Variants of RMSProp and Adagrad with Logarithmic Regret Bounds

Mahesh Chandra Mukkamala [1 2] **Matthias Hein** [1]

Abstract

Adaptive gradient methods have become recently very popular, in particular as they have been shown to be useful in the training of deep neural networks. In this paper we have analyzed RMSProp, originally proposed for the training of deep neural networks, in the context of online convex optimization and show $\sqrt{T}$-type regret bounds. Moreover, we propose two variants SC-Adagrad and SC-RMSProp for which we show logarithmic regret bounds for strongly convex functions. Finally, we demonstrate in the experiments that these new variants outperform other adaptive gradient techniques or stochastic gradient descent in the optimization of strongly convex functions as well as in training of deep neural networks.

1. Introduction

There has recently been a lot of work on adaptive gradient algorithms such as Adagrad (Duchi et al., 2011), RMSProp (Hinton et al., 2012), ADADELTA (Zeiler, 2012), and Adam (Kingma & Bai, 2015). The original idea of Adagrad to have a parameter specific learning rate by analyzing the gradients observed during the optimization turned out to be useful not only in online convex optimization but also for training deep neural networks. The original analysis of Adagrad (Duchi et al., 2011) was limited to the case of all convex functions for which it obtained a data-dependent regret bound of order $O(\sqrt{T})$ which is known to be optimal (Hazan, 2016) for this class. However, a lot of learning problems have more structure in the sense that one optimizes over the restricted class of strongly convex functions. It has been shown in (Hazan et al., 2007) that one can achieve much better logarithmic regret bounds for the class of strongly convex functions.

[1]Department of Mathematics and Computer Science, Saarland University, Germany [2]IMPRS-CS, Max Planck Institute for Informatics, Saarbrücken, Germany . Correspondence to: Mahesh Chandra Mukkamala <mmahesh.chandra873@gmail.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

The goal of this paper is twofold. First, we propose SC-Adagrad which is a variant of Adagrad adapted to the strongly convex case. We show that SC-Adagrad achieves a logarithmic regret bound for the case of strongly convex functions, which is data-dependent. It is known that such bounds can be much better in practice than data independent bounds (Hazan et al., 2007),(McMahan, 2014). Second, we analyze RMSProp which has become one of the standard methods to train neural networks beyond stochastic gradient descent. We show that under some conditions on the weighting scheme of RMSProp, this algorithm achieves a data-dependent $O(\sqrt{T})$ regret bound. In fact, it turns out that RMSProp contains Adagrad as a special case for a particular choice of the weighting scheme. Up to our knowledge this is the first theoretical result justifying the usage of RMSProp in online convex optimization and thus can at least be seen as theoretical support for its usage in deep learning. Similarly, we then propose the variant SC-RMSProp for which we also show a data-dependent logarithmic regret bound similar to SC-Adagrad for the class of strongly convex functions. Interestingly, SC-Adagrad has been discussed in (Ruder, 2016), where it is said that "it does not to work". The reason for this is that SC-Adagrad comes along with a damping factor which prevents potentially large steps in the beginning of the iterations. However, as our analysis shows this damping factor has to be rather large initially to prevent large steps and should be then monotonically decreasing as a function of the iterations in order to stay adaptive. Finally, we show in experiments on three datasets that the new methods are competitive or outperform other adaptive gradient techniques as well as stochastic gradient descent for strongly convex optimization problem in terms of regret and training objective but also perform very well in the training of deep neural networks, where we show results for different networks and datasets.

2. Problem Statement

We first need some technical statements and notation and then introduce the online convex optimization problem.

2.1. Notation and Technical Statements

We denote by $[T]$ the set $\{1, \dots, T\}$. Let $A \in \mathbb{R}^{d\times d}$ be a symmetric, positive definite matrix. We denote as

$$\langle x, y\rangle_A = \langle x, Ay\rangle = \sum_{i,j=1}^{d} A_{ij}x_iy_j, \quad \|x\|_A \quad = \sqrt{\langle x, x\rangle_A}$$

Note that the standard Euclidean inner product becomes $\langle x, y\rangle = \sum_i x_iy_i = \langle x, y\rangle_{\mathbb{I}}$ While we use here the general notation for matrices for comparison to the literature. All positive definite matrices A in this paper will be diagonal matrices, so that the computational effort for computing inner products and norms is still linear in d. The Cauchy-Schwarz inequality becomes, $\langle x, y\rangle_A \leq \|x\|_A \|y\|_A$. We further introduce the element-wise product $a\odot b$ of two vectors. Let $a, b \in \mathbb{R}^d$, then $(a \odot b)_i = a_ib_i$ for $i = 1, \dots, d$.

Let $A \in \mathbb{R}^{d\times d}$ be a symmetric, positive definite matrix, $z \in \mathbb{R}^d$ and $C \subset \mathbb{R}^d$ a convex set. Then we define the **weighted projection** $P_C^A(z)$ of z onto the set C as

$$P_C^A(z) = \arg\min_{x\in C} \|x - z\|_A^2 . \tag{1}$$

It is well-known that the weighted projection is unique and non-expansive.

Lemma 2.1 *Let $A \in \mathbb{R}^{d\times d}$ be a symmetric, positive definite matrix and $C \subset \mathbb{R}^d$ be a convex set. Then*

$$\left\|P_C^A(z) - P_C^A(y)\right\|_A \leq \|z - y\|_A .$$

Lemma 2.2 *For any symmetric, positive semi-definite matrix $A \in \mathbb{R}^{d\times d}$ we have*

$$\langle x, Ax\rangle \leq \lambda_{max}(A)\langle x, x\rangle \leq \mathrm{tr}(A)\langle x, x\rangle \tag{2}$$

where $\lambda_{max}(A)$ is the maximum eigenvalue of matrix A and $\mathrm{tr}(A)$ denotes the trace of matrix A.

2.2. Problem Statement

In this paper we analyze the online convex optimization setting, that is we have a convex set C and at each round we get access to a (sub)-gradient of some continuous convex function $f_t : C \to \mathbb{R}$. At the t-th iterate we predict $\theta_t \in C$ and suffer a loss $f_t(\theta_t)$. The goal is to perform well with respect to the optimal decision in hindsight defined as

$$\theta^* = \arg\min_{\theta\in C} \sum_{t=1}^{T} f_t(\theta).$$

The adversarial regret at time $T \in \mathbb{N}$ is then given as

$$R(T) = \sum_{t=1}^{T} (f_t(\theta_t) - f_t(\theta^*)).$$

We assume that the adversarial can choose from the class of convex functions on C, for some parts we will specialize this to the set of strongly convex functions.

Definition 2.1 *Let C be a convex set. We say that a function $f : C \to \mathbb{R}$ is μ-strongly convex, if there exists $\mu \in \mathbb{R}^d$ with $\mu_i > 0$ for $i = 1, \dots, d$ such that for all $x, y \in C$,*

$$\begin{aligned} f(y) \;&\geq\; f(x) + \langle \nabla f(x), y - x\rangle + \|y - x\|^2_{\mathrm{diag}(\mu)} \\ &= f(x) + \langle \nabla f(x), y - x\rangle + \sum_{i=1}^{d} \mu_i(y_i - x_i)^2. \end{aligned}$$

Let $\zeta = \min_{i=1,\dots,d} \mu_i$, then this function is ζ-strongly convex (in the usual sense), that is

$$f(y) \;\geq\; f(x) + \langle \nabla f(x), y - x\rangle + \zeta \|x - y\|^2 .$$

Note that the difference between our notion of componentwise strong convexity and the usual definition of strong convexity is indicated by the bold font versus normal font. We have two assumptions:

- **A1:** It holds $\sup_{t\geq 1} \|g_t\|_2 \leq G$ which implies the existence of a constant G_∞ such that $\sup_{t\geq 1} \|g_t\|_\infty \leq G_\infty$.
- **A2:** It holds $\sup_{t\geq 1} \|\theta_t - \theta^*\|_2 \leq D$ which implies the existence of a constant D_∞ such that $\sup_{t\geq 1} \|\theta_t - \theta^*\|_\infty \leq D_\infty$.

One of the first methods which achieves the optimal regret bound of $O(\sqrt{T})$ for convex problems is online projected gradient descent (Zinkevich, 2003), defined as

$$\theta_{t+1} = P_C(\theta_t - \alpha_t g_t) \tag{3}$$

where $\alpha_t = \frac{\alpha}{\sqrt{t}}$ is the step-size scheme and g_t is a (sub)-gradient of f_t at θ_t. With $\alpha_t = \frac{\alpha}{t}$, online projected gradient descent method achieves the optimal $O(\log(T))$ regret bound for strongly-convex problems (Hazan et al., 2007). We consider Adagrad in the next subsection which is one of the popular adaptive alternative to online projected gradient descent.

2.3. Adagrad for convex problems

In this section we briefly recall the main result for the Adagrad. The algorithm for Adagrad is given in Algorithm 1. If the adversarial is allowed to choose from the set of all possible convex functions on $C \subset \mathbb{R}^d$, then Adagrad achieves the regret bound of order $O(\sqrt{T})$ as shown in (Duchi et al., 2011). This regret bound is known to be optimal for this class, see e.g. (Hazan, 2016). For better comparison to our results for RMSProp, we recall the result from (Duchi

Algorithm 1 Adagrad

Input: $\theta_1 \in C\,, \delta > 0, v_0 = 0 \in \mathbb{R}^d$
for $t = 1$ **to** T **do**
  $g_t \in \partial f_t(\theta_t)$
  $v_t = v_{t-1} + (g_t \odot g_t)$
  $A_t = \text{diag}(\sqrt{v_t}) + \delta\mathbb{I}$
  $\theta_{t+1} = P_C^{A_t}\left(\theta_t - \alpha A_t^{-1} g_t\right)$
end for

et al., 2011) in our notation. For this purpose, we introduce the notation, $g_{1:T,i} = (g_{1,i}, g_{2,i}, .., g_{T,i})^T$, where $g_{t,i}$ is the i-th component of the gradient $g_t \in \mathbb{R}^d$ of the function f_t evaluated at θ_t.

Theorem 2.1 *(Duchi et al., 2011) Let Assumptions **A1**, **A2** hold and let θ_t be the sequence generated by Adagrad in Algorithm 1, where $g_t \in \partial f_t(\theta_t)$ and $f_t : C \to \mathbb{R}$ is an arbitrary convex function, then for stepsize $\alpha > 0$ the regret is upper bounded as*

$$R(T) \leq \frac{D_\infty^2}{2\alpha} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \alpha \sum_{i=1}^{d} \|g_{1:T,i}\|_2 \,.$$

The effective step-length of Adagrad is on the order of $\frac{\alpha}{\sqrt{t}}$. This can be seen as follows; first note that $v_{T,i} = \sum_{t=1}^{T} g_{t,i}^2$ and thus $(A_t)^{-1}$ is a diagonal matrix with entries $\frac{1}{\sqrt{v_{t,i}}+\delta}$. Then one has

$$\begin{aligned} \alpha(A_T^{-1})_{ii} &= \frac{\alpha}{\sqrt{\sum_{t=1}^{T} g_{t,i}^2} + \delta} \\ &= \frac{\alpha}{\sqrt{T}} \frac{1}{\sqrt{\frac{1}{T}\sum_{t=1}^{T} g_{t,i}^2} + \frac{\delta}{\sqrt{T}}} \end{aligned} \tag{4}$$

Thus an alternative point of view of Adagrad, is that it has a decaying stepsize $\frac{\alpha}{\sqrt{t}}$ but now the correction term becomes the running average of the squared derivatives plus a vanishing damping term. However, the effective stepsize has to decay faster to get a logarithmic regret bound for the strongly convex case. This is what we analyze in the next section, where we propose SC-Adagrad for strongly convex functions.

3. Strongly convex Adagrad (SC-Adagrad)

The modification SC-Adagrad of Adagrad which we propose in the following can be motivated by the observation that the online projected gradient descent (Hazan et al., 2007) uses stepsizes of order $\alpha = O(\frac{1}{T})$ in order to achieve the logarithmic regret bound for strongly convex functions. In analogy with the derivation in the previous section, we still have $v_{T,i} = \sum_{t=1}^{T} g_{t,i}^2$. But now we modify $(A_t)^{-1}$ and set it as a diagonal matrix with entries $\frac{1}{v_{t,i}+\delta_t}$. Then one has

$$\alpha(A_T^{-1})_{ii} = \frac{\alpha}{\sum_{t=1}^{T} g_{t,i}^2 + \delta_t} = \frac{\alpha}{T} \frac{1}{\frac{1}{T}\sum_{t=1}^{T} g_{t,i}^2 + \frac{\delta_T}{T}}. \tag{5}$$

Again, we have in the denominator a running average of the observed gradients and a decaying damping factor. In this way, we get an effective stepsize of order $O(\frac{1}{T})$ in SC-Adagrad. The formal method is presented in Algorithm 2. As just derived the only difference of Adagrad and SC-Adagrad is the definition of the diagonal matrix A_t. Note

Algorithm 2 SC-Adagrad

Input: $\theta_1 \in C\,, \delta_0 > 0, v_0 = 0 \in \mathbb{R}^d$
for $t = 1$ **to** T **do**
  $g_t \in \partial f_t(\theta_t)$
  $v_t = v_{t-1} + (g_t \odot g_t)$
  Choose $0 < \delta_t \leq \delta_{t-1}$ element wise
  $A_t = \text{diag}(v_t) + \text{diag}(\delta_t)$
  $\theta_{t+1} = P_C^{A_t}\left(\theta_t - \alpha A_t^{-1} g_t\right)$
end for

also that we have defined the damping factor δ_t as a function of t which is also different from standard Adagrad. The constant δ in Adagrad is mainly introduced due to numerical reasons in order to avoid problems when $g_{t,i}$ is very small for some components in the first iterations and is typically chosen quite small e.g. $\delta = 10^{-8}$. For SC-Adagrad the situation is different. If the first components $g_{1,i}, g_{2,i}, \ldots$ are very small, say of order ϵ, then the update is $\frac{\epsilon}{\epsilon^2+\delta_t}$ which can become extremely large if δ_t is chosen to be small. This would make the method very unstable and would lead to huge constants in the bounds. This is probably why in (Ruder, 2016), the modification of Adagrad where one "drops the square-root" did not work. A good choice of δ_t should be initially roughly on the order of 1 and it should decay as $v_{t,i} = \sum_{t=1}^{T} g_{t,i}^2$ starts to grow. This is why we propose to use

$$\delta_{t,i} = \xi_2 e^{-\xi_1 v_{t,i}}, \quad i = 1, \ldots, d,$$

for $\xi_1 > 0$, $\xi_2 > 0$ as a potential decay scheme as it satisfies both properties for sufficiently large ξ_1 and ξ_2 chosen on the order of 1. Also, one can achieve a constant decay scheme for $\xi_1 = 0\,, \xi_2 > 0$. We will come back to this choice after the proof. In the following we provide the regret analysis of SC-Adagrad and show that the optimal logarithmic regret bound can be achieved. However, as it is data-dependent it is typically significantly better in practice than data-independent bounds.

3.1. Analysis

For any two matrices $A, B \in \mathbb{R}^{d\times d}$, we use the notation $\bullet$ to denote the inner product i.e $A \bullet B = \sum_i \sum_j A_{ij}B_{ij}$. Note that $A \bullet B = \mathrm{tr}(A^T B)$.

Lemma 3.1 *[Lemma 12 (Hazan et al., 2007)] Let A, B be positive definite matrices, let $A \succeq B \succ 0$ then*

$$A^{-1} \bullet (A - B) \leq \log\Big(\frac{|A|}{|B|}\Big) \tag{6}$$

where $|A|$ denotes the determinant of the matrix A

Lemma 3.2 *Let Assumptions **A1**, **A2** hold, then for $T \geq 1$ and A_t, δ_t as defined in the SC-Adagrad algorithm we have,*

$$\sum_{t=1}^{T} \langle g_t, A_t^{-1} g_t\rangle \leq \sum_{i=1}^{d} \log\left(\frac{\|g_{1:T,i}\|^2 + \delta_{T,i}}{\delta_{1,i}}\right) - \sum_{i=1}^{d}\sum_{t=2}^{T} \frac{\delta_{t,i} - \delta_{t-1,i}}{\|g_{1:t,i}\|^2 + \delta_{t,i}}$$

Theorem 3.1 *Let Assumptions **A1**, **A2** hold and let θ_t be the sequence generated by the SC-Adagrad in Algorithm 2, where $g_t \in \partial f_t(\theta_t)$ and $f_t : C \to \mathbb{R}$ is an arbitrary μ-strongly convex function ($\mu \in \mathbb{R}^d_+$) where the stepsize fulfills $\alpha \geq \max_{i=1,\ldots,d} \frac{G^2_\infty}{2\mu_i}$. Furthermore, let $\delta_t > 0$ and $\delta_{t,i} \leq \delta_{t-1,i} \forall t \in [T], \forall i \in [d]$, then the regret of SC-Adagrad can be upper bounded for $T \geq 1$ as*

$$R(T) \leq \frac{D^2_\infty \,\mathrm{tr}(\mathrm{diag}(\delta_1))}{2\alpha} + \frac{\alpha}{2}\sum_{i=1}^{d} \log\Big(\frac{\|g_{1:T,i}\|^2 + \delta_{T,i}}{\delta_{1,i}}\Big) + \frac{1}{2}\sum_{i=1}^{d} \inf_{t\in[T]} \Big(\frac{(\theta_{t,i} - \theta^*_i)^2}{\alpha} - \frac{\alpha}{\|g_{1:t,i}\|^2 + \delta_{t,i}}\Big)(\delta_{T,i} - \delta_{1,i})$$

For constant δ_t i.e $\delta_{t,i} = \delta > 0 \,\forall t \in [T]$ and $\forall i \in [d]$ then the regret of SC-Adagrad is upper bounded as

$$R(T) \leq \frac{D^2_\infty d\delta}{2\alpha} + \frac{\alpha}{2}\sum_{i=1}^{d} \log\Big(\frac{\|g_{1:T,i}\|^2 + \delta}{\delta}\Big) \tag{7}$$

For ζ-strongly convex function choosing $\alpha \geq \frac{G^2_\infty}{2\zeta}$ we obtain the above mentioned regret bounds.

Note that the first and the last term in the regret bound can be upper bounded by constants. Only the second term depends on T. Note that $\|g_{1:T,i}\|^2 \leq TG^2$ and as δ_t is monotonically decreasing, the second term is on the order of $O(\log(T))$ and thus we have a logarithmic regret bound. As the bound is data-dependent, in the sense that it depends on the observed sequence of gradients, it is much tighter than a data-independent bound.

The bound includes also the case of a non-decaying damping factor $\delta_t = \delta = \xi_2$ ($\xi_1 = 0$). While a rather large constant damping factor can work well, we have noticed that the best results are obtained with the decay scheme

$$\delta_{t,i} = \xi_2 e^{-\xi_1 v_{t,i}}, \quad i = 1, \ldots, d.$$

where $\xi_1 > 0$, $\xi_2 > 0$, which is what we use in the experiments. Note that this decay scheme for $\xi_1, \xi_2 > 0$ is adaptive to the specific dimension and thus increases the adaptivity of the overall algorithm. For completeness we also give the bound specialized for this decay scheme.

Corollary 3.1 *In the setting of Theorem 3.1 choose $\delta_{t,i} = \xi_2 e^{-\xi_1 v_{t,i}}$ for $i = 1, \ldots, d$ for some $\xi_1 > 0, \xi_2 > 0$. Then the regret of SC-Adagrad can be upper bounded for $T \geq 1$ as*

$$R(T) \leq \frac{dD^2_\infty \xi_2}{2\alpha} - \frac{\alpha}{2}\log(\xi_2 e^{-\xi_1 G^2_\infty}) + \frac{\alpha}{2}\sum_{i=1}^{d} \log\Big(\|g_{1:T,i}\|^2 + \xi_2\, e^{-\xi_1 \|g_{1:T,i}\|^2}\Big) + \frac{\alpha\xi_1\xi_2}{2\big(\log(\xi_2\,\xi_1) + 1\big)}\sum_{i=1}^{d}\Big(1 - e^{-\xi_1\|g_{1:T,i}\|^2}\Big)$$

Unfortunately, it is not obvious that the regret bound for our decaying damping factor is better than the one of a constant damping factor. Note, however that the third term in the regret bound of Theorem 3.1 can be negative. It thus remains an interesting question for future work, if there exists an optimal decay scheme which provably works better than any constant one.

4. RMSProp and SC-RMSProp

RMSProp is one of the most popular adaptive gradient algorithms used for the training of deep neural networks (Schaul et al., 2014; Dauphin et al., 2015; Daniel et al., 2016; Schmidhuber, 2015). It has been used frequently in computer vision (Karpathy & Fei-Fei, 2016) e.g. to train the latest InceptionV4 network (Szegedy et al., 2016a;b). Note that RMSProp outperformed other adaptive methods like Adagrad order Adadelta as well as SGD with momentum in a large number of tests in (Schaul et al., 2014). It has been argued that if the changes in the parameter update are approximately Gaussian distributed, then the matrix A_t can be seen as a preconditioner which approximates the diagonal of the Hessian (Daniel et al., 2016). However, it is fair to say that despite its huge empirical success in practice and some first analysis in the literature, there is so far no rigorous theoretical analysis of RMSProp. We will analyze RMSProp given in Algorithm 3 in the framework of of online convex optimization.

Algorithm 3 RMSProp

Input: $\theta_1 \in C\,, \delta > 0,\ \alpha > 0, v_0 = 0 \in \mathbb{R}^d$
for $t = 1$ **to** T **do**
  $g_t \in \partial f_t(\theta_t)$
  $v_t = \beta_t v_{t-1} + (1-\beta_t)(g_t \odot g_t)$
  Set $\epsilon_t = \frac{\delta}{\sqrt{t}}$ and $\alpha_t = \frac{\alpha}{\sqrt{t}}$
  $A_t = diag(\sqrt{v_t}) + \epsilon_t I$
  $\theta_{t+1} = P_C^{A_t}\big(\theta_t - \alpha_t A_t^{-1} g_t\big)$
end for

First, we will show that RMSProp reduces to Adagrad for a certain choice of its parameters. Second, we will prove for the general convex case a regret bound of $O(\sqrt{T})$ similar to the bound given in Theorem 2.1. It turns out that the convergence analysis requires that in the update of the weighted cumulative squared gradients (v_t), it has to hold

$$1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t},$$

for some $0 < \gamma \leq 1$. This is in contrast to the original suggestion of (Hinton et al., 2012) to choose $\beta_t = 0.9$. It will turn out later in the experiments that the constant choice of β_t leads sometimes to divergence of the sequence, whereas the choice derived from our theoretical analysis always leads to a convergent scheme even when applied to deep neural networks. Thus we think that the analysis in the following is not only interesting for the convex case but can give valuable hints how the parameters of RMSProp should be chosen in deep learning.

Before we start the regret analysis we want to discuss the sequence v_t in more detail. Using the recursive definition of v_t, we get the closed form expression

$$v_{t,i} = \sum_{j=1}^{t} (1-\beta_j) \prod_{k=j+1}^{t} \beta_k g_{j,i}^2.$$

With $\beta_t = 1 - \frac{1}{t}$ one gets, $v_{t,i} = \sum_{j=1}^{t} \frac{1}{j} \prod_{k=j+1}^{t} \frac{k-1}{k} g_{j,i}^2$, and using the telescoping product ones gets $\prod_{k=j+1}^{t} \frac{k-1}{k} = \frac{j}{t}$ and thus

$v_{t,i} = \frac{1}{t} \sum_{j=1}^{t} g_{j,i}^2$.

If one uses additionally the stepsize scheme $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\epsilon_t = \frac{\delta}{\sqrt{T}}$, then we recover the update scheme of Adagrad, see (4), as a particular case of RMSProp. We are not aware of that this correspondence of Adagrad and RMSProp has been observed before.

The proof of the regret bound for RMSProp relies on the following lemma.

Lemma 4.1 *Let Assumptions **A1** and **A2** and suppose that* $1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t}$ *for some* $0 < \gamma \leq 1$, *and* $t \geq 1$. *Also for* $t > 1$ *suppose* $\sqrt{(t-1)}\epsilon_{t-1} \leq \sqrt{t}\epsilon_t$, *then*

$$\sum_{t=1}^{T} \frac{g_{t,i}^2}{\sqrt{t\, v_{t,i}} + \sqrt{t}\epsilon_t} \leq \frac{2(2-\gamma)}{\gamma} \Big(\sqrt{T\, v_{T,i}} + \sqrt{T}\epsilon_T \Big).$$

Corollary 4.1 *Let Assumptions **A1**, **A2** hold and suppose that* $1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t}$ *for some* $0 < \gamma \leq 1$, *and* $t \geq 1$. *Also for* $t > 1$ *suppose* $\sqrt{(t-1)}\epsilon_{t-1} \leq \sqrt{t}\epsilon_t$, *and set* $\alpha_t = \frac{\alpha}{\sqrt{t}}$, *then*

$$\sum_{t=1}^{T} \frac{\alpha_t}{2} \left\langle g_t, A_t^{-1} g_t \right\rangle \leq \frac{\alpha(2-\gamma)}{\gamma} \sum_{i=1}^{d} \Big(\sqrt{T\, v_{T,i}} + \sqrt{T}\epsilon_T \Big)$$

With the help of Lemma 4.1 and Corollary 4.1 we can now state the regret bound for RMSProp.

Theorem 4.1 *Let Assumptions **A1**, **A2** hold and let* θ_t *be the sequence generated by RMSProp in Algorithm 3, where* $g_t \in \partial f_t(\theta_t)$ *and* $f_t : C \to \mathbb{R}$ *is an arbitrary convex function and* $\alpha_t = \frac{\alpha}{\sqrt{t}}$ *for some* $\alpha > 0$ *and* $1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t}$ *for some* $0 < \gamma \leq 1$. *Also for* $t > 1$ *let* $\sqrt{(t-1)}\epsilon_{t-1} \leq \sqrt{t}\epsilon_t$, *then the regret of RMSProp can be upper bounded for* $T \geq 1$ *as*

$$R(T) \leq \Big(\frac{D_\infty^2}{2\alpha} + \frac{\alpha(2-\gamma)}{\gamma} \Big) \sum_{i=1}^{d} \Big(\sqrt{T v_{T,i}} + \sqrt{T}\epsilon_T \Big)$$

Note that for $\beta_t = 1 - \frac{1}{t}$, that is $\gamma = 1$, and $\epsilon_t = \frac{\delta}{T}$ where RMSProp corresponds to Adagrad we recover the regret bound of Adagrad in the convex case, see Theorem 2.1, up to the damping factor. Note that in this case

$$\sqrt{T v_{T,i}} = \sqrt{\sum_{j=1}^{T} g_{j,i}^2} = \left\| g_{1:T,i} \right\|_2.$$

4.1. SC-RMSProp

Similar to the extension of Adagrad to SC-Adagrad, we present in this section SC-RMSProp which achieves a logarithmic regret bound.

Note that again there exist choices for the parameters of SC-RMSProp such that it reduces to SC-Adagrad. The correspondence is given by the choice

$$\beta_t = 1 - \frac{1}{t}, \quad \alpha_t = \frac{\alpha}{t}, \quad \epsilon_t = \frac{\delta_t}{t},$$

for which again it follows $v_{t,i} = \frac{1}{t} \sum_{j=1}^{t} g_{j,i}^2$ with the same argument as for RMSProp. Please see Equation (5) for the correspondence. Moreover, with the same argument as for SC-Adagrad we use a decay scheme for the damping factor

$$\epsilon_{t,i} = \xi_2 \frac{e^{-\xi_1\, t\, v_{t,i}}}{t}, \quad i = 1, \ldots, d. \text{ for } \xi_1 \geq 0\,,\ \xi_2 > 0$$

Algorithm 4 SC-RMSProp

Input: $\theta_1 \in C$, $\delta_0 = 1$, $v_0 = 0 \in \mathbb{R}^d$
for $t = 1$ **to** T **do**
$g_t \in \partial f_t(\theta_t)$
$v_t = \beta_t v_{t-1} + (1-\beta_t)(g_t \odot g_t)$
Set $\epsilon_t = \frac{\delta_t}{t}$ where $\delta_{t,i} \leq \delta_{t-1,i}$ for $i \in [d]$ and $\alpha_t = \frac{\alpha}{t}$
$A_t = \text{diag}(v_t + \epsilon_t)$
$\theta_{t+1} = P_C^{A_t}\left(\theta_t - \alpha_t A_t^{-1} g_t\right)$
end for

The analysis of SC-RMSProp is along the lines of SC-Adagrad with some overhead due to the structure of v_t.

Lemma 4.2 *Let $\alpha_t = \frac{\alpha}{t}$, $1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t}$ and A_t as defined in SC-RMSProp, then it holds for all $T \geq 1$,*

$$\sum_{t=1}^{T} \frac{\alpha_t}{2} \left\langle g_t, A_t^{-1} g_t \right\rangle \leq \frac{\alpha}{2\gamma} \sum_{i=1}^{d} \log\left(\frac{T\left(v_{T,i} + \epsilon_{T,i}\right)}{\epsilon_{1,i}}\right) + \frac{\alpha}{2\gamma} \sum_{t=2}^{T} \sum_{i=1}^{d} \frac{-t\epsilon_{t,i} + (t-1)\epsilon_{t-1,i}}{t v_{t,i} + t\epsilon_{t,i}} + \frac{\alpha}{2\gamma} \sum_{i=1}^{d} \frac{(1-\gamma)(1+\log T)}{\inf_{j \in [1,T]} j v_{j,i} + j\epsilon_{j,i}}$$

Note that for $\gamma = 1$ and the choice $\epsilon_t = \frac{\delta_t}{t}$ this reduces to the result of Lemma 3.2.

Lemma 4.3 *Let $\epsilon_t \leq \frac{1}{t}$ and $1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t}$ for some $1 \geq \gamma > 0$. Then it holds,*

$$\sum_{t=2}^{T} \frac{\alpha}{2} \left((tA_t)^{-1} \bullet \left(\frac{-\text{diag}(\epsilon_t) + \beta_t \text{diag}(\epsilon_{t-1})}{(1-\beta_t)}\right)\right) \leq \frac{\alpha}{2\gamma} \sum_{t=2}^{T} \sum_{i=1}^{d} \frac{-t\epsilon_{t,i} + (t-1)\epsilon_{t-1,i}}{t v_{t,i} + t\epsilon_{t,i}} + \frac{\alpha}{2\gamma} \sum_{i=1}^{d} \frac{(1-\gamma)(1+\log T)}{\inf_{j \in [1,T]} j v_{j,i} + j\epsilon_{j,i}}$$

Theorem 4.2 *Let Assumptions **A1**, **A2** hold and let θ_t be the sequence generated by SC-RMSProp in Algorithm 4, where $g_t \in \partial f_t(\theta_t)$ and $f_t : C \to \mathbb{R}$ is an arbitrary μ-strongly convex function ($\mu \in \mathbb{R}^d_+$) with $\alpha_t = \frac{\alpha}{t}$ for some $\alpha \geq \frac{(2-\gamma)G_\infty^2}{2\min_i \mu_i}$ and $1 - \frac{1}{t} \leq \beta_t \leq 1 - \frac{\gamma}{t}$ for some $0 < \gamma \leq 1$. Furthermore, set $\epsilon_t = \frac{\delta_t}{t}$ and assume $1 \geq \delta_{t,i} > 0$ and $\delta_{t,i} \leq \delta_{t-1,i} \forall t \in [T], \forall i \in [d]$, then the regret of SC-RMSProp can be upper bounded for $T \geq 1$ as*

$$R(T) \leq \frac{D_\infty^2 \,\text{tr}(\text{diag}(\delta_1))}{2\alpha} + \frac{\alpha}{2\gamma} \sum_{i=1}^{d} \log\left(\frac{T v_{T,i} + \delta_{T,i}}{\delta_{1,i}}\right) + \frac{1}{2} \sum_{i=1}^{d} \inf_{t \in [T]} \left(\frac{(\theta_{t,i} - \theta_i^*)^2}{\alpha} - \frac{\alpha}{\gamma(t v_{t,i} + t\epsilon_{t,i})}\right)(\delta_{T,i} - \delta_{1,i}) + \frac{\alpha}{2\gamma} \sum_{i=1}^{d} \frac{(1-\gamma)(1+\log T)}{\inf_{j \in [1,T]} j v_{j,i} + j\epsilon_{j,i}}$$

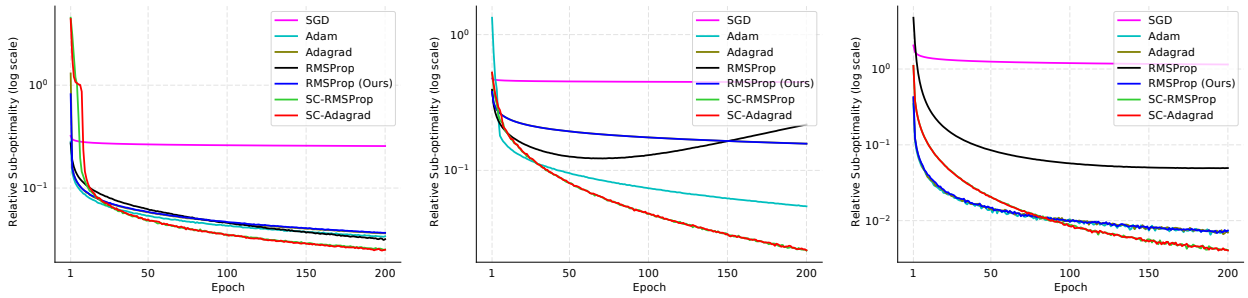
Note that the regret bound reduces for $\gamma = 1$ to that of SC-Adagrad. For $0 < \gamma < 1$ a comparison between the bounds is not straightforward as the $v_{t,i}$ terms cannot be compared. It is an interesting future research question whether it is possible to show that one scheme is better than the other one potentially dependent on the problem characteristics.

5. Experiments

The idea of the experiments is to show that the proposed algorithms are useful for standard learning problems in both online and batch settings. We are aware of the fact that in the strongly convex case online to batch conversion is not tight (Hazan & Kale, 2014), however that does not necessarily imply that the algorithms behave generally suboptimal. We compare all algorithms for a strongly convex problem and present relative suboptimality plots, $\log_{10}\left(\frac{f(x_t) - p^*}{p^*}\right)$, where p^* is the global optimum, as well as separate regret plots, where we compare to the best optimal parameter in hindsight for the fraction of training points seen so far. On the other hand RMSProp was originally developed by (Hinton et al., 2012) for usage in deep learning. As discussed before the fixed choice of β_t is not allowed if one wants to get the optimal $O(\sqrt{T})$ regret bound in the convex case. Thus we think it is of interest to the deep learning community, if the insights from the convex optimization case transfer to deep learning. Moreover, Adagrad and RMSProp are heavily used in deep learning and thus it is interesting to compare their counterparts SC-Adagrad and SC-RMSProp developed for the strongly convex case also in deep learning. The supplementary material contains additional experiments on various neural network models.

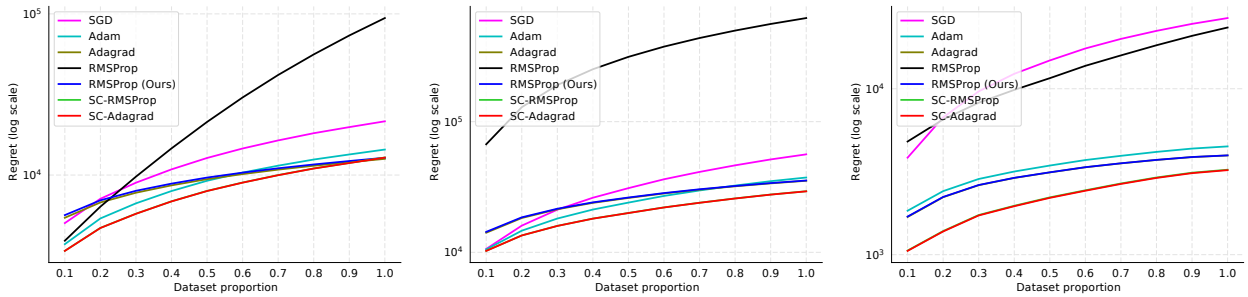
Datasets: We use three datasets where it is easy, difficult and very difficult to achieve good test performance, just in order to see if this influences the performance. For this purpose we use MNIST (60000 training samples, 10 classes), CIFAR10 (50000 training samples, 10 classes) and CIFAR100 (50000 training samples, 100 classes). We refer to (Krizhevsky, 2009) for more details on the CIFAR datasets.

Algorithms: We compare 1) Stochastic Gradient Descent (SGD) (Bottou, 2010) with $O(1/t)$ decaying step-size for the strongly convex problems and for non-convex problems we use a constant learning rate, 2) Adam (Kingma & Bai, 2015) , is used with step size decay of $\alpha_t = \frac{\alpha}{\sqrt{t}}$ for strongly convex problems and for non-convex problems we use a constant step-size. 3) Adagrad, see Algorithm 1, remains the same for strongly convex problems and non-convex



(a) CIFAR10 (b) CIFAR100 (c) MNIST

Figure 1. Relative Suboptimality vs Number of Epoch for L2-Regularized Softmax Regression



(a) CIFAR10 (b) CIFAR100 (c) MNIST

Figure 2. Regret (log scale) vs Dataset Proportion for Online L2-Regularized Softmax Regression

problems. 4) RMSProp as proposed in (Hinton et al., 2012) is used for both strongly convex problems and non-convex problems with $\beta_t = 0.9\ \forall t \geq 1$. 5) RMSProp (Ours) is used with step-size decay of $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_t = 1 - \frac{\gamma}{t}$. In order that the parameter range is similar to the original RMSProp ((Hinton et al., 2012)) we fix as $\gamma = 0.9$ for all experiment (note that for $\gamma = 1$ RMSProp (Ours) is equivalent to Adagrad), 6) SC-RMSProp is used with stepsize $\alpha_t = \frac{\alpha}{t}$ and $\gamma = 0.9$ as RMSProp (Ours) 7) SC-Adagrad is used with a constant stepsize α. The decaying damping factor for both SC-Adagrad and SC-RMSProp is used with $\xi_1 = 0.1, \xi_2 = 1$ for convex problems and we use $\xi_1 = 0.1, \xi_2 = 0.1$ for non-convex deep learning problems. Finally, the numerical stability parameter δ used in Adagrad, Adam, RMSProp is set to 10^{-8} as it is typically recommended for these algorithms.

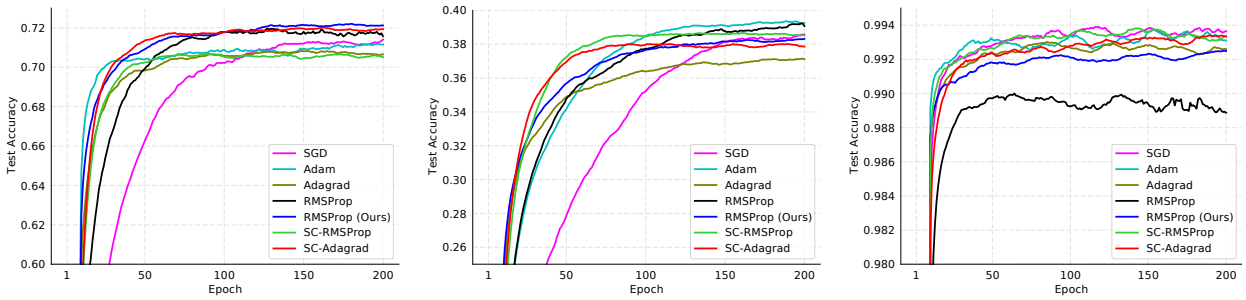
Setup: Note that all methods have only one varying parameter: the stepsize α which we choose from the set of $\{1, 0.1, 0.01, 0.001, 0.0001\}$ for all experiments. By this setup no method has an advantage just because it has more hyperparameters over which it can optimize. The optimal rate is always chosen for each algorithm separately so that one achieves either best training objective or best test performance after a fixed number of epochs.

Strongly Convex Case - Softmax Regression: Given the training data $(x_i, y_i)_{i\in[m]}$ and let $y_i \in [K]$. we fit a linear model with cross entropy loss and use as regularization the squared Euclidean norm of the weight parameters. The objective is then given as

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{e^{\theta_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{K} e^{\theta_j^T x_i + b_j}}\right) + \lambda\sum_{k=1}^{K}\|\theta_k\|^2$$

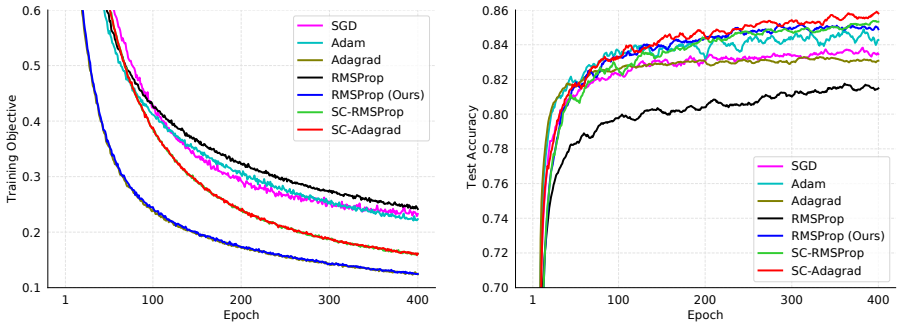
All methods are initialized with zero weights. The regularization parameter was chosen so that one achieves the best prediction performance on the test set. The results are shown in Figure 1. We also conduct experiments in an online setting, where we restrict the number of iterations to the number of training samples. Here for all the algorithms, we choose the stepsize resulting in best regret value at the end. We plot the Regret (in log scale) vs dataset proportion seen, and as expected SC-Adagrad and SC-RMSProp outperform all the other methods across all the considered datasets. Also, RMSProp (Ours) has a lower regret values than the original RMSProp as shown in Figure 2.

Convolutional Neural Networks: Here we test a 4-layer CNN with two convolutional (32 filters of size 3×3) and one fully connected layer (128 hidden units followed by 0.5 dropout). The activation function is ReLU and after



(a) CIFAR10 (b) CIFAR100 (c) MNIST

Figure 3. Test Accuracy vs Number of Epochs for 4-layer CNN



(a) Training Objective (b) Test Accuracy

Figure 4. Plots of ResNet-18 on CIFAR10 dataset

the last convolutional layer we use max-pooling over a 2×2 window and 0.25 dropout. The final layer is a softmax layer and the final objective is cross-entropy loss. This is a pretty simple standard architecture and we use it for all datasets. The results are shown in Figure 3. Both RMSProp (Ours) and SC-Adagrad perform better than all the other methods in terms of test accuracy for CIFAR10 dataset. On both CIFAR100 and MNIST datasets SC-RMSProp is very competitive.

Residual Network: We also conduct experiments for ResNet-18 network proposed in (He et al., 2016a) where the residual blocks are used with modifications proposed in (He et al., 2016b) on CIFAR10 dataset. We report the results in Figures 4. SC-Adagrad, SC-RMSProp and RMSProp (Ours) have the best performance in terms of test Accuracy and RMSProp (Ours) has the best performance in terms of training objective along with Adagrad.

We also test all the algorithms on a simple 3-layer Multilayer perceptron which we include in the supplementary material. Given these experiments, we think that SC-RMSProp and SC-Adagrad are valuable new adaptive gradient techniques for deep learning.

6. Conclusion

We have analyzed RMSProp originally proposed in the deep learning community in the framework of online convex optimization. We show that the conditions for convergence of RMSProp for the convex case are different than what is used by (Hinton et al., 2012) and that this leads to better performance in practice. We also propose variants SC-Adagrad and SC-RMSProp which achieve logarithmic regret bounds for the strongly convex case. Moreover, they perform very well for different network models and datasets and thus they are an interesting alternative to existing adaptive gradient schemes. In the future we want to explore why these algorithms perform so well in deep learning tasks even though they have been designed for the strongly convex case.

Acknowledgements

We would like to thank Shweta Mahajan and all the reviewers for their insightful comments.

References

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Daniel, C., Taylor, J., and Nowozin, S. Learning step size controllers for robust neural network training. In *AAAI*, 2016.

Dauphin, Y., de Vries, H., and Bengio, Y. Equilibrated adaptive learning rates for non-convex optimization. In *NIPS*, 2015.

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

Hazan, E. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2:157–325, 2016.

Hazan, E. and Kale, S. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

Hazan, E., Agarwal, A., and Kale, S. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016b.

Hinton, G., Srivastava, N., and Swersky, K. Lecture 6d - a separate, adaptive learning rate for each connection. Slides of Lecture Neural Networks for Machine Learning, 2012.

Karpathy, A. and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2016.

Kingma, D. P. and Bai, J. L. Adam: a method for stochastic optimization. *ICLR*, 2015.

Koushik, Jayanth and Hayashi, Hiroaki. Improving stochastic gradient descent with feedback. *arXiv preprint arXiv:1611.01505*, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

McMahan, H Brendan. A survey of algorithms and analysis for adaptive online learning. *arXiv preprint arXiv:1403.3465*, 2014.

Ruder, S. An overview of gradient descent optimization algorithms. preprint, arXiv:1609.04747v1, 2016.

Schaul, T., Antonoglou, I., and Silver, D. Unit tests for stochastic optimization. In *ICLR*, 2014.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Szegedy, C., Ioffe, S., and Vanhoucke, V. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR Workshop*, 2016a.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016b.

Zeiler, M. D. ADADELTA: An adaptive learning rate method. preprint, arXiv:1212.5701v1, 2012.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.