Studying generalization in first-order optimization algorithms

Manjot Singh, Kshiteej Jitesh Sheth
e-mail: manjot.singh@epfl.ch, kshiteej.sheth@epfl.ch

***Abstract*—Generalization in deep neural networks has been of great interest in machine learning community in recent years. Many theoretical bounds have been proven to quantify and explain generalization in deep learning, several experimental observations have also been made. In this work primarily we experimentally investigate one of such bounds based on the PAC-Bayesian framework proposed in the literature by (4) to compare and explain generalization behaviour of models obtained using different first-order optimization algorithms and obtain partial positive evidence to justify the effectiveness of those bounds. We also briefly explore another notion in the literature - variance of gradients, which has been proposed to capture a certain notion of flatness of local minima that has been said to lead to better generalization. However our experiments suggest that there is not such a direct relationship between variance of gradients and generalization.**

I. Introduction

Deep neural networks have had a tremendous success in the recent years. Most of the times, solutions found by deep neural networks do not generalise well, even if they are able to minimize the training error. For linear models, a more concrete measure of generalization is based on norms and margin based measures (4). An empirically motivated measure, *sharpness* proposed by (3), is an other complexity measure for neural networks (robustness of the training error to perturbations in the parameters).

In this project, we use the PAC-Bayes framework from (4) to compare generalisation(using norms and sharpness) in different first order algorithms. Motivated from (2) and (7), we also study *variance of gradients* as an additional complexity measure. In section II we describe the definitions and literature on the PAC-Bayesian bounds and variance of gradients. Section III is dedicated to details regarding the experimental setup. Section IV discusses the results obtained. Section V highlights the future directions and conclusions.

II. Complexity Measures

A. PAC Bayesian framework

In (4), the authors present the following PAC-Bayes based generalization bound which holds with probability $1-\delta$ over the training set-

$$\mathbb{E}_{v\sim\mathcal{N}(0,\sigma)^n}[L(f_{w+v})] - \widehat{L}(f_w) \leq \mathbb{E}_{v\sim\mathcal{N}(0,\sigma)^n}[\widehat{L}(f_{w+v})] - \widehat{L}(f_w) + 4\sqrt{\frac{1}{m}(\frac{||w||_2^2}{2\sigma^2} + \ln\frac{2m}{\delta})}$$

where $f_w(x)$ or f_w is the function computed by the neural network with weight vector $w \in \mathbb{R}^n$. $L(.)$ and $\widehat{L}(.)$ denote the true and training error respectively, m is the size of the training set and $\mathcal{N}(0,\sigma)^n$ denotes the n dimensional spherical gaussian with mean zero and variance σ^2. The upper bound is composed of two parts, first term corresponds to the *expected sharpness bound* $\mathbb{E}_{v\sim\mathcal{N}(0,\sigma)^n}[\widehat{L}(f_{w+v})] - \widehat{L}(f_w)$ and second term, $4\sqrt{\frac{1}{m}(\frac{||w||_2^2}{2\sigma^2} + \ln\frac{2m}{\delta})}$ is referred to as the *norm bound*.

The sharpness bound can be understood as the expected change in the training error after perturbing the weights by gaussian noise of mean zero and variance σ^2, measuring robustness to perturbations in the parameter space. In (4), the authors study why norm or sharpness bounds by themselves are not enough to explain generalization behavior of models whereas the above bound gives a clear way to think about capacity control in terms of both norm and sharpness. However in the above bound, norm and sharpness interact with each other depending on σ and thus choosing an appropriate σ becomes important to get a tight generalization bound.

B. Variance of Gradients

In (2), the authors propose that towards the end of the training phase, the magnitude of *variance of gradients* captures a certain flatness of local minima which is known to be positively correlated with generalization performance. For a fixed weight vector w, let $g(w,d)$ denote the stochastic gradient of the loss function at weights w for a random data point d uniformly sampled from the training set. Then the variance of gradients at w is defined as:

$$\mathbb{E}_d\left[\|g(w,d) - \mathbb{E}_d[g(w,d)]\|_2^2\right] \quad (1)$$

which, as per (7), is equal to

$$\mathbb{E}_d\left[\|g(w,d)\|_2^2\right] - \|\mathbb{E}_d[g(w,d)]\|_2^2 \qquad (2)$$

Thus, our goal in this project is to explore empirically any relationship between the variance of gradients and the generalization behaviour of the model learnt using different optimisation algorithms.

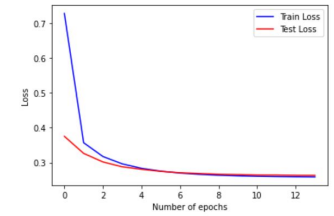
III. Experiments

The goal of our experiments is to explore whether the PAC bayesian bounds and variance of gradients can be used to explain generalization behavior or not. More concretely, we consider the task of multi-class classification on the MNIST data set using a feed-forward Neural Network with 3 fully connected layers. Our training and test sets have sizes 60,000 and 10,000 examples respectively. We further split the training set into training and a validation set of size 54,000 and 6,000 respectively. We then train the model using 3 different algorithms - Adam, SGD and RMSProp, using the negative log likelihood as the loss function. We use Pytorch (5) functions for implementation of each optimisation algorithm. For each algorithm, we consider a fixed batch size of 64 and train them for 14 epochs. For simplicity we only optimize the learning rate. We do a grid-search over different range of values and choose the one with lowest validation error. The best learning rates for each algorithm and the set of learning rates tested are

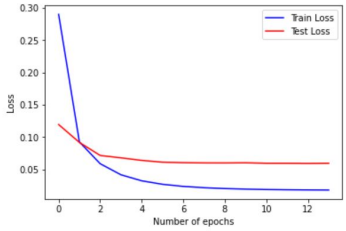
1) **SGD-** We select $1e-1$ out of $\{1e-1, 3e-2, 3e-3, 3e-4, 3e-5\}$.
2) **Adam-** We select $2e-5$ out of $\{1e-1, 3e-2, 3e-3, 3e-4, 2e-5\}$.
3) **RMSProp-** We select $3e-6$ out of $\{3e-2, 3e-3, 3e-4, 3e-5, 3e-6\}$.

The learning curves of each algorithm i.e. training and test loss for an entire duration of training are plotted in figure 1.

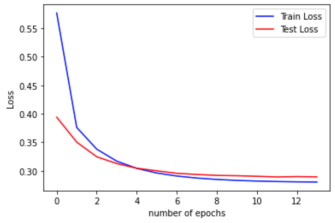
PAC-Bayes bounds Implementation We first train the model using a given first-order optimization algorithm. Then, we use the trained model to compute the norm and sharpness bounds for the following values of σ i.e. $\{1e-3, 3e-3, 6e-3, 9e-3, 3e-2, 6e-2, 9e-2\}$. To compute the sharpness bound, we perturb the trained model using gaussian noise with mean 0 and std. dev σ. The term $\mathbb{E}_{v\sim\mathcal{N}(0,\sigma)^n}[\widehat{L}(f_{w+v})]$, for computing the sharpness bound, is estimated by taking an empirical average of $\widehat{L}(f_{w+v})$ using 20 samples of v from $\mathcal{N}(0,\sigma)^n$. Taking an empirical average for more than 20 times was computationally expensive, thereby, we also calculate the standard deviation associated with $\widehat{L}(f_{w+v})$ to reason about the precision of this empirical estimate of the true $\mathbb{E}_{v\sim\mathcal{N}(0,\sigma)^n}[\widehat{L}(f_{w+v})]$. For calculating the norm bound, we access all the model weights using PyTorch's in-built functions, calculate the l_2-norm of the weight vector and use $\delta = 0.01$ and $m =$ size of the training set.



(a) Adam



(b) SGD



(c) RMSProp

Fig. 1: Training and Test error (negative log likelihood) for each algorithm versus the number of epochs run

Implementation of Variance of Gradients - Our goal in this section is to explore whether low magnitude of variance of gradient at the end of the training phase correspond to better generalization or not. However, in (2), the authors suggest that variance of gradients appear to be positively correlated with each hyperparameter. Thus, for each algorithm, we also explore how variance of gradients change with different learning rates while we fix other hyperparameters for simplicity. More precisely, for each algorithm we consider 3 learning rates (refer to Table II for the exact values for each algorithm) and we train the model using that algorithm and that particular learning rate. At the end of each epoch, we calculate the variance of gradients as per equation 2, keeping the current weights fixed. To calculate the expectations in equation 2 by averaging over all data points, we

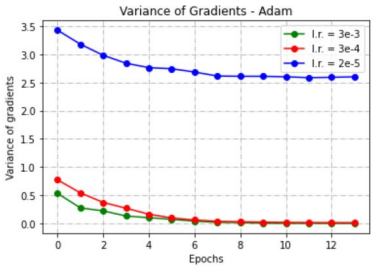
instead, average over all mini-batches each of size 64. For each algorithm and each learning rate we also record the training and test error obtained at the end of training.

Averaging over multiple Seeds - To reduce variability in our results, we run all of the above described experiments for seed values $\{12345, 1234, 123\}$ and average all numerical values over these 3 seeds.

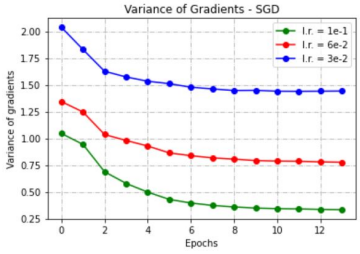
IV. Results and Discussion

PAC-Bayes bounds - Since different algorithms converge to different training error, we use the difference in test-train error found at the end of training to compare generalization behaviour of different algorithms. From Table I, we observe that Adam has the best generalization behaviour in terms of having the lowest (test-train) error gap, followed by RMSProp and SGD. Now, in order to test the hypothesis that a lower (norm+sharpness) bound is equivalent to better generalization, the estimate of sharpness needs to be accurate enough. For $\sigma = 3e-3$, the standard deviation of the sharpness estimate is roughly an order of magnitude smaller than the estimate of sharpness itself for all three algorithms, thus we can use this $\sigma = 3e-3$ for analysis. We observe that for this σ Adam has the lowest value of norm+sharpness followed by RMSProp and SGD, which is evidence in support of the hypothesis. This analysis also holds for $\sigma = 6e-3, 9e-3$. Even though sharpness estimates are fairly accurate for these values of σ, the PAC-Bayes bound at these values of sigma is dominated by the norm bound. Note that for higher values of σ such as $\sigma = 9e-2$ the sharpness estimates are not accurate enough as the corresponding standard deviations are comparable to the estimate itself. It can also be noted that the empirical (norm+sharpness) bound is orders of magnitude higher than the training loss itself and thus this upper bound(refer to PAC-Bayes eqn. above) is a weak upper bound for generalization error.

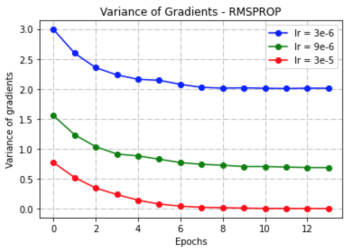
Variance of Gradients - For each algorithm, from fig. 2 we can observe that for higher learning rates, the variance of gradients converge to a lower value. However from Table II we can observe that for all three algorithms, as the learning rate increase the gap between test and train error increases, indicating poor generalization. This suggests that the variance of gradients is clearly sensitive to the learning rate and it converges to a lower value for a higher learning rate but not necessarily with better generalization.



(a) Adam



(b) SGD



(c) RMSProp

Fig. 2: Variance of gradients for each algorithm as a function of the number of epochs run

V. Conclusion and Future Directions

From the previous section, we observe that we have positive evidence to support the hypothesis that the PAC-Bayes based upper bound is indicative of generalization behaviour. However our sharpness estimates clearly need more than 20 samples of the perturbation to obtain a better estimate of sharpness for high values of σ. We also highlighted that empirically the norm+sharpness bound is orders of magnitude higher than the training error in almost all our experiments which suggests to investigate tighter upper bounds on generalization error. In this direction, (1) present a margin-based multiclass generalization bound for neural networks which could also be investigated. It is also important to perform our experiments at a larger scale, for larger models and other datasets. To the best our knowledge, not many have explored the connection between gradient variance and generalisation gap except (2), but a recent paper (6) tells that both curvature and noise are essential to estimate generalization gap.

References

[1] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6241–6250, Red Hook, NY, USA, 2017. Curran Associates Inc.

[2] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations(ICLR)*, 2020.

[3] Nitish S. 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(ICLR)*, 2017.

[4] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *NIPS Neural Information Processing Systems*.

[5] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Neural Information Processing Systems (NIPS)*, 2017.

[6] Valentin Thomas, Fabian Pedregosa, Bart van Merri enboer, Pierre-Antoine Mangazol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics(AISTATS)*, 2020.

[7] Chong Wang, Xi Chen, Alex Smola, and Eric P. Xing. Variance reduction for stochastic gradient optimization. In *NIPS Neural Information Processing Systems*, 2013.

VI. Appendix

We open source the implementation of our experiments at OptML_Project.

Code - The code is commented. The instructions to run the experiments are mentioned in ReadME. It's easy to run the experiments on Colab by just uploading the notebook and all the utility files with it. One can change model hyper-parameter values and any other variables by changing their values in the notebook to do any exploration.

On the next page, there are two tables, table I and table II which contains all the results on two complexity measures i.e. norms and sharpness and variance of gradients for each optimization algorithm respectively.

TABLE I: The table shows the results for norm and sharpness measures over the whole range of sigma values for different optimization algorithms. Std.Dev. represents the standard deviation associated with the perturbed training error for computing sharpness. N+S refers to norm plus sharpness. On the right side is the test-train error/gap for each algorithm.

Method	Measure	Sigma(σ)							Test-Train gap
		1e-3	3e-3	6e-3	9e-3	3e-2	6e-2	9e-2	
SGD	Norm	339.79	113.26	56.63	37.75	11.32	5.66	3.77	0.04
	Sharpn.	-2.6e-5	9.3e-4	3.6e-3	8.1e-3	2.4e-1	4.69	18.17	
	Std.Dev	1.5e-5	6.7e-5	2.5e-4	7.3e-4	7.5e-2	1.02	4.75	
	N+S	339.79	113.26	56.63	37.76	11.57	10.35	21.95	
RMSpr.	Norm	332.32	110.77	55.38	36.92	11.07	5.53	3.69	0.009
	Sharpn.	-9.8e-5	1.5e-3	6.6e-3	1.6e-2	4.5e-1	4.95	15.96	
	Std.Dev	5.4e-5	2.5e-4	1.2e-3	2.9e-3	1.5e-1	1.86	5.18	
	N+S	332.32	110.77	55.39	36.94	11.52	10.49	19.66	
Adam	Norm	328.71	109.57	54.78	36.52	10.95	5.47	3.65	0.004
	Sharpn.	-1.8e-5	1.8e-3	7.9e-3	1.9e-2	5.0e-1	5.54	17.20	
	Std.Dev	7.1e-5	3.6e-4	1.5e-3	3.8e-3	1.5e-1	1.89	5.15	
	N+S	328.71	109.57	54.79	36.54	11.46	11.02	20.82	

TABLE II: This table shows results for variance of gradients at the end of training for each optimisation algorithm. On the right side is the test-train error/gap for each algorithm.

Method	Learning Rate	Var. of Grad.	Test-Train gap
SGD	1e-1	0.337	0.0415
	6e-2	0.77	0.032
	3e-2	1.44	0.017
RMSprop	3e-5	0.0068	0.054
	9e-6	0.68	0.032
	3e-6	2.01	0.0092
Adam	3e-3	0.002	0.067
	3e-4	0.014	0.05
	2e-5	2.6	0.0047