A Variant of Gradient Descent Algorithm Based on Gradient Averaging

Saugata Purkayastha
Assam Don Bosco University, India

Sukannya Purkayastha
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 $J : \mathbb{R}^n \rightarrow \mathbb{R}$ be a $C^1$ function. The classical Gradient Descent method is given by the following algorithm:

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

where $\alpha$ is the constant step size, otherwise known as the learning rate, $\nabla J(\theta)$ stands for the gradient of the function $J$ and $\theta$ 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 occurring 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:

$$\theta = \theta - \alpha \frac{1}{2}(\nabla J(\theta) + \nabla J(\theta - \alpha \nabla J(\theta)))$$ (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 $J : \mathbb{R}^n \rightarrow \mathbb{R}$ is $C^1$ i.e. the function $J$ has continuous first order partial derivatives.

(ii) the gradient $\nabla J$ of the function $J : \mathbb{R}^n \rightarrow \mathbb{R}$ 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
    $\theta_{n+1} = \theta_n - \alpha \cdot \nabla J(\theta_n; (x_i, y_i))$
    $\theta_{n+1} = \theta_n - \alpha \cdot \frac{\nabla J(\theta_n) + \nabla J(\theta_{n+1})}{2}$
end
```
4. Convergence Analysis

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

\[ θ_{n+1} = θ_n - \alpha \frac{1}{2} (∇J(θ_n) + ∇J(θ_{n+1})) \]  

(2)

with

\[ \bar{θ}_{n+1} = θ_n - α ∇J(θ_n) \]  

(3)

We note that if ∇J(θ_n) = 0, then both θ_{n+1} and θ_{n+1} reduces to θ_n, i.e. θ_n is the optimal value. We may thus assume that ∇J(θ_n) ≠ 0 and hence ∥∇J(θ_n)∥ > 0. We first establish that the sequence \{J(θ_n)\} where \{θ_n\} as defined by scheme (2) is monotonic decreasing in the following result. As such, by the Monotone Convergence Theorem, it follows that \{J(θ_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 \( α \leq \frac{1}{2L} \) the sequence \{J(θ_n)\} where \{θ_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 \( ε_0 > 0 \), there is some \( δ_0 > 0 \) such that when \( ∥α∇J(θ_n)∥ < δ_0 \), one has

\[ ∥∇J(θ_{n+1}) - ∇J(θ_n)∥ < Lδ_0 = ε_0 \]  

(4)

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

\[ f(t) = J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) \]  

(5)

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

\[ f'(t) = (∇J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1})))) \cdot (-α \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) \]  

(6)

where “\( \cdot \)” stands for the usual dot product. Now by Fundamental Theorem of Calculus, we get

\[ J(θ_{n+1}) - J(θ_n) = \int_0^1 (∇J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1})))) \cdot (-α \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) \, dt \]  

(7)

As by assumption \( J \) is \( C^1 \), hence for \( ε = \frac{2δ}{αt} - (ε_0 + \frac{δ_0}{α}) \) there is a \( δ > 0 \) such that whenever

\[ ∥αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))∥ < δ, \]  

one gets

\[ ∥∇J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) - ∇J(θ_n)∥ < ε \]  

(8)

Further, we have the following two inequalities:

\[ ∇J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) (∇J(θ_n) + ∇J(θ_{n+1})) \]

\[ = (∇J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) - ∇J(θ_n)) (∇J(θ_n) + ∇J(θ_{n+1})) + ∇J(θ_n)(∇J(θ_n) + ∇J(θ_{n+1})) \]

\[ ≥ ∇J(θ_n)(∇J(θ_n) + ∇J(θ_{n+1})) - ∥∇J(θ_n - αt \frac{1}{2}(∇J(θ_n) + ∇J(θ_{n+1}))) - ∇J(θ_n)∥ ∥∇J(θ_n) + ∇J(θ_{n+1})∥ \]
and
\[
\nabla J(\theta_n)(\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})) = \|\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})\|^2 - \nabla J(\bar{\theta}_{n+1})(\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1}))
\]
\[
\quad = \|\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})\|^2 - (\nabla J(\bar{\theta}_{n+1}) - \nabla J(\theta_n) + \nabla J(\theta_n))(\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1}))
\]
\[
\quad \geq \|\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})\|^2 - \|\nabla J(\bar{\theta}_{n+1}) - \nabla J(\theta_n) + \nabla J(\theta_n)\| \|\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})\|
\]
\[
\quad \geq \|\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})\|^2 - (\|\nabla J(\bar{\theta}_{n+1}) - \nabla J(\theta_n)\| + \|\nabla J(\theta_n)\|) \|\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1})\|
\]

Now using equation (8) and the above two inequalities in equation (7), one has
\[
J(\theta_{n+1}) - J(\theta_n) \leq -\alpha t_0 \left(\frac{4\delta^2}{\alpha^2} - (\epsilon_0 + \frac{2\delta}{\alpha}) \frac{2\delta}{\alpha} - \epsilon \frac{2\delta}{\alpha^2}\right) \leq 0
\]

Thus \(\{J(\theta_n)\}\) is a monotonic decreasing sequence. The proof will be complete once we show that \(\epsilon > 0\). We note that
\[
\|\alpha t \frac{1}{2}(\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1}))\| \leq \|\alpha t \frac{1}{2}(\nabla J(\theta_n))\| + \|\alpha t \frac{1}{2}(\nabla J(\bar{\theta}_{n+1}))\| < \frac{1}{2} t_0 + \|\alpha t \frac{1}{2}(\nabla J(\bar{\theta}_{n+1}))\|
\]
i.e. the norm of the sum given by the first expression of the inequality is bounded by the last expression of the inequality with the minimum being \(\frac{1}{2} \alpha \) which is attained when \(\nabla J(\bar{\theta}_{n+1}) = 0\).

Under the present hypothesis, it is possible only when \(\theta_{n+1}\) is the minimizer as \(\theta_{n+1}\) represents the iterates of GD as shown in equation (7). Thus, we clearly have \(\delta > \frac{1}{2} \alpha_0\) as \(\|\alpha t \frac{1}{2}(\nabla J(\theta_n) + \nabla J(\bar{\theta}_{n+1}))\| < \delta\). Now simple calculation yields \(\epsilon > \frac{1}{\alpha} (\delta_0 - \frac{\delta_0}{2\alpha}) > \frac{\delta_0}{2\alpha} > 0\) since \(\alpha < \frac{1}{\delta_0}\) 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 data points.

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 \(f(x) = \frac{1}{2} x^T Q x\). Then \(\nabla f = Q x\) and \(\nabla^2 f = Q\), the Hessian of \(f\). Further, let the \(n\) eigenvalues of \(Q\) be given by \(\lambda_i\) 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]:

\[
x_{t+1} = \sum_{i=1}^{n} (1 - \alpha \lambda_i)^{t+1} < e_i, x_0 > e_i
\]
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 \( \nabla f \), 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 \( \alpha \), reduces to \( x - \alpha \nabla f(x) \) (upon neglecting terms having higher powers of \( \alpha \)) 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.

![Graphs](image)

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

![Graphs](image)

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MOM</th>
<th>NAG</th>
<th>Grad-Avg</th>
<th>SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boston Housing</strong></td>
<td>14.31</td>
<td>2.31</td>
<td><strong>3.35</strong></td>
<td>3.36</td>
</tr>
<tr>
<td><strong>CBM</strong></td>
<td>0.76</td>
<td>0.48</td>
<td><strong>0.32</strong></td>
<td>0.31</td>
</tr>
</tbody>
</table>

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


A Variant of Gradient Descent Algorithm Based on Gradient Averaging


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 Variant of Gradient Descent Algorithm Based on Gradient Averaging

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SG-MOM</th>
<th>NAG</th>
<th>Grad-Avg</th>
<th>SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston Housing</td>
<td>38.60</td>
<td>8.70</td>
<td>6.64</td>
<td>6.61</td>
</tr>
<tr>
<td>CBM</td>
<td>2.42</td>
<td>1.22</td>
<td>0.51</td>
<td>0.51</td>
</tr>
</tbody>
</table>

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 under-
lying 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.