Non-Gaussian behaviour of Stochastic Gradient Noise

Candidate Number: 1040706
Department of Mathematics
University of Oxford

Abstract

In recent years, there has been a growing interest in Stochastic Gradient Descent (SGD) and its modifications in the field of machine learning, mainly due to its computational efficiency. It is often assumed that gradient noise follows Gaussian distribution in large data-sets by invoking the *classical* Central Limit Theorem. However, the results in this report show that this is far from true, in fact we show that stochastic gradient noise (SGN) follows an α-stable distribution, which is a family of heavy tailed distribution where α is a tail index. We carry out further analysis by converting the SGD into a stochastic differential equation (SDE) driven by Levy motion for a sufficiently small learning rate. These types of SDEs shed more light on the metastability theory which states that particles can jump from narrow minimas in polynomial time and seem to prefer wider minimas. For validation, we vary the choice of the activation function and learning rate to show that SGN is far from Gaussian and its distribution is very sensitive to these changes.

1 Introduction

In deep learning, we usually deal with an Optimization Problem where we are trying to find an unknown parameter θ^* which minimizes an objective function L i.e.:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^p} \left\{ L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell^{(i)}(\theta) \right\} \tag{1}$$

where $\theta \in \mathbb{R}^p$ usually denotes the weights and biases in a neural network and $\ell^{(i)}(\theta)$ is the loss function associated to the i^{th} observation in a data-set (used for training of size n). The objective function is taken over the whole data-set which is usually non-convex.

When used to minimize the objective function, a standard Batch Gradient Descent (also known as Steepest Gradient Descent) would perform the following iterations:

$$\theta_{k+1}^{(l)} = \theta_k^{(l)} - \eta \nabla L(\theta_k^{(l)}) \tag{2}$$

where η is a user defined learning rate (also referred to as step-size) and $l = 1, \dots D$ refers to the update in the l^{th} layer with total of D layers.

Evaluating the sum of gradients ∇L may require expensive evaluation of loss function gradients $\nabla \ell^{(i)}(\theta)$ especially when your training data-set is large and no simple function exists. This is why we try to make use of *Stochastic Gradient Descent*[1](SGD) [1] instead of Batch Gradient Descent since it replaces the actual gradient (calculated over the entire dataset) by estimate gradients (calculated from a randomly selected subset of the data). The following algorithm shows SGD in action; for simplicity, we ignore the index l for keeping track of layers in the net.

[1]here we refer SGD as *Mini-Batch Stochastic Gradient Descent*

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

1. Randomly choose an initial vector θ_0. Also choose a learning rate η and $epoch$
2. At each iteration $k \in \{1, \ldots, epoch\}$ define a random mini-batch $B_k \subseteq \{1, \ldots, n\}$ of size $|B_k| = b << n$ without replacement which contains indices of training data. Define:

$$G_k = \nabla L_{B_k}(\theta) = \frac{1}{b} \sum_{i \in B_k} \nabla \ell^{(i)}(\theta)$$

3. Update unknown parameters as:

$$\theta_{k+1} = \theta_k - \eta \nabla L_{B_k}(\theta_k) \tag{3}$$

4. Repeat steps 2 and 3

Given we have chosen our learning rate η to be sufficiently small, by Robbins-Seigmund Theorem [2], SGD converges to a global minimum when the objective function is convex almost surely or else converges to a local minimum. Many modifications to the vanilla SGD have been proposed and implemented which adjust the learning rate η at each iteration such as AdaDelta [3] and Adam [4] which try to reduce the stochastic noise and smooth out the path to the minima by taking weighted average of the previous gradients. However, all these algorithms make a very naive underlying assumption that this stochastic noise follows a Gaussian distribution. More formally, if we define *Stochastic Gradient Noise* (SGN) as $U_k(\theta) = \nabla L_{B_k}(\theta) - \nabla L(\theta)$, then it is assumed that

$$U_k(\theta) \sim N(0, \sigma^2 I)$$

where $U_k(\theta)$ is a multivariate Gaussian random variable and each element is independent with variance σ^2.

The reason for making this assumption is that for sufficiently large mini-batch size b, by the *classical* Central Limit Theorem (CLT), U_k can be seen as the sum of independent and identically distributed (i.i.d) random variables, with *finite* variance. Then by equation (3), we can express $\theta_{k+1} = \theta_k - \eta(U_k(\theta_k) + \nabla L(\theta_k))$. If we define $Z_k = [\sqrt{\sigma^2} I]^{-1} U_k$ to be standard normal, we get:

$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k) + \sqrt{\eta}\sqrt{\eta\sigma^2} Z_k \tag{4}$$

Due to this Gaussian assumption, it allows us to transition from SGD to a continuous *Stochastic differential equation* (SDE) for a sufficiently small η driven by Brownian motion:

$$d\theta_t = -\nabla L(\theta_t) dt + \sqrt{\eta\sigma^2} dB_t \tag{5}$$

where B_t defines the standard Brownian motion with a property that each increment is continuous[2]. Although the assumption that random variables have a finite variance seems reasonable, but in fact in many natural applications, this does not hold. For example in finance, the underlying assumption of the Black-Scholes model of option pricing is that returns are Gaussian; but as Mandelbrot pointed out in his book [5], the distribution is in fact *heavy-tailed*. The probability that the option price is 5-7 standard deviations out of the money is much higher than normal-distribution can predict. Similar observations can be spotted for neural networks under SGD as seen in Figure 1 of [6]. Actual SGN seems to show more heavy-tailed behaviour than anticipated. Therefore, finite variance assumption does not seem valid.

2 Transition from SGD to Levy driven SDEs

It appears that gradient noise follows more of a *Cauchy Distribution*. More generally, it makes more sense to assume that U_k has a probability density $p(x)$ that is asymptotic to power law tail

$$P(U_k > x) \sim 1/x^{\alpha+1} \text{ as } |x| \to \infty \tag{6}$$

for $\alpha \in (0, 2)$ where α is the *tail-index* controlling the thickness of the tail of the distribution. It is clear that if $\alpha < 2$, then the second moment of U_k i.e. $\mathbb{E}[U_k^\alpha]$ is *not* finite. The distribution which has a similar tail behaviour is the *Stable Distribution*:

[2]The increments $B_1 - B_0, B_2 - B_1, \ldots, B_n - B_{n-1}, \ldots$ are continuous

Definition 1 *: Stable Distribution*

Let X_0, X_1, X_2 be i.i.d random variables. We say they are stable if there exists $a, b, c > 0$ and $d \in \mathbb{R}$ such that $aX_1 + bX_2$ has the same distribution as $cX_0 + d$.

Now we can use the *Generalised Central Limit Theorem*, to show that SGN follows an α-stable distribution.

Theorem 1 *: Generalised Central Limit Theorem (GCLT)*

Let $\alpha < 2$ be the tail-index and let random variables $\{X_i\}_{i=1}^{\infty}$ have probability density $p(x)$ which is asymptotically proportional to $1/|x|^{\alpha+1}$ at the tails. Define $S_n = X_1 + \cdots + X_n$ then:

$$\frac{S_n - \mu_n}{\sigma_n} \xrightarrow[n\to\infty]{} \alpha\text{-stable distribution}$$

where σ_n is width of the distribution and μ_n is the shifting factor.

Special cases: When $\alpha = 0.5, 1, 2$, then we get Levy, Cauchy and Gaussian distribution respectively. In general, the probability density functions for all $\alpha \in (0, 2]$ do not have a closed form solution except for these special cases.

2.1 Assumptions

We will now assume that:

1. $$[U_k(\theta)]_i \sim S\alpha S(\sigma) \quad i = 1, \ldots, n$$
   where σ is a scale parameter and $S\alpha S(\sigma)$ represents the *Symmetric Stable distribution*[3] with tail index α. Note that the variance of each element $[U_k(\theta)]_i$ is not necessarily finite anymore (which was assumed earlier). Now the update for the weights in SGD can be expressed as:
   $$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k) + \eta^{\frac{1}{\alpha}} \left(\eta^{\frac{\alpha-1}{\alpha}} \sigma\right) S_k \tag{7}$$
   where the last term signifies the stochastic noise in the system and $S_k \sim S\alpha S(1)$.
   Just as before, for a sufficiently small learning rate η, we can consider converting the discrete case equation (7) into a SDE of the form:
   $$d\theta_t = -\nabla L(\theta_t) dt + \left(\eta^{\frac{\alpha-1}{\alpha}} \sigma\right) d\mathcal{L}_t^{\alpha} \tag{8}$$
   where $\mathcal{L}_t^{\alpha}$ is an independent α-stable Levy motion of p-dimension. It is a replacement of the Brownian Motion in equation (5). Equation (8) is known as *Levy Driven SDE*. Note, when $\alpha = 2$, $\mathcal{L}_t^{\alpha}$ is just a translated version of the standard Brownian motion $\sqrt{2}B_t$ thus giving finite variance for this case.
2. The objective function L is smooth and it contains r local minima $\{m_i\}_{i=1}^{n}$ each enclosed by local maxima $\{s_i\}_{i=1}^{r-1}$. Further, it is assumed that m_i or s_i can not be *saddle points* and thus the Hessian matrix is non-singular (only positive definite or negative definite at critical points).

Comments

Levy driven SDEs have a fundamentally different behaviour to Brownian Motion SDEs. This is due to the fact that the main difference between B_t and $\mathcal{L}_t^{\alpha}$ is that $\mathcal{L}_t^{\alpha}$ can have countable number of discontinuities i.e. "jumps".

The smoothness assumption makes sense because Levy driven SDEs (8) are split into deterministic term and stochastic noise (i.e. some kind of random perturbation to simulate a real life situation). In-order for SDE to replicate our SGD in equation (7), it is necessary for gradients of objective function to exist and be defined.

[3] a more general α stable distribution has four parameters $(\alpha, \beta, \mu, \sigma)$ which determines the concentration about tails, asymmetry, location and scaling of the distribution respectively. $S\alpha S(\sigma)$ is a simple case where $\beta = 0$ and we assume $\mu = 0$ i.e. on average our SGN is centred around zero.

2.2 So why are Levy SDEs better than Brownian SDEs?

As suggested in [7], the *basin* of the objective function is the neighbourhood of a local minimum m_i i.e. a p-dimensional ball $B_i = \{|x - m_i| \leq \delta\}$ for some small $\delta > 0$. So if our initial guess for the parameter $\theta_0 \in B_i$ is in the basin, then we are interested in the minimum time $T^i_{\theta_0}$ it takes to climb out of a valley and transition to another basin. More formally, we are looking for

$$T^i_{\theta_0} = \inf\{t > 0 : \theta_t \in \cup_{j \neq i} B_j\} \tag{9}$$

Equation (9) is related to *Metastability* phenomenon [8] which states that the first exit time $T^i_{\theta_0}$ of θ_t should only take *polynomial time* and should depend only on the *width* of the basin not it's *depth*. However, Brownian Motion takes *exponential time* due to it's dependency on the depth of the valley. From [9], we see that Levy Motion SDEs do not depend on depth which gives it a huge advantage over Brownian motion to climb out of deep valleys into other basins in shorter amount of time. This is mainly due to the fact that $\mathcal{L}^\alpha_t$ has countably many discontinuities which allows it to jump out of deep valleys. This result can help us understand why SGD prefers to choose a *wide-minima*. $\mathcal{L}^\alpha_t$ prevents you from getting stuck in a local minimum even with a fixed learning rate η. We can encounter problems if we happen to initiate our weights in the basin of global minimum which means, after polynomial time, there is a chance that we escape this basin (this is the desired basin we want to be in) and transition to a neighbouring one (i.e. local minimum).

2.3 Concerns

Here, we state some of our concerns we have regarding Sections 2.1 and 2.2

1. Although we have assumed that our objective function $L : \mathbb{R}^p \to \mathbb{R}$ has no saddle points, [10] highlights the fact that non-convex error surfaces in high dimensions are *exponentially* more likely to have saddle points than local minima as p increases. Although this result is for stochastic Gaussian noise, it can be shown that this holds for $S\alpha S$ distributions too:

   Sketch of the proof

   Suppose the objective function L has m critical points $\{x_i\}_{i=1}^m$. From multi-variable calculus, it is a common result that a Hessian matrix at a critical point $H(x_i)$ is *Symmetric Positive Definite* if x_i is a local minimum (and Symmetric Negative Definite if x_i is a local maximum). Since $H(x_i)$ is diagonalisable, we can find its similar matrix $D = diag(d_1, \ldots, d_p)$ with $d_1, \ldots, d_p > 0$ being all positive eigenvalues. Due to the complexity of the deep neural net and assumption that SGN is centred and symmetric at 0 which can be seen by looking at the cumulative density function (CDF) of $S\alpha S$ implies that $P(d_i > 0) = \frac{1}{2}\ \forall i$. Since each d_i is relatively independent to each other due to high non-linearity of Hessian, the probability that a given critical point x_i is a minimum is:

   $$P(minima) = P(d_1 > 0, \ldots, d_p > 0) = P(d_1 > 0) \times \cdots \times P(d_p > 0) = \frac{1}{2^p}$$

   With similar logic, $P(maxima) = \frac{1}{2^p}$. Then the probability for x_i being a saddle point is:

   $$P(saddle) = 1 - P(minima) - P(maxima) = 1 - \frac{1}{2^{p-1}} \xrightarrow[p \to \infty]{} 1$$

   leading to saddle points exponentially more likely to occur than local minima and maxima.

   Saddle Points can be problematic because in their basin, the gradient can be small, leading to negligible updates to the weights and this could cease the training. [10] also shows that huge training errors are found if weights are initialised near the saddle points. Thus assuming there are no saddle points in the error surface in a complex problem does not seem to be valid.

2. Throughout the paper, we have analyzed SGD by approximating it by a continuous time approach. But we ask ourselves, the validity of such approximations, i.e. how small should learning rate be. More formally, does there exist an upper bound $\tilde{\eta}$ such that $\forall \eta \leq \tilde{\eta}$, this is a well conditioned approximation. Based on this condition, we will be able to say metastability condition translate well even in a discrete case of SGD. Theorem 2 of [7] attempts to address

this question by finding this upper bound $\tilde{\eta}$, but since research in Levy driven SDEs are quite recent, some underlying assumptions have been made to get to this result. Secondly, what if $\tilde{\eta}$ is so small that the approximation of SGD by SDE is ill-conditioned for practical use. A lot of research still needs to be carried out before we can certainly agree with all the results we have shown in this report.

3 Setting up the Experiment

We want to investigate the tail behaviour of SGN and its effect by varying different factors. We try to answer two questions:

a) Does the choice of different activation functions have a big impact on the distribution?

b) Does choosing a different learning rate η effect the distribution?

For these models, we look at two popular activation functions, Rectified Linear Unit *ReLu* and *Sigmoid*:

$$a_1(x) = max(x, 0) \qquad a_2(x) = \frac{1}{1 + e^{-x}} \tag{10}$$

We consider a fully connected network (FCN) with a single hidden layer with 100 neurons and no biases on Fashion-MNIST datasets which is randomly split into training and testing images of size $60K$ and $10K$ respectively[4]. We keep our objective function fixed to *Mean Squared Error*. We train our model using vanilla SGD as discussed in the whole of the report, without introducing momentum nor weight decay. In order to estimate the tail index α, we will make use of a recent estimator as proposed in [11] in the form of following theorem:

Theorem 2 *Let $X_i \sim S\alpha S(\sigma)$ where $i = 1, \ldots, K$ and $K = K_1 \times K_2$. Let $Y_i = \sum_{j=1}^{K_1} X_{j+(i-1)K_1}$ for $i \in [1, K_2]$. Then the estimator.*

$$\frac{\hat{1}}{\alpha} = \frac{1}{logK_1} \left(\frac{1}{K_2} \sum_{i=1}^{K_2} log|Y_i| - \frac{1}{K} \sum_{i=1}^{K} log|X_i| \right) \tag{11}$$

converges to $1/\alpha$ almost surely as $K_2 \to \infty$

Sketch of Algorithm: The implementation of the following models is in Python (without the use of tensorflow or pytorch since they didn't have the functionality required). They are available on the Github: `https://anonymous.4open.science/r/58487305-8c01-4b0d-8c54-ae8c375c3aff/`[5]

1. Set batch size $b = 500$, learning rate $\eta = 0.1$ and $epoch = 500$
2. Initiate the algorithm by randomly choosing weight θ_0
3. for $k = 1, \ldots, epoch$:
   - Compute full gradient $\nabla L_D(\theta_k)$ over the training data-set $D = \{1, \ldots n\}$ of size $n = 60,000$
   - Sample the data-set D into mini-batches $B_k^i \subset D$ for $i = 1, \ldots, \lceil \frac{n}{b} \rceil$ of size b without replacement ($B_k^i \cap B_k^j = \emptyset$ for $i \neq j$).
   - Compute the Stochastic gradient over each mini-batch $\nabla_{B_k^i} L(\theta_k)$
   - Compute stochastic gradient noise $U_k^i = \nabla L_D(\theta_k) - \nabla L_{B_k^i}(\theta_k)$ which is then flattened into a vector of length K
   - Then using equation (11), we can calculate $\hat{\alpha}_k$ where K_1 is set to be divisor of K closest to $\sqrt{K}$ and $K_2 = K/K_1$
4. After keeping track of each $\hat{\alpha}_k$, we can plot the graphs.
5. Follow the above steps for testing data-set (where $n = 10,000$)

Above we have described the model for question a) but this can be slightly modified to give output for b) by changing learning rate η at each iteration taking values between $\eta \in [0.001, 0.1]$. The reason epoch is so small is due to limited computing power.

[4] Infact, I have also carried out tests on MNIST data-set with exciting results, but due to the limited report size, I am unable to show the results here.

[5] To keep my identity anonymous, I have made use of *Anonymous Github* software.

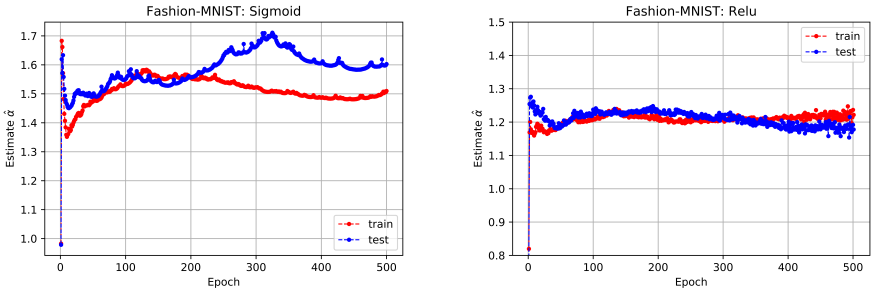

Figure 1: Estimation of α after varying activation function

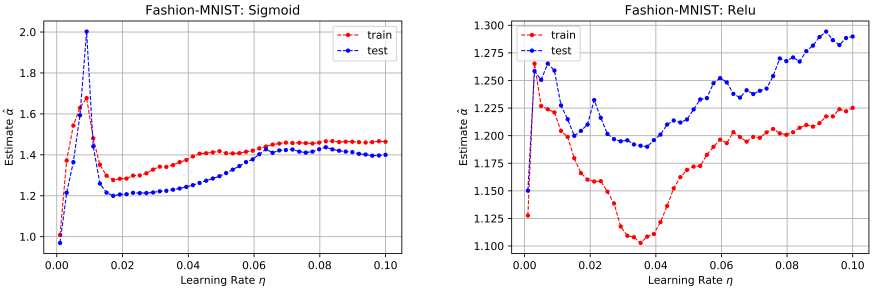

Figure 2: Estimation of α after varying η

4 Results

Effects of varying the activation function: By looking at both graphs in Figure 1 we see that stochastic gradient noise is far from Gaussian, clearly showing the assumption made by many to be false. Using sigmoid activation function, does give less heavy tails (since α is closer to 2) but we see huge swings especially in the test data-set. But with the epoch getting larger, the train and test data seems to settle to tail index of 1.5 and 1.6 respectively. Whereas ReLu graph shows SGN following a much more heavy tailed distribution with lower variance than sigmoid. This could be because ReLu causes all the negative neurons to die off when you calculate its gradient and setting all positive neurons to 1; so there is not much variation when calculating the SGN. Overall, it can be seen that the value of epoch has a much smaller effect on ReLu than it does on sigmoid.

Effects of varying the learning rate η: If you focus on sigmoid graph in Figure 2, you can see that for $\eta \in [0.01 - \epsilon, 0.01 + \epsilon]$ for $\epsilon > 0$ and small, the value of tail index is around 2, which means that we have found a neighbourhood of η, where infact the SGN behaves like a Gaussian. But, this neighbourhood is very small, and α jumps back to more heavy tail distribution as η increases. Infact, for small η, we see huge fluctuations in the tail index jumping from 1 to 2 in just increment of 0.01 in η. Perhaps, there does not exist a single value of α which can be used to approximate noise. On the other hand, ReLu graphs give a much more heavy tailed distribution; more-so for $\eta \in [0.001, 0.04]$ which is close to Cauchy distribution. Overall, it can be seen that tail index of the stochastic noise is very sensitive to η and the activation function.

5 Conclusion

In this report, we showed that gradient noise in deep learning is highly non-Gaussian and follows a more SαS distribution with $\alpha < 2$. We also examined SGD by considering them as Levy Driven SDE to shed more light on the metastability theory and why SDEs seem to prefer a wider minima. We also expressed our concerns that analysis of SDEs might not be valid when explaining behaviour in discretized process like SGD since they are very sensitive to learning rate η. Also the assumption that critical points of objective function are not saddle points is not suitable in majority of situations and more research needs to be done in that area. Furthermore, by our numerics, we see a large oscillation in α due to both epochs and learning rate η, thus, it is possible to consider the tail index as a function of time and other parameters which changes its value accordingly; this we leave for our future endeavours.

Acknowledgements

This report is mainly inspired by the papers *'A tail-index analysis of stochastic gradient noise in deep neural network'* [6] and *'Identifying and attacking the saddle point problem in high-dimensional non-convex optimization'* [10].

References

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier, 1971.

[3] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Benoit B Mandelbrot. *Fractals and scaling in finance: Discontinuity, concentration, risk. Selecta volume E.* Springer Science & Business Media, 2013.

[6] Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. *arXiv preprint arXiv:1901.06053*, 2019.

[7] Thanh Huy Nguyen, Umut Şimşekli, Mert Gürbüzbalaban, and Gaël Richard. First exit time analysis of stochastic gradient descent under heavy-tailed gradient noise. *arXiv preprint arXiv:1906.09069*, 2019.

[8] Larissa Serdukova, Yayun Zheng, Jinqiao Duan, and Jürgen Kurths. Stochastic basins of attraction for metastable states. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(7):073117, 2016.

[9] Ilya Pavlyukevich. Cooling down lévy flights. *Journal of Physics A: Mathematical and Theoretical*, 40(41):12299, 2007.

[10] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[11] Mohammad Mohammadi, Adel Mohammadpour, and Hiroaki Ogata. On estimating the tail index and the spectral measure of multivariate α-stable distributions. *Metrika*, 78(5):549–561, 2015.