Adaptive Optimizers with Sparse Group Lasso for Neural Networks in CTR Prediction

Yun Yue, Yongchao Liu, Suo Tong, Minghao Li, Zhen Zhang, Chunyang Wen, Huanjun Bao, Lihong Gu, Jinjie Gu, Yixiang Mu

Ant Group, No. 556 Xixi Road, Xihu district, Hangzhou, Zhejiang Province, China
{yueyun.yy, yongchao.ly, tongsuo.ts, chris.lmh, elliott.zz, chengfu.wcy, alex.bao, lihong.glh, jinjie.gujj, yixiang.myx}@antgroup.com

Abstract. We develop a novel framework that adds the regularizers of the sparse group lasso to a family of adaptive optimizers in deep learning, such as Momentum, Adagrad, Adam, AMSGrad, AdaHessian, and create a new class of optimizers, which are named Group Momentum, Group Adagrad, Group Adam, Group AMSGrad and Group AdaHessian, etc., accordingly. We establish theoretically proven convergence guarantees in the stochastic convex settings, based on primal-dual methods. We evaluate the regularized effect of our new optimizers on three large-scale real-world ad click datasets with state-of-the-art deep learning models. The experimental results reveal that compared with the original optimizers with the post-processing procedure which uses the magnitude pruning method, the performance of the models can be significantly improved on the same sparsity level. Furthermore, in comparison to the cases without magnitude pruning, our methods can achieve extremely high sparsity with significantly better or highly competitive performance.

Keywords: adaptive optimizers · sparse group lasso · DNN models · online optimization.

1 Introduction

With the development of deep learning, deep neural network (DNN) models have been widely used in various machine learning scenarios such as search, recommendation and advertisement, and achieved significant improvements. In the last decades, different kinds of optimization methods based on the variations of stochastic gradient descent (SGD) have been invented for training DNN models. However, most optimizers cannot directly produce sparsity which has been proven effective and efficient for saving computational resource and improving model performance especially in the scenarios of very high-dimensional data. Meanwhile, the simple rounding approach is very unreliable due to the inherent low accuracy of these optimizers.

In this paper, we develop a new class of optimization methods, that adds the regularizers especially sparse group lasso to prevalent adaptive optimizers, and retains the characteristics of the respective optimizers. Compared with the

original optimizers with the post-processing procedure which use the magnitude pruning method, the performance of the models can be significantly improved on the same sparsity level. Furthermore, in comparison to the cases without magnitude pruning, the new optimizers can achieve extremely high sparsity with significantly better or highly competitive performance. In this section, we describe the two types of optimization methods, and explain the motivation of our work.

1.1 Adaptive Optimization Methods

Due to the simplicity and effectiveness, adaptive optimization methods [20,17,4,27,8,19,26] have become the de-facto standard algorithms used in deep learning. There are multiple variants, but they can be represented using the general update formula [19]:

$$x_{t+1} = x_t - \alpha_t m_t / \sqrt{V_t}, \tag{1}$$

where α_t is the step size, m_t is the first moment term which is the weighted average of gradient g_t and V_t is the so called second moment term that adjusts updated velocity of variable x_t in each direction. Here, $\sqrt{V_t} := V_t^{1/2}$, $m_t/\sqrt{V_t} := \sqrt{V_t}^{-1} \cdot m_t$. By setting different m_t, V_t and α_t , we can derive different adaptive optimizers including Momentum [17], Adagrad [4], Adam [8], AMSGrad [19] and AdaHessian [26], etc. See Table 1.

Table 1. Adaptive optimizers with choosing different m_t, V_t and α_t.

Optimizer	$\boldsymbol{m_t}$	$\boldsymbol{V_t}$	$\boldsymbol{\alpha_t}$
Sgd	g_t	$\mathbb{I}$	$\frac{\alpha}{\sqrt{t}}$
Momentum	$\gamma m_{t-1} + g_t$	$\mathbb{I}$	α
Adagrad	g_t	$\mathrm{diag}(\sum_{i=1}^{t} g_i^2)/t$	$\frac{\alpha}{\sqrt{t}}$
Adam	$\beta_1 m_{t-1} + (1-\beta_1) g_t$	$\beta_2 V_{t-1} + (1-\beta_2)\mathrm{diag}(g_t^2)$	$\frac{\alpha\sqrt{1-\beta_2^t}}{1-\beta_1^t}$
AMSGrad	$\beta_1 m_{t-1} + (1-\beta_1) g_t$	$\max(V_{t-1}, \beta_2 V_{t-1} + (1-\beta_2)\mathrm{diag}(g_t^2))$	$\frac{\alpha\sqrt{1-\beta_2^t}}{1-\beta_1^t}$
AdaHessian	$\beta_1 m_{t-1} + (1-\beta_1) g_t$	$\beta_2 V_{t-1} + (1-\beta_2) D_t^2$ *	$\frac{\alpha\sqrt{1-\beta_2^t}}{1-\beta_1^t}$

* $D_t = \mathrm{diag}(H_t)$, where H_t is the Hessian matrix.

1.2 Regularized Optimization Methods

Follow-the-regularized-leader (FTRL) [12,11] has been widely used in click-through rates (CTR) prediction problems, which adds ℓ_1-regularization (lasso) to logistic regression and can effectively balance the performance of the model and the sparsity of features. The update formula [11] is:

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|x - x_s\|_2^2 + \lambda_1 \|x\|_1, \tag{2}$$

where $g_{1:t} = \sum_{s=1}^{t} g_s$, $\frac{1}{2}\sum_{s=1}^{t} \sigma_s \|x - x_s\|_2^2$ is the strong convex term that stabilizes the algorithm and $\lambda_1 \|x\|_1$ is the regularization term that produces sparsity. However, it doesn't work well in DNN models since one input feature can correspond to multiple weights and lasso only can make single weight zero hence can't effectively delete features.

To solve above problem, [16] adds the ℓ_{21}-regularization (group lasso) to FTRL, which is named G-FTRL. [25] conducts the research on a group lasso method for online learning that adds ℓ_{21}-regularization to the algorithm of Dual Averaging (DA) [15], which is named DA-GL. Even so, these two methods cannot be applied to other optimizers. Different scenarios are suitable for different optimizers in the deep learning fields. For example, MOMENTUM [17] is typically used in computer vision; ADAM [8] is used for training transformer models for natural language processing; and ADAGRAD [4] is used for recommendation systems. If we want to produce sparsity of the model in some scenario, we have to change optimizer which probably influence the performance of the model.

1.3 Motivation

Eq. (1) can be rewritten into this form:

$$x_{t+1} = \arg\min_x m_t \cdot x + \frac{1}{2\alpha_t} \| \sqrt{V_t}^{\frac{1}{2}} (x - x_t) \|_2^2. \tag{3}$$

Furthermore, we can rewrite Eq. (3) into

$$x_{t+1} = \arg\min_x m_{1:t} \cdot x + \sum_{s=1}^{t} \frac{1}{2\alpha_s} \| Q_s^{\frac{1}{2}} (x - x_s) \|_2^2, \tag{4}$$

where $m_{1:t} = \sum_{s=1}^{t} m_s$, $\sum_{s=1}^{t} Q_s / \alpha_s = \sqrt{V_t}/\alpha_t$. It is easy to prove that Eq. (3) and Eq. (4) are equivalent using the method of induction. The matrices Q_s can be interpreted as generalized learning rates. To our best knowledge, V_t of Eq. (1) of all the adaptive optimization methods are diagonal for the computation simplicity. Therefore, we consider Q_s as diagonal matrices throughout this paper.

We find that Eq. (4) is similar to Eq. (2) except for the regularization term. Therefore, we add the regularization term $\Psi(x)$ to Eq. (4), which is the sparse group lasso penalty also including ℓ_2-regularization that can diffuse weights of neural networks. The concrete formula is:

$$\Psi_t(x) = \sum_{g=1}^{G} \left(\lambda_1 \|x^g\|_1 + \lambda_{21} \sqrt{d_{x^g}} \|A_t^{\frac{1}{2}} x^g\|_2 \right) + \lambda_2 \|x\|_2^2, \tag{5}$$

where λ_1, λ_{21}, λ_2 are regularization parameters of ℓ_1, ℓ_{21}, ℓ_2 respectively, G is the total number of groups of weights, x^g is the weights of group g and d_{x^g} is the size of group g. In DNN models, each group is defined as the set of outgoing weights from a unit which can be an input feature, or a hidden neuron, or a bias unit (see, e.g., [22]). A_t can be arbitrary positive matrix satisfying $A_{t+1} \succeq A_t$, e.g., $A_t = \mathbb{I}$.

In Section 2.1, we let $A_t = (\sum_{s=1}^{t} \frac{Q_s^g}{2\alpha_s} + \lambda_2 \mathbb{I})$ just for solving the closed-form solution directly, where Q_s^g is a diagonal matrix whose diagonal elements are part of Q_s corresponding to x_g. The ultimate update formula is:

$$x_{t+1} = \arg\min_x m_{1:t} \cdot x + \sum_{s=1}^{t} \frac{1}{2\alpha_s} \|Q_s^{\frac{1}{2}}(x - x_s)\|_2^2 + \Psi_t(x). \tag{6}$$

1.4 Outline of Contents

The rest of the paper is organized as follows. In Section 1.5, we introduce the necessary notations and technical background.

In Section 2, we present the closed-form solution of Eq. (4) and the algorithm of general framework of adaptive optimization methods with sparse group lasso. We prove the algorithm is equivalent to adaptive optimization methods when regularization terms vanish. In the end, we give two concrete examples of the algorithm.[1]

In Section 3, we derive the regret bounds of the method and convergence rates.

In Section 4, we validate the performance of new optimizers in the public datasets.

In Section 5, we summarize the conclusion.

Appendices A-D contain technical proofs of our main results and Appendix E includes the additional details of the experiments of Section 4.

1.5 Notations and Technical Background

We use lowercase letters to denote scalars and vectors, and uppercase letters to denote matrices. We denote a sequence of vectors by subscripts, that is, $x_1, \ldots, x_t$, and entries of each vector by an additional subscript, e.g., $x_{t,i}$. We use the notation $g_{1:t}$ as a shorthand for $\sum_{s=1}^{t} g_s$. Similarly we write $m_{1:t}$ for a sum of the first moment m_t, and $f_{1:t}$ to denote the function $f_{1:t}(x) = \sum_{s=1}^{t} f_s(x)$. Let $M_t = [m_1 \cdots m_t]$ denote the matrix obtained by concatenating the vector sequence $\{m_t\}_{t\geq 1}$ and $M_{t,i}$ denote the i-th row of this matrix which amounts to the concatenation of the i-th component of each vector. The notation $A \succeq 0$ (resp. $A \succ 0$) for a matrix A means that A is symmetric and positive semidefinite (resp. definite). Similarly, the notations $A \succeq B$ and $A \succ B$ mean that $A - B \succeq 0$ and $A - B \succ 0$ respectively, and both tacitly assume that A and B are symmetric. Given $A \succeq 0$, we write $A^{\frac{1}{2}}$ for the square root of A, the unique $X \succeq 0$ such that $XX = A$ ([12], Section 1.4).

Let $\mathcal{E}$ be a finite-dimension real vector space, endowed with the Mahalanobis norm $\|\cdot\|_A$ which is denoted by $\|\cdot\|_A = \sqrt{\langle \cdot, A\cdot \rangle}$ as induced by $A \succ 0$. Let $\mathcal{E}^*$ be the vector space of all linear functions on $\mathcal{E}$. The dual space $\mathcal{E}^*$ is endowed with the dual norm $\|\cdot\|_A^* = \sqrt{\langle \cdot, A^{-1}\cdot \rangle}$.

[1] The codes will be released if the paper is accepted.

Let $\mathcal{Q}$ be a closed convex set in $\mathcal{E}$. A continuous function $h(x)$ is called *strongly convex* on $\mathcal{Q}$ with norm $\|\cdot\|_H$ if $\mathcal{Q} \subseteq \operatorname{dom} h$ and there exists a constant $\sigma > 0$ such that for all $x, y \in \mathcal{Q}$ and $\alpha \in [0, 1]$ we have

$$h(\alpha x + (1-\alpha)y) \leq \alpha h(x) + (1-\alpha)h(y) - \frac{1}{2}\sigma\alpha(1-\alpha)\|x-y\|_H^2.$$

The constant σ is called the *convexity parameter* of $h(x)$, or the *modulus* of strong convexity. We also denote by $\|\cdot\|_h = \|\cdot\|_H$. Further, if h is differentiable, we have

$$h(y) \geq h(x) + \langle \nabla h(x), y - x\rangle + \frac{\sigma}{2}\|x-y\|_h^2.$$

We use online convex optimization as our analysis framework. On each round $t = 1, \ldots, T$, a convex loss function $f_t : \mathcal{Q} \mapsto \mathbb{R}$ is chosen, and we pick a point $x_t \in \mathcal{Q}$ hence get loss $f_t(x_t)$. Our goal is minimizing the *regret* which is defined as the quantity

$$\mathcal{R}_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x\in\mathcal{Q}} \sum_{t=1}^{T} f_t(x). \tag{7}$$

Online convex optimization can be seen as a generalization of stochastic convex optimization. Any regret minimizing algorithm can be converted to a stochastic optimization algorithm with convergence rate $O(\mathcal{R}_T/T)$ using an online-to-batch conversion technique [9].

In this paper, we assume $\mathcal{Q} \equiv \mathcal{E} = \mathbb{R}^n$, hence we have $\mathcal{E}^* = \mathbb{R}^n$. We write $s^T x$ or $s \cdot x$ for the standard inner product between $s, x \in \mathbb{R}^n$. For the standard Euclidean norm, $\|x\| = \|x\|_2 = \sqrt{\langle x, x\rangle}$ and $\|s\|_* = \|s\|_2$. We also use $\|x\|_1 = \sum_{i=1}^n |x^{(i)}|$ and $\|x\|_\infty = \max_i |x^{(i)}|$ to denote ℓ_1-norm and ℓ_∞-norm respectively, where $x^{(i)}$ is the i-th element of x.

2 Algorithm

2.1 Closed-form Solution

We will derive the closed-form solution of Eq. (6) with specific A_t and Algorithm 1 with slight modification in this section. We have the following theorem.

Theorem 1. *Given* $A_t = (\sum_{s=1}^{t} \frac{Q_s^g}{2\alpha_s} + \lambda_2 \mathbb{I})$ *of Eq.* (5)*,* $z_t = z_{t-1} + m_t - \frac{Q_t}{\alpha_t} x_t$ *at each iteration* $t = 1, \ldots, T$ *and* $z_0 = \mathbf{0}$*, the optimal solution of Eq.* (6) *is updated accordingly as follows:*

$$x_{t+1} = (\sum_{s=1}^{t} \frac{Q_s}{\alpha_s} + 2\lambda_2 \mathbb{I})^{-1} \max(1 - \frac{\sqrt{d_{x_t^g}}\lambda_{21}}{\|\tilde{s}_t\|_2}, 0) s_t \tag{8}$$

where the i*-th element of* s_t *is defined as*

$$s_{t,i} = \begin{cases} 0 & \text{if } |z_{t,i}| \leq \lambda_1, \\ \operatorname{sign}(z_{t,i})\lambda_1 - z_{t,i} & \text{otherwise,} \end{cases} \tag{9}$$

$\tilde{s}_t$ is defined as

$$\tilde{s}_t = (\sum_{s=1}^{t} \frac{Q_s}{2\alpha_s} + \lambda_2 \mathbb{I})^{-1} s_t \tag{10}$$

and $\sum_{s=1}^{t} \frac{Q_s}{\alpha_s}$ is the diagonal and positive definite matrix.

The proof of Theorem 1 is given in Appendix A. Here $\tilde{s}_t$ can be considered as the weighted average of s_t. We slightly modify (8) where we replace $\tilde{s}_t$ with s_t in practical algorithms. Our purpose is that the ℓ_{21}-regularization does not depend on the second moment terms and other hyperparameters such as α_s and λ_2. The empirical experiment will also show that the algorithm using s_t can improve accuracy over using $\tilde{s}_t$ in the same level of sparsity in Section 4.4. Therefore, we get Algorithm 1. Furthermore, we have the following theorem which shows the relationship between Algorithm 1 and adaptive optimization methods. The proof is given in Appendix B.

Algorithm 1 Generic framework of adaptive optimization methods with sparse group lasso

Input: parameters $\lambda_1, \lambda_{21}, \lambda_2$
$x_1 \in \mathbb{R}^n$, step size $\{\alpha_t > 0\}_{t=0}^{T}$, sequence of functions $\{\phi_t, \psi_t\}_{t=1}^{T}$, initialize $z_0 = \mathbf{0}, V_0 = \mathbf{0}$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(x_t)$
  $m_t = \phi_t(g_1, \ldots, g_t)$ and $V_t = \psi_t(g_1, \ldots, g_t)$
  $\frac{Q_t}{\alpha_t} = \frac{\sqrt{V_t}}{\alpha_t} - \frac{\sqrt{V_{t-1}}}{\alpha_{t-1}}$
  $z_t \leftarrow z_{t-1} + m_t - \frac{Q_t}{\alpha_t} x_t$
  for $i \in \{1, \ldots, n\}$ **do**
    $s_{t,i} = \begin{cases} 0 & \text{if } |z_{t,i}| \le \lambda_1 \\ \text{sign}(z_{t,i})\lambda_1 - z_{t,i} & \text{otherwise.} \end{cases}$
  end for
  $x_{t+1} = (\frac{\sqrt{V_t}}{\alpha_t} + 2\lambda_2 \mathbb{I})^{-1} \max(1 - \frac{\sqrt{d_{x_t^g}}\lambda_{21}}{\|s_t\|_2}, 0) s_t$
end for

Theorem 2. *If regularization terms of Algorithm 1 vanish, Algorithm 1 is equivalent to Eq.* (1)*.*

2.2 Concrete Examples

Using Algorithm 1, we can easily derive the new optimizers based on Adam [8], Adagrad [4] which we call Group Adam, Group Adagrad respectively.

Group Adam The detail of the algorithm is given in Algorithm 2. From Theorem 2, we know that when λ_1, λ_2, λ_{21} are all zeros, Algorithm 2 is equivalent to Adam [8].

Algorithm 2 Group Adam

Input: parameters $\lambda_1, \lambda_{21}, \lambda_2, \beta_1, \beta_2, \epsilon$
$x_1 \in \mathbb{R}^n$, step size α, initialize $z_0 = \mathbf{0}, \hat{m}_0 = \mathbf{0}, \hat{V}_0 = \mathbf{0}, V_0 = \mathbf{0}$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(x_t)$
  $\hat{m}_t \leftarrow \beta_1 \hat{m}_{t-1} + (1-\beta_1) g_t$
  $m_t = \hat{m}_t / (1-\beta_1^t)$
  $\hat{V}_t \leftarrow \beta_2 \hat{V}_{t-1} + (1-\beta_2)\mathrm{diag}(g_t^2)$
  $V_t = \hat{V}_t / (1-\beta_2^t)$
  $Q_t = \begin{cases} \sqrt{V_t} - \sqrt{V_{t-1}} + \epsilon \mathbb{I} & t = 1 \\ \sqrt{V_t} - \sqrt{V_{t-1}} & t > 1 \end{cases}$
  $z_t \leftarrow z_{t-1} + m_t - \frac{1}{\alpha} Q_t x_t$
  for $i \in \{1, \ldots, n\}$ **do**
    $s_{t,i} = -\mathrm{sign}(z_{t,i}) \max(|z_{t,i}| - \lambda_1, 0)$
  end for
  $x_{t+1} = (\frac{\sqrt{V_t} + \epsilon \mathbb{I}}{\alpha} + 2\lambda_2 \mathbb{I})^{-1} \max(1 - \frac{\sqrt{d_{x_t^g}} \lambda_{21}}{\|s_t\|_2}, 0) s_t$
end for

Algorithm 3 Group Adagrad

Input: parameters $\lambda_1, \lambda_{21}, \lambda_2, \epsilon$
$x_1 \in \mathbb{R}^n$, step size α, initialize $z_0 = \mathbf{0}, V_0 = \mathbf{0}$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(x_t)$
  $m_t = g_t$
  $V_t = \begin{cases} V_{t-1} + \mathrm{diag}(g_t^2) + \epsilon \mathbb{I} & t = 1 \\ V_{t-1} + \mathrm{diag}(g_t^2) & t > 1 \end{cases}$
  $Q_t = \sqrt{V_t} - \sqrt{V_{t-1}}$
  $z_t \leftarrow z_{t-1} + m_t - \frac{1}{\alpha} Q_t x_t$
  for $i \in \{1, \ldots, n\}$ **do**
    $s_{t,i} = -\mathrm{sign}(z_{t,i}) \max(|z_{t,i}| - \lambda_1, 0)$
  end for
  $x_{t+1} = (\frac{\sqrt{V_t}}{\alpha} + 2\lambda_2 \mathbb{I})^{-1} \max(1 - \frac{\sqrt{d_{x_t^g}} \lambda_{21}}{\|s_t\|_2}, 0) s_t$
end for

Group Adagrad The detail of the algorithm is given in Algorithm 3. Similarly, from Theorem 2, when λ_1, λ_2, λ_{21} are all zeros, Algorithm 3 is equivalent to ADAGRAD [4]. Furthermore, we can find that when $\lambda_{21} = 0$, Algorithm 3 is equivalent to FTRL [11]. Therefore, GROUP ADAGRAD can also be called GROUP FTRL from the research of [16].

Similarly, GROUP MOMENTUM, GROUP AMSGRAD, GROUP ADAHESSIAN, etc., can be derived from MOMENTUM [17], AMSGRAD [19], ADAHESSIAN [26], etc., with the same framework and we will not list the details.

3 Convergence and Regret Analysis

Using the framework developed in [15,24,4], we have the following theorem providing the bound of the regret.

Theorem 3. *Let the sequence* $\{x_t\}$ *be defined by the update* (6) *and*

$$x_1 = \arg\min_{x \in \mathcal{Q}} \frac{1}{2} \|x - c\|_2^2, \tag{11}$$

where c *is an arbitrary constant vector. Suppose* $f_t(x)$ *is convex for any* $t \geq 1$ *and there exists an optimal solution* x^* *of* $\sum_{t=1}^T f_t(x)$*, i.e.,* $x^* = \arg\min_{x \in \mathcal{Q}} \sum_{t=1}^T f_t(x)$*,*

which satisfies the condition

$$\langle m_{t-1}, x_t - x^* \rangle \geq 0, \quad t \in [T], \tag{12}$$

where m_t is the weighted average of the gradient $f_t(x_t)$ and $[T] = \{1, \ldots, T\}$ for simplicity. Without loss of generality, we assume

$$m_t = \gamma m_{t-1} + g_t, \tag{13}$$

where $\gamma < 1$ and $m_0 = 0$. Then

$$\mathcal{R}_T \leq \Psi_T(x^*) + \sum_{t=1}^{T} \frac{1}{2\alpha_t} \|Q_t^{\frac{1}{2}}(x^* - x_t)\|_2^2 + \frac{1}{2} \sum_{t=1}^{T} \|m_t\|_{h_{t-1}^*}^2, \tag{14}$$

where $\|\cdot\|_{h_t^*}$ is the dual norm of $\|\cdot\|_{h_t}$. h_t is 1-strongly convex with respect to $\|\cdot\|_{\sqrt{V_t}/\alpha_t}$ for $t \in [T]$ and h_0 is 1-strongly convex with respect to $\|\cdot\|_2$.

The proof of Theorem 3 is given in Appendix C. Since in most of adaptive optimizers, V_t is the weighted average of $\text{diag}(g_t^2)$, without loss of generality, we assume $\alpha_t = \alpha$ and

$$V_t = \eta V_{t-1} + \text{diag}(g_t^2), \quad t \geq 1, \tag{15}$$

where $V_0 = 0$ and $\eta \leq 1$. Hence, we have the following lemma whose proof is given in Appendix D.1.

Lemma 1. *Suppose V_t is the weighted average of the square of the gradient which is defined by (15), $\alpha_t = \alpha$, m_t is defined by (13) and one of the following conditions:*

1. *$\eta = 1$,*
2. *$\eta < 1$, $\eta \geq \gamma$ and $\kappa V_t \succeq V_{t-1}$ for all $t \geq 1$ where $\kappa < 1$.*

is satisfied. Then we have

$$\sum_{t=1}^{T} \|m_t\|_{(\frac{\sqrt{V_t}}{\alpha_t})^{-1}}^2 < \frac{2\alpha}{1-\nu} \sum_{i=1}^{d} \|M_{T,i}\|_2, \tag{16}$$

where $\nu = \max(\gamma, \kappa)$ and d is the dimension of x_t.

We can always add $\delta^2 \mathbb{I}$ to V_t at each step to ensure $V_t \succ 0$. Therefore, $h_t(x)$ is 1-strongly convex with respect to $\|\cdot\|_{\sqrt{\delta^2 \mathbb{I} + V_t}/\alpha_t}$. Let $\delta \geq \max_{t \in [T]} \|g_t\|_\infty$, for $t > 1$, we have

$$\begin{aligned} \|m_t\|_{h_{t-1}^*}^2 &= \left\langle m_t, \alpha_t (\delta^2 \mathbb{I} + V_{t-1})^{-\frac{1}{2}} m_t \right\rangle \leq \left\langle m_t, \alpha_t \left(\text{diag}(g_t^2) + \eta V_{t-1}\right)^{-\frac{1}{2}} m_t \right\rangle \\ &= \left\langle m_t, \alpha_t V_t^{-\frac{1}{2}} m_t \right\rangle = \|m_t\|_{(\frac{\sqrt{V_t}}{\alpha_t})^{-1}}^2. \end{aligned} \tag{17}$$

For $t = 1$, we have

$$\begin{aligned}\|m_1\|^2_{h_0^*} &= \left\langle m_1, \alpha_1(\delta^2\mathbb{I} + \mathbb{I})^{-\frac{1}{2}} m_1 \right\rangle \leq \left\langle m_1, \alpha_1 \left(\mathrm{diag}^{-\frac{1}{2}}(g_1^2)\right) m_1 \right\rangle \\ &= \left\langle m_1, \alpha_1 V_1^{-\frac{1}{2}} m_1 \right\rangle = \|m_1\|^2_{(\frac{\sqrt{V_1}}{\alpha_1})^{-1}}.\end{aligned} \tag{18}$$

From (17), (18) and Lemma 1, we have

Lemma 2. *Suppose V_t, m_t, α_t, ν, d are defined the same as Lemma 1,* $\max_{t\in[T]} \|g_t\|_\infty \leq \delta$, $\|\cdot\|^2_{h_t^*} = \left\langle \cdot, \alpha_t(\delta^2\mathbb{I} + V_t)^{-\frac{1}{2}} \cdot \right\rangle$ *for* $t \geq 1$ *and* $\|\cdot\|^2_{h_0^*} = \left\langle \cdot, \alpha_1 \left((\delta^2+1)\mathbb{I}\right)^{-\frac{1}{2}} \cdot \right\rangle$. *Then*

$$\sum_{t=1}^{T} \|m_t\|^2_{h_{t-1}^*} < \frac{2\alpha}{1-\nu} \sum_{i=1}^{d} \|M_{T,i}\|_2. \tag{19}$$

Therefore, from Theorem 3 and Lemma 2, we have

Corollary 1. *Suppose V_t, m_t, α_t, h_t^*, ν, d are defined the same as Lemma 2, there exist constants G, D_1, D_2 such that* $\max_{t\in[T]} \|g_t\|_\infty \leq G \leq \delta$, $\|x^*\|_\infty \leq D_1$ *and* $\max_{t\in[T]} \|x_t - x^*\|_\infty \leq D_2$. *Then*

$$\mathcal{R}_T < dD_1 \left(\lambda_1 + \lambda_{21}(\frac{\sqrt{T}G}{2\alpha} + \lambda_2)^{\frac{1}{2}} + \lambda_2 D_1 \right) + dG \left(\frac{D_2^2}{2\alpha} + \frac{\alpha}{(1-\nu)^2} \right) \sqrt{T}. \tag{20}$$

The proof of Corollary 1 is given in D.2. Furthermore, from Corollary 1, we have

Corollary 2. *Suppose m_t is defined as* (13), $\alpha_t = \alpha$ *and satisfies the condition* (19). *There exist constants G, D_1, D_2 such that $tG^2\mathbb{I} \succeq V_t$,* $\max_{t\in[T]} \|g_t\|_\infty \leq G$, $\|x^*\|_\infty \leq D_1$ *and* $\max_{t\in[T]} \|x_t - x^*\|_\infty \leq D_2$. *Then*

$$\mathcal{R}_T < dD_1 \left(\lambda_1 + \lambda_{21}(\frac{\sqrt{T}G}{2\alpha} + \lambda_2)^{\frac{1}{2}} + \lambda_2 D_1 \right) + dG \left(\frac{D_2^2}{2\alpha} + \frac{\alpha}{(1-\nu)^2} \right) \sqrt{T}. \tag{21}$$

Therefore, we know that the regret of the update (6) is $O(\sqrt{T})$ and can achieve the optimal convergence rate $O(1/\sqrt{T})$ under the conditions of Corollary 1 or Corollary 2.

4 Experiments

4.1 Experiment Setup

We test the algorithms on three different large-scale real-world datasets with different neural network structures. These datasets are various display ads logs for the purpose of predicting ads CTR. The details are as follows.

a) The Avazu CTR dataset [2] contains approximately 40M samples and 22 categorical features over 10 days. In order to handle categorical data, we use the one-hot-encoding based embedding technique (see, e.g., [23], Section 2.1 or [13], Section 2.1.1) and get 9.4M features in total. For this dataset, the samples from the first 9 days (containing 8.7M one-hot features) are used for training, while the rest is for testing. Our DNN model follows the basic structure of most deep CTR models. Specifically, the model comprises one embedding layer, which maps each one-hot feature into 16-dimensional embeddings, and four fully connected layers (with output dimension of 64, 32, 16 and 1, respectively) in sequence.
b) The iPinYou dataset[2] [7] is another real-world dataset for ad click logs over 21 days. The dataset contains 16 categorical features[3]. After one-hot encoding, we get a dataset containing 19.5M instances with 1033.1K input dimensions. We keep the original train/test splitting scheme, where the training set contains 15.4M samples with 937.7K one-hot features. We use Outer Product-based Neural Network (OPNN) [18], and follow the standard settings of [18], i.e., one embedding layer with the embedding dimension of 10, one product layer and three hidden layers of size 512, 256, 128 respectively where we set dropout rate at 0.5.
c) The third dataset is the Criteo Display Ads dataset [3] which contains approximately 46M samples over 7 days. There are 13 integer features and 26 categorical features. After one-hot encoding of categorical features, we have a total of 33.8M features. We split the dataset into 7 partitions in chronological order and select the earliest 6 parts for training which contains 29.6M features and the rest for testing though the dataset has no timestamp. We use Deep & Cross Network (DCN) [23] and choose the following settings[4]: one embedding layer with embedding dimension 8, two deep layers of size 64 each, and two cross layers.

For the convenience of discussion, we use MLP, OPNN and DCN to represent the aforementioned three datasets coupled with their corresponding models. It is obvious that the embedding layer has most of parameters of the neural networks when the features have very high dimension, therefore we just add the regularization terms to the embedding layer. Furthermore, each embedding

[2] We only use the data from season 2 and 3 because of the same data schema.
[3] See `https://github.com/Atomu2014/Ads-RecSys-Datasets/` for details.
[4] Limited by training resources available, we don't use the optimal hyperparameter settings of [23].

vector is considered as a group, and a visual comparison between ℓ_1, ℓ_{21} and mixed regularization effect is given in Fig. 2 of [22].

We treat the training set as the streaming data, hence we train 1 epoch with a batch size of 512 and do the validation. The experiments are conducted with 4-9 workers and 2-3 parameter servers in the TensorFlow framework [1], which depends on the different sizes of the datasets. According to [5], area under the receiver-operator curve (AUC) is a good measurement in CTR estimation and AUC is widely adopted as the evaluation criterion in classification problems. Thus we choose AUC as our evaluation criterion. We explore 5 learning rates from 1e-5 to 1e-1 with increments of 10× and choose the one with the best AUC for each new optimizer in the case of no regularization terms (It is equivalent to the original optimizer according to Theorem 2). The details are listed in Table 5 of Appendix E. All the experiments are run 5 times repeatedly and tested statistical significance using t-test. Without loss of generality, we choose two new optimizers to validate the performance, which are Group Adam and Group AdaGrad.

4.2 Adam vs. Group Adam

First, we compare the performance of the two optimizers on the same sparsity level. We set $\lambda_1 = \lambda_2 = 0$ and choose different values of λ_{21} of Algorithm 2, i.e., Group Adam, and achieve the same sparsity with Adam that uses the magnitude pruning method. Since we should delete the entire embedding vector which the feature corresponds to, not a single weight, and the amount of the features will dynamically increase as the training goes on, our method is different from the commonly used method [28]. Concretely, our method works in three steps. The first step sorts the norm of embedding vector from largest to smallest, and keeps top N embedding vectors which depend on the sparsity when finishing the first phase of training. In the second step we fine-tune the model. Since some new or deleted features will appear in the model after training with new data, in the last step we need to prune the model again to ensure that the desired sparsity is reached. We use the schedule of keeping 0%, 10%, 20%, 30% training samples to fine tune, and choose the best one. The details are listed in Table 8 of Appendix E.

Table 2 reports the average results of the two optimizers in the three datasets. Note that Group Adam significantly outperforms Adam on the AUC metric in the same sparsity level for most experiments especially under extreme sparsity. (60% experiments show statistically significant with 90% confidence level, and 87.5% experiments show statistically significant with 90% confidence level when sparsity level is less than 5%). Furthermore, as shown in Figure 1, the same ℓ_{21}-regularization strength λ_{21} has different effects of sparsity and accuracy on different datasets. The best choice of λ_{21} depends on the dataset as well as the application (For example, if the memory of serving resource is limited, sparsity might be relatively more important). One can trade off accuracy to get more sparsity by increasing the value of λ_{21}.

Next, we compare the performance of Adam without post-processing procedure, i.e., no magnitude pruning, and Group Adam under extremely high

Table 2. AUC for the two optimizers and sparsity (feature rate) in parentheses. The best AUC for each dataset on each sparsity level is bolded. The p-value of the t-test of AUC is also listed.

λ_{21}	MLP			OPNN			DCN		
Group Adam	Adam	Group Adam	P-Value	Adam	Group Adam	P-Value	Adam	Group Adam	P-Value
1e-4	0.7457 (0.974)	**0.7461** (0.974)	0.470	0.7551 (0.078)	**0.7595** (0.078)	0.086	0.8018 (0.518)	**0.8022** (0.518)	0.105
5e-4	0.7464 (0.864)	**0.7468** (0.864)	0.466	0.7491 (0.039)	**0.7573** (0.039)	0.091	0.8017 (0.062)	**0.8019** (0.062)	0.487
1e-3	0.7452 (0.701)	**0.7468** (0.701)	0.058	0.7465 (0.032)	**0.7595** (0.032)	0.014	0.8017 (0.018)	0.8017 (0.018)	0.943
5e-3	0.7457 (0.132)	**0.7464** (0.132)	0.335	0.7509 (0.018)	**0.7561** (0.018)	0.041	0.7995 (4.2e-3)	**0.8007** (4.2e-3)	9.11e-3
1e-2	0.7444 (0.038)	**0.7466** (0.038)	0.014	0.7396 (9.2e-3)	**0.7493** (9.2e-3)	0.031	0.7972 (2.5e-3)	**0.7999** (2.5e-3)	5.97e-7

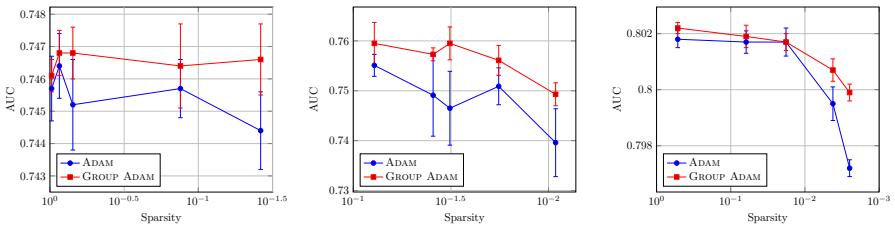

Fig. 1. AUC across different sparsity on two optimizers for the three datasets. MLP, OPNN and DCN are in left, middle, right column respectively. The x-axis is sparsity (number of non-zero features whose embedding vectors are not equal to **0** divided by the total number of features present in the training data). The y-axis is AUC. Error bars represent one standard deviation.

sparsity. We search regularization terms according to AUC and the values are listed in Table 6 of Appendix E. In general, good default settings of λ_2 is 1e-5. The results are shown in Table 3. Note that compared with Adam, Group Adam with appropriate regularization terms can achieve significantly better or highly competitive performance with producing extremely high sparsity.

4.3 Adagrad vs. Group Adagrad

We compare the performance of Adagrad without magnitude pruning and Group Adagrad under extremely high sparsity. The regularization terms we choose are listed in Table 7 of Appendix E. The results are also shown in Table 3. Again note that in comparison to Adagrad, Group Adagrad can not only achieve significantly better or highly competitive performance of AUC, but also effectively and efficiently reduce the dimensions of the features.

Table 3. AUC for three datasets and sparsity (feature rate) in parentheses. The best value for each dataset is bolded. The p-value of t-test is also listed.

Dataset	Adam	Group Adam	P-Value	Adagrad	Group Adagrad	P-Value
MLP	0.7458 (1.000)	**0.7486** (**0.018**)	1.10e-3 (2.69e-11)	0.7453 (1.000)	**0.7469** (**0.063**)	0.106 (1.51e-9)
OPNN	0.7588 (0.827)	**0.7617** (**0.130**)	0.289 (6.20e-11)	0.7556 (0.827)	**0.7595** (**0.016**)	0.026 (< 2.2e-16)
DCN	**0.8021** (1.000)	0.8019 (**0.030**)	0.422 (1.44e-11)	0.7975 (1.000)	**0.7978** (**0.040**)	0.198 (3.94e-11)

4.4 Discussion

In this section we will compare the performance of s_t with $\tilde{s}_t$ discussed in Section 2.1, i.e., using $\tilde{s}_t$ means that replacing $\|\tilde{s}_t\|$ with $\|s_t\|$ in line 10 of Algorithm 1. Furthermore, we will discuss the hyperparameters of ℓ_1-regularization, ℓ_{21}-regularization and emdedding dimension to show how these hyperparameters affect the effects of regularization. Without loss of generality, all experiments are conducted on DCN using Group Adam. The default settings of regularization terms are all zeros, unless otherwise specified.

s_t vs. $\tilde{s}_t$ We choose ℓ_{21}-regularization of s_t and $\tilde{s}_t$ from 10 points in different sparsity levels. The details are listed in Table 9 of Appendix E. As shown in Figure 2, the algorithm using s_t outperforms the one using $\tilde{s}_t$ in the same level of sparsity.

ℓ_1 vs. ℓ_{21} From lines 8 and 10 of Algorithm 1, we know that if z_t has the same elements, the values of ℓ_1 and ℓ_{21}, i.e., λ_1 and λ_{21}, have the same regularization effects. However, this situation almost cannot happen in reality. We compare the regularization performance with the same values of λ_1 and λ_{21}. The results are shown in Figure 3. It is obvious that ℓ_{21}-regularization is much more effective than ℓ_1-regularization in producing sparsity. Therefore, if we just need to produce a sparse model, tuning λ_{21} while keeping $\lambda_1 = 0$ is usually a simple but effective choice.

Embedding Dimension Table 4 reports the average results of different embedding dimensions, whose regularization terms are same to DCN of Table 6 of Appendix E. Note that the sparsity increases with the growth of the embedding dimension. The reason is that the square root of the embedding dimension is the multiplier of ℓ_{21}-regularization.

5 Conclusion

In this paper, we propose a novel framework that adds the regularization terms to a family of adaptive optimizers for producing sparsity of DNN models. We

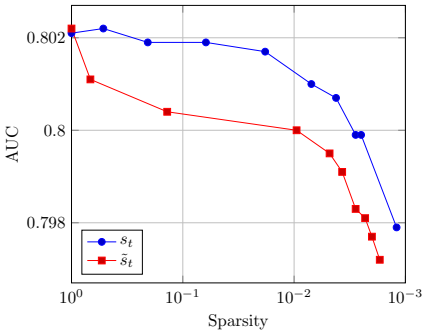

Fig. 2. AUC across different sparsity (feature rate) on two methods. The legend is the algorithms using s_t and $\tilde{s}_t$. The x-axis is sparsity. The y-axis is AUC.

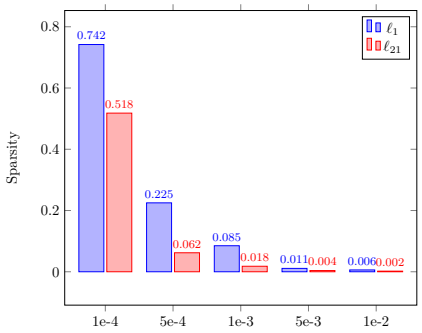

Fig. 3. The sparsity (feature rate) across different values of regularized terms. The legend is the regularized terms. The x-axis is the values of regularized terms. The y-axis is sparsity.

Table 4. The sparsity (feature rate) for different embedding dimensions and AUC in parentheses. The best results are bolded.

Embedding Dimension	Group Adam
4	0.074 (0.8008)
8	0.030 (0.8019)
16	0.012 (**0.8020**)
32	**0.008** (0.8011)

apply this framework to create a new class of optimizers. We provide closed-form solutions and algorithms with slight modification. We built the relation between new and original optimizers, i.e., our new optimizers become equivalent with the corresponding original ones, once the regularization terms vanish. We theoretically prove the convergence rate of the regret and also conduct empirical evaluation on the proposed optimizers in comparison to the original optimizers with and without magnitude pruning. The results clearly demonstrate the advantages of our proposed optimizers in both getting significantly better performance and producing sparsity. Finally, it would be interesting in the future to investigate the convergence in non-convex settings and evaluate our optimizers on more applications from fields such as compute vision, natural language processing and etc.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P.A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y.,

Zheng, X.: Tensorflow: A system for large-scale machine learning. In: Keeton, K., Roscoe, T. (eds.) 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016. pp. 265–283. USENIX Association (2016), https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

2. Avazu: Avazu click-through rate prediction (2015), https://www.kaggle.com/c/avazu-ctr-prediction/data
3. Criteo: Criteo display ad challenge (2014), http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset
4. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**, 2121–2159 (2011). https://doi.org/10.5555/1953048.2021068, https://dl.acm.org/doi/10.5555/1953048.2021068
5. Graepel, T., Candela, J.Q., Borchert, T., Herbrich, R.: Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 13–20. Omnipress (2010), https://icml.cc/Conferences/2010/papers/901.pdf
6. Gupta, V., Koren, T., Singer, Y.: Shampoo: Preconditioned stochastic tensor optimization. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 1837–1845. PMLR (2018), http://proceedings.mlr.press/v80/gupta18a.html
7. iPinYou: ipinyou global rtb bidding algorithm competition (2013), https://www.kaggle.com/lastsummer/ipinyou
8. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations. ICLR '15, San Diego, CA, USA (2015)
9. Littlestone, N.: From on-line to batch learning. In: Rivest, R.L., Haussler, D., Warmuth, M.K. (eds.) Proceedings of the Second Annual Workshop on Computational Learning Theory, COLT 1989, Santa Cruz, CA, USA, July 31 - August 2, 1989. pp. 269–284. Morgan Kaufmann (1989), http://dl.acm.org/citation.cfm?id=93365
10. McMahan, H.B.: Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. AISTATS '11, vol. 15, pp. 525–533. PMLR, Fort Lauderdale, FL, USA (2011)
11. McMahan, H.B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., Chikkerur, S., Liu, D., Wattenberg, M., Hrafnkelsson, A.M., Boulos, T., Kubica, J.: Ad click prediction: a view from the trenches. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1222–1230. KDD '13, ACM, Chicago, Illinois, USA (2013)
12. McMahan, H.B., Streeter, M.J.: Adaptive bound optimization for online convex optimization. In: COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010. pp. 244–256. Omnipress (2010), http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf#page=252
13. Naumov, M., Mudigere, D., Shi, H.M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C., Azzolini, A.G., Dzhulgakov, D., Mallevich, A., Cherniavskii, I., Lu, Y., Krishnamoorthi, R., Yu, A., Kondratenko, V., Pereira, S., Chen, X., Chen, W., Rao, V., Jia, B., Xiong, L., Smelyanskiy, M.: Deep learning recommendation

model for personalization and recommendation systems. CoRR **abs/1906.00091** (2019), http://arxiv.org/abs/1906.00091
14. Nesterov, Y.E.: Smooth minimization of non-smooth functions. Math. Program. **103**, 127–152 (2005)
15. Nesterov, Y.E.: Primal-dual subgradient methods for convex problems. Math. Program. **120**(1), 221–259 (2009). https://doi.org/10.1007/s10107-007-0149-x, https://doi.org/10.1007/s10107-007-0149-x
16. Ni, X., Yu, Y., Wu, P., Li, Y., Nie, S., Que, Q., Chen, C.: Feature selection for facebook feed ranking system via a group-sparsity-regularized training algorithm. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 2085–2088. CIKM '19, ACM, Beijing, China (2019)
17. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics **4**(5), 1–17 (1964). https://doi.org/10.1016/0041-5553(64)90137-5
18. Qu, Y., Cai, H., Ren, K., Zhang, W., Yu, Y., Wen, Y., Wang, J.: Product-based neural networks for user response prediction. In: Bonchi, F., Domingo-Ferrer, J., Baeza-Yates, R., Zhou, Z., Wu, X. (eds.) IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain. pp. 1149–1154. IEEE Computer Society (2016). https://doi.org/10.1109/ICDM.2016.0151, https://doi.org/10.1109/ICDM.2016.0151
19. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: Proceedings of the 6th International Conference on Learning Representations. ICLR '18, OpenReview.net, Vancouver, BC, Canada (2018)
20. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
21. Rockafellar, R.T.: Convex Analysis. Princeton Landmarks in Mathematics and Physics, Princeton University Press (1970)
22. Scardapane, S., Comminiello, D., Hussain, A., Uncini, A.: Group sparse regularization for deep neural networks. Neurocomputing **241**, 43–52 (2016). https://doi.org/10.1016/j.neucom.2017.02.029, https://doi.org/10.1016/j.neucom.2017.02.029
23. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: Proceedings of the ADKDD'17, Halifax, NS, Canada, August 13 - 17, 2017. pp. 12:1–12:7. ACM (2017). https://doi.org/10.1145/3124749.3124754, https://doi.org/10.1145/3124749.3124754
24. Xiao, L.: Dual averaging method for regularized stochastic learning and online optimization. Journal of Machine Learning Research **11**, 2543–2596 (2010). https://doi.org/10.5555/1756006.1953017, https://dl.acm.org/doi/10.5555/1756006.1953017
25. Yang, H., Xu, Z., King, I., Lyu, M.R.: Online learning for group lasso. In: Proceedings of the 27th International Conference on Machine Learning. pp. 1191–1198. ICML '10, Omnipress, Haifa, Israel (2010)
26. Yao, Z., Gholami, A., Shen, S., Keutzer, K., Mahoney, M.W.: ADAHESSIAN: an adaptive second order optimizer for machine learning. CoRR **abs/2006.00719** (2020), https://arxiv.org/abs/2006.00719
27. Zeiler, M.D.: Adadelta: An adaptive learning rate method. CoRR **abs/1212.5701** (2012), https://arxiv.org/abs/1212.5701
28. Zhu, M., Gupta, S.: To prune, or not to prune: Exploring the efficacy of pruning for model compression. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings. OpenReview.net (2018), https://openreview.net/forum?id=Sy1iIDkPM