An Empirical Exploration of Gradient Correlations in Deep Learning

Daniel Rothchild, Roy Fox, Noah Golmant, Joseph Gonzalez, Michael Mahoney
Kai Rothauge, Ion Stoica, Zhewei Yao
`{drothchild, royf, noah.golmant, jegonzal}@eecs.berkeley.edu`
`mmahoney@stat.berkeley.edu`
`{rothauge, istoica, zheweiy}@eecs.berkeley.edu`

Abstract

We introduce the mean and RMS dot product between normalized gradient vectors as tools for investigating the structure of loss functions and the trajectories followed by optimizers. We show that these quantities are sensitive to well-understood properties of the optimization algorithm, and we argue that investigating these quantities in detail can provide insight into properties that are less well understood. Using these tools, we observe that variance in the gradients of the loss function can be mostly explained by a small number of dimensions, and we compare results when training networks within the subspace spanned by the first few gradients to those obtained by training within a randomly chosen subspace.

1 Introduction

Fueled by growing success at addressing real-world problems, the practice of deep learning has far outpaced our theoretical understanding of model training. The shape of the empirical loss function that machine learning models are trained on depends on the training data and on the model's functional form. Currently, the main methods in deep learning that try to improve training by making assumptions about the loss shape are momentum and adaptive per-parameter learning rates: momentum encodes the general notion that the best direction along which to optimize doesn't change drastically from one step to the next, and adaptive per-weight learning rates encode the idea that rescaling weight dimensions to make first-order optimization easier will not affect the quality of the resulting solution. These ideas have been very successful in some applications, but they inform the training process in only very general terms. A better understanding of how these factors impact the loss function would allow us to tailor a training algorithm to particular properties of the training set and model, instead of using a generic method with hyperparameters tuned empirically.

With the eventual goal of uncovering the structure imparted onto a loss function by the data and the model architecture, we seek in this work to explore what information can be gleaned by analyzing a set of gradients measured on a loss function instead of simply using those gradients to carry out optimization. We define two quantities below that are sensitive to the non-local shape of the loss function, and we explore how these quantities depend on the model, data, and optimization trajectory.

2 Measuring Shape with Gradient Correlations

For a set of points in parameter space (e.g. randomly chosen, or chosen along an optimization trajectory), we use the mean and RMS dot product between pairs of normalized gradients to measure if the loss tends to slope in the same direction or along the same dimensions, respectively. Let $\mathscr{L}(\theta)$ be a loss function defined

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

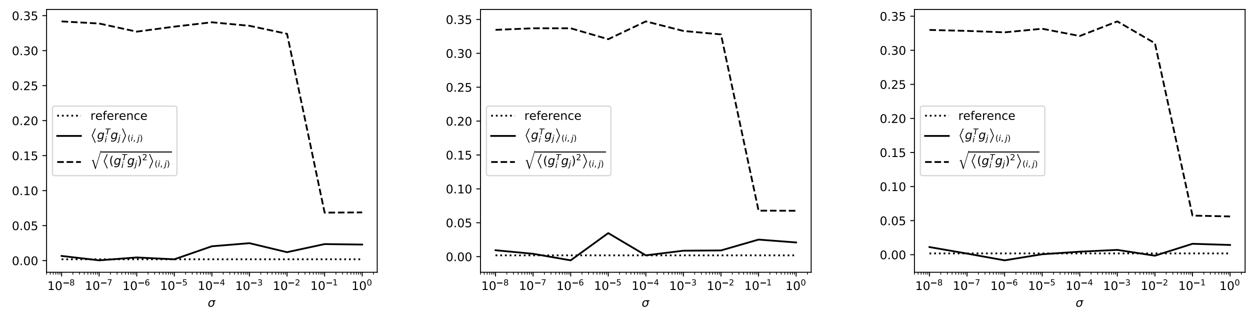

Figure 1: μ and α computed on gradients measured at parameters drawn from a normal distribution centered at 0 with standard deviation on the x-axis. Left: un-modified data; Center: randomized labels; Right: images replaced with white noise.

over parameters θ. We define, for normalized gradient vectors $g_i \equiv \nabla_\theta \mathscr{L}(\theta_i)/||\nabla_\theta \mathscr{L}(\theta_i)||_2$:

$$\mu \equiv \langle g_i \cdot g_j \rangle_{(i,j)}\,, \qquad \alpha \equiv \sqrt{\langle (g_i \cdot g_j)^2 \rangle_{(i,j)}}$$

where $\langle \cdot \rangle_{(i,j)}$ denotes the expectation over all pairs of gradients. For $g_{i,j}$ uniformly distributed on the unit sphere of dimension d, $\mathbb{E}[\alpha] = 1/\sqrt{d}$ (see Appendix §A for a proof). If $\alpha < 1/\sqrt{d}$, the g_i are unusually close to being perpendicular, and if $\alpha > 1/\sqrt{d}$, the variance in the g_i is largely explained by fewer dimensions than d. We include this reference value as a dotted line in all plots of μ and α.

Motivated in part by a desire to emulate the way deep learning is implemented in practice, in our experiments we compute mini-batches gradients. For very nearby points in parameter space, we therefore expect α and μ to be higher than otherwise, but still less than 1.

Figure 1 plots α and μ for a cross-entropy loss defined over a larger LeNet-like architecture (see Appendix §C for details) on the CIFAR-10 dataset. Unless otherwise noted, all results presented are for this experimental setup. The g_i are computed at parameter vectors chosen from a random normal distribution centered at 0 with standard deviation given on the x-axis. As expected, μ is close to zero, since the loss doesn't consistently slope in the same direction, even very close to 0. However, α is much higher than the reference value $1/\sqrt{d}$, demonstrating that the gradients are mostly explained by a low-dimensional subspace of the full weight space. For large σ, α decreases; this may be because the loss looks qualitatively different once the parameter settings are no longer "reasonably" small, but we have no good explanation for this observation.

Either some structure in the training data or the functional form of the model could drive this low-dimensionality. To distinguish between these two possibilities, Figure 1 shows results of the same computation where the labels are randomly shuffled and where the images in CIFAR-10 are replaced by white noise. If the structure in the data were causing the low-dimensionality, we would expect replacing the data with white noise to have a large effect on α. However, the three plots in Figure 1 are nearly indistinguishable, suggesting that the model architecture, and not the training data, is the main determiner of loss function's gross shape. Since the useful minima of the loss are highly data-dependent, we conclude that the low-dimensionality of the loss is unlikely to help when searching for useful minima. We investigate this hypothesis further in §4.

3 Gradient Correlations Along Trajectories

Computing μ and α on randomly chosen points in parameter space only provides limited insight, since most of parameter space is not explored during training. We therefore define $\mu_t(k)$ and $\alpha_t(k)$ over an ordered set $\{g_i\}$ of normalized gradients (or parameter updates) measured along an optimization trajectory:

$$\mu_t(k) \equiv \langle g_i \cdot g_{i+k} \rangle_i\,, \qquad \alpha_t(k) \equiv \sqrt{\langle (g_i \cdot g_{i+k})^2 \rangle_i}$$

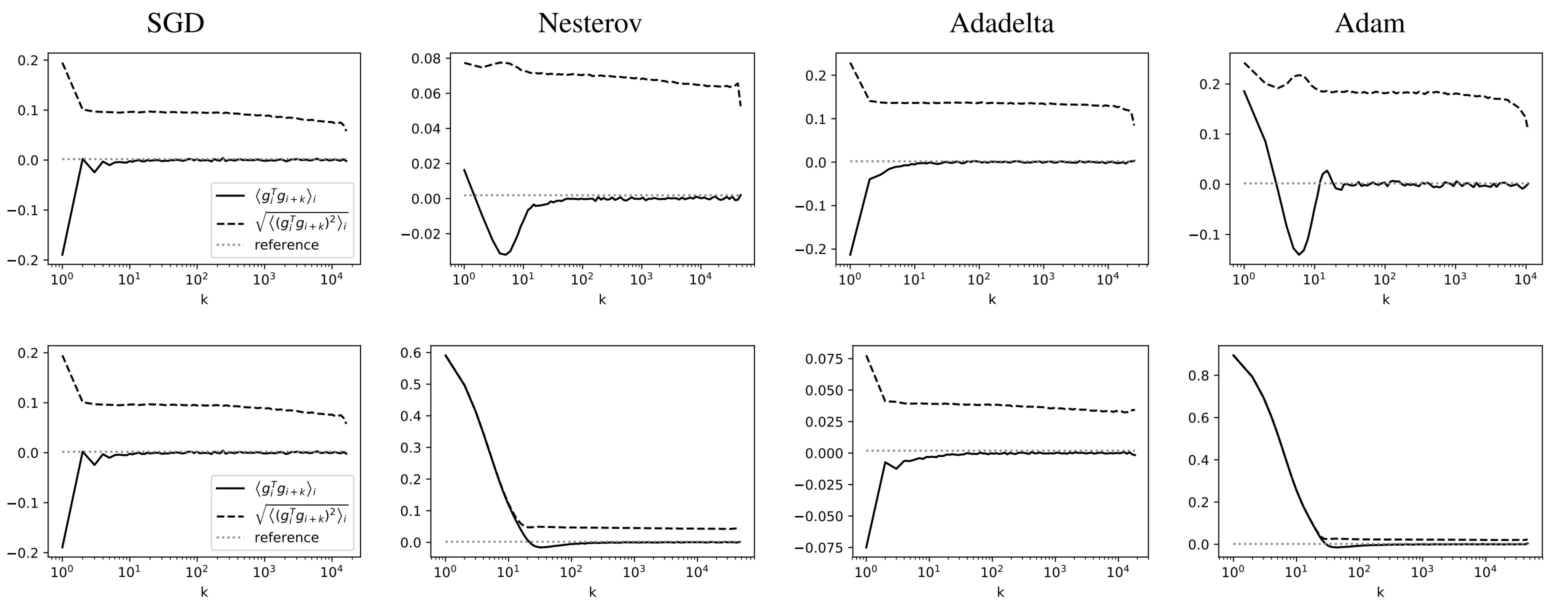

Figure 2: $\alpha_t(k)$ and $\mu_t(k)$ plotted vs. k for four optimizers: SGD, SGD with Nesterov momentum, Adadelta, and Adam. The top row computes α_t and μ_t using the gradients calculated at each point in the optimization trajectory. The bottom row uses the actual parameter updates.

The top row of Figure 2 plots $\mu_t(k)$ and $\alpha_t(k)$ computed on the gradients along the optimization path for four different optimization algorithms. The second row of the figure shows the analogous plots computed the parameter updates that the optimizer actually took (including momentum and per-parameter learning rates).

These plots clearly recover the main differences between the four optimization methods. In all plots, α is much higher than the reference value, as expected from Figure 1. For optimizers with per-parameter learning rates (Adam [Kingma and Ba, 2014] and Adadelta [Zeiler, 2012]), α is significantly lower when computed on parameter updates (second row), confirming that the per-parameter learning rates successfully reshape the loss to have less elongated level surfaces. Similarly, for plots computed on parameter updates, optimizers with momentum (Nesterov and Adam) yield very high α and μ for low k, as expected. For optimizers without momentum, μ is large and negative for $k \approx 1$, confirming that, without momentum, optimizers tend to bounce around the loss surface, potentially hindering their progress towards a minimum.

Some additional features of the plots in Figure 2 may be less obvious, but could give some insight into the shape of the loss function or the performance of the optimizers. For example, even for optimizers with per-parameter learning rates, α is still much higher than the reference value. This could indicate that the optimizer is not properly adapting each learning rate, causing the scaled loss surface to remain elongated. Or it may indicate that the loss is sufficiently far from a paraboloid at the scale of the typical parameter update norm that even an optimal rescaling does not lead to isotropic parameter updates. Lastly, it could be that the reference value is too simple of a metric to compare to, since even an optimally rescaled paraboloid does not have spherical level surfaces (unless the paraboloid is perfectly aligned with the coordinate axes).

Another feature of these plots is that, for large k, α tends to decrease. This happens because, for fixed k, α tends to decrease as optimization progresses, and for large k, α averages dot products between gradients from the very beginning and end of training. The decrease in α could result from the loss surface becoming flatter closer to a minimum. It could also be a manifestation of over-fitting if the decrease is due to the gradients on different mini-batches having higher variance.

Lastly, we point out that, for the optimizers with momentum, plots of μ in the top row of Figure 2 indicate that the gradient tends to switch direction just as the correlation in parameter updates (second row) decreases. This may indicate that momentum does not prevent the optimizer from bouncing around, but rather just

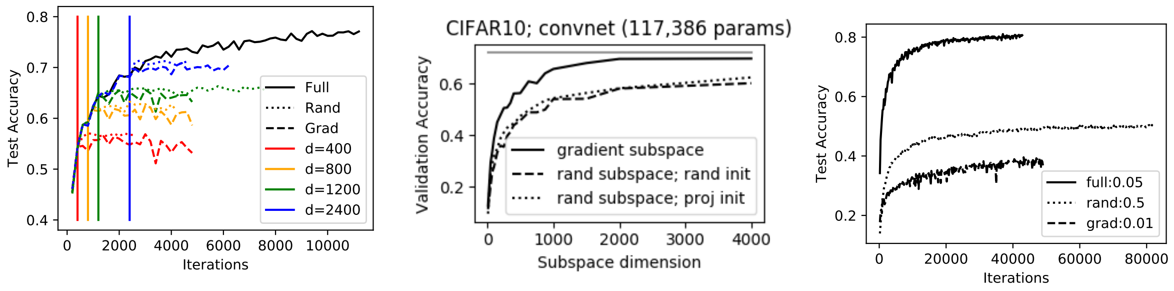

Figure 3: Three comparisons between gradient subspace training and random subspace training. Left: models trained with Adam. After d normal training iterations, all future gradient update are projected onto the subspace spanned by the first d gradients (dashed) or d random vectors (dotted). In this comparison, the dotted line trains in twice as many dimensions as the dashed line. Center: A fully convolutional network trained on CIFAR-10. Each point in the plot is a model fully trained as in the left plot of this figure with a subspace dimension given on the x-axis. The horizontal line shows final performance of an unconstrained model. Right: learning curves for vanilla mini-batch SGD, where gradients are projected into a random 1,000-dimensional subspace (dotted) or the subspace spanned by the first 1,000 gradients when SGD is run from a different initialization point (dashed). The solid line represents unconstrained training. The best learning rate (shown in the legend) is chosen empirically for each curve.

postpones the switching of direction by a few iterations. That μ for parameter updates dips slightly below 0 is additional evidence for this interpretation.

Further investigation is needed to distinguish between interpretations for each of these observations. However, we believe this discussion illustrates that α and μ can provide an interesting perspective on the shape of the loss surface as well as the optimization trajectories followed by different algorithms.

4 Subspace Training

We attempt to take advantage of the low-dimensional structure of the gradients that we observe by restricting training to a low-dimensional subspace spanned by the first few gradients. We compare our method to a similar method, investigated by Li et al. [2018], that restricts training to a randomly chosen low-dimensional subspace. Figure 3 compares these methods in three ways. Although there is no perfect way to compare the two methods, we conclude from these plots that, consistent with our interpretation of Figure 1, training within the subspace spanned by the first few gradient updates does not give a significant advantage in terms of converged test accuracy over training in a randomly chosen subspace of the same dimension. Interestingly, training in the subspace spanned by the first few gradient obtained from elsewhere on the loss actually performs much worse than training in a randomly chosen subspace (right-most plot in Figure 3). If, as is argued above, the steepness of the loss in certain dimensions is intrinsic to the model architecture, then the only difference between projecting onto a gradient subspace and onto a random subspace is that the former will make the parameter updates point in much steeper directions, potentially hindering optimization. And in practice, a random subspace admits larger learning rates before diverging than a gradient subspace.

Conclusion

We introduce the mean and RMS dot product between normalized gradient vectors as tools for gaining insight into the structure of loss functions and into optimization trajectories. Using these plots, we present a preliminary analysis of how the data and model affect the shape of the loss surface, and we conclude that the gross shape of the loss is determined mostly by the functional form of the model and not by the data over which the loss is defined. We also use these plots to investigate the shape of the areas of the loss function probed by different optimization methods. Momentum and per-parameter learning rates have the expected effects on α and μ, but several details in these plots are not fully explained.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 1752814. This research is also supported by NSF CISE Expeditions Award CCF-1730628, as well as by gifts from Alibaba, Amazon Web Services, Ant Financial, Arm, CapitalOne, Ericsson, Facebook, Google, Huawei, Intel, Microsoft, Scotiabank, Splunk and VMware.

References

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *CoRR*, abs/1804.08838, 2018. URL `http://arxiv.org/abs/1804.08838`.

Martin von Gagern, April 2015. URL `https://tinyurl.com/y7txn2n7`.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012.

Appendices

A Expectation of α

Proof that $\mathbb{E}\alpha = 1/D$ for gradients randomly distributed on the unit sphere of dimension D:

Consider the dot product between two uncorrelated normalized gradients g_1 and g_2. The angle θ between g_1 and g_2 is drawn from a distribution with PDF given by von Gagern [2015]:

$$f(\theta) = \frac{\Gamma(D/2)}{\sqrt{\pi}\Gamma((D-1)/2]} \sin^{D-2}\theta, \qquad \theta \in [0, \pi]$$

So the expected value of $(g_1^T g_2)^2$ is

$$\int_0^{\pi} \frac{\Gamma(D/2)}{\sqrt{\pi}\Gamma((D-1)/2)} \sin^{D-2}\theta \cos^2\theta d\theta = \frac{1}{D} \tag{1}$$

B PCA on Gradients

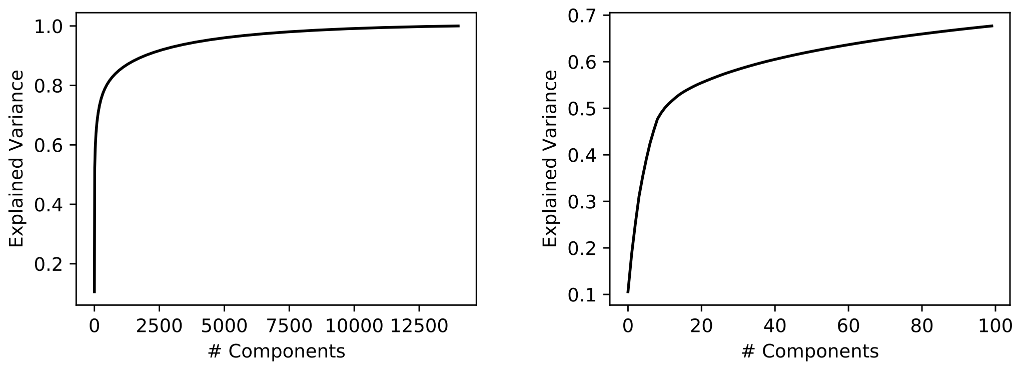

Figure 4: Explained variance for the principal components of the gradients obtained when running mini-batch SGD. Right plot is a zoom of the left.

C Model Architecture

The model used almost exclusively throughout the paper is a convolutional ReLU network with two 5x5 convolution/max pool layers followed by two fully connected layers. The convolutional layers have 20 and 50 channels, respectively; the first fully connected layer has 200 nodes; and the second is the output, which for CIFAR-10 has 10 nodes for the 10 classes.