OPT2020: 12th Annual Workshop on Optimization for Machine Learning

Quasi-Newton's method in the class gradient defined high-curvature subspace

Mark Tuddenham MARK.TUDDENHAM@SOTON.AC.UK
Adam Prügel-Bennett APB@ECS.SOTON.AC.UK
Jonathan Hare JSH2@ECS.SOTON.AC.UK
University of Southampton, Southampton, SO17 1BJ, United Kingdom

Abstract

Classification problems using deep learning have been shown to have a high-curvature subspace in the loss landscape equal in dimension to the number of classes. Moreover, this subspace corresponds to the subspace spanned by the logit gradients for each class. An obvious strategy to speed up optimisation would be to use Newton's method in the high-curvature subspace and stochastic gradient descent in the co-space. We show that a naive implementation actually slows down convergence and we speculate why this might be.

1. Introduction

Optimisation is often not a prime consideration for practitioners training deep neural networks. This is because there are many options where the large majority of choices optimise well but getting the best optimisation needs a seemingly arbitrary choice of (almost always first-order) update method, learning rate, decay schedule, momentum, *etc.*

The ideal combination of these options has mainly been brute-force searched for each individual problem: see the research done on speed training *e.g.* DAWNBench [2]. This shows it is difficult to choose the ideal combination when faced with a new problem or a new network architecture. Moreover, as this ideal choice is often dependant on the curvature of the loss landscape, second-order methods, which are designed to capture the local curvature, should be more powerful and robust than first-order methods.

1.1. Second-order methods

Directions of high curvature limit the step size that can be taken in gradient descent. To see this, consider a quadratic minimum defined by a Hessian, H,

$$f(\theta) = \frac{1}{2}(\theta - \theta^*)^\mathsf{T} H\,(\theta - \theta^*). \tag{1}$$

We can write the Hessian matrix as its eigen-decomposition

$$H = \sum_i \lambda_i v_i\, v_i^\mathsf{T} \tag{2}$$

where λ_i is a measure of the curvature in the direction v_i. For θ^* to be a minimum we require $\lambda_i > 0$ for all i. If we perform gradient descent, $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla f(\theta^{(t)}) = \theta^{(t)} - \eta H (\theta^{(t)} - \theta^*)$, then

$$\theta^{(t+1)} - \theta^* = (\mathbf{I} - \eta H) \left(\theta^{(t)} - \theta^*\right) = (\mathbf{I} - \eta H)^t \left(\theta^{(1)} - \theta^*\right) \tag{3}$$

$$= \sum_i v_i (1 - \eta \lambda_i)^t v_i^\mathsf{T} \left(\theta^{(1)} - \theta^*\right). \tag{4}$$

Thus gradient descent will diverge exponentially fast unless $\eta \leq 2/\lambda_{\text{max}}$.

Second-order methods are a form of preconditioned gradient methods using second derivatives; these methods work well for ill-conditioned problems as the Hessian "automatically normalize[s] the ill-conditioned problem by stretching and contracting"[8] the directions corresponding to its eigenvectors. However, the difficulty in calculating accurate second-order information holds this class of methods back from improving general deep learning optimisation. If there are only a small number of directions in weight space with high curvature, then we only need to precondition the gradient in those directions.

1.2. A second-order method

Gur-Ari et al. [6] show that gradient descent is mostly contained in a small subspace, so although the problem is high-dimensional, the optimisation is limited by a low-dimensional high-curvature subspace. That subspace, spanned by the logit gradients, intersects with the top-C eigenvectors of the Hessian [4, 7]. If there are a small number of identifiable directions in weight space with high curvature, then we can significantly speed up optimisation by performing Newton's method in this high-curvature subspace and a standard first-order method in the low-curvature dual subspace.

1.3. Setting

We consider a classification problem with data, $\mathcal{D} = \left\{(x^\mu, y^\mu) \,\middle|\, \mu \in [1, N]\right\}$ where y^μ is a C-dimensional one hot vector, as in Fort and Ganguli [3].

We will look at some deep network with parameters, θ, that calculates logits, z_k^μ, which gives class probabilities, p_k^μ, via a softmax. We will train this network with a cross-entropy loss, $\mathcal{L}^\mu(\theta)$ for an input $(x^\mu, y^\mu) \in \mathcal{D}$. Then, for any single parameter θ_α we have its gradient, $g_\alpha^\mathcal{B}$,[1] for some minibatch

$$g_\alpha^\mathcal{B} = \frac{\partial \mathcal{L}(\theta)}{\partial \theta_\alpha} = \frac{1}{|\mathcal{B}|} \sum_{\mu \in \mathcal{B}} \sum_{k=1}^{C} (p_k^\mu - y_k^\mu) \frac{\partial z_k^\mu}{\partial \theta_\alpha}, \tag{5}$$

and so the Hessian has components

$$H_{\alpha\beta}^\mathcal{B} = \frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta_\alpha \partial \theta_\beta} = \frac{1}{|\mathcal{B}|} \sum_{\mu \in \mathcal{B}} \sum_{k=1}^{C} \sum_{\ell=1}^{C} p_k^\mu \left(\delta_{k\ell} - p_\ell^\mu\right) \frac{\partial z_k^\mu}{\partial \theta_\alpha} \frac{\partial z_\ell^\mu}{\partial \theta_\beta}$$

$$+ \frac{1}{|\mathcal{B}|} \sum_{\mu \in \mathcal{B}} \sum_{k=1}^{C} (p_k^\mu - y_k^\mu) \frac{\partial^2 z_k^\mu}{\partial \theta_\alpha \partial \theta_\beta} \tag{6}$$

1. We derive this as the negative of what Fort and Ganguli [3] state, however, this disparity only affects the parts of the Hessian that we ignore.

Fort and Ganguli [3] claim that the logits are almost a linear function of the weights *i.e.*

$$\frac{\partial^2 z_k^\mu}{\partial\theta_\alpha \partial\theta_\beta} \approx 0 \tag{7}$$

and so the second term of the Hessian, eq. (6), can be ignored [1]. This term will be exactly 0 for a piecewise linear network *e.g.* with ReLUs.

1.4. Class gradients

Fort and Ganguli [3] define the class gradient as

$$c_k = \frac{1}{|\mathcal{D}_k|} \sum_{\mu\in\mathcal{D}_k} \boldsymbol{\nabla} z_k^\mu \tag{8}$$

where $\mathcal{D}_k$ is the set of training examples belonging to class k. The logit gradients can be decomposed as

$$\boldsymbol{\nabla} z_k^\mu = y_k^\mu \, c_k + \epsilon_k^\mu. \tag{9}$$

Substituting the logit gradient decomposition, eq. (9), into the first term of the Hessian, eq. (6), we get

$$H \approx \sum_{k=1}^{C} \left(\frac{1}{|\mathcal{B}|} \sum_{\mu\in\mathcal{B}} y_k^\mu \, p_k^\mu (1-p_k^\mu) \right) c_k \, c_k^\mathsf{T} + \frac{1}{|\mathcal{B}|} \sum_{\mu\in\mathcal{B}} \sum_{k=1}^{C} \sum_{\ell=1}^{C} p_k^\mu \left(\delta_{k\ell} - p_\ell^\mu\right) \epsilon_k^\mu \, \epsilon_\ell^\mu \tag{10}$$

where, following Fort and Ganguli [3], we have ignored the second term in the Hessian and the cross term — as long as a term is sufficiently small the eigensystem of the Hessian remains sufficiently unchanged [1]. Thus a low-rank approximation of the Hessian is the first term in eq. (10), now denoted $H^{\text{Low Rank}}$.

2. Quasi-Newton's method

Assuming that the class gradients are orthogonal we have

$$H^{\text{Low Rank}} = \sum_{k=1}^{C} \lambda_k v_k \, v_k^\mathsf{T}, \tag{11}$$

where $v_k = c_k/|c_k|$ and

$$\lambda_k = \frac{1}{|\mathcal{B}|} \sum_{\mu\in\mathcal{B}} y_k^\mu \, p_k^\mu \, (1-p_k^\mu) \, |c_k|^2. \tag{12}$$

As this is a sum of non-negative elements it is positive definite which is a necessary condition for Newton's method to move downhill. As $H^{\text{Low Rank}}$ is singular it has no inverse, however, we can define a generalised inverse

$$(H^{\text{Low Rank}})^\dagger = \sum_{k=1}^{C} \frac{1}{\lambda_k} v_k \, v_k^\mathsf{T}. \tag{13}$$

We can use this generalised inverse to perform Newton's method in the high-curvature subspace, $\theta' \leftarrow \theta - (H^{\text{Low Rank}})^{\dagger} g^{\mathcal{B}}$. To perform stochastic gradient descent in the orthogonal subspace we can use the projection operator

$$\mathbf{P} = \mathbf{I} - \sum_{k=1}^{C} v_k \, v_k^{\mathsf{T}} \tag{14}$$

applied to the gradient $\theta' \leftarrow \theta - \eta \mathbf{P} g^{\mathcal{B}}$, where η is a learning rate. Then, puttting these two steps together, we get the update equation

$$\theta' \leftarrow \theta - \eta g^{\mathcal{B}} - \sum_{k=1}^{C} \left(\frac{1}{\lambda_k} - \eta \right) (v_k^{\mathsf{T}} g^{\mathcal{B}}) \, v_k. \tag{15}$$

To lower the stochasticity of the estimation of the eigensystem of $H^{\text{Low Rank}}$ we use an exponential moving average and root mean square for the eigenvalues

$$c_k^{(t)} = \frac{1-\gamma}{1-\gamma^t} \sum_{i=1}^{t} \gamma^{t-i} c_k^{\mathcal{B}^{(i)}}, \qquad c_k^{\mathcal{B}^{(t)}} = \frac{1}{|\mathcal{B}_k^{(t)}|} \sum_{\mu \in \mathcal{B}_k^t} \boldsymbol{\nabla} z_k^{\mu}, \tag{16}$$

$$\lambda_k^{(t)} = \sqrt{\frac{1-\gamma}{1-\gamma^t} \sum_{i=1}^{t} \gamma^{t-i} (\lambda_k^{\mathcal{B}^{(i)}})^2}, \qquad \lambda_k^{\mathcal{B}^{(t)}} = \frac{1}{|\mathcal{B}^{(t)}|} \sum_{\mu \in \mathcal{B}^{(t)}} y_k^{\mu} \, p_k^{\mu} \, (1 - p_k^{\mu}) \, |c_k^{\mathcal{B}^{(t)}}|^2, \tag{17}$$

where $\mathcal{B}_k^{(t)}$ is the set of examples in the t^{th} minibatch, $\mathcal{B}^{(t)}$, that belong to class k. This gives us a hyperparameter, γ, which needs to take into account how rapidly the high-curvature subspace changes during learning; Gur-Ari et al. [6] show that the subspace does not markedly change during training, thus, we can set γ close to one. We initialise $c_k^{(0)} = \mathbf{0}$ but $\lambda_k^{(0)} = 1$ since we use $1/\lambda_k$.

3. Comparison to SGD

Figure 1 shows the typical loss when training a ResNet9 (6.5M parameters) with normal SGD and with the quasi-Newton's method, eq. (15) on CIFAR-10. We can see that the final performance is similar; however, the quasi-Newton's method is more volatile, slower to improve in general, and does end on average with lower accuracy, see appendix A. One would expect to be able to use a higher learning rate with the quasi-Newton's method as that applies to the low-curvature subspace, however, this is not true, and both methods diverge at $\eta > 10^{-1}$.

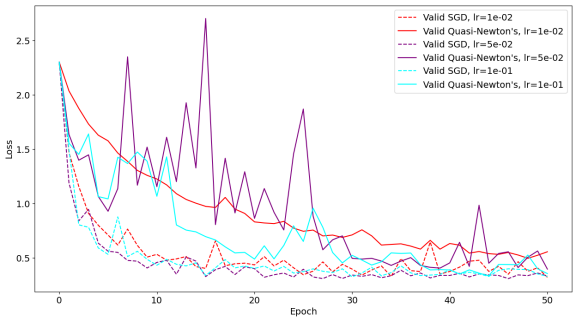


Figure 1: Validation loss for both SGD and quasi-Newton's method

4. Too much noise

One obvious reason to explore is If a curve is noisy, then Hessian will capture the noise as local curvature meaning that our estimation will be spurious and oppose learning.

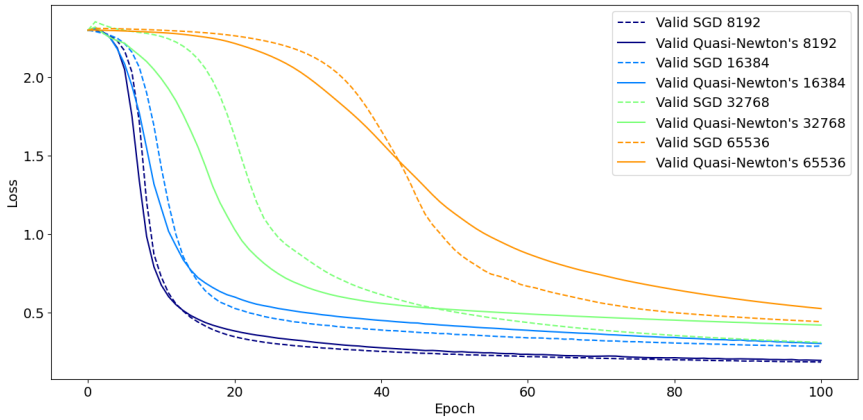


Figure 2: Plots for different batch sizes on MNIST. $\eta = 10^{-2}$

While we have taken steps to reduce noise, *e.g.* using the exponential moving average, there might still be too much stochasticity in each minibatch given that the minibatch error is an approximation of the training error which itself is an approximation of the generalisation error. Indeed, Granziol [5] shows that "the extremal eigenvalues of the batch Hessian are larger than those of the empirical Hessian" due to the noise in the minibatch.

To see if inter-batch noise causes the lack of improvement, we have trained a small CNN on MNIST so that full-batch training is feasible. From fig. 2 we can see that, allthough our quasi-Newton's method may start learning faster than SGD, there comes a crossover point where SGD overtakes and continues to have better performance regardless of the batch size[2]. Smaller batches have their crossover point in the first epoch due to the increased number of steps and so are not shown in fig. 2 to illustrate this effect. Thus, we can conclude that the noise incurred due to batch training is not a root of the performance deficit.

5. Discussion

The poor performance of the proposed method is both disappointing and mysterious. We are currently examining this and it is a work in progress. We have outlined a few potential explanations for this inconsistency between theory and praxis below.

5.1. Non-intersection

If the class vector subspace does not totally cover the top eigenvector subspace, then we are still leaving at least one high-curvature direction not preconditioned. This would also explain the inability to increase the learning rate.

It is interesting to note from fig. 3 that, knowing C, it seems a reasonable cut-off point for the top-curvature subspace, but if given only the eigenspectrum it would be difficult to guess the number of classes. This might indicate that C is not the best cut off point for the high-curvature subspace. It is possible that the class vectors actually have no particular relation to the high-curvature subspace

2. Since MNIST only has 50,000 training samples the batch size of 65,535 is just full batch.

but are simply an arbitrary way of splitting the cumulative gradient into some approximately orthogonal component vectors. Gur-Ari et al. [6], however, argue that it is only the top-C eigenvectors that are preserved over time. See Appendix C for more detail on the intersection of these subspaces.

5.2. Gradient information loss

It may be that the structure and non-convexity of the landscape is such that rapid gradient descent actually slows down the search. That is, this method pushes the low-curvature dimensions into a flat region of similar loss, where there is less gradient information, before they can be optimised into a low-loss region, thus slowing down the overall optimisation.

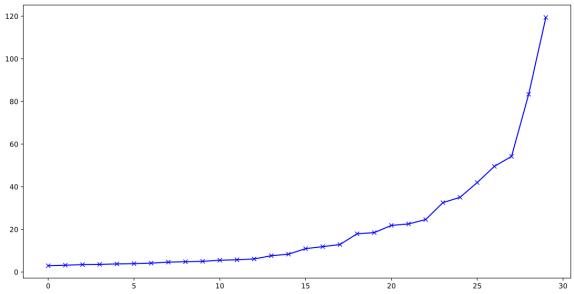


Figure 3: Top 30 true eigenvalues of a small 60k parameter CNN.

6. Conclusions

We are currently investigating the different explanations of why this naive method offers no improvement. Common experience is that, when done right, optimisation strategies that compensate for different curvatures in the loss landscape can lead to a significant speedup. Deep learning, however, rules out many traditional methods because of the vast dimensionality of the search space. Despite the failure of our naive implementation, the recognition that there exists a relatively low-dimensional subspace of high curvature which can be identified relatively easily suggests that it may be possible to improve on current methods.

References

[1] Florent Benaych-Georges and Raj Rao Nadakuditi. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics*, 227:494–521, 5 2011. ISSN 10902082. doi: 10.1016/j.aim.2011.02.007.

[2] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100(101):102, 2017.

[3] Stanislav Fort and Surya Ganguli. Emergent properties of the local geometry of neural loss landscapes. 10 2019. URL http://arxiv.org/abs/1910.05929.

[4] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:4039–4052, 1 2019. URL http://arxiv.org/abs/1901.10159.

[5] Diego Granziol. Curvature is key: Sub-sampled loss surfaces and the implications for large batch training. 6 2020. URL http://arxiv.org/abs/2006.09092.

[6] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018. URL http://arxiv.org/abs/1812.04754.

[7] Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:8819–8830, 1 2019. URL http://arxiv.org/abs/1901.08244.

[8] Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. Adahessian: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719*, 2020.

Appendix A. Comparison of SGD and quasi-Newton's

η	SGD	QN
1e-1	88	92
5e-2	87	92
1e-2	82	90

Table 1: Accuracy (%)

η	SGD	QN
1e-1	0.3574	0.3304
5e-2	0.3985	0.3346
1e-2	0.5362	0.3411

Table 2: Loss

Appendix B. On the Hessian

The main disadvantages of most second-order methods are the iteration time complexity, as computing the Hessian is $\mathcal{O}(N^2)$ and then computing its inverse is $\mathcal{O}(N^3)$, and the increased space complexity, $\mathcal{O}(N^2)$, to store the Hessian. The quasi-Newton's method avoids these pitfalls as it computes an approximation of the inverse Hessian in the same timescale as the gradient, $\mathcal{O}(N)$ and, since our Hessian approximation is formed from outer products, we can calculate a Hessian-vector-product without the quadratic space cost ($H = hh^\mathsf{T} \rightarrow Hz = (h^\mathsf{T} z)h$). However, it is still infeasible in its current form because it requires C backward passes and storing c_k means we require $C + 1$ times as much space as SGD.

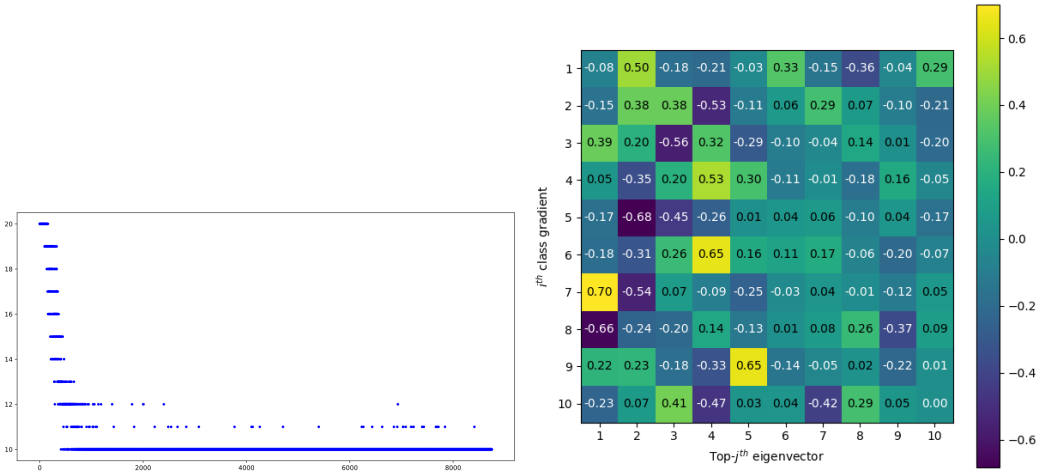
Incidentally, it is slightly odd to talk of a Hessian in a landscape with a discontinuous derivative. With ReLU activations we have a continuous piecewise linear surface which has zero Hessians almost everywhere. But even in one dimension, if we approximate a quadratic by a piecewise linear curve, then, although the second derivative is zero almost everywhere, we can still diverge using gradient descent if the step size is too big since the function is non-locally curved.

Appendix C. Extent of subspace intersection

Although there is general consensus that the class vector directions align with the top eigenvectors of the Hessian, we experimentally check to what degree this is true.since this is a probable reason for the worse performance of the quasi-Newton's method.

We collect the gradient directions for each class in the minibatch, $c_k^{\mathcal{B}^{(t)}}$ while training a simple 60k parameter CNN on CIFAR-10. At the end of training (50 epochs) we calculate the full eigensystem of the trained model. We can then plot the correlation and check whether the directions are largely stable.

The cosine-maximised linear assignment for the top-C eigenvectors to the class gradient directions is ≈ 0.36. This is a significant correlation since random 60k-dimensional vectors have an expected cosine of order 10^{-3}.



(*a*) Rank of the combined matrix for each batch

(*b*) Cosine matrix

Figure 4: Similarity measures between the class gradients and the top-C eigenvectors

We can also look at the rank of the combined matrix

$$[v_1, \ldots, v_C,\ e_0, \ldots, e_C], \tag{18}$$

where e_i is the top i^{th} true eigenvector of the fully trained model. If the subspace spanned by the top-C eigenvectors is the same as that which is spanned by the class gradients then the rank of this matrix should be C. We see that the rank quickly declines from $2C$, *i.e.*no intersection, to consistently being C. However, there are several batches when the rank increases by one or two implying that these batches have a greater amount of noise.

The rank of the combined matrix is a strong indicator that it is covered, but the noise may introduce spurious results due to the precision error in calculating the rank.

Appendix D. Hyperparameters

For all the experiments we choose a learning rate of 0.1 (unless otherwise specified), basic momentum of with a decay parameter of 0.9, and a gamma 0.9 for the quasi-Newton's method, and 0 weight decay or regularization. These values were not optimised.