Scalable Adaptive Stochastic Optimization Using Random Projections

Gabriel Krummenacher

Brian McWilliams

Yannic Kilcher

Joachim M. Buhmann

Nicolai Meinshausen

Abstract

Adaptive stochastic gradient methods such as ADA
d 
GRAD have gained popularity in particular for training deep neural networks. The most commonly used and studied variant maintains a diagonal matrix approximation to second order information by accumulating past gradients which are used to tune the step size adaptively. In certain situations the full-matrix variant of ADA
d 
GRAD is expected to attain better performance, however in high dimensions it is computationally impractical. We present ADA-LR and RADA
d 
GRAD two computationally efficient approximations to full-matrix ADA
d 
GRAD based on randomized dimensionality reduction. They are able to capture dependencies between features and achieve similar performance to full-matrix ADA
d 
GRAD but at a much smaller computational cost. We show that the regret of ADA-LR is close to the regret of full-matrix ADA
d 
GRAD which can have an up-to exponentially smaller dependence on the dimension than the diagonal variant. Empirically, we show that ADA-LR and RADA
d 
GRAD perform similarly to full-matrix ADA
d 
GRAD. On the task of training convolutional neural networks as well as recurrent neural networks, RADA
d 
GRAD achieves faster convergence than diagonal ADA
d 
GRAD.

1 Introduction

Recently, adaptive stochastic optimization algorithms have gained popularity for large-scale convex and non-convex optimization problems. Among these, ADA
d 
GRAD [10] and its variants [22] have received particular attention and have proven among the most successful algorithms for training deep networks. Although these problems are inherently highly non-convex, recent work has begun to explain the success of such algorithms [3, 5].

ADA
d 
GRAD adaptively sets the learning rate for each dimension by means of a time-varying proximal regularizer. The most commonly studied and utilised version considers only a diagonal matrix proximal term. As such it incurs almost no additional computational cost over standard stochastic
gradient descent (SGD). However, when the data has low effective rank the regret of ADA\textsubscript{GRAD} may have a much worse dependence on the dimensionality of the problem than its full-matrix variant (which we refer to as ADA\textsubscript{FULL}). Such settings are common in high dimensional data where there are many correlations between features and can also be observed in the convolutional layers of neural networks. The computational cost of ADA\textsubscript{FULL} is substantially higher than that of ADA\textsubscript{GRAD} – it requires computing the inverse square root of the matrix of gradient outer products to evaluate the proximal term which grows with the cube of the dimension. As such it is rarely used in practise.

In this work we propose two methods that approximate the proximal term used in ADA\textsubscript{FULL} drastically reducing computational and storage complexity with little adverse affect on optimization performance. First, in Section 3.1 we develop ADA\textsubscript{LR}, a simple approximation using random projections. This procedure reduces the computational complexity of ADA\textsubscript{FULL} by a factor of \( p \) but retains similar theoretical guarantees. In Section 3.2 we systematically profile the most computationally expensive parts of ADA\textsubscript{LR} and introduce further randomized approximations resulting in a truly scalable algorithm, RADA\textsubscript{GRAD}. In Section 3.3 we outline a simple modification to RADA\textsubscript{GRAD} – reducing the variance of the stochastic gradients – which greatly improves practical performance. Finally we perform an extensive comparison between the performance of RADA\textsubscript{GRAD} with several widely used optimization algorithms on a variety of deep learning tasks. For image recognition with convolutional networks and language modeling with recurrent neural networks we find that RADA\textsubscript{GRAD} and in particular its variance-reduced variant achieves faster convergence.

1.1 Related work

Motivated by the problem of training deep neural networks, very recently many new adaptive optimization methods have been proposed. Most computationally efficient among these are first order methods similar in spirit to ADA\textsubscript{GRAD}, which suggest alternative normalization factors \[ 22 \] [29] \[ 7 \]. Several authors propose efficient stochastic variants of classical second order methods such as LBFGS \[ 6 \] [21]. Efficient algorithms exist to update the inverse of the Hessian approximation by applying the matrix-inversion lemma or directly updating the Hessian-vector product using the “double-loop” algorithm but these are not applicable to ADA\textsubscript{GRAD} style algorithms. In the convex setting these methods can show great theoretical and practical benefit over first order methods but have yet to be extensively applied to training deep networks.

On a different note, the growing zoo of variance reduced SGD algorithms \[ 20 \] [8] [19] has shown vastly superior performance to ADA\textsubscript{GRAD}-style methods for standard empirical risk minimization and convex optimization. Recent work has aimed to move these methods into the non-convex setting \[ 11 \]. Notably, \[ 23 \] combine variance reduction with second order methods.

Most similar to RADA\textsubscript{GRAD} are those which propose factorized approximations of second order information. Several methods focus on the natural gradient method \[ 2 \] which leverages second order information through the Fisher information matrix. \[ 15 \] approximate the inverse Fisher matrix using a sparse graphical model. \[ 9 \] use low-rank approximations whereas \[ 27 \] propose an efficient Kronecker product based factorization. Concurrently with this work, \[ 13 \] propose a randomized preconditioner for SGD. However, their approach requires access to all of the data at once in order to compute the preconditioning matrix which is impractical for training deep networks. \[ 24 \] propose a theoretically motivated algorithm similar to ADA\textsubscript{LR} and a faster alternative based on Oja’s rule to update the SVD.

Fast random projections. Fast random projections are low-dimensional embeddings \( \Pi : \mathbb{R}^p \rightarrow \mathbb{R}^\tau \) which preserve – up to a small distortion – the geometry of a subspace of vectors. We concentrate on the class of structured random projections, among which the Subsampled Randomized Fourier Transform (SRFT) has particularly attractive properties \[ 16 \]. The SRFT consists of a preconditioning step after which \( \tau \) columns of the new matrix are subsampled uniformly at random as \( \Pi = \sqrt{p/\tau} S \Theta D \) with the definitions: \( i \) \( S \in \mathbb{R}^{\tau \times p} \) is a subsampling matrix. \( i i \) \( D \in \mathbb{R}^{p \times p} \) is a diagonal matrix whose entries are drawn independently from \( \{-1, 1\} \). \( i i i \) \( \Theta \in \mathbb{R}^{p \times p} \) is a unitary discrete Fourier transform (DFT) matrix. This formulations allows very fast implementations using the fast Fourier transform (FFT), for example using the popular FFTW package\(^*\). Applying the FFT to a \( p \)-dimensional vector can be achieved in \( O(p \log \tau) \) time. Similar structured random projections

\[ * \text{http://www.fftw.org/} \]
have gained popularity as a way to speed up [25] and robustify [28] large-scale linear regression and for distributed estimation [18, 17].

1.2 Problem setting

The problem considered by [10] is online stochastic optimization where the goal is, at each step, to predict a point \( \beta_t \in \mathbb{R}^p \) which achieves low regret with respect to a fixed optimal predictor, \( \beta^{\text{opt}} \), for a sequence of (convex) functions \( F_t(\beta) \). After \( T \) rounds, the regret can be defined as
\[
R(T) = \sum_{t=1}^{T} F_t(\beta_t) - \sum_{t=1}^{T} F_t(\beta^{\text{opt}}).
\]
Initially, we will consider functions \( F_t \) of the form \( F_t(\beta) = f_t(\beta) + \varphi(\beta) \) where \( f_t \) and \( \varphi \) are convex loss and regularization functions respectively. Throughout, the vector \( g_t \in \nabla f_t(\beta_t) \) refers to a particular subgradient of the loss function. Standard first order methods update \( \beta_t \) at each step by moving in the opposite direction of \( g_t \) according to a step-size parameter, \( \eta_t \). The ADAGRAD family of algorithms [10] instead use an adaptive learning rate which can be different for each feature. This is controlled using a time-varying proximal term which we briefly review. Defining
\[
G_t = \sum_{i=1}^{t} g_i g_i^\top \quad \text{and} \quad H_t = \delta I_p + (G_{t-1} + g_t g_t^\top)^{1/2},
\]
the ADA-FULL proximal term is given by
\[
\psi_t(\beta) = \frac{1}{2} \langle \beta, H_t \beta \rangle.
\]
Clearly when \( p \) is large, constructing \( G \) and finding its root and inverse at each iteration is impractical. In practice, rather than the full outer product matrix, ADAGRAD uses a proximal function consisting of the diagonal of \( G_t \). The ADA-FULL proximal term is computational cheaper, it is unable to capture dependencies between coordinates in the gradient terms. Despite this, ADAGRAD has been found to perform very well empirically. One reason for this is modern high-dimensional datasets are typically also very sparse. Under these conditions, coordinates in the gradient are approximately independent.

2 Stochastic optimization in high dimensions

ADAGRAD has attractive theoretical and empirical properties and adds essentially no overhead above a standard first order method such as SGD. It begs the question, what we might hope to gain by introducing additional computational complexity. In order to motivate our contribution, we first present an analogue of the discussion in [11] focussing on when data is high-dimensional and dense. We argue that if the data has low-rank (rather than sparse) structure ADA-FULL can effectively adapt to the intrinsic dimensionality. We also show in Section 3.1 that ADA-LR has the same property.

First, we review the theoretical properties of ADAGRAD algorithms, borrowing the \( g_{1:T,j} \) notation [10].

**Proposition 1.** ADAGRAD and ADA-FULL achieve the following regret (Corollaries 6 & 11 from [10]) respectively:
\[
R_D(T) \leq 2\|\beta^{\text{opt}}\|_\infty \sum_{j=1}^{P} \|g_{1:T,j}\| + \delta\|\beta^{\text{opt}}\|_1, \quad R_F(T) \leq 2\|\beta^{\text{opt}}\| \cdot \text{tr}(G_{T+1}^{1/2}) + \delta\|\beta^{\text{opt}}\|. \tag{1}
\]

The major difference between \( R_D(T) \) and \( R_F(T) \) is the inclusion of the final full-matrix and diagonal proximal term, respectively. Under a sparse data generating distribution ADAGRAD achieves an up-to exponential improvement over SGD which is optimal in a minimax sense [11]. While data sparsity is often observed in practise in high-dimensional datasets (particularly web/text data) many other problems are dense. Furthermore, in practise applying ADAGRAD to dense data results in a learning rate which tends to decay too rapidly. It is therefore natural to ask how dense data affects the performance of ADA-FULL.

For illustration, consider when the data points \( x_t \) are sampled i.i.d. from a Gaussian distribution \( P_X = \mathcal{N}(\mathbf{0}, \Sigma) \). The resulting variable will clearly be dense. A common feature of high dimensional data is low effective rank defined for a matrix \( \Sigma \) as \( r(\Sigma) = \text{tr}(\Sigma)/\|\Sigma\| \leq \text{rank}(\Sigma) \leq p \). Low effective rank implies that \( r \ll p \) and therefore the eigenvalues of the covariance matrix decay quickly. We will consider distributions parameterised by covariance matrices \( \Sigma \) with eigenvalues \( \lambda_j(\Sigma) = \lambda_0 j^{-\alpha} \) for \( j = 1, \ldots, p \).
Functions of the form \( F_t(\beta) = F_t(\beta^T x_t) \) have gradients \( \| g_t \| \leq M \| x_t \| \). For example, the least squares loss \( F_t(\beta^T x_t) = \frac{1}{2}(y_t - \beta^T x_t)^2 \) has gradient \( g_t = x_t(y_t - x_t^T \beta) = x_t \varepsilon_t \), such that \( \| \varepsilon_t \| \leq M \). Let us consider the effect of distributions parametrised by \( \Sigma \) on the proximal terms of full, and diagonal ADAGRAD. Plugging \( X \) into the proximal terms of \( [1] \) and taking expectations with respect to \( P_X \) we obtain for ADAGRAD and ADA-FULL respectively:

\[
\mathbb{E} \sum_{j=1}^{p} \| g_{1: T, j} \| \leq \sum_{j=1}^{p} M^2 \mathbb{E} \sum_{t=1}^{T} \sum_{j=1}^{p} x_{i, j}^2 \leq p M \sqrt{T}, \quad \mathbb{E} \text{tr}(\sum_{t=1}^{T} g_t g_t^T)^{1/2} \leq M \sqrt{T} \lambda_0 \sum_{j=1}^{p} j^{-\alpha/2},
\]

where the first inequality is from Jensen and the second is from noticing the sum of \( T \) squared Gaussian random variables is a \( \chi^2 \) random variable. We can consider the effect of fast-decaying spectrum: for \( \alpha \geq 2 \), \( \sum_{j=1}^{p} j^{-\alpha/2} = O(\log p) \) and for \( \alpha \in (1, 2) \), \( \sum_{j=1}^{p} j^{-\alpha/2} = O(p^{1-\alpha/2}) \).

When the data (and thus the gradients) are dense, yet have low effective rank, ADA-FULL is able to adapt to this structure. On the contrary, although ADAGRAD is computationally practical, in the worst case it may have exponentially worse dependence on the data dimension (\( p \) compared with \( \log p \)). In fact, the discrepancy between the regret of ADA-FULL and that of ADAGRAD is analogous to the discrepancy between ADAGRAD and SGD for sparse data.

### Algorithm 1 ADA-LR

**Input:** \( \eta > 0, \delta \geq 0, \tau \)

1: for \( t = 1 \ldots T \) do
2: \( \text{Receive } g_t = \nabla f_t(\beta_t) \).
3: \( G_t = G_{t-1} + g_t g_t^T \).
4: \( \text{Project: } \tilde{G}_t = G_t \Pi \).
5: \( \text{QR} = \tilde{G}_t \{ \text{QR-decomposition} \} \).
6: \( U, \Sigma, V = B \{ \text{SVD} \} \).
7: \( \beta_{t+1} = \beta_t - \delta \Sigma^{1/2} + \delta I)^{-1} V^T g_t \).
8: end for

**Output:** \( \beta_T \)

### Algorithm 2 RADAGrad

**Input:** \( \eta > 0, \delta \geq 0, \tau \)

1: for \( t = 1 \ldots T \) do
2: \( \text{Receive } g_t = \nabla f_t(\beta_t) \).
3: \( \text{Project: } \tilde{g}_t = \Pi g_t \).
4: \( G_t = G_{t-1} + g_t \tilde{g}_t^T \).
5: \( Q_t, R_t \leftarrow \text{qr_update}(Q_{t-1}, R_{t-1}, g_t, \tilde{g}_t) \).
6: \( B = G_t^T Q_t \).
7: \( U, \Sigma, W = B \{ \text{SVD} \} \).
8: \( V = W Q_t^T \).
9: \( \gamma_t = \eta(g_t - V V^T g_t) \).
10: \( \beta_{t+1} = \beta_t - \delta \Sigma^{1/2} + \delta I)^{-1} V^T g_t - \gamma_t \).
11: end for

**Output:** \( \beta_T \)

### 3 Approximating ADA-FULL using random projections

It is clear that in certain regimes, ADA-FULL provides stark optimization advantages over ADAGRAD in terms of the dependence on \( p \). However, ADA-FULL requires maintaining a \( p \times p \) matrix, \( G \) and computing its square root and inverse. Therefore, computationally the dependence of ADA-FULL on \( p \) scales with the cube which is impractical in high dimensions.

A naïve approach would be to simply reduce the dimensionality of the gradient vector, \( \tilde{g}_t \in \mathbb{R}^T = \Pi g_t \). ADA-FULL is now directly applicable in this low-dimensional space, returning a solution vector \( \beta_t \in \mathbb{R}^T \) at each iteration. However, for many problems, the original coordinates may have some intrinsic meaning or in the case of deep networks, may be parameters in a model. In which case it is important to return a solution in the original space. Unfortunately in general it is not possible to recover such a solution from \( \beta \). \([31]\).

Instead, we consider a different approach to maintaining and updating an approximation of the ADAGRAD matrix while retaining the original dimensionality of the parameter updates \( \beta \) and gradients \( g \).
3.1 Randomized low-rank approximation

As a first approach we approximate the inverse square root of $G_t$ using a fast randomized singular value decomposition (SVD) [15]. We proceed in two stages: First we compute an approximate basis $Q$ for the range of $G_t$. Then we use $Q$ to compute an approximate SVD of $G_t$ by forming the smaller dimensional matrix $B = Q^T G_t$ and then compute the low-rank SVD $U \Sigma V^T = B$. This is faster than computing the SVD of $G_t$ directly if $Q$ has few columns.

An approximate basis $Q$ can be computed efficiently by forming the matrix $\tilde{G}_t = G_t \Pi$ by means of a structured random projection and then constructing an orthonormal basis for the range of $\tilde{G}_t$ by QR-decomposition. The randomized SVD allows us to quickly compute the square root and pseudo-inverse of the proximal term $H_t$ by setting $H_t^{-1} = V (\Sigma^{1/2} + \delta I)^{-1} V^T$. We call this approximation Ada-LR and describe the steps in full in Algorithm 1.

In practice, using a structured random projection such as the SRFT leads to an approximation of the original matrix, $G_t$ of the form $\|G_t - QQ^T G_t\| \leq \epsilon$, with high probability [15] where $\epsilon$ depends on $\tau$, the number of columns of $Q$; $p$ and the $\tau^{th}$ singular value of $G_t$. Briefly, if the singular values of $G_t$ decay quickly and $\tau$ is chosen appropriately, $\epsilon$ will be small (this is stated more formally in Proposition 2). We leverage this result to derive the following regret bound for Ada-LR (see Proposition 2 for proof).

**Proposition 2.** Let $\sigma_k$ be the $k$th largest singular value of $G_t$. Setting the projection dimension as $4 \left( \sqrt{k} + \sqrt{8 \log(k) \epsilon} \right)^2 \leq \tau \leq p$ and defining $\epsilon = \sqrt{1 + 7p/\tau} \cdot \sigma_k$. With failure probability at most $O\left(k^{-1}\right)$ Ada-LR achieves regret $R_{LR}(T) \leq 2\epsilon \|G_t\|_{2} + (2\tau \sqrt{\epsilon} + \delta) \beta^{opt}$

Due to the randomized approximation we incur an additional $2\tau \sqrt{\epsilon} \|\beta^{opt}\|$ compared with the regret of Ada-FULL (eq. [1]). So, under the earlier stated assumption of fast decaying eigenvalues we can use an identical argument as in eq. (2) to similarly obtain a dimension dependence of $O\left(\log p + \tau\right)$. Approximating the inverse square root decreases the complexity of each iteration from $O\left(p^3\right)$ to $O\left(\tau p^2\right)$. We summarize the cost of each step in Algorithm 1 and contrast it with the cost of Ada-FULL in Table A.1 in Section A. Even though Ada-LR removes one factor of $p$ from the runtime of Ada-FULL it still needs to store the large matrix $G_t$. This prevents Ada-LR from being a truly practical algorithm. In the following section we propose a second algorithm which directly stores a low dimensional approximation to $G_t$ that can be updated cheaply. This allows for an improvement in runtime to $O\left(\tau^2 p\right)$.

3.2 RADAGrad: A faster approximation

From Table A.1 the expensive steps in Algorithm 1 are the update of $G_t$ (line 3), the random projection (line 4) and the projection onto the approximate range of $G_t$ (line 6). In the following we propose RADAGrad, an algorithm that reduces the complexity to $O\left(\tau^2 p\right)$ by only approximately solving some of the expensive steps in Ada-LR while maintaining similar performance in practice.

To compute the approximate range $Q$, we do not need to store the full matrix $G_t$. Instead we only require the low dimensional matrix $G_t = G_t \Pi$. This matrix can be computed iteratively by setting $G_t \in \mathbb{R}^{p \times \tau} = G_{t-1} + g_t (\Pi g_t)^T$. This directly reduces the cost of the random projection to $O(p \log \tau)$ since we only project the vector $g_t$ instead of the matrix $G_t$, it also makes the update of $G_t$ faster and saves storage.

We then project $\tilde{G}_t$ on the approximate range of $G_t$ and use the SVD to compute the inverse square root. Since $G_t$ is symmetric its row and column space are identical so little information is lost by projecting $\tilde{G}_t$ instead of $G_t$ on the approximate range of $G_t$. The advantage is that we can now compute the SVD in $O\left(\tau^3\right)$ and the matrix-matrix product on line 6 in $O\left(\tau^2 p\right)$. See Algorithm 2 for the full procedure.

The most expensive steps are now the QR decomposition and the matrix multiplications in steps 6 and 8 (see Algorithm 2 and Table A.1). Since at each iteration we only update the matrix $\tilde{G}_t$ with

---

3 This idea is similar to bilinear random projections [14].
the rank-one matrix $g_t g_t^\top$ we can use faster rank-1 QR-updates \cite{12} instead of recomputing the full QR decomposition. To speed up the matrix-matrix product $G_t^\top Q$ for very large problems (e.g. backpropagation in convolutional neural networks), a multithreaded BLAS implementation can be used.

3.3 Practical algorithms

Here we outline several simple modifications to the RADA\textsc{Grad} algorithm to improve practical performance.

Corrected update. The random projection step only retains at most $\tau$ eigenvalues of $G_t$. If the assumption of low effective rank does not hold, important information from the $p - \tau$ smallest eigenvalues might be discarded. RADA\textsc{Grad} therefore makes use of the corrected update

$$\beta_{t+1} = \beta_t - \eta V(\Sigma^{1/2} + \delta I)^{-1}V^\top g_t - \gamma_t,$$

where $\gamma_t = \eta(I - VV^\top)g_t$. $\gamma_t$ is the projection of the current gradient onto the space orthogonal to the one captured by the random projection of $G_t$. This ensures that important variation in the gradient which is poorly approximated by the random projection is not completely lost. Consequently, if the data has rank less than $\tau$, $\|\gamma\| \approx 0$. This correction only requires quantities which have already been computed but greatly improves practical performance.

Variance reduction. Variance reduction methods based on SVRG \cite{20} obtain lower-variance gradient estimates by means of computing a "pivot point" over larger batches of data. Recent work has shown improved theoretical and empirical convergence in non-convex problems \cite{1} in particular in combination with ADA\textsc{Grad}.

We modify RADA\textsc{Grad} to use the variance reduction scheme of SVRG. The full procedure is given in Algorithm 3 in Section B. The majority of the algorithm is as RADA\textsc{Grad} except for the outer loop which computes the pivot point, $\mu$ every epoch which is used to reduce the variance of the stochastic gradient (line 4). The important additional parameter is $m$, the update frequency for $\mu$. As in \cite{1} we set this to $m = 50$. Practically, as is standard practise we initialise RADA-VR by running ADA\textsc{Grad} for several epochs.

We study the empirical behaviour of ADA-LR, RADA\textsc{Grad} and its variance reduced variant in the next section.

4 Experiments

4.1 Low effective rank data

We compare the performance of our proposed algorithms against both the diagonal and full-matrix ADA\textsc{Grad} variants in the idealised setting where the data is dense but has low effective rank. We generate binary classification data with $n = 1000$ and $p = 125$. The data is sampled i.i.d. from a Gaussian distribution $\mathcal{N}(\mu_c, \Sigma)$ where $\Sigma$ has with rapidly decaying eigenvalues $\lambda_j(\Sigma) = \lambda_0 \beta^{-\alpha}$ with $\alpha = 1.3$, $\lambda_0 = 30$. Each of the two classes has a different mean, $\mu_c$.

For each algorithm learning rates are tuned using cross validation. The results for 5 epochs are averaged over 5 runs with different permutations of the data set and instantiations of the random projection for ADA-LR and RADA\textsc{Grad}. For the random projection we use an oversampling factor so $\Pi \in \mathbb{R}^{(10+\tau)\times p}$ to ensure accurate recovery of the top $\tau$ singular values and then set the values of $\lambda_{[\tau:p]}$ to zero \cite{16}.

Figure 1: Comparison of: (a) Loss and (b) the largest eigenvalues (normalised by their sum) of the proximal term on simulated data.
The results of our experiments can be seen in Figure 2, where we show the objective value during training and evaluating performance on a validation set. We found that RADA variants consistently outperform both ADAGRAD and the combination of ADAGRAD+SVRG on these tasks. In particular combining RADA with variance reduction results in the largest improvement for training although both RADA variants quickly converge to very similar values for test accuracy.

4.2 Non-convex optimization in neural networks

Here we compare RADA against ADAGRAD and the combination of ADAGRAD+SVRG on the task of optimizing several different neural network architectures.

Convolutional Neural Networks. We used modified variants of standard convolutional network architectures for image classification on the MNIST, CIFAR-10 and SVHN datasets. These consist of three $5 \times 5$ convolutional layers generating 32 channels with ReLU non-linearities, each followed by $2 \times 2$ max-pooling. The final layer was a dense softmax layer and the objective was to minimize the categorical cross entropy.

We used a batch size of 8 and trained the networks without momentum or weight decay, in order to eliminate confounding factors. Instead, we used dropout regularization ($p = 0.5$) in the dense layers during training. Step sizes were determined by coarsely searching a log scale of possible values and evaluating performance on a validation set. We found RADA to have a higher impact with convolutional layers than with dense layers, due to the higher correlations between weights. Therefore, for computational reasons, RADA was only applied on the convolutional layers. The last dense classification layer was trained with ADAGRAD. In this setting ADAGRAD+SVRG is computationally infeasible. The number of parameters in the convolutional layers is between 50-80k.

Simply storing the full $G$ matrix using double precision would require more memory than is available on top-of-the-line GPUs.

The results of our experiments can be seen in Figure 2 where we show the objective value during training and the test accuracy. We find that both RADA variants consistently outperform both ADAGRAD and the combination of ADAGRAD+SVRG on these tasks. In particular combining RADA with variance reduction results in the largest improvement for training although both RADA variants quickly converge to similar values for test accuracy.
For all models, the learning rate selected by RADA\textsc{Grad} is approximately an order of magnitude larger than the one selected by ADA\textsc{Grad}. This suggests that RADA\textsc{Grad} can make more aggressive steps than ADA\textsc{Grad}, which results in the relative success of RADA\textsc{Grad} over ADA\textsc{Grad}, especially at the beginning of the experiments.

We observed that RADA\textsc{Grad} performed 5-10× slower than ADA\textsc{Grad} per iteration. This can be attributed to the lack of GPU-optimized SVD and QR routines. These numbers are comparable with other similar recently proposed techniques [24]. However, due to the faster convergence we found that the overall optimization time of RADA\textsc{Grad} was lower than for ADA\textsc{Grad}.

**Recurrent Neural Networks.**

We trained the strongly-typed variant of the long short-term memory network (T-LSTM, [4]) for language modeling, which consists of the following task: Given a sequence of words from an original text, predict the next word. We used pre-trained GLO\textsc{Ve} embedding vectors [30] as input to the T-LSTM layer and a softmax over the vocabulary (10k words) as output. The loss is the mean categorical cross-entropy. The memory size of the T-LSTM units was set to 256. We trained and evaluated our network on the Penn Treebank dataset [26]. We subsampled strings of length 20 from the dataset and asked the network to predict each word in the string, given the words up to that point. Learning rates were selected by searching over a log scale of possible values and measuring performance on a validation set.

We compared RADA\textsc{Grad} with ADA\textsc{Grad} without variance reduction. The results of this experiment can be seen in Figure 3. During training, we found that RADA\textsc{Grad} consistently outperforms ADA\textsc{Grad}: RADA\textsc{Grad} is able to both quicker reduce the training loss and also reaches a smaller value ($5.62 \times 10^{-4}$ vs. $1.52 \times 10^{-3}$, a 2.7× reduction in loss). Again, we found that the selected learning rate is an order of magnitude higher for RADA\textsc{Grad} than for ADA\textsc{Grad}. RADA\textsc{Grad} is able to exploit the fact that T-LSTMs perform type-preserving update steps which should preserve any low-rank structure present in the weight matrices. The relative improvement of RADA\textsc{Grad} over ADA\textsc{Grad} in training is also reflected in the test loss ($1.15 \times 10^{-2}$ vs. $3.23 \times 10^{-2}$, a 2.8× reduction).

\section{Discussion}

We have presented ADA-LR and RADA\textsc{Grad} which approximate the full proximal term of ADA\textsc{Grad} using fast, structured random projections. ADA-LR enjoys similar regret to ADA-FULL and both methods achieve similar empirical performance at a fraction of the computational cost. Importantly, RADA\textsc{Grad} can easily be modified to make use of standard improvements such as variance reduction. Using variance reduction in combination in particular has stark benefits for non-convex optimization in convolutional and recurrent neural networks. We observe a marked improvement over widely-used techniques such as ADA\textsc{Grad} and SVRG, the combination of which has recently been proven to be an excellent choice for non-convex optimization [1].

Furthermore, we tried to incorporate exponential forgetting schemes similar to RMS\textsc{Prop} and ADAM into the RADA\textsc{Grad} framework but found that these methods degraded performance. A downside of such methods is that they require additional parameters to control the rate of forgetting.

Optimization for deep networks has understandably been a very active research area. Recent work has concentrated on either improving estimates of second order information or investigating the effect of variance reduction on the gradient estimates. It is clear from our experimental results that a thorough
study of the combination provides an important avenue for further investigation, particularly where parts of the underlying model might have low effective rank.

Acknowledgements. We are grateful to David Balduzzi, Christina Heinze-Deml, Martin Jaggi, Aurelien Lucchi, Nishant Mehta and Cheng Soon Ong for valuable discussions and suggestions.

References


Supplementary Information for Scalable Adaptive Stochastic Optimization Using Random Projections

A Computational Complexity

Table A.1: Comparison of computational complexity in big O notation between ADA-FULL, ADA-LR and RADAGrad.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Line</th>
<th>ADA-FULL</th>
<th>ADA-LR</th>
<th>RADAGrad</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi g_t$</td>
<td>3</td>
<td>$p^2$</td>
<td>$p^2$</td>
<td>$p^3$</td>
</tr>
<tr>
<td>$G_t g_t^\top$</td>
<td>$p^2 \log \tau$</td>
<td>$p^2 \log \tau$</td>
<td>$p^2 \log \tau$</td>
<td>$p^2 \log \tau$</td>
</tr>
<tr>
<td>QR-decomp</td>
<td>5</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
</tr>
<tr>
<td>$Q^\top G_t$</td>
<td>6</td>
<td>$\tau p^2$</td>
<td>$\tau p^2$</td>
<td>$\tau p^2$</td>
</tr>
<tr>
<td>SVD</td>
<td>7</td>
<td>$p^3$</td>
<td>$\tau^2 p$</td>
<td>$\tau^3$</td>
</tr>
<tr>
<td>$Q W$</td>
<td>8</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
</tr>
<tr>
<td>$\beta_{t+1}$</td>
<td>10</td>
<td>$p^2$</td>
<td>$\tau p$</td>
<td>$\tau p$</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$p^3$</td>
<td>$\tau p^2$</td>
<td>$\tau^2 p$</td>
</tr>
</tbody>
</table>

B RADA-VR: RADAGrad with variance reduction.

Algorithm 3 RADA-VR

Input: $\eta > 0$, $\delta \geq 0$, $\tau$, $S$ number of epochs, $m$ iterations per epoch, initial $\beta_0$

1: for $s = 1 \ldots S$ do
2: $\mu = \nabla \sum_{i=1}^n f_i(\beta_0^s)$
3: for $t = 1 \ldots m - 1$ do
4: Compute VR gradient: $g_t = \nabla f_t(\beta_t^s) - \nabla f_t(\beta_0^s) + \mu$
5: Project: $g_t = Q g_t$
6: $G_t = G_{t-1} + g_t g_t^\top$
7: $Q_t, R_t \leftarrow qr\_update(Q_{t-1}, R_{t-1}, g_t, \tilde{g}_t)$
8: $B = G_t^\top Q$
9: $U, \Sigma, W = B \{SVD\}$
10: $V = W Q^\top$
11: $\beta_{t+1}^s = \beta_t^s - \eta V (\Sigma^{1/2} + \delta I)^{-1} V^\top g_t - \gamma_t$
12: end for
13: $\beta_t^{s+1} = \beta_{t+1}^s$
14: end for

Output: $\beta_m$

C Analysis

C.1 Regret bound for ADA-LR

The following proof is based on the proof for Theorem 7 in [10]. The key difference is that instead of having the square root and (pseudo-)inverse of the full matrix $G_t \cdot G_t^{1/2}$ and $S_t^i$ we have the approximate square root and inverse based on the randomized SVD [10]: $\tilde{S}_t = (Q Q^\top G_t)^{1/2}$ and $\tilde{S}_t^i = (Q Q^\top G_t)^{-1/2}$. Essentially we use the proximal function $\psi_t = \langle x, \tilde{S}_t x \rangle$ or $\psi_t = \langle x, \tilde{H}_t x \rangle$ where we set $\tilde{H}_t = \delta I + \tilde{S}_t$. Here $Q$ is the approximate basis for the range of the matrix $G_t$ [16].

We first state the following facts about the relationship between $G$ and $G^{-1/2}$.

11
Lemma 3. Defining \( \hat{G}^{-1/2} = (QQ^T G)^{-1/2} \) we have

(I) \( \hat{G}^{-1/2} G = (G^{-1}(QQ^T)G^{1/2}) \).

(II) \( \text{tr}(G^{-1}(QQ^T)G^{1/2}) = \text{tr}(\hat{G}^{1/2}) \).

We also require the following Lemma which bounds the sequence of proximal terms by the trace of the final \( G^{-1/2} \).

Lemma 4 (Based on Lemma 10 in [10]).

\[
\sum_{t=1}^{T} \langle g_t, \hat{G}^{-1/2}_t g_t \rangle \leq 2 \sum_{t=1}^{T} \langle g_t, \hat{G}^{-1/2}_t g_t \rangle = 2 \text{tr}(\hat{G}^{1/2}_T). \tag{3}
\]

We are now ready to prove Proposition 2.

Proof of Proposition 2. Inspecting Lemma 6:

\[
R(T) \leq \frac{1}{\eta} \psi_T(\beta^{\text{opt}}) + \frac{\eta}{2} \sum_{t=1}^{T} \| f'_t(\beta_t) \|_{\psi_{T-1}}^2,
\]

we first bound the term \( \sum_{t=1}^{T} \| f'_t(\beta_t) \|_{\psi_{T-1}}^2 \).

From [10, Proof of Theorem 7] we have that the squared dual norm associated with \( \psi_t \) is

\[
\| x \|_{\psi_t}^2 = \langle x, (\delta I + (QQ^T G_t)^{-1/2})^{-1} x \rangle
\]

and thus it is clear that \( \| g_t \|_{\psi_{T-1}}^2 \leq \langle g_t, (QQ^T G_t)^{-1/2} g_t \rangle \). Lemma 8 shows that \( \| g_t \|_{\psi_{T-1}}^2 \leq \langle g_t, \hat{S}_t g_t \rangle \) as long as \( \delta \geq \| g_t \|_2 \). Lemma 4 then implies that

\[
\sum_{t=1}^{T} \| f'_t(\beta_t) \|_{\psi_{T-1}}^2 \leq 2 \text{tr}(\hat{G}^{1/2}_T).
\]

We now bound \( 2 \text{tr}(\hat{G}^{1/2}_T) \) by \( 2(\text{tr}(G^{1/2}_T) + \tau \sqrt{\epsilon}) \):

\[
\text{tr}(\hat{G}^{1/2}_T) - \text{tr}(G^{1/2}_T) = \text{tr}(\hat{G}^{1/2}_T - G^{1/2}_T) \tag{4}
\]

\[
= \sum_{j=1}^{\tau} \left( \lambda_j(\hat{G}^{1/2}_T) - \lambda_j(G^{1/2}_T) \right) - \sum_{j=\tau+1}^{p} \lambda_j(G^{1/2}_T) \tag{5}
\]

\[
\leq \sum_{j=1}^{\tau} \left( \lambda_1(\hat{G}^{1/2}_T) - \lambda_1(G^{1/2}_T) \right) \tag{6}
\]

since \( \lambda_j(\hat{G}_T) = 0, \forall j > \tau \).

Now, using the reverse triangle inequality and Theorem 5 we obtain

\[
\sum_{j=1}^{\tau} \left( \lambda_1(\hat{G}^{1/2}_T) - \lambda_1(G^{1/2}_T) \right) \leq \sum_{j=1}^{\tau} \| \hat{G}^{1/2}_T - G^{1/2}_T \|_2 \tag{7}
\]

\[
\leq \sum_{j=1}^{\tau} \sqrt{\epsilon} \tag{8}
\]

\[
\leq \tau \sqrt{\epsilon} \tag{9}
\]
It remains to show that $\psi_T(\beta^{opt})$ in Lemma 6 is bounded by $\left(\delta + \sqrt{\epsilon + \text{tr}(G_{T}^{1/2})}\right)\|\beta^{opt}\|^2$ to get the statement of Theorem 2:

$$\psi_T(\beta^{opt}) = \langle \beta^{opt}, \delta I + (QQ^\top G_T)^{1/2} \beta^{opt} \rangle$$

$$\leq \|\beta^{opt}\|^2 \| (QQ^\top G_T)^{1/2} \|_2 + \delta \|\beta^{opt}\|^2$$

$$\leq \|\beta^{opt}\|^2 \left(\sqrt{\epsilon + \|G_T^{1/2}\|} \right) + \delta \|\beta^{opt}\|^2$$

where we again use the reverse triangle inequality and Theorem 3 as above.

Finally, plugging this into the statement of Lemma 6 and setting $\eta = \|\beta^{opt}\|_2$ (as in Corollary 11 in [10]) we get the expression for the regret of ADA-LR as stated in Theorem 2.

C.2 Proofs of supporting results

Proof of Lemma 3
By direct computation we have for (I)

$$\tilde{G}^{-1/2} G = (QQ^\top G)^{-1/2} G$$

$$= ((QQ^\top G)^{-1} G^2)^{1/2}$$

$$= (G^{-1}(QQ^\top)^{-1} G^2)^{1/2}$$

$$= (G^{-1}(QQ^\top)^2)^{1/2}.$$ 

and for (II)

$$\text{tr}((G^{-1}(QQ^\top)^2)^{1/2}) = \text{tr}((Q^\top GQ)^{1/2})$$

$$= \text{tr}((QQ^\top G)^{1/2})$$

$$= \text{tr}(\tilde{G}^{1/2}).$$

Proof of Lemma 4
We set up the following proof by induction. In the base case:

$$\langle g_1, \tilde{G}_1^{-1/2}, g_1 \rangle = \text{tr}(\tilde{G}_1^{-1/2} g_1 g_1^\top) = \text{tr}(\tilde{G}_1^{1/2}) \leq 2\text{tr}(\tilde{G}_1^{1/2}),$$

where we have used (II).

Now, assuming that the lemma is true for $T - 1$, we get:

$$\sum_{t=1}^{T} \langle g_t, \tilde{G}_t^{-1/2}, g_t \rangle \leq 2 \sum_{t=1}^{T} \langle g_t, \tilde{G}_{T-1}^{-1/2} g_t \rangle + \langle g_T, \tilde{G}_{T}^{-1/2} g_T \rangle.$$ 

Now using that $\tilde{G}_{T-1}^{-1/2}$ does not depend on $t$ and (II):

$$\sum_{t=1}^{T-1} \langle g_t, \tilde{G}_{T-1}^{-1/2} g_t \rangle = \text{tr}(\tilde{G}_{T-1}^{-1/2} G_{T-1}) = \text{tr}(\tilde{G}_{T-1}^{1/2}).$$

Therefore we get

$$\sum_{t=1}^{T} \langle g_t, \tilde{G}_t^{-1/2}, g_t \rangle \leq 2\text{tr}(\tilde{G}_{T-1}^{1/2}) + \langle g_T, \tilde{G}_{T}^{-1/2} g_T \rangle.$$ 

We can rewrite

$$\text{tr}(\tilde{G}_{T-1}^{1/2}) = \text{tr}((Q_{T-1} Q_{T-1}^\top G_T - Q_{T-1} Q_{T-1}^\top G_T g_T g_T^\top)^{1/2})$$

Now since $\text{range}(Q_{T-1}) \subset \text{range}(Q_T)$ and Proposition 8.5 in [16] we can use Lemma 7 with $\nu = 1$ and $g = g_t$ to obtain:

$$2\text{tr}(\tilde{G}_{T-1}^{1/2}) + \langle g_T, \tilde{G}_{T}^{-1/2}, g_T \rangle \leq 2\text{tr}(\tilde{G}_T^{1/2})$$
D Supporting Results

**Theorem 5** (SRFT approximation error (Theorem 11.2 in [16])). Defining \( \epsilon = \sqrt{1 + \frac{7p}{\tau} \cdot \sigma_{k+1}} \), the following holds with failure probability at most \( O \left( k^{-5} \right) \)

\[
\| G_t - QQ^T G_t \|_2 \leq \epsilon,
\]

where \( \sigma_{k+1} \) is the \( k \)th largest singular value of \( G_t \), and \( 4 \left[ \sqrt{k} + \sqrt{8 \log(kn)} \right]^2 \leq \tau \leq p \).

**Lemma 6** (Proposition 2 from [10]).

\[
R(T) := \sum_{t=1}^{T} f_t(\beta_t) + \varphi(\beta_t) - f_t(\beta^o) - \varphi(\beta^o) \leq \frac{1}{\eta} \psi_T(\beta^o) + \frac{\eta}{2} \sum_{t=1}^{T} \| f'_t(\beta_t) \|_2^2
\]

**Lemma 7** (Lemma 8 from [10]). Let \( B \succeq 0 \). For any \( \nu \) such that \( B - \nu gg^T \succeq 0 \) the following holds

\[
2\text{tr}((B - \nu gg^T)^{1/2}) \leq 2\text{tr}(B^{1/2}) - \nu \text{tr}(B^{1/2}gg^T)
\]

**Lemma 8** (Lemma 9 from [10]). Let \( \delta \geq \| g \|_2 \) and \( A \succeq 0 \), then

\[
\langle g, (\delta I + A^{1/2})^{-1} g \rangle \leq \langle g, ((A + gg^T)^{1/2})^{1/2} g \rangle
\]