How does mini-batching affect curvature information for second order deep learning optimization?

Diego Granziol, Xingchen Wan, Stefan Zohren, Stephen Roberts,
Timur Garipov, Dmitry Vetrov, Andrew Gordon Wilson, BinXin Ru, Sebastien Ehrhardt
{diego, zohren, sjrob}@robots.ox.ac.uk
Machine Learning Research Group, University of Oxford

Abstract

In this paper we consider the effect of using a sub-sample of the data-set on the Hessian spectrum of deep neural networks. We evaluate the perturbations between the full data-set Hessian and batch Hessian analytically, by considering the noise matrix to be a Gaussian Orthogonal Ensemble and using random matrix theory. We also evaluate the differences empirically using iterative methods and the Pearlmutter trick. We show that the batch spectrum is broader, having larger in magnitude both positive and negative maximal eigenvalues. We show that despite un-realistic assumptions our theoretical predictions hold in practice.

1 Introduction

In cases where the optimisation is plagued by locally dependent, tightly coupled parameters, or large variations in scale along different vectors in the parameter space, second-order methods can be expected to significantly outperform first order methods [11]. Many flavours of second order methods have been developed over the years, from iterative methods such as Conjugate Gradient techniques using methods for cheap Hessian vector products [18], generalized Gauss-Newton approximation to the Hessian such as [10, 13], sign changing of the Hessian negative eigenvalues [15, 16], to the recent KFAC algorithm, which proposes a full-rank empirical Fisher matrix approximation [12], and amortizing the cost over several iterations and its variants [6].
In order to remain effective compared to first order counterparts, such as stochastic gradient descent (SGD) [1], second order methods typically operate on a subset of the data (a mini-batch) as opposed to the full dataset [11, 1]. Some authors advocate the use of a smaller batch size for the Hessian vector products than the gradient [4].

Recent convergence analysis on batch second order methods, explicitly requires the sub-sampled Hessian to have the same spectrum as that of the full dataset [19][1]. In general, the spectrum of the matrix sum $\boldsymbol{A} + \boldsymbol{B}$, where $\boldsymbol{B}$ is some perturbing noise matrix, is not unbiased estimate of the spectrum of $\boldsymbol{A}$ [2, 3].

Given the prevalence of mini-batching in second order deep learning, we investigate the effects of the noise incurred due to mini-batching on the spectrum. We do this theoretically, by assuming the noise to be a random matrix and using the law of free addition to compute the perturbed spectrum. We also investigate this phenomenon practically, by using iterative spectral methods with cheap Hessian vector multiplications to approximate the spectra of both the full data-set Hessian and mini-batch Hessian at the same point in weight-space, for different real networks and datasets.

[1] Lemma 1 p.15

Preprint. Under review.

2 Problem Statement

For an input $\boldsymbol{x} \in \mathbb{R}^{d_x}$ and output $\boldsymbol{y} \in \mathbb{R}^{d_y}$ we have a given prediction function $h(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^P \rightarrow \mathbb{R}^{d_y}$, we consider the family of prediction functions parameterised by a weight vector $\boldsymbol{w}$, i.e., $\mathcal{H} := \{h(\cdot;\boldsymbol{w}) : \boldsymbol{w} \in \mathbb{R}^P\}$ with a loss $l(h(\boldsymbol{x};\boldsymbol{w})\boldsymbol{y}) : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \rightarrow \mathbb{R}$. Our true risk is

$$R_{true}(\boldsymbol{w}) = \int_{\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} l(h(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y}) dP(\boldsymbol{x},\boldsymbol{y}) = \mathbb{E}[l(h(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y})] \tag{1}$$

Where the expectation operator $\mathbb{E}$, is taken with respect to the data-generating distribution $P(\boldsymbol{x},\boldsymbol{y})$. The corresponding gradient $\boldsymbol{g}_{true} = \nabla R_{true} \in \mathbb{R}^{1\times P}$ and Hessian $\boldsymbol{H}_{true} = \nabla\nabla R_{true} \in \mathbb{R}^{P\times P}$. For a data-set of size N, we only have access to the empirical risk R_{emp} and the gradients $\boldsymbol{g}_{emp} = \nabla R_{emp}$ and Hessians $\boldsymbol{H}_{emp} = \nabla\nabla R_{emp}$ thereof.

$$R_{emp}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} l(h(x_i;\boldsymbol{w}),y_i) \tag{2}$$

In general, using the entire dataset N for each evaluation of the gradient or Hessian is prohibitively expensive, hence a mini-batching procedure is employed, where we use a batch of size B to compute an approximation to the loss R_{batch}, its derivative ∇R_{batch} and Hessian $\nabla\nabla R_{batch}$.

$$R_{batch}(\boldsymbol{w}) = \frac{1}{B}\sum_{i=1}^{B} l(h(x_i;\boldsymbol{w}),y_i) \tag{3}$$

2.1 Second order methods

All second order methods solve the minimisation problem for the risk, R associated with parameters $\boldsymbol{w}$ and perturbation $\boldsymbol{\delta w}$,

$$\begin{aligned} \boldsymbol{w}^* &= \text{argmin}_{\boldsymbol{w}} R(\boldsymbol{w}+\boldsymbol{\delta w}) \\ R(\boldsymbol{w}+\boldsymbol{\delta w}) &\approx R(\boldsymbol{w}) + \nabla R^T \boldsymbol{\delta w} + \tfrac{1}{2}\boldsymbol{\delta w}^T \boldsymbol{H} \boldsymbol{\delta w} \end{aligned} \tag{4}$$

Where if the risk is not convex, instead of the Hessian $\boldsymbol{H} \in \mathbb{R}^{n\times n} = \nabla\nabla R(\boldsymbol{w})$, a surrogate positive definite approximation to the Hessian $\tilde{\boldsymbol{H}}$ is employed so to make sure the minimum is lower bounded. Examples in the literature include the Generalised Gauss-Newton approximation [10, 13], the Fisher information matrix [12], or the Hessian with the sign of the negative eigenvalues reversed [15, 16]. Its solution is

$$\boldsymbol{\delta}* = -\tilde{\boldsymbol{H}}^{-1}\nabla R(\boldsymbol{w}) = -\sum_{i}^{P} \frac{1}{\lambda_i}\boldsymbol{u}_i\boldsymbol{u}_i^T \nabla R(\boldsymbol{w}) \tag{5}$$

Where $\boldsymbol{u}_i$ correspond to the generalised Hessian eigenvectors. The parameters are updated with $\wp = \wp - \alpha\boldsymbol{\delta}*$. Where α is the global learning rate. Hence as can be seen, if the eigenvalue,eigenvector pairs of $\boldsymbol{H}_{batch}$ differ significantly from $\boldsymbol{H}_{emp}$, then the update $\boldsymbol{\delta}*$ can also differ drastically.

3 The difference between the Empirical and Mini-batch Hessian

In this section we derive theoretical results on the difference between the the eigenvalues of $\boldsymbol{H}_{emp}$ and $\boldsymbol{H}_{batch}$ under some assumptions. Consider the elements of the noise matrix $\boldsymbol{N} = \boldsymbol{H}_{batch} - \boldsymbol{H}_{emp}$

Assumption 1. *The moments of the the elements of $\boldsymbol{H}_{true}$ under the data generating distribution are finite*

Assumption 2. *The data is identically and independently distributed*

Theorem 1. *Under assumption 1, the elements of the matrix $\boldsymbol{N}$ converge to the normal random variables $\mathcal{N}(0, \frac{N-B}{B}\sigma_{j,k}^2)$, where $\sigma_{j,k}^2$ is the variance of the j,k'th element of $\boldsymbol{H}_{true}$ under the data generating distribution.*

Proof. The difference between our empirical and true Hessian is given as

$$[\nabla\nabla R_{batch}(\boldsymbol{w}) - \nabla\nabla R_{emp}(\boldsymbol{w})]_{j,k} = \left(\frac{1}{B}\sum_{i=1}^{B} - \frac{1}{N}\sum_{i=1}^{N}\right)\frac{\partial^2 l(h(x_i;\boldsymbol{w}),y_i)}{\partial \boldsymbol{w}_j \partial \boldsymbol{w}_k} \tag{6}$$

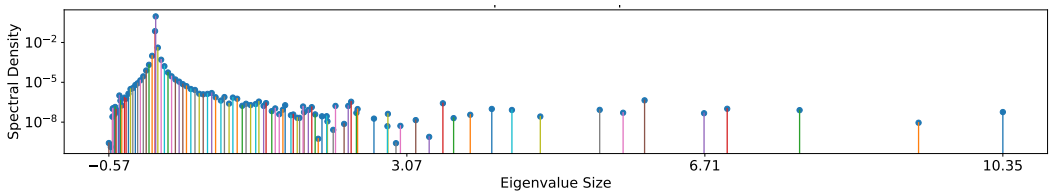

Figure 1: Full data Hessian spectrum for CIFAR-100 VGG16 Epoch 200

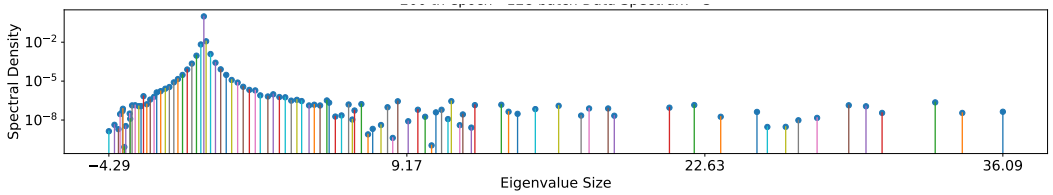

Figure 2: mini-batch data Hessian spectrum for CIFAR-100 VGG16 Epoch 200

which by the central limit theorem [21] converges to a normal random variable, using assumption 2 this variable is zero mean. □

Assumption 3. *We assume that the elements of $\boldsymbol{N}$ are i.i.d Gaussian with common variance $\frac{N-B}{B}\sigma^2$, such a matrix is known as the Gaussian Orthogonal Ensemble (GOE)*

Assumption 4. $\boldsymbol{H}_{emp}$ *is low rank*

Remark. Without assumption 3, the best we can do is use Weyl's inequalites [22] and show that the difference in eigenvalues between the batch and full Hessian is bounded by

$$||\lambda_i - \lambda_i'|| \leq ||\boldsymbol{N}|| = \frac{N-B}{BN}\langle\sigma^2\rangle P^2 \tag{7}$$

where $\langle\sigma^2\rangle P^2$ is arithmetic mean of the $\sigma_{j,k}$. Assumption 4 allows us to use perturbation theory and prove the following theorem.

Theorem 2. *The extremal eigenvalues $[\lambda_1', \lambda_n']$ of the matrix sum H_{batch} (such that assumptions 3 and 4 are satisfied), where H_{emp} has extremal eigenvalues $[\lambda_1, \lambda_n]$, are given by*

$$|\lambda_i'| = \left\{ \begin{array}{ll} |\lambda_i + \frac{N-B}{BN} P \frac{\sigma^2}{\lambda_i}|, & \text{if } |\lambda_i| > \sqrt{\frac{P(N-B)}{BN}}\sigma \\ 2\sqrt{\frac{P(N-B)}{BN}}\sigma, & \text{otherwise} \end{array} \right\}. \tag{8}$$

Proof. See Appendix. □

Remark. Hence for neural networks, where the number of network parameters P is often in the tens or hundreds of millions, and the batch size is typically in the hundreds, hence under the proposed framework, the element variance under the data-generating distribution must be small to not significantly alter the spectrum $P/B << \lambda_i/\sigma^2$

4 Experiments

In order to perform spectral analysis on the Hessian of state of the art deep neural networks with tens of millions of parameters we avoid the infeasible $\mathcal{O}(P^3)$ eigen-decomposition and use the iterative stochastic GPU powered Lanczos quadrature algorithm [5, 14]. We use the Pearlmutter trick [18] for Hessian vector products, with computational cost $\mathcal{O}(PN)$ per iterative step. the Total complexity is $\mathcal{O}(PNm)$ with memory cost of $\mathcal{O}(Pm)$. We exploit Lanczos's relationship to Gaussian quadrature using random vectors to allow us to learn a discrete approximation to the spectral density [14]. We use $m = 100$ for our Experiments. We train a 16 Layer VGG CNN [20] with $P = 15291300$ parameters and 110 Layer ResNet [7] with $P = 1169972$ parameters, architectures on the CIFAR-100 datasets using SGD, with a learning rate of $0.1/0.05$ respectively for $[225, 300]$ epochs with a linearly

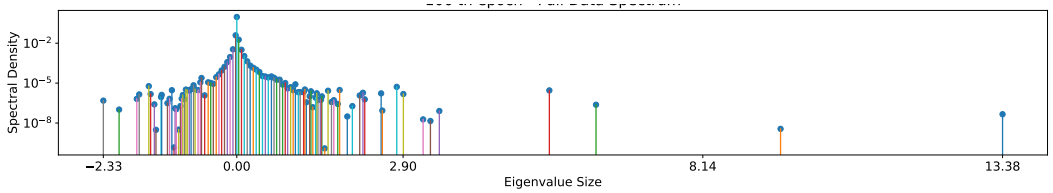

Figure 5: Full data Hessian spectrum for PreResNet110 CIFAR-100 Epoch 100

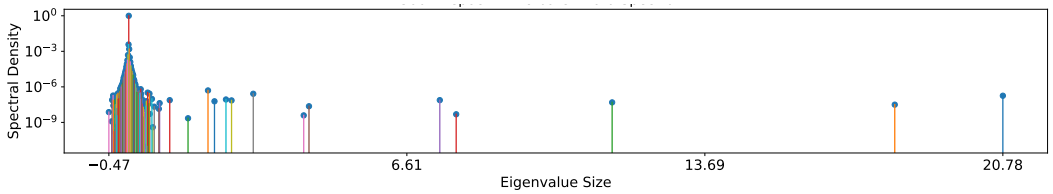

Figure 6: mini-batch data Hessian spectrum for PreResNet110 CIFAR-100 Epoch 100

decaying learning rate from epochs $[126, 161]$ with momentum $\rho = 0.9$ and data-augmentation on Pytorch [17]. We estimate σ^2 for a given point in weight-space $\boldsymbol{w}$ using the methodology advocated in [9], where since the data is i.i.d (assumption 2), hence the individual Hessia per datum are i.i.d.

$$\sigma^2 \approx \frac{1}{NP^2}\sum_{i=1}^{N}||\boldsymbol{H}_i - \boldsymbol{H}_{emp}||^2 = \frac{1}{NP}\sum_{i=1}^{N}\mathbb{E}_v \boldsymbol{v}^t(\boldsymbol{H}_i - \boldsymbol{H}_{emp})(\boldsymbol{H}_i - \boldsymbol{H}_{emp})\boldsymbol{v} \tag{9}$$

Where $\boldsymbol{v}$ is zero mean unit variance random vector, we use the gaussian random vector. The expectation over the set of vectors is approximated with a monte carlo sum, this is known as stochastic trace estimation [8].

4.1 CIFAR-100 CNN Experiments

We display the weighted Ritz value plot for the VGG-16 $\boldsymbol{H}_{emp}$ at epoch 200 in Figure 1 and a $B = 128$ counterpart in Figure 2. Here we see the the spectral broadening taking effect, with the maximum eigenvalue going from 10.35 to 36.09, where the theoretical prediction would be 28.19. We plot the change of σ^2_{vgg} over epoch in Figure 3. We display the weighted Ritz value plot for the PreResNet-110 $\boldsymbol{H}_{emp}$ at epoch 200 in Figure 5 and a $B = 256$ counterpart in Figure 6. Here we see the the spectral broadening taking effect, with the maximum eigenvalue going from 13.38 to 20.78, where the theoretical prediction would be 24.41. We plot the change of σ^2_{resnet} over epoch in Figure 4.

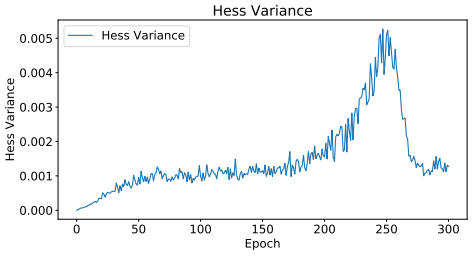

Figure 3: Variance VGG Cifar-100

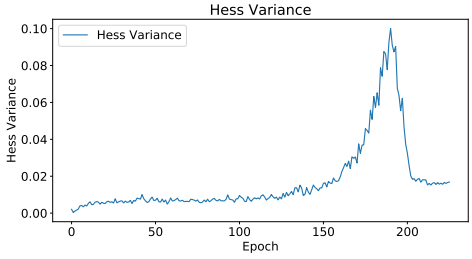

Figure 4: Variance PreResNet110 Cifar-100

5 Conclusion

We investigate the spectral perturbations of mini-batching under a random matrix theory framework and show their effects to be non-negligible. We test our predictions and find them to be in close agreement. This could have implications for the convergence of stochastic second order deep learning methods.

References

[1] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[2] J. Bun, R. Allez, J.-P. Bouchaud, M. Potters, et al. Rotational invariant estimator for general noisy matrices. *IEEE Trans. Information Theory*, 62(12):7475–7490, 2016.

[3] J. Bun, J.-P. Bouchaud, and M. Potters. Cleaning large correlation matrices: tools from random matrix theory. *Physics Reports*, 666:1–109, 2017.

[4] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

[5] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

[6] T. George, C. Laurent, X. Bouthillier, N. Ballas, and P. Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In *Advances in Neural Information Processing Systems*, pages 9550–9560, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[9] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[10] J. Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

[11] J. Martens. *Second-order optimization for neural networks*. PhD thesis, University of Toronto, 2016.

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

[13] J. Martens and I. Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

[14] G. Meurant and Z. Strakoš. The lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, 15:471–542, 2006.

[15] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[16] R. Pascanu, Y. N. Dauphin, S. Ganguli, and Y. Bengio. On the saddle point problem for non-convex optimization. *arXiv preprint arXiv:1405.4604*, 2014.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[18] B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

[19] F. Roosta-Khorasani and M. W. Mahoney. Sub-sampled newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] C. Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, pages 583–602, Berkeley, Calif., 1972. University of California Press.

[22] H. Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479, Dec 1912.