A Variant of Gradient Descent Algorithm Based on Gradient Averaging

Saugata Purkayastha SAU.PUR9@GMAIL.COM
Assam Don Bosco University, India

Sukannya Purkayastha PURKAYASTHASUKANNYA020@GMAIL.COM
Indian Institute of Technology Kharagpur, India

Abstract

In this work, we study an optimizer, `Grad-Avg` to optimize error functions. We establish the convergence of the sequence of iterates of `Grad-Avg` mathematically to a minimizer (under boundedness assumption). We apply `Grad-Avg` along with some of the popular optimizers on regression as well as classification tasks. In regression tasks, it is observed that the behaviour of `Grad-Avg` is almost identical with Stochastic Gradient Descent (SGD). We present a mathematical justification of this fact. In case of classification tasks, it is observed that the performance of `Grad-Avg` can be enhanced by suitably scaling the parameters. Experimental results demonstrate that `Grad-Avg` converges faster than the other state-of-the-art optimizers for the classification task on two benchmark datasets.

1. Introduction

Gradient descent (GD) method [10] is one of the most popular algorithms to optimise an error function. Let $J : \mathbb{R}^n \to \mathbb{R}$ be a C^1 function. The classical Gradient Descent method is given by the following algorithm:

$$\theta = \theta - \alpha \nabla J(\theta)$$

where α is the constant step size, otherwise known as the learning rate ,$\nabla J(\theta)$ stands for the gradient of the function J and θ is the set of parameters. Although gradient descent algorithm is guaranteed to converge to the global minima (a local minima) for convex functions (non-convex functions), in case of large datasets the convergence can be slow. To overcome this, a variant of GD known as Stochastic Gradient Descent (SGD) [13] is applied. SGD, unlike GD avoids redundant computations but has a tendency to overshoot the minima in the process. Both GD and SGD, albeit being efficient in convex optimization, proved to be inefficient in case of non-convex surfaces because of the existence of *saddle points*, as observed in [3]. Keeping this in mind two more variants of GD, namely SGD with momentum [14] and Nesterov Accelerated Gradient algorithm [12] are commonly used in case of non-convex optimization. One is referred to [14] for a detailed discussion on the above mentioned optimizers. Further, [16] contains an excellent account of various analytical concepts occuring in the context of machine learning. We define the proposed optimizer, `Grad-Avg` motivated by the Heun's method [1] with the following iterative scheme:

$$\theta = \theta - \alpha \frac{1}{2}(\nabla J(\theta) + \nabla J(\theta - \alpha \nabla J(\theta))) \tag{1}$$

Thus we note that in the present optimizer the update of the parameter is done by considering the average of the gradients calculated at the previous position and the position of the parameter as suggested by the GD algorithm. The intuition lies in the fact that if the gradient becomes zero at a position which is not the minimum (say at a saddle point instead), even then the update of the parameter continues by virtue of the average of the gradients as mentioned above. The concept of gradient averaging is a known concept in this field. For instance [4] introduced a new optimization method using this concept. Also [6] used the average of model parameters obtained by letting the model to converge at multiple local minimas before making the final prediction . Our work, however uses the averaging of the gradients of the model parameter at every timestep to update the parameter's position.
Contributions: In this work, we develop a new optimizer, `Grad-Avg` based on gradient descent algorithms. We obtain its convergence for $\alpha \leq \frac{1}{3L}$ where L is the Lipschitz constant (under boundedness assumption). We propose a justification of its similar behaviour with GD for regression tasks. We empirically demonstrate its efficiency over other popular optimizers for classification task.

2. Assumptions

In section 1, we have introduced the proposed optimizer. Below we mention the assumptions which are necessary to establish the convergence of the optimizer:

(i) the function $J : \mathbb{R}^n \to \mathbb{R}$ is C^1 i.e. the function J has continuous first order partial derivatives.

(ii) the gradient ∇J of the function $J : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous i.e. there exists some $L > 0$ (known as the Lipschitz constant) such that $\|\nabla J(x) - \nabla J(y)\| < L \|x - y\|$ for all $x, y \in \mathbb{R}^n$.

Further, we make use of Monotone Convergence Theorem which states that every bounded below monotonic decreasing sequence of real numbers converges to the infimum. A detailed account of these concepts can be found in [7] and [15].

3. Our Proposed Algorithm

Algorithm 1: Algorithm for `Grad-Avg`

```
Data: $(x_i, y_i)$: Input, Output pair
Require: Information of gradient, $\nabla$ for function, $J$
Require: Learning rate, $\alpha$
Require: Initial parameter values, $\theta_0$
for t in range (epochs) do
    $\overline{\theta_{n+1}} = \theta_n - \alpha \cdot \nabla(J(\theta_n); (x_i, y_i));$
    $\theta_{n+1} = \theta_n - \alpha \cdot \frac{\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})}{2}$
end
```

4. Convergence Analysis

To analyze the convergence of equation (1), we first rewrite equation (1) in the following way:

$$\theta_{n+1} = \theta_n - \alpha\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})) \tag{2}$$

with

$$\overline{\theta_{n+1}} = \theta_n - \alpha\nabla J(\theta_n) \tag{3}$$

We note that if $\nabla J(\theta_n) = 0$, then both $\overline{\theta_{n+1}}$ and θ_{n+1} reduces to θ_n i.e. θ_n is the optimal value. We may thus assume that $\nabla J(\theta_n) \neq 0$ and hence $\|\nabla J(\theta_n)\| > 0$. We first establish that the sequence $\{J(\theta_n)\}$ where $\{\theta_n\}$ as defined by scheme (2) is monotonic decreasing in the following result. As such, by the Monotone Convergence Theorem , it follows that $\{J(\theta_n)\}$ converges to the infimum provided it is bounded.

Theorem 1 *Let $J : \mathbb{R}^n \to \mathbb{R}$ be a C^1 function such that ∇J is Lipschitz continuous with Lipschitz constant L. Then for $\alpha \leq \frac{1}{3L}$the sequence $\{J(\theta_n)\}$ where $\{\theta_n\}$ is defined by scheme (2) is monotonic decreasing.*

First of all, using the fact that $J : \mathbb{R}^n \to \mathbb{R}$, is Lipschitz continuous, we observe that for $\epsilon_0 > 0$, there is some $\delta_0 > 0$ such that when $\|\alpha\nabla J(\theta_n)\| < \delta_0$, one has

$$\left\|\nabla J(\overline{\theta_{n+1}}) - \nabla J(\theta_n)\right\| < L\delta_0 = \epsilon_0 \tag{4}$$

Let us now consider $f : \mathbb{R} \to \mathbb{R}$ and define

$$f(t) = J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))) \tag{5}$$

for $t \in \mathbb{R}$. Then by chain rule we get,

$$f'(t) = (\nabla J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})))) \cdot (-\alpha\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))) \tag{6}$$

where " $\cdot$ " stands for the usual dot product. Now by Fundamental Theorem of Calculus, we get

$$\begin{aligned} J(\theta_{n+1}) - J(\theta_n) \quad &= \textstyle\int_0^1(\nabla J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})))) \\ &\quad (-\alpha\tfrac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})))dt \end{aligned} \tag{7}$$

As by assumption J is C^1, hence for $\epsilon = \frac{2\delta}{\alpha t} - (\epsilon_0 + \frac{\delta_0}{\alpha})$ there is a $\delta > 0$ such that whenever $\left\|\alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\right\| < \delta$, one gets

$$\left\|\nabla J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))) - \nabla J(\theta_n)\right\| < \epsilon \tag{8}$$

Further, we have the following two inequalities:

$$\begin{aligned} &\nabla J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})))(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})) \\ &= (\nabla J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})) - \nabla J(\theta_n))(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})) + \nabla J(\theta_n)(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}) \\ &\geq \nabla J(\theta_n)(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})) - \left\|\nabla J(\theta_n - \alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})) - \nabla J(\theta_n)\right\| \left\|\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})\right\| \end{aligned}$$

and

$$\begin{aligned}
&\nabla J(\theta_n)(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\\
&= \left\|\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})\right\|^2 - \nabla J(\overline{\theta_{n+1}})(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\\
&= \left\|\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})\right\|^2 - (\nabla J(\overline{\theta_{n+1}}) - \nabla J(\theta_n) + \nabla J(\theta_n))(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\\
&\geq \left\|\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})\right\|^2 - \left\|\nabla J(\overline{\theta_{n+1}}) - \nabla J(\theta_n) + \nabla J(\theta_n))\right\| \left\|(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\right\|\\
&\geq \left\|\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}})\right\|^2 - (\left\|\nabla J(\overline{\theta_{n+1}}) - \nabla J(\theta_n)\right\| + \|\nabla J(\theta_n))\|) \left\|(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\right\|
\end{aligned}$$

Now using equation (8) and the above two inequalities in equation (7), one has

$$J(\theta_{n+1}) - J(\theta_n) \leq -\frac{\alpha}{2}[\frac{4\delta^2}{(\alpha t)^2} - (\epsilon_0 + \frac{\delta_0}{\alpha})\frac{2\delta}{\alpha t} - \epsilon\frac{2\delta}{\alpha t}] \qquad \leq 0$$

Thus $\{J(\theta_n)\}$ is a monotonic decreasing sequence. The proof will be complete once we show that $\epsilon > 0$. We note that

$$\left\|\alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\right\| \leq \left\|\alpha t\frac{1}{2}(\nabla J(\theta_n)\right\| + \left\|\alpha t\frac{1}{2}\nabla J(\overline{\theta_{n+1}}))\right\| < \frac{1}{2}t\delta_0 + \left\|\alpha t\frac{1}{2}\nabla J(\overline{\theta_{n+1}}))\right\|$$

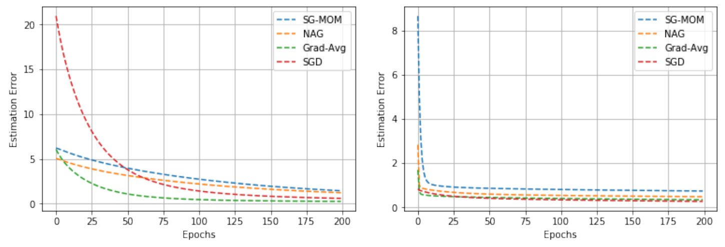
i.e. the norm of the sum given by the first expression of the above inequality is bounded by the last expression of the inequality with the minimum being $\frac{t}{2}\alpha_0$ which is attained when $\nabla J(\overline{\theta_{n+1}}) = 0$. Under the present hypothesis, it is possible only when $\overline{\theta_{n+1}}$ is the minimizer as $\overline{\theta_{n+1}}$ represents the iterates of GD as shown in equation (7). Thus, we clearly have $\delta > \frac{t}{2}\alpha_0$ as $\left\|\alpha t\frac{1}{2}(\nabla J(\theta_n) + \nabla J(\overline{\theta_{n+1}}))\right\| < \delta$. Now simple calculation yields $\epsilon > \frac{1}{\alpha}(\delta_0 - \frac{\delta_0}{2}) > \frac{\delta_0}{2\alpha} > 0$ since $\alpha < \frac{1}{3L}$ by hypothesis.

5. Numerical Experiments and Discussion

We perform experiments on two regression datasets, Boston Housing Dataset [5] and Condition Based Maintenance of Naval Propulsion Plants Data Set (CBM) [2]. The datasets contain N = 506 and 13 features and N = 11,934 and 15 features respectively where N is the number of datapoints. We perform experiments on two different initialization strategies of the neural network weights, the details of which are furnished in Appendix C. For the classification task, we make use of two of the standard benchmark datasets: the well-known **MNIST dataset** [8] and a harder drop-in replacement **MNIST Fashion dataset** [17]. Further details about the datasets and the learning models are provided in Appendix A and Appendix B.
From Figure 1 and Table 1, it is clear that the performance of `Grad-Avg` is almost identical with SGD in case of Regression Tasks. The reason for this lies in the *MSE* loss with which the model is trained in regression tasks. We present a mathematical justification of this phenomena considering a general C^2 function represented by a quadratic form. Let $f : \mathbb{R}^n \to \mathbb{R}$ be given by $f(x) = \frac{1}{2}x^TQx$. Then $\nabla f = Qx$ and $\nabla^2 f = Q$, the Hessian of f. Further, let the n eigenvalues of Q be given by λ_i for $i = 1, 2, \ldots, n$. Then the iterates for Gradient Descent Algorithm for f starting at x_0 are given in the following way [9]:

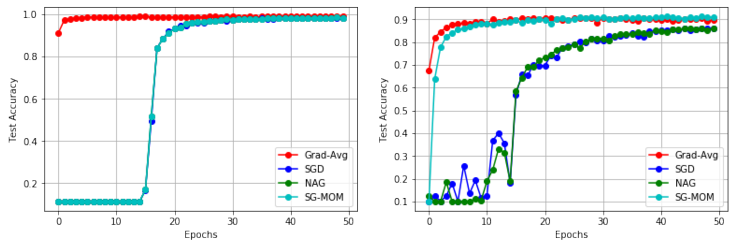
$$x_{t+1} = \sum_{i=1}^{n}(1 - \alpha\lambda_i)^{t+1} < e_i, x_0 > e_i \tag{9}$$

with e_i being the standard basis vectors. The above will certainly converge for $0 < \alpha < \frac{2}{L}$ where $L = max|\lambda_i|$ [12]. Moreover, Gradient Descent for f converges to a local minimizer or diverges to negative infinity with random initialization depending on whether each critical point of f is a local minimizer or a strict saddle provided $0 < \alpha < \frac{1}{L}$ with L being the Lipschitz constant of ∇f, as noted in [9]. It can be seen by the application of Taylor's theorem that the iterates of `Grad-Avg` for f are generated by $x - \alpha\nabla f(x) - O(\alpha^2)$ which for the above bound on α, reduces to $x - \alpha\nabla f(x)$ (upon neglecting terms having higher powers of α) which is just the SGD scheme. Hence the sequence of iterates of `Grad-Avg` coincides with equation (9) above in this case. Now in view of Theorem 1 and the discussion of the preceding paragraph, it follows that `Grad-Avg` converges to the local minimizer of f almost identically as SGD. In case of classification tasks, we note that while Figure 2 indicates that the performance of `Grad-Avg` is distinct as compared to other optimizers, for MNIST dataset there is yet some scope of improvement. We achieve this passing over to MNIST fashion dataset while keeping the hyper-parameters unchanged as indicated in Figure 2. We attribute this improvement to the complexity of the dataset.



(a) Boston Housing Dataset (b) CBM dataset

Figure 1: Mean Square Error plots for normal initialization of weights on CBM and Boston Housing Dataset



(a) MNIST Dataset (b) MNIST Fashion Dataset

Figure 2: Test set accuracy comparison on MNIST and MNIST Fashion dataset

Dataset	MOM	NAG	`Grad-Avg`	SGD
Boston Housing	14.31	2.31	**3.35**	3.36
CBM	0.76	0.48	**0.32**	0.31

Table 1: MSE on the test set for both the regression datasets using normal initialization.'MOM' stands for 'SGD with Momentum'

6. Conclusion

We have observed that for a regression task, `Grad-Avg` performs almost identically as SGD while for classification tasks, `Grad-Avg` outperforms other optimizers for large datasets applied to complex models. Overall the performance of `Grad-Avg` is comparable to that of SGD. This together with the fact that SGD surprisingly performs better than most of the standard optimizers irrespective of the task as noted in [11], makes `Grad-Avg` *as good as* SGD.

References

[1] Kendall E Atkinson. *An introduction to numerical analysis*. John wiley & sons, 2008.

[2] Andrea Coraddu, Luca Oneto, Aessandro Ghio, Stefano Savio, Davide Anguita, and Massimo Figari. Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1):136–153, 2016.

[3] 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.

[4] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[5] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. 1978.

[6] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[7] Erwin Kreyszig. *Introductory functional analysis with applications*, volume 1. wiley New York, 1978.

[8] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7:23, 2010.

[9] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

[10] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251:254, 2012.

[11] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

[12] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[13] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[14] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[15] Theodore Shifrin. *Multivariable mathematics: linear algebra, multivariable calculus, and manifolds*. John Wiley & Sons, 2005.

[16] Tuyen Trung Truong and Tuan Hang Nguyen. Backtracking gradient descent method for general c^1 functions, with applications to deep learning. *arXiv preprint arXiv:1808.05160*, 2018.

[17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Appendix A. Datasets

We discuss the various datasets used in detail here:

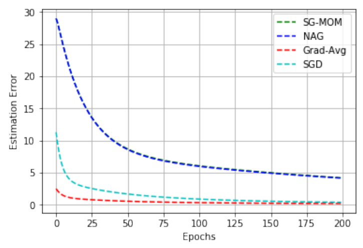
- **Boston Housing Dataset**: The Boston Housing Dataset [13] consists of data collected by US Censor Service concerning houses in the areas of Boston Housing mass. The dataset consists of various attributes such as per capita-income, proportion of non-retail business acres per town, etc. The price of a particular house given these attributes is to be predicted.

- **CBM Dataset**: Condition Based Maintenance of Naval Propulsion Plants Data Set (CBM) [2] consists of data from experiments carried out by a numerical simulator of a naval vessel (Frigate) characterized by a Gas Turbine (GT) propulsion plant. The GT propulsion plant is characterised by various components such as, Lever position, ship speed, Gas Turbine Shaft Torque, etc. Based on these attributes, the Gas Turbine Compressor Decay co-efficient is to be calculated.

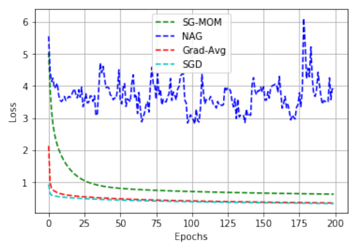
The datasets were split into $80\% - 20\%$ randomly for training and testing. For the classification task, we make use of two of the standard benchmark datasets: the well-known **MNIST dataset** [8] and a harder drop-in replacement **MNIST fashion dataset** [17]. Both of these datasets consist of 28*28 grayscale images. Each image needs to be classified into one of the 10 classes. For the experiments, we have used the pre-defined train-test split that consists of $60,000$ images for training and $10,000$ images for testing.

Appendix B. Hyper-parameters and Learning Models

- **Regression Task**: We use a single layer Neural Network as our learning model for the experiments. The model is parameterized by a set of weights, w and bias, b. For all the experiments, we use the same set of hyper-parameters. The learning rate is set to be $5e - 5$, momentum is set at 0.9 for SG-MOM and NAG, batch size is set to be 50 and the models are trained for 200 epochs.

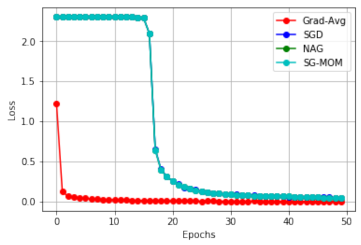


(a) Boston Housing Dataset

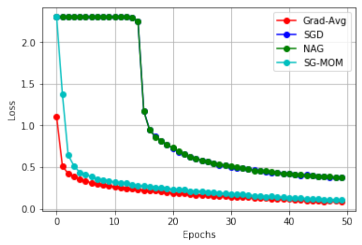


(b) CBM Dataset

Figure 3: Mean Square Error Plots on Boston Housing and CBM dataset



(a) MNIST dataset



(b) MNIST Fashion Dataset

Figure 4: Error plots on MNIST and MNIST Fashion Dataset

- **Classification Task**: We utilize a Convolutional Neural Network (CNN) as our learning model. The CNN consists of a convolutional layer with kernel size of 3 pixels having a 20 channel output. The output is followed by a max-pooling layer of 2 pixels. The output is then passed through a convolution layer of 5 pixels having a 50 channel output, followed by a max-pooling layer of the same configuration as before. The output is then flattened and connected to a layer of 128 neurons which is connected to a 10-neuron output layer.We train the model for 50 epochs with a learning rate of 0.01 for all the optimizers. The momentum value is set at 0.9. The batch size is set to be 128.

Appendix C. Error Plots

We show the error plots obtained for the Uniform Initialization of Weights for the Regression task on Boston Housing and CBM dataset in Figure 3.

Thus it is clear that `Grad-Avg` behaves identically with SGD in case of Uniform Initialization too on the same pair of datasets as considered under Normal Initialization. The justification which

Dataset	SG-MOM	NAG	`Grad-Avg`	SGD
Boston Housing	38.60	8.70	6.64	**6.61**
CBM	2.42	1.22	**0.51**	**0.51**

Table 2: MSE on the test set for both the regression datasets for uniform initialization.'SG-MOM' stands for 'SGD with Momentum'

we have provided in section 5 for Normal Initialization remains valid in this case also as the underlying error function is MSE. Further, owing to the same reason it may be noted that the convergence of `Grad-Avg` to a local minimizer is almost guaranteed (in the sense of [9]) irrespective of the choice of initialization.
Finally from the error plots, on the classification task as shown in Figure 4 on the same pair of datasets as considered in section 5, it is evident that `Grad-Avg` converges faster to the minimizer as compared to the other optimizers for the training set which further validates our claim of faster convergence of `Grad-Avg`.