SM3: Memory Efficient Adaptive Optimization

Jaime Campos Salas, Rahul Das, Zhixiang Hu

I. Introduction

The use of adaptive gradient-based optimizers is widespread in machine learning, the most popular of which are Adam, Adagrad, and Adafactor. These optimizers utilize cumulative second-order statistics to tune the learning rate of each parameter during the optimization process, presenting numerous advantages, most importantly, constraining the time and space requirements of the methods to be linear in the number of parameters.

However, the existing optimizers still present significant memory overhead when training models with billions of parameters. This places a limitation on the size of the model and on the batch size during training, which can severely effect the accuracy of the model. By improving on the memory overhead of existing optimizers, we can train ever more complex and accurate models, which is especially pertinent to the field of natural language processing. Anil et al. [2] present a novel adaptive optimization method which retains the benefits of conventional per-parameter adaptivity while significantly reducing memory requirements.

The optimization algorithm, presented below in pseudocode ($SM3$-I), utilizes covers over the parameters, i.e. a collection of k nonempty set $\{S_r\}_{r=1}^{k}$ such that $S_r \subseteq [d]$ and $\cup_r S_r = [d]$. For each set S_r in the cover, the algorithm maintains a cumulative sum $\mu_t(r)$ of the maximal variance over all gradient entries $j \in S_r$. Then, for each parameter i, the minimum is taken over all variables $\mu_t(r)$ associated with sets which cover i ($S_r \ni i$). The square root of this minimum, $\sqrt{\nu_t(i)}$, is used to determine the learning rate corresponding to the i'th gradient entry. As such, the algorithm is named Square-root of Minima of Sums of Maxima of Squared-gradients Method, or simply, $SM3$.

SM3-I(η):
- **Input:** learning rate η
- initialize $w_1 = 0$; $\forall r \in [k]$: $\mu_0(r) = 0$;
- **for** $t = 1, \ldots, T$ **do**
  - receive gradient $g_t = \nabla \ell_t(w_t)$
  - **for** $r = 1, \ldots, k$ **do**
    - set $\mu_t(r) \leftarrow \mu_{t-1}(r) + \max_{j \in S_r} g_t^2(j)$
  - **end**
  - **for** $i = 1, \ldots, d$ **do**
    - set $\nu_t(r) \leftarrow \min_{r: S_r \ni i} \mu_t(r)$
  - **end**
  - update $w_{t+1}(i) \leftarrow w_t(i) - \eta g_t(i) / \sqrt{\nu_t(i)}$
- **end**

They also propose a variant of, $SM3$-II, which they claim provides a tighter upper bound on the cumulative gradient squares, as compared to $SM3$-I.

SM3-II(η):
- **Input:** learning rate η
- initialize $w_1 = 0$; $\forall r \in [k]$: $\mu'_0(r) = 0$;
- **for** $t = 1, \ldots, T$ **do**
  - receive gradient $g_t = \nabla \ell_t(w_t)$
  - initialize $\mu'_t(r) = 0$ $\forall r \in [k]$
  - **for** $r = 1, \ldots, d$ **do**
    - $\nu'_t(i) \leftarrow \min_{r: S_r \ni i} \mu'_{t-1}(r) + g_t^2(i)$
    - $w'_{t+1}(i) \leftarrow w_t(i) - \eta g_t(i) / \sqrt{\nu'_t(i)}$
    - **for** *all* $r : S_r \ni i$ **do**
      - $\mu'_t(r) \leftarrow \max\{\mu'_t(r), \nu'_t(i)\}$
    - **end**
  - **end**
- **end**

II. Experimental Setup

In the following sections, we will compare the performance of $SM3$ with other adaptive (Adam, Adagrad) and non-adaptive (SGD) optimizers.

To test the optimizers, we consider a language modelling task, specifically dependency parsing. For this task, the model must parse the sentence into a tree structure by correctly labelling each word with the correct part-of-speech and predicting the relationship between words in the sentence. The model used is a feed-forward neural network with an embedding layer and two hidden layers, which produces a categorical output.

The dataset used is the Penn Treebank WSJ[1], which is split into training (∼40,000 sentences) and test (∼2,500 sentences) sets. Each optimizer considered is initialized with a learning rate $= 0.01$. For evaluation, we consider the training loss, training categorical accuracy, CPU usage, and runtime. To evaluate the model's accuracy in the task of dependency parsing, we use macro (sentence-based) Labelled Attachment Score (LAS) and Unlabelled Attachment Score (UAS), which provide useful information about how frequently the model correctly labels words and their relations to other words in the sentence.

[1] https://www.dropbox.com/s/la8oxeu4aladhpg/data.zip?dl=0

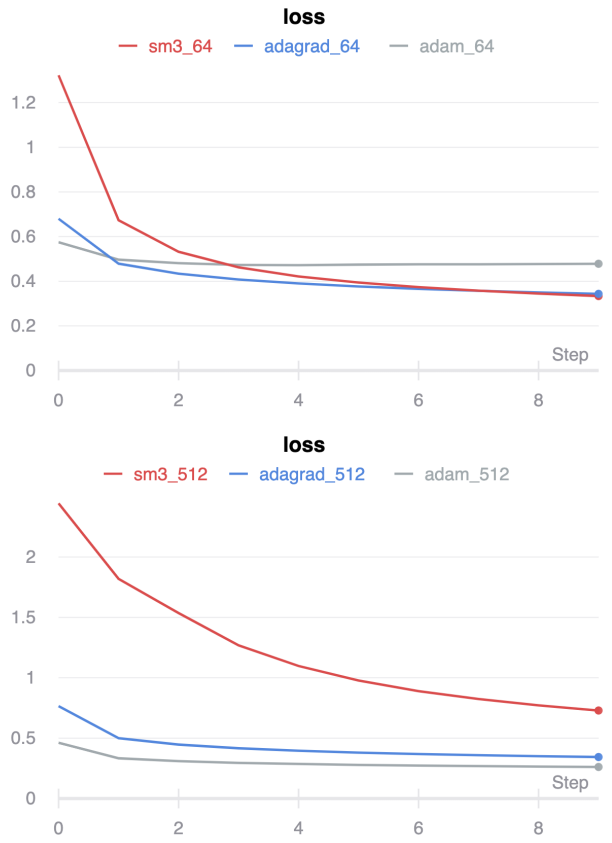

Fig. 1. Comparison of SM3, Adagrad, and Adam across 10 training epochs. Upper figure shows training with batch size=64, and lower figure shows batch size=512

III. Experimental Results

We first consider the training loss over the course of 10 training steps (epochs), as shown in Fig. 1. It can be seen that the $SM3$-optimized models have significantly higher training loss at the end of the first few epochs, after which the results are comparable with Adam and Adagrad. In the case of small batch size (64), training loss decreases rapidly, eventually reaching the level of Adagrad and outperforming Adam. However, when training with large batch size (512), convergence is slower and significantly worse than both Adam and Adagrad.

Optimizer	Batch Size	CPU%	Runtime (mins)
SM3	64	38.3	8.5
SGD	64	40.6	9.1
Adagrad	64	42.8	10.3
Adam	64	53.7	13.1
SM3	512	46.6	2.8
SGD	512	46.8	2.9
Adagrad	512	54.1	3.9
Adam	512	58.1	4.0

TABLE I
Comparison of SM3, SGD, Adagrad, and Adam based on CPU usage and runtime (training)

We now consider the time requirements of the various optimizers. Table III shows the CPU% and runtime when training the model for 10 epochs. $SM3$ presents significant time savings over other optimizers, up . to 25% in some cases. From the results, it is clear that $SM3$ is less computationally expensive than the adaptive optimizers Adam and Adagrad, and is comparable to standard SGD.

Optimizer	LAS	UAS
SM3	0.732	0.783
SGD	0.727	0.778
Adagrad	0.713	0.769
Adam	0.745	0.794

TABLE II
Performance of SM3, SGD, Adagrad, and Adam on test set. Metrics used are Macro Labelled Attachment Score (LAS) and Macro Unlabelled Attachment Score (UAS)

Finally, we evaluate the actual performance of the models trained with different optimizers. The metrics used measure the frequency with which the model accurately labels words and identifies the parent word ('head') in the dependency tree. Specifically, LAS measures the percentage of the time that words are assigned the correct head AND label, and UAS measures the percentage of the time that words are assigned the correct head. The dependency parser used for this task performs adequately as compared to state-of-the-art dependency parsers, which often achieve LAS and UAS values exceeding 0.900. From Table II, it can be seen that $SM3$ performs as well or better than other optimizers, with the exception of Adam.

IV. Evaluation

The experiments results show that $SM3$ presents significant savings over the usage of CPU. For example, there's a 28.68% reduction when compared to Adam in the case of batch size 64. This verifies the authors' claim that $SM3$ has memory efficiency over other adaptive algorithms. The better utilization of memory allowed the CPU to be used more efficiently.

When we compare the losses of different algorithms, we also found that as the number of epoch increases, $SM3$ tends to converge to the performance of Adagrad. This is also consistent with the result shown in Proposition 1 from the paper (or Proposition 3 in section VI below) that $SM3$ has a convergence bound that is of the same order of magnitude as Adagrad's.

V. Conclusion

Based on experiments with dataset, SM3 exhibits superior efficiency and performance compared to existing adaptive optimizers and SGD.

VI. Analysis of $SM3$-I and $SM3$-II

Here we give a summary of the convergence properties of the algorithms and show that $SM3-II$ has a tighter regret bound than $SM3-I$. We start with some preliminary results proved in [1, Proposition 3] and [2, Claim 2, Proposition 3].

Proposition 1. *Assume that the loss functions $\ell_1, \ell_2, ...$ are convex, and let $w_1, w_2, ...$ be the iterates generated by $SM3-I$ or $SM3-II$. Let $H_t = diag(\nu_t^{1/2})$. Assume $max_t \|w_t - w^*\|_\infty^2 \leq D$. Then for any $w^* \in \mathbb{R}^d$, the regret of $SM3-I$ or $SM3-II$ is bounded by:*

$$\frac{1}{2\eta}\sum_{t=1}^{T}(\|w_t - w^*\|_{H_t}^2 - \|w_{t+1} - w^*\|_{H_t}^2) + \frac{\eta}{2}\sum_{t=1}^{T}(\|g_t\|_{H_t}^*)^2$$

where $\|x\|_H^2$ is defined as $\sqrt{x^T H x}$ and $\|x\|_H^*$ as $\sqrt{x^T H^{-1} x}$.

Proof. See Duchi et., 2011: Proposition 3. □

Proposition 2. *For any $i \in [d]$, the sequences $\nu_1(i)$, $\nu_2(i)$, ..., from $SM3-I$ and $\nu_1^{'}(i)$, $\nu_2^{'}(i)$, ..., from $SM3-II$ are monotonically increasing. Fix a sequence of gradients g_1, g_2, ... we have that for all t, i that $\sum_{s=1}^{t} g_s^2(i) \leq \nu_t^{'}(i) \leq v_t(i)$.*

Proof. See Anil et., 2019: Claim 2 and Proposition 3. □

Proposition 3. *Given the setup in Proposition 1, let $w_1, w_2, ...$ be the iterates generated by $SM3-I$ and $w_1^{'}, w_2^{'}, ...$ be the iterates from $SM3-II$. Let also $H_t^{'}$ be the diagonal matrix contains the sequence values $\nu_1^{'}(i)$, $\nu_2^{'}(i)$, ..., and set the learning rate $\eta = D$. Then we have the following convergence bounds for the two algorithms:*

$$\sum_{t=1}^{T}(\ell_t(w_t^{'}) - \ell_t(w^*)) \leq$$
$$\frac{3}{2}D\sum_{t=1}^{d}\sqrt{\sum_{t=1}^{T}\left[g_t^2(i) + \min_{r:S_r \ni i}\max\{\mu_{t-1}^{'}(r), \nu_{t-1}^{'}(i)\}\right]}$$
$$\leq \sum_{t=1}^{T}(\ell_t(w_t) - \ell_t(w^*)) \leq$$
$$\frac{3}{2}D\sum_{t=1}^{T}\sqrt{\min_{r:S_r \ni i}\sum_{t=1}^{T}\max_{j \in S_r} g_t^2(j)}$$

with $\mu_0(r) = \nu_0(r) = \mu_0^{'}(r) = \nu_0^{'}(r) = 0$.

Proof. From Proposition 1, we have:

$$\sum_{t=1}^{T}(\ell_t(w_t) - \ell_t(w^*)) \leq$$
$$\frac{1}{2\eta}\sum_{t=1}^{T}(\|w_t - w^*\|_{H_t}^2 - \|w_{t+1} - w^*\|_{H_t}^2) + \frac{\eta}{2}\sum_{t=1}^{T}(\|g_t\|_{H_t}^*)^2$$

The first term on the RHS can be bounded as follows:

$$\begin{aligned}2\eta(I) &\leq \sum_{t=1}^{T}(\nu_t^{1/2} - \nu_{t-1}^{1/2}) \cdot (w_t - w^*)^2 \\ &\leq \sum_{t=1}^{T}(\nu_t^{1/2} - \nu_{t-1}^{1/2}) \cdot (\|w_t - w^*\|_\infty^2 \mathbf{1}_d) \\ &\leq D^2(\nu_T^{1/2} \cdot \mathbf{1}_d) = D^2 \operatorname{Tr}(H_T)\end{aligned}$$

For the second term, let $\gamma_t(i) = \sum_{s=1}^{t} g_s^2(i)$ and consider the positive definite diagonal matrix $G_t = diag(\gamma_t^{1/2})$. Use the results from [3, Lemma 2] with $\Phi(G) = \operatorname{Tr}(G)$, we have:

$$\begin{aligned}\sum_{t=1}^{T}(\|g_t\|_{G_t}^*)^2 &\leq \sum_{t=1}^{T}(\|g_t\|_{G_T}^*)^2 + \operatorname{Tr}(G_T) \\ &= \gamma_T^{-1/2} \cdot \gamma_T = 2\operatorname{Tr}(G_T).\end{aligned}$$

Now from Proposition 2, we know for all t, $H_t \succeq H_t^{'} \succeq G_t$. Then we have:

$$\frac{2}{\eta}(II) \leq \sum_{t=1}^{T}(\|g_t\|_{G_t}^*)^2 \leq 2\operatorname{Tr}(G_T) \leq 2\operatorname{Tr}(H_T^{'}) \leq 2\operatorname{Tr}(H_T)$$

Then we have:

$$\sum_{t=1}^{T}\ell_t(w_t) - \ell_t(w^*) \leq \left(\frac{D^2}{2\eta} + \eta\right)\operatorname{Tr}(H_T) = \frac{3}{2}D\operatorname{Tr}(H_T)$$
$$\sum_{t=1}^{T}\ell_t(w_t^{'}) - \ell_t(w^*) \leq \frac{3}{2}D\operatorname{Tr}(H_T^{'}) \leq \frac{3}{2}D\operatorname{Tr}(H_T)$$

Plug in the definitions for ν_T and $\nu_T^{'}$ we then recover the results stated in the Proposition. □

VII. BIBLIOGRAPHY

[1] J. Duchi, E. Hazan, and Y. Singer. *Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[2] R. Anil, V. Gupta, T. Koren, Y. Singer. *Memory-Efficient Adaptive Optimization.* https://arxiv.org/abs/1901.11150.

[3] V. Gupta, T. Koren, and Y. Singer. *Shampoo: Preconditioned stochastic tensor optimization. In Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1842–1850, 2018.