャー行列の次元が $\hat{\boldsymbol{F}}_\ell \in \mathbb{R}^{d_\ell d_{\ell-1}\times d_\ell d_{\ell-1}}$ なのに対して、クロネッカー因子の次元は $G_\ell \in \mathbb{R}^{d_\ell\times d_\ell}, A_{\ell-1} \in \mathbb{R}^{d_{\ell-1}\times d_{\ell-1}}$ であり、逆行列の計算量が $\mathcal{O}(N^3)$ から $\mathcal{O}(N^{3/2})$ に低減できることが分かる。

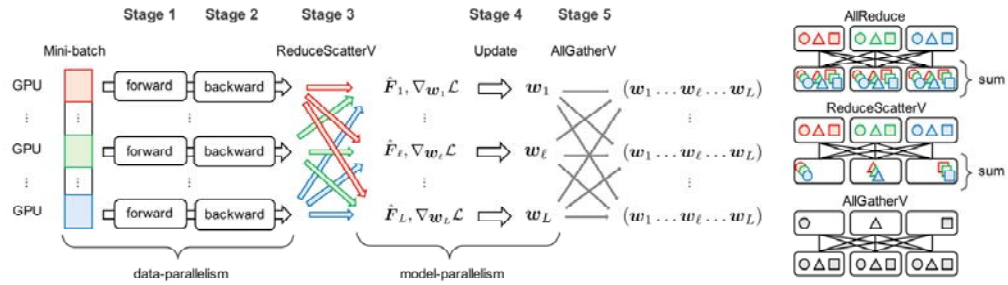

図 3: 分散並列の自然勾配近似法

4 巨大行列の分散並列解法

上記の近似解法を用いただけでは、巨大な行列計算のオーバーヘッドは十分に低減できない。ただし、層ごとのブロック対角近似を行い、各層を分散並列処理する方法をとると、通信を全く行うことなく行列計算の時間を並列数に反比例して低減させることができる。分散並列深層学習では、データを複数の GPU に分散させることで一気にたくさんのデータを学習させる方式が一般的である。このようなデータ並列化では、GPU 数に反比例して反復数（計算時間）が減少する。図 3 にデータ並列の自然勾配近似法の計算と通信の様子を示す。層ごとのブロック対角化によるフィッシャー行列 F の分散並列計算では、まず通常のミニバッチ勾配降下法と同様に forward と backward の計算を行う。その後、ReduceScatterV の集団通信によってデータ並列からモデル並列への切り替えを行い、上記の反復数の減少は維持したまま行列計算のオーバーヘッドをさらに削減することができる。さらに、フィッシャー行列で前処理された勾配ベクトルを層ごとに異なる GPU で更新し、全 GPU へ全層の情報を集約するために AllGatherV を用いて通信する。ここで、勾配降下法と自然勾配法の通信量の違いに着目すると、勾配の ReduceScatterV と AllGatherV は勾配降下法でも行う通信であるため、自然勾配法特有の通信はフィッシャー行列の ReduceScatterV のみであることが分かる。

5 ImageNet への適用例

深層学習において巨大なフィッシャー行列を計算するオーバーヘッドを定量的に評価するために ImageNet を用いた ResNet-50 の学習を行った。これは、単に自然勾配法の収束性や汎化性能を評価するというよりは、クロネッカー分解と分散並列化を行うことで勾配降下法と比べたときのオーバーヘッドをどこまで低減できているかを検証することが目的である。

計算は産総研の ABCI 上で 2048GPU まで使用して、1GPU あたり 1 プロセスで実行した。バッチサイズは GPU あたり 32 とし、4,096 から 131,072 まで 2 のべき乗で増大させ、top-1 validation accuracy が 75%を超えるまで学習を行った。自然勾配法は勾配降下法に比べて強い正則化を用いる必要があるため、通常の画像の random crop、水平方向の回転、白色化、バッチ正規化 [6] などに加え、ラベル平滑化 [11]、重み減衰の改良、重みの正規化などを行った。

これまで、自然勾配法は勾配降下法と比べて収束は早いものの、汎化性能が低くなることが懸念されていた。また、データ並列による分散並列学習ではバッチサイズの増大により、汎化性能が低下することも報告されている [7]。これらの疑念を払拭するために大きなバッチサイ

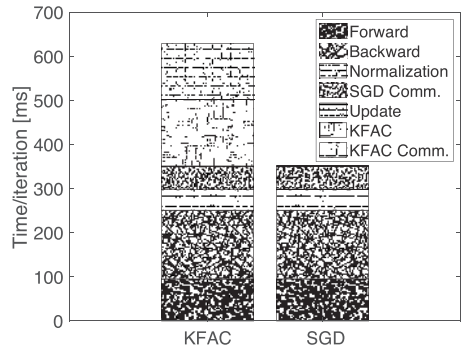

図 4: 自然勾配近似法 (KFAC) と勾配降下法 (SGD) の計算時間の内訳

ズにおける自然勾配法による ImageNet[3] の学習結果を表 1 に示す。これまで、ImageNet による ResNet-50 の分散並列学習は勾配降下法を用いて盛んに行われてきたが [1, 4, 9, 10]、いずれも収束までに 90 epoch を要していた。表 1 にあるように、自然勾配法を用いることでバッチサイズ 16,384 までは同等の精度まで 35 epoch で収束させることができる。さらに、バッチサイズを増大させた場合でもその効果はある程度認められ、過去最大のバッチサイズである 131,072 を用いても 100 epoch で同じ精度まで収束させることができた。

図 4 に自然勾配近似法 (KFAC) と勾配降下法 (SGD) の計算時間の内訳の比較を示す。凡例を見ると分かるように、Forward、Backward、(Batch) Normalization、AllReduce (SGD Comm.)、Update の計算は両手法に共通の部分である。自然勾配近似法ではこれに加え、フィッシャー行列のクロネッカー因子の期待値の計算及びその行列分解の計算 (KFAC) とそれらの行列の通信 (KFAC Comm.) がある。図 4 では、このオーバーヘッドの部分が全体の半分程度を占めているが、これは分散並列化により計算時間を大幅に低減させた後での結果である。

6 まとめ

これまでの深層学習では、巨大な行列の計算をできる限りさける傾向があったが、高速なアルゴリズムと分散並列処理により、そのオーバーヘッドを大幅に低減できることが分かった。収束性を向上させることができる二次最適化で用いられるヘッセ行列やフィッシャー行列、継続学習における致命的忘却を抑制するのに用いられるフィッシャー行列、学習率やバッチサイズなどのハイパーパラメータの予測に用いられるヘッセ行列などはいずれも巨大な密行列で

表 1: ImageNet, ResNet-50 の学習におけるバッチサイズと収束までの反復数の関係

Batch size	Epoch	Iteration	Accuracy
4,096	35	10,948	$75.1 \pm 0.09\%$
8,192	35	5,434	$75.2 \pm 0.05\%$
16,384	35	2,737	$75.2 \pm 0.03\%$
32,768	45	1,760	$75.3 \pm 0.13\%$
65,536	60	1,173	$75.0 \pm 0.09\%$
131,072	100	978	$75.0 \pm 0.06\%$

あり、ここで述べた計算技術を用いて精度を損なうことなく僅かなオーバーヘッドで計算できる。

謝辞

本研究は JSPS 科研費 JP18H03248, JP19J13477 の助成を受けたものである。本研究は、JST、CREST、JPMJCR19F5 の支援を受けたものである。本研究は、学際大規模情報基盤共同利用・共同研究拠点の支援による (課題番号: jh190043-NAHI)。本研究の一部は HPCI システム利用研究課題の成果によるものである (課題番号:hp190112)。

参考文献

[1] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes. *http://arxiv.org/abs/1711.04325*, 2017.

[2] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. page 8, 2012.

[4] Priya Goyal, Piotr Dollar, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *http://arxiv.org/abs/1706.02677*, 2017.

[5] W. Hackbusch. A sparse matrix arithmetic based on H-matrices, part I: Introduction to H-matrices. *Computing*, 62:89–108, 1999.

[6] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. page 11, 2015.

[7] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[8] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

[9] Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala, Yoshiki Tanaka, and Yuichi Kageyama. ImageNet/ResNet-50 Training in 224 Seconds. page 8, 2018.

[10] Chris Ying, Sameer Kumar, Dehao Chen, Tao Wang, and Youlong Cheng. Image Classification at Supercomputer Scale. *arXiv:1811.06992 [cs, stat]*, November 2018.

[11] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond Empirical Risk Minimization. *arXiv:1710.09412 [cs, stat]*, October 2017.