RMSProp and equilibrated adaptive learning rates for non-convex optimization

Yann N. Dauphin YANN@DAUPHIN.IO
Harm de Vries MAIL@HARMDEVRIES.COM
Junyoung Chung ELECEGG@GMAIL.COM
Yoshua Bengio BENGIOY@IRO.MONTREAL.CA
Dept. IRO, Universit de Montral. Montral (QC), H3C 3J7, Canada

Abstract

Parameter-specific adaptive learning rate methods are computationally efficient ways to reduce the ill-conditioning problems encountered when training large deep networks. Following recent work that strongly suggests that most of the critical points encountered when training such networks are saddle points, we find how considering the presence of negative eigenvalues of the Hessian could help us design better suited adaptive learning rate schemes, i.e., diagonal preconditioners. We show that the optimal preconditioner is based on taking the absolute value of the Hessian's eigenvalues, which is not what Newton and classical preconditioners like Jacobi's do. In this paper, we propose a novel adaptive learning rate scheme based on the equilibration preconditioner and show that RMSProp approximates it, which may explain some of its success in the presence of saddle points. Whereas RMSProp is a biased estimator of the equilibration preconditioner, the proposed stochastic estimator, ESGD, is unbiased and only adds a small percentage to computing time. We find that both schemes yield very similar step directions but that ESGD sometimes surpasses RMSProp in terms of convergence speed, always clearly improving over plain stochastic gradient descent.

1. Introduction

One of the challenging aspects of deep learning is the optimization of the training criterion over millions of parameters: the difficulty comes from both the size of these neural networks and because the training objective is non-convex in the parameters. Stochastic gradient descent (SGD) has remained the method of choice for most practitioners of neural networks since the 80's, in spite of a rich literature in numerical optimization and it remains unclear how to best exploit second-order structure of the objective function when training deep networks. One of the questions regarding SGD is how to set the learning rate adaptively, both over time and for different parameters, and several methods have been proposed (Schaul et al., 2013), including the RMSProp method studied in this paper. Because of the large number of parameters, storing the full Hessian (or other full-rank preconditioner) or even a low-rank approximation is not practical, while these adaptive learning rate methods can be seen as estimating a diagonal preconditioner.

On the other hand, recent work (Dauphin et al., 2014; Choromanska et al., 2014) has brought theoretical and empirical evidence suggesting that local minima are with high probability not the main obstacle to optimizing large and deep neural networks, contrary to what was previously believed: instead, saddle points are the most prevalent critical points on the optimization path (except when we approach the value of the global minimum). These saddle points can considerably slow down training, mostly because the objective function tends to be flat in many directions and ill-conditioned in the neighborhood of these saddle points. This raises the question: can we take advantage of the saddle structure to design good and computationally efficient preconditioners? This suggests diagonal preconditioners, i.e. specifying how to set local learning rates adaptively.

In this paper, we bring these two threads together: we first show that RMSProp provides a biased estimator of the diagonal of the absolute value of the Hessian (Section 4), which would correspond to the equilibration preconditioner. We then show that when the objective function is non-convex, the inverse of the absolute value of the Hessian is the best possible preconditioner (Section 5), being as good or better than using the Hessian (Newton's method). Practically, we find that when there are negative eigenvalues of the Hessian, the equilibration preconditioner is much more efficient at improving the condition number. This last result also serves as a justification for the method presented by Dauphin et al. (2014), which however involved

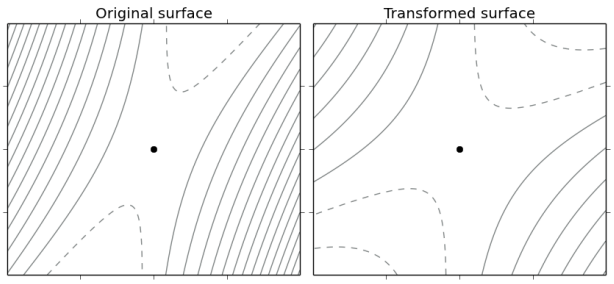

Figure 1. Contour lines before (left) and after equilibration preconditioning (right).

either the full Hessian or a low-rank approximation. Bringing these two contributions together justifies exploring estimators of the equilibration preconditioner such as RMSProp, and suggests that such preconditioners are particularly well-suited to efficiently improve the condition number when the objective function is non-convex, in the neighborhood of ill-conditioned saddle points. However, understanding the way in which RMSProp estimates the quantity of interest and the bias it incurs in doing so suggests that better estimators could be devised. Section 6 proposes such an unbiased alternative to estimate the equilibration preconditioner. In Section 7 we setup experiments to validate the above claims, and Section 8 presents the results, which confirm the claims. In addition, a surprising observation was made: these preconditioners yield trajectories that are very different from SGD; they go through much better behaved paths where the Hessian is dominated much more by its diagonal. This may be because they can tolerate more saturated hidden units, corresponding to stronger non-linearities, which is consistent with the observed ability of RMSProp and of the new stochastic equilibration variant, equilibrated SGD (ESGD), to train neural nets to a point that corresponds to strongly non-linear input-output relationships, compared to SGD training.

2. Preconditioning

The loss functions of neural networks notoriously exhibit pathological curvature (Martens, 2010). The loss surface may be highly curved in some directions while being flat in others. This is problematic for gradient descent since a single learning rate is not suitable for all search directions. A high learning rate will do well for directions of low curvature, but will diverge for those of high curvature. Some methods have recently been proposed to tune a separate learning rate for each parameter but they assume convexity of the function (Zeiler, 2012; Duchi et al., 2011). It is not clear that applying these methods to neural networks will produce good results.

Preconditioning is a geometric solution to the problem of pathological curvature that does not require convexity. Preconditioning aims to locally transform the optimization landscape so that its curvature is equal in all directions, as illustrated in Figure 1 for the equilibration preconditioner studied here. Consider the problem of minimizing the function over parameters $\Delta\theta \in \mathbb{R}^N$

$$f(\theta + \Delta\theta) = f(\theta) + \nabla f^T \Delta\theta + \Delta\theta^T \mathbf{H} \Delta\theta.$$

The critical points are the solutions to the secant equation $\nabla f(\theta + \Delta\theta) = \nabla f + \mathbf{H}\Delta\theta = 0$. It is clear that multiplying each side by a non-singular matrix $\mathbf{D}^{-1}$ does not change the solution. This transformed function has derivative $\mathbf{D}^{-1}\nabla f$ and Hessian $\mathbf{D}^{-1}\mathbf{H}$. The key idea of preconditioning is to choose a matrix $\mathbf{D}$ that reduces the condition number of the transformed Hessian $\mathbf{D}^{-1}\mathbf{H}$, such that a gradient descent update on this transformed surface

$$\theta^t = \theta^{t-1} + \mathbf{D}^{-1}\nabla f(\theta^{t-1}).$$

is close to the optimum of the quadratic approximation. One perfect preconditioning matrix is $\mathbf{H}^{-1}$ which gives us the famous Newton step. However, computing this matrix exactly is usually intractable for neural networks, and therefore diagonal approximations are typically preferred. Such diagonal preconditioners are commonly known as adaptive learning rates in the neural network literature.

The most well-known approximation is the Jacobi preconditioner where $\mathbf{H}$ is approximated by its diagonal

$$\mathbf{D}^{\text{Jacobi}} = |\text{diag}(\mathbf{H})|.$$

where $|\cdot|$ is the absolute value. There has been some success in applying this preconditioner to neural networks using a Gauss-Newton approximation of the Hessian (LeCun et al., 2012). However, the Hessian of neural networks are most likely indefinite and the Jacobi has not been found to be competitive for indefinite matrices (Bradley & Murray, 2011). The evidence for the presence of both positive eigenvalues and negative eigenvalues (i.e., for saddle points) on the training trajectory of deep neural networks is both theoretical and empirical (Dauphin et al., 2014; Choromanska et al., 2014). In this paper, we will investigate the qualities that make a good preconditioner for deep neural networks. As discussed in Section 5, preconditioners which approximate the diagonal of the Hessian are not appropriate in the neighborhood of saddle points, whereas the best preconditioners in this region involve taking the absolute value of the Hessian.

3. Equilibration

Equilibration is a type of preconditioning that has been found to work well for indefinite matrices (Bradley & Murray, 2011). A matrix $\mathbf{H}$ is row equilibrated if all its rows

are of equal norm. This can be achieved by the diagonal preconditioning matrix with entries

$$\mathbf{D}_i^{\text{Equilibration}} = \|\mathbf{H}_{i,\cdot}\|.$$

For the 2-norm, equilibration can be reformulated as

$$\mathbf{D}^{\text{Equilibration}} = \sqrt{\text{diag}(\mathbf{H}^2)}.$$

which reveals its relation to the absolute value of the Hessian: changing the order of the square root and diagonal gives us the diagonal of the absolute Hessian $\text{diag}(\sqrt{\mathbf{H}^2})$. As we discuss in Section 5, this link can help explain the success of equilibration for indefinite matrices, and clarifies the relation between equilibration and the Jacobi preconditioner. The key difference resides in the use of $\mathbf{H}^2$ to gather curvature information instead of $\mathbf{H}$. Like the absolute value of the Hessian, the squared Hessian does not have negative curvature, and this may allow equilibration to side-step some of the problems associated with non-convexity. One issue with indefinite preconditioning matrices is that they can transform directions of descent into directions of ascent.

The equilibration matrix can be estimated using the matrix-free estimator $(\mathbf{Hv})^2$ where the square is element-wise and $\mathbf{v} \sim \mathcal{N}(0, 1)$. As shown by Bradley & Murray (2011), this estimator is unbiased, i.e.,

$$\text{diag}(\mathbf{H}^2) = \text{E}[(\mathbf{Hv})^2]. \tag{1}$$

Since multiplying the Hessian by a vector can be done efficiently without ever computing the Hessian, this method can be efficiently used in the context of neural networks using the R-operator (Schraudolph, 2002). The R-operator computation only uses gradient-like computations and costs about the same as two backpropagations. We will adopt this estimator to propose a stochastic equilibration preconditioner for neural networks, called ESGD.

4. RMSProp approximates equilibration

In this section we uncover an unsuspected link between RMSProp and equilibration. RMSProp is an adaptative learning rate method that has found much success in practice (Tieleman & Hinton, 2012; Korjus et al.; Carlson et al., 2015). Tieleman & Hinton (2012) propose to normalize the gradients by an exponential moving average of the magnitude of the gradient for each parameter:

$$\mathbf{v}^t = \alpha \mathbf{v}^{t-1} + (1 - \alpha)(\nabla f)^2 \tag{2}$$

where $0 < \alpha < 1$ denotes the decay rate. The update step is given by

$$\theta^t = \theta^{t-1} + \epsilon \frac{\nabla f(\theta^{t-1})}{\sqrt{\mathbf{v}_t} + \lambda}.$$

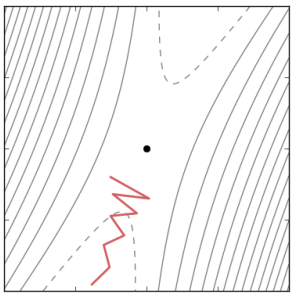

Figure 2. Gradient descent oscillates randomly around the critical point in the high curvature directions.

where ϵ is the learning rate and λ is a damping factor. Although it is observed in practice that normalizing gradients helps the optimization process, there was until this paper no published theoretical justification for RMSProp. Here, we show that RMSProp is in fact a biased estimator of the equilibration preconditioner.

We observe that the vector $\mathbf{v}_t$ estimates the expectation

$$\mathbf{v}^t \approx \text{E}[(\nabla f(\theta^t))^2] \tag{3}$$

We will see that this expectation gathers curvature information related to equilibration similarly to Equation 1. Consider the N-dimensional function

$$f(\theta + \Delta\theta) = f(\theta) + \nabla f^T \Delta\theta + \frac{1}{2}\Delta\theta^T \mathbf{H} \Delta\theta.$$

which we take to be quadratic for simplicity. A non-quadratic function will incur an approximation error. Differentiating this quantity by $\Delta\theta$ gives us the secant equation

$$\nabla f(\theta + \Delta\theta) = \nabla f(\theta) + \mathbf{H}\Delta\theta$$

which can be re-ordered to give us an expression for the product

$$\mathbf{H}\Delta\theta = \nabla f(\theta + \Delta\theta) - \nabla f(\theta).$$

Without loss of generality we take the root of the Taylor expansion to be the nearest critical point θ^*, i.e., with the property $\nabla f(\theta^*) = 0$, hence

$$\begin{aligned} \mathbf{H}\Delta\theta^{t*} &= \nabla f(\theta^* + \Delta\theta^{t*}) - \nabla f(\theta^*). \\ &= \nabla f(\theta^* + \Delta\theta^{t*}) \\ &= \nabla f(\theta^t) \end{aligned} \tag{4}$$

where $\Delta\theta^{t*} = \theta^t - \theta^*$ and θ^t is the value of the parameters at the timestep t. Note that this is a re-ordering of the familiar Newton step $\Delta\theta^{t*} = \mathbf{H}^{-1}\nabla f(\theta^t)$.

Combining this with the expression of RMSProp (Equation 3) we have $\mathbf{v}^t = \text{E}[(\nabla f(\theta^t))^2] = \text{E}[(\mathbf{H}\Delta\theta^{t*})^2]$. This is the form of the estimator for equilibration described in Equation 1. The main difference is that elements of the vector $\Delta\theta^{t*}$ may not be Gaussian distributed with zero mean

and unit variance, in which case, RMSProp will incur a bias in its estimation of the equilibration preconditioner. However, we have found the bias to be comparatively small in practice even for very high-dimensional problems (Figure 4). The reason we believe that elements of $\Delta\theta^{t*}$ are randomly distributed is two-fold. First, the sampling noise introduced by SGD lets RMSProp jump in random directions around the saddle point. Second, the nature of gradient descent also causes random fluctuations around the critical point θ^* as pictured in Figure 2. This effect is especially pronounced in directions of high curvature and it is the reason why methods like momentum can be effective. This same phenomenon is what helps RMSProp gather curvature information. Since $\Delta\theta^{t*} = \theta^t - \theta^*$ is centered at the critical point this causes $\Delta\theta^{t*}$ to oscillate roughly randomly. Thus RMSProp can be thought as a biased estimator of equilibration and this may explain some of its success in optimizing in the presence of negative eigenvalues of the Hessian, as is the case when training neural networks (Dauphin et al., 2014; Choromanska et al., 2014).

5. The absolute Hessian is better for non-convex problems

Here we will explain why using the absolute value of the Hessian instead of the Hessian is useful for non-convex problems. First, we show that the absolute value of the Hessian is the only symmetric positive definite ideal preconditioner. Second, we will show theoretically that the absolute value of the Hessian provides a better second order step than the Hessian. Third, we will show that when approximating the Hessian to obtain curvature information you will consistently underestimate the curvature in a given direction. This can lead to taking a completely wrong step. This problem is avoided completely by the absolute value of the Hessian.

5.1. The symmetric positive-definite preconditioner

A preconditioner $\mathbf{D}$ is called ideal if it maximally reduces the condition number $\kappa(\mathbf{HD}) = 1$. Another useful property for a preconditioner to have is to be positive definite. If the preconditioner is not positive definite then multiplying it by the gradient can actually flip the sign of the gradient. This can make the optimizer take steps that increase the objective instead of decreasing it. The absolute value of the Hessian is unique in that it is the only symmetric positive definite preconditioner.

Theorem 1. *Let $\mathbf{H}$ be a symmetric non-singular matrix in $\mathbb{R}^{N\times N}$. Then it holds that the matrix $|\mathbf{H}|^{-1}$ is the only symmetric positive definite ideal preconditioner.*

Proof. By definition the condition number is given by

$$\kappa(\mathbf{AM}) = \|(\mathbf{AM})^{-1}\|\|\mathbf{AM}\|.$$

where $\mathbf{M}$ is a symmetric preconditioning matrix and $\|\cdot\|$ is the Frobenius norm without loss of generality. We can recover the ideal preconditioning by minimizing the squared condition number over $\mathbf{M}$, giving us

$$\begin{aligned}&= \frac{\partial \text{tr}((\mathbf{M}^{-1}\mathbf{A}^{-1})^2)}{\partial \mathbf{M}}\|\mathbf{AM}\|^2 + \|\mathbf{M}^{-1}\mathbf{A}^{-1}\|^2\frac{\partial \text{tr}((\mathbf{A}^T\mathbf{M}^T)^2)}{\partial \mathbf{M}}\\ &= -\mathbf{M}^{-2}\mathbf{A}^{-2}\mathbf{M}^{-1}\|\mathbf{AM}\|^2 + \|\mathbf{M}^{-1}\mathbf{A}^{-1}\|^2\mathbf{MA}^2,\end{aligned}$$

Setting to zero on the left side we have

$$\begin{aligned}\mathbf{M}^{-2}\mathbf{A}^{-2}\mathbf{M}^{-1} &= \mathbf{MA}^2\|\mathbf{M}^{-1}\mathbf{A}^{-1}\|^2\|\mathbf{AM}\|^{-2}\\ \mathbf{M}^{-2}\mathbf{A}^{-2} &= \mathbf{MA}^2\mathbf{M}\|\mathbf{M}^{-1}\mathbf{A}^{-1}\|^2\|\mathbf{AM}\|^{-2}.\end{aligned}$$

$\mathbf{MA}^2\mathbf{M}$ is symmetric so we have $\mathbf{M}^{-2}\mathbf{A}^{-2} = \mathbf{A}^{-2}\mathbf{M}^{-2}$. This implies that $\mathbf{A}$ and $\mathbf{M}$ are commuting matrices and they must have the same eigenvectors. $\|\mathbf{M}^{-1}\mathbf{A}^{-1}\| = \|\mathbf{AM}\|^{-1} = \alpha$ is just a scaling factor. Leveraging this we have $\mathbf{M}^{-2}\mathbf{A}^{-2} = \mathbf{M}^2\mathbf{A}^2$. Proving $\mathbf{M} = \sqrt[4]{\mathbf{A}^{-4}} = |\mathbf{A}|^{-1}$, the other solutions all differ by sign flips of the absolute value of the eigenvalues. □

5.2. Comparison to the Newton step

The fact that the absolute value of the Hessian is positive definite and the Hessian may be indefinite directly affects optimization. The theoretical study of non-convex optimization is still very much an open question. It is not clear how to show a faster convergence rate for a non-convex problem. However, we can show that at each step the absolute value of the Hessian decreases the function equally or more than the Newton step. This difference arises because an indefinite Hessian will lead to the wrong step in directions of negative curvature. The result is given by Theorem 2.

Lemma 1. *Let $\mathbf{A}$ be a non-singular square matrix in $\mathbb{R}^{N\times N}$, and $\mathbf{x} \in \mathbb{R}^N$ be some vector. Then it holds that $\mathbf{x}^\top\mathbf{Ax} \leq \mathbf{x}^\top|\mathbf{A}|\mathbf{x}$, where $|\mathbf{A}|$ is the matrix obtained by taking the absolute value of each of the eigenvalues of $\mathbf{A}$.*

Proof. Let $\mathbf{q}_1, \dots \mathbf{q}_N$ be the different eigenvectors of $\mathbf{A}$ and $\lambda_1, \dots \lambda_N$ the corresponding eigenvalues. We have $\mathbf{x}^\top\mathbf{Ax} = \sum_i \lambda_i(\mathbf{x}^\top\mathbf{q}_i)^2$. It holds that $\sum_i \lambda_i(\mathbf{x}^\top\mathbf{q}_i)^2 \leq |\sum_i \lambda_i(\mathbf{x}^\top\mathbf{q}_i)^2|$ and by the triangular inequality $|\mathbf{x}^\top\mathbf{Ax}| \leq \sum_i |\lambda_i(\mathbf{x}^\top\mathbf{q}_i)^2|$ which is equal to $\mathbf{x}^\top|\mathbf{A}|\mathbf{x}$. □

Lemma 2. *Let $\mathbf{A}$ be a non-singular square matrix in $\mathbb{R}^{N\times N}$, and $\mathbf{v} \in \mathbb{R}^N$ be some vector. Then it holds that $\|\mathbf{Av}\| = \||\mathbf{A}|\mathbf{v}\|$, where $|\mathbf{A}|$ is the matrix obtained by taking the absolute value of each of the eigenvalues of $\mathbf{A}$.*

Proof. The norm is given by $\|\mathbf{A}\mathbf{v}\|^2 = \|\sum_i^N \lambda_i \mathbf{q}_i \mathbf{q}_i^T \mathbf{v}\|^2$ where λ_i is the eigenvalue of the eigenvector $\mathbf{q}_i$ of $\mathbf{A}$. The eigenvectors are orthogonal thus $\|(\sum_i^N \lambda_i \mathbf{q}_i \mathbf{q}_i^T \mathbf{v})^T (\sum_i^N \lambda_i \mathbf{q}_i \mathbf{q}_i^T \mathbf{v})\|^2$ simplifies to $\sum_i^N \|\lambda_i (\mathbf{q}_i \mathbf{q}_i^T)\mathbf{v}\|^2$. We can pull the eigenvalues out of the squared norms giving us $\sum_i^N \lambda_i^2 \|(\mathbf{q}_i \mathbf{q}_i^T)\nabla f\|^2$. Squaring the eigenvalues is invariant to the sign, which proves that the proposition. □

Theorem 2. *Let the matrix $\mathbf{H} \in \mathbb{R}^{N \times N}$ be the Hessian of the function f with gradient ∇f. Then it holds that taking a gradient step with $|\mathbf{H}|^{-1}$ minimizes the function equally or better than the Newton step, that is $f(\theta - |\mathbf{H}|^{-1}\nabla f) \leq f(\theta - \mathbf{H}^{-1}\nabla f)$.*

Proof. We can express any function f using its Taylor expansion

$$f(\theta + \Delta\theta) = f(\theta) + \nabla f^T \Delta\theta + \frac{1}{2}\Delta\theta^T \mathbf{H} \Delta\theta + o(\|\Delta\theta\|^3).$$

The Newton step for this function is given by $\Delta\theta_N = -\mathbf{H}^{-1}\nabla f$ and the step with the absolute Hessian is $\Delta\theta_A = -|\mathbf{H}|^{-1}\nabla f$. We can recover the error after the Newton step by taking $\Delta\theta = \Delta\theta_N$ in the previous equation

$$f(\theta+\Delta\theta_N) = f(\theta)+\nabla f^T \Delta\theta_N + \frac{1}{2}\Delta\theta_N^T \mathbf{H} \Delta\theta_N + o(\|\Delta\theta_N\|^3).$$

From Lemma 2 we have $o(\|\Delta\theta_N\|^3) = o(\|\Delta\theta_A\|^3)$, and we observe that

$$\begin{aligned}
\Delta\theta_N^T \mathbf{H} \Delta\theta_N &= \Delta\theta^T \mathbf{H}^{-1}\mathbf{H}\mathbf{H}^{-1}\Delta\theta \\
&= \Delta\theta^T \mathbf{H} \Delta\theta \\
&= \Delta\theta^T |\mathbf{H}|^{-1}\mathbf{H}|\mathbf{H}|^{-1}\Delta\theta \\
&= \Delta\theta_A^T \mathbf{H} \Delta\theta_A.
\end{aligned}$$

This gives us

$$f(\theta+\Delta\theta_N) = f(\theta)+\nabla f^T \Delta\theta_N + \frac{1}{2}\Delta\theta_A^T \mathbf{H} \Delta\theta_A + o(\|\Delta\theta_A\|^3).$$

Furthermore, it follows from Lemma 1 that

$$\begin{aligned}
\nabla f^T \Delta\theta_N &= -\Delta\theta^T \mathbf{H}^{-1}\Delta\theta \\
&\geq -\Delta\theta^T |\mathbf{H}|^{-1}\Delta\theta \\
&= \nabla f^T \Delta\theta_A.
\end{aligned}$$

Then it must hold that $f(\theta - |\mathbf{H}|^{-1}\nabla f) \leq f(\theta - \mathbf{H}^{-1}\nabla f)$. □

This formalizes the intuitions of (Dauphin et al., 2014). Thus the use of the absolute Hessian is justified for second order methods. As we shall see this also extends to even crude approximations of the second order step.

5.3. Better approximation of the curvature

As argued below, Krylov subspace approximations based on indefinite Hessians will tend to underestimate curvature. Therefore, even for approximate methods it is preferable to work with a quantity close to the absolute Hessian. Take the diagonal Jacobi approximation

$$\mathbf{D}^{\text{Jacobi}} = \sqrt{\text{diag}(\mathbf{H})^2}.$$

The elements $\mathbf{D}_i^{\text{Jacobi}}$ estimate the curvature of the euclidean axis $\mathbf{V} = \text{I}$. More precisely, they are equivalent to the Raleigh quotient in the direction of each axis direction $\mathbf{D}_i^{\text{Jacobi}} = |R(\mathbf{H}, \mathbf{V}_i)| = |\sum_j^N \lambda_j \mathbf{q}_{ji}^2|$ where λ_i and $\mathbf{q}_i$ are the eigenvalues and eigenvectors of $\mathbf{H}$. We can see that the terms associated with negative eigenvalues λ_i will cancel the positive terms. This can gravely underestimate the curvature and lead to bad preconditioning near saddle points. Specifically, this makes high curvature directions have even more curvature. This is problematic because of the proliferation of saddle points in neural networks (Dauphin et al., 2014).

This problem is avoided by equilibration methods like RMSProp. The curvature information

$$\mathbf{D}^{\text{Equilibration}} = \sqrt{\text{diag}(\mathbf{H}^2)}$$

is given by the Raleigh quotient of the squared Hessian $\mathbf{D}^{\text{Equilibration}} = \sqrt{R(\mathbf{H}^2, \mathbf{V}_i)} = \sqrt{\sum_j \lambda_j^2 \mathbf{q}_{ji}^2}$. Thus the eigenvalues are all positive and will not cancel, ensuring that the preconditioner will not enlarge already high curvature directions. We observe experimentally that the elements in the Jacobi matrix are much closer to 0 than the equilibration matrix. This suggests that negative curvature does cancel out other curvatures in practice. This explains why equilibration has been found to work well for indefinite matrices (Bradley & Murray, 2011).

As a first step, we have verified this claim experimentally for random neural networks. The neural networks have 1 hidden layer of a 100 sigmoid units with zero mean unit-variance Gaussian distributed inputs, weights and biases. The output layer is a softmax with the target generated randomly. We also give results for similarly sampled logistic regressions. We compare the capacity of the Jacobi and equilibration preconditioners to reduce the condition number of the Hessian. Reducing the condition number of the Hessian makes optimization much faster and it has been found to be very important in practice. In the convex case, the condition number is tied directly to the convergence rate (Wright & Nocedal, 1999). LeCun et al. (2012) proposed a Gauss-Newton approximation to the Jacobi preconditioner for neural networks. However, it is arguably less successful in the literature than RMSProp. We will see that this can be attributed to the better preconditioning properties given by

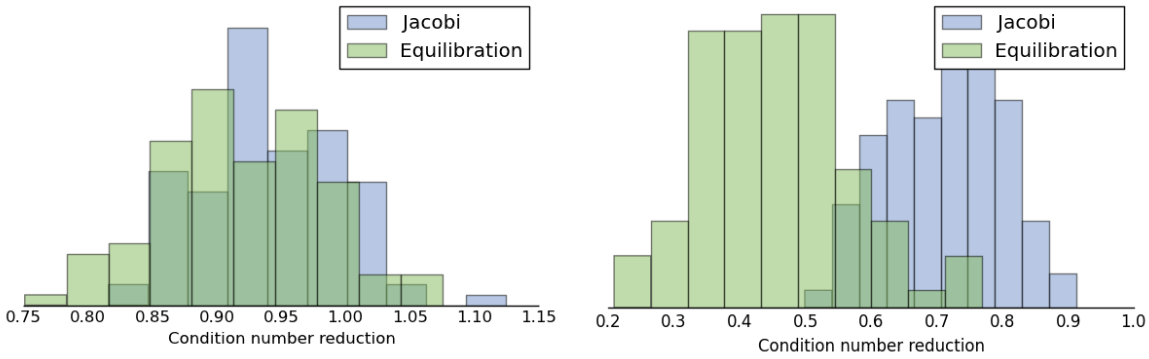

Figure 3. Histogram of the reduction of the condition number for random Hessians (lower is better). On the left are positive-definite Hessians of logistic regressions and on the right indefinite Hessians of neural networks. Equilibration makes a much bigger difference in the non-convex case. This confirms that approximating the information from the absolute value of the Hessian leads to better results.

approximating the absolute Hessian. Figure 3 gives the histograms of the condition number reductions. We obtained these graphs by sampling a hundred networks and computing the ratio of the condition number before and after preconditioning. On the left we have the convex case, and on the right the non-convex case. We clearly observe that the Jacobi and equilibration method are closely matched for the convex case. However, in the non-convex case the Jacobi does not properly handle negative curvature and we see a striking difference between the two methods. This confirms that properly addressing the negative curvature that arises near saddle points is beneficial. As we will see in Section 8 these results extend to practical high-dimensional problems.

6. Implementation

We propose to build a scalable algorithm for preconditioning neural networks using equilibration. This method will estimate the same curvature information $\sqrt{\text{diag}(\mathbf{H}^2)}$ with the unbiased estimator described in Equation 1. It is prohibitive to compute the full expectation at each learning step. Instead we will simply update our running average at each learning step much like RMSProp. The pseudo-code is given in Algorithm 1. The cost of this is one product with the Hessian which is roughly the cost of two additional gradient calculations and the cost of sampling a vector from a random Gaussian. In practice we greatly amortize the cost by only performing the update every 20 iterations. This brings the cost of equilibration very close to that of regular SGD. The only added hyper-parameter is the damping λ. We find that a good setting for that hyper-parameter is $\lambda = 10^{-2}$ and it is robust over the tasks we considered.

In the interest of comparison, we will evaluate SGD preconditioned with the Jacobi preconditioner. This will allow us to verify the claims that the equilibration preconditioner is better suited for non-convex problems. Bekas et al. (2007) show that the diagonal of a matrix can be recovered by the expression

$$\text{diag}(\mathbf{H}) = \mathrm{E}[\mathbf{v} \odot \mathbf{H}\mathbf{v}] \tag{5}$$

where $\mathbf{v}$ are random vectors with entries ± 1 and $\odot$ is the element-wise product. We use this estimator to precondition SGD in the same fashion as that described in Algorithm 1. The computational complexity is the same as equilibrated SGD.

Algorithm 1 Equilibrated Gradient Descent

Require: Function $f(\theta)$ to minimize, learning rate ϵ and damping factor λ
$\mathbf{D} \leftarrow 0$
for $i = k \rightarrow K$ **do**
$\mathbf{v} \sim \mathcal{N}(0, 1)$
$\mathbf{D} \leftarrow \mathbf{D} + (\mathbf{H}\mathbf{v})^2$
$\theta \leftarrow \theta - \epsilon \frac{\nabla f(\theta)}{\sqrt{\mathbf{D}/k + \lambda}}$
end for

7. Experimental setup

We aim to confirm experimentally the theoretical results proposed in the previous sections. First, we measure how well RMSProp estimates the equilibration matrix $\sqrt{\text{diag}(\mathbf{H}^2)}$. Second, we evaluate the equilibrated SGD proposed in Algorithm 1. Finally, we want to confirm that there is a significant difference in performance between the Jacobi preconditioner and equilibration methods for high-dimensional non-convex problems.

In these experiments, we consider the challenging opti-

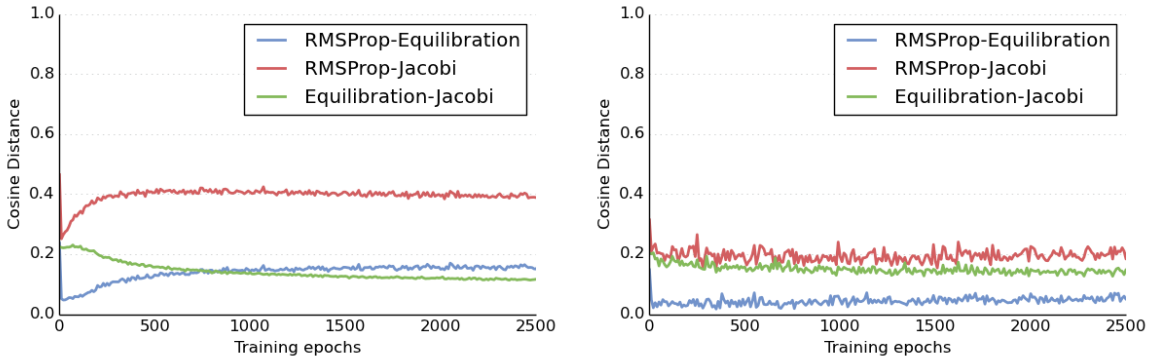

Figure 4. Cosine distance between the diagonals estimated by each method during the training of a deep auto-encoder trained on MNIST (left) and CURVES (right). We can see that RMSProp successfully estimates $\sqrt{\text{diag}(\mathbf{H})^2}$.

mization benchmark of training very deep neural networks. Following (Martens, 2010; Sutskever et al., 2013; Vinyals & Povey, 2011), we train deep auto-encoders which have to reconstruct their input under the constraint that one layer is very low-dimensional. This makes the reconstruction task difficult because it requires the optimizer to finely tune the parameters. The low-dimensional bottleneck layer learns a low-dimensional map similar in principle to non-linear Principal Component Analysis. The networks have up to 11 layers of sigmoidal hidden units and have on the order of a million parameters. We use the standard network architectures described in (Martens, 2010) for the MNIST and CURVES dataset. Both of these datasets have 784 input dimensions and 60,000 and 20,000 examples respectively.

We tune the hyper-parameters of the optimization methods with random search. We have sampled the learning rate from a logarithmic scale between $[0.1, 0.01]$ for stochastic gradient descent (SGD) and equilibrated SGD (ESGD). The learning rate for RMSProp and the Jacobi preconditioner are sampled from $[0.001, 0.0001]$. We note that the learning rate for RMSProp is smaller than ESGD because the RMSProp vector is multiplied by the small variance $\mathrm{E}[(\Delta\theta_j^{t*})^2]$. The damping factor λ used before dividing the gradient is taken from either $\{10^{-4}, 10^{-5}, 10^{-6}\}$ while the exponential decay rate of RMSProp is taken from either $\{0.9, 0.95\}$. The networks are initialized using the sparse initialization described in (Martens, 2010). We initialize 15 connections per neuron with a zero mean unit variance Gaussian and the others are set to zero. The minibatch size for all methods in 200. We do not make use of momentum in these experiments in order to evaluate the strength of each preconditioning method on its own. Similarly we do not use any regularization because we are only concerned with optimization performance. For these reasons, we report training error in our graphs.

The networks and algorithms were implemented using Theano (Bastien et al., 2012), simplifying the use of the R-operator in Jacobi and equilibrated SGD. All experiments were run on GPU's.

8. Results

8.1. Measuring the bias of RMSProp

In Section 4 we argued that under certain conditions RMSProp estimates the diagonal of the absolute Hessian. In this section we verify to what extent these conditions are valid, and experimentally determine the bias of RMSProp in estimating the equilibration matrix. We train deep autoencoders with RMSProp and measure every 10 epochs the equilibration matrix $\mathbf{D}^{\text{Equilibration}} = \sqrt{\text{diag}(\mathbf{H}^2)}$ and Jacobi matrix $\mathbf{D}^{\text{Jacobi}} = \sqrt{\text{diag}(\mathbf{H})^2}$ using 100 samples of the unbiased estimators described in Equations 1, respectively. We then measure the pairwise differences between these quantities in terms of the cosine distance $\text{cosine}(u, v) = 1 - \frac{u \cdot v}{\|u\|\|v\|}$, which measures the angle between two vectors and ignores their norms. Note further that the cosine distance ranges from 0 (zero degree angle) to 2 (180 degrees angle), and that the cosine distance between two preconditioners should not exceed one for preconditioned SGD to converge.

Figure 4 shows the resulting cosine distances over training on MNIST and CURVES. For the latter dataset we observe that RMSProp remains remarkably close (around 0.05) to equilibration, while it is significantly different from Jacobi (in the order of 0.2). The same order of difference is observed when we compare equilibration and Jacobi, confirming the observations of Section 5 that both quantities are rather different in practice. For the MNIST dataset we see that RMSProp fairly well estimates $\sqrt{\text{diag}(\mathbf{H})^2}$ in the beginning of training, but then quickly diverges. After 1000

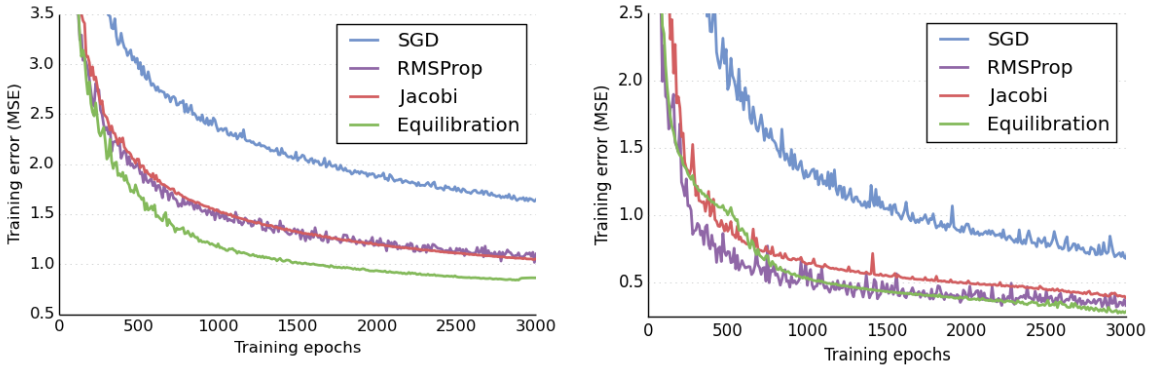

Figure 5. Learning Curves for deep auto-encoders on MNIST and CURVES comparing the different preconditioned SGD methods.

epochs this difference has exceeded the difference between Jacobi and equilibration, and RMSProp no longer successfully estimates the equilibration. Interestingly, at the same time that RMSProp starts diverging, we observe in Figure 5 that also the performance of the optimizer drops in comparison to ESGD. This suggests that the success of RMSProp as a optimizer is tied to its success in estimating the equilibration matrix.

8.2. Comparison of preconditioned SGD methods

We present the optimization curves of SGD and its preconditioned variants in Figure 5. We observe that the preconditioned methods beat SGD on both problems. Our results for MNIST show that the proposed equilibrated SGD significantly outperforms both RMSProp and Jacobi SGD, and the difference in performance becomes especially notable after 250 epochs. The gap with classical SGD is even more substantial, and we observe a convergence speed that is approximately three times faster. ESGD also performs best for CURVES, although the difference with RMSProp and Jacobi SGD is not as significant as for MNIST. These results correlate very well with the observation in the previous section. Figure 4 shows that the Jacobi is closer to equilibration on CURVES compared to MNIST. This explains their closer performance during optimization and may be because the Hessian on CURVES has less negative curvature. These graphs also justify the use of RMSProp as a preconditioning method.

8.3. Measuring diagonal dominance

We also measured the degree of diagonal dominance

$$d(\mathbf{H}) = \frac{\sqrt{\sum_i H_{ii}^2}}{\|\mathbf{H}\|_F} \cdot 100 \qquad (6)$$

of the Hessian during training. Fig. 6 shows the results on MNIST for the preconditioned SGD methods. Remarkably, there is a significant degree of diagonal dominance. It exceeds 15% for all optimization methods, which is bigger than we would expect from a random symmetric matrix. For example, if all elements of a matrix were independently drawn from a distribution with zero mean and unit variance, then the expected degree of diagonal dominance is $E[d(\mathbf{H})] = \frac{1}{\sqrt{N}} \cdot 100$ where N is the dimension of the parameters. For the million parameters of our deep autoencoders, the expected degree of diagonal dominance is only 0.01%. The Hessian is thus considerably diagonally dominant, which is good news for diagonally preconditioned SGD methods, since most ill-conditioning can then be removed by the diagonal of the absolute Hessian.

Another surprising observation from Fig. 6 is that the preconditioned SGD methods follow a very different optimization trajectory than SGD. They have tendency to move to part of the parameter space where the Hessian is much more diagonally dominant. We speculate that preconditioned SGD methods, in contrast to standard SGD, are still able to optimize in regions of parameter space which correspond to many saturated hidden units.

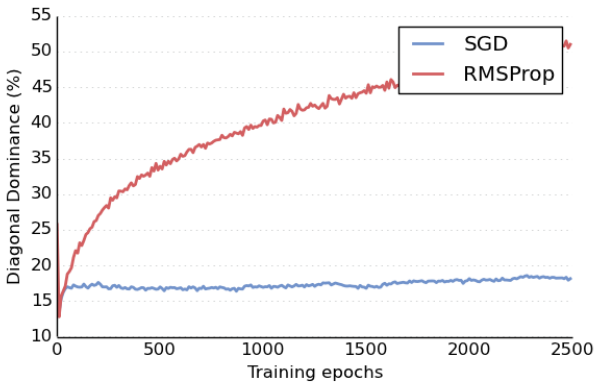

Figure 6. Degree of diagonal dominance during training for a deep auto-encoder on MNIST. The preconditioned SGD methods have a tendency to move to parts of the paremter space where the Hessian becomes more diagonally dominant.

9. Conclusion

We have proposed a new preconditioning method for neural networks dubbed equilibrated SGD (ESGD), which we found to be closely related to RMSProp. We have shown that this method approximates the absolute value of the Hessian and enjoys better performance on non-convex problems.

References

Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian J., Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

Bekas, Costas, Kokiopoulou, Effrosyni, and Saad, Yousef. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11):1214–1229, 2007.

Bradley, Andrew M and Murray, Walter. Matrix-free approximate equilibration. *arXiv preprint arXiv:1110.2805*, 2011.

Carlson, David, Cevher, Volkan, and Carin, Lawrence. Stochastic spectral descent for restricted boltzmann machines. 2015.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Grard Ben, and LeCun, Yann. The loss surface of multilayer networks, 2014.

Dauphin, Yann, Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, Ganguli, Surya, and Bengio, Yoshua. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS'2014*, 2014.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

Korjus, Kristjan, Kuzovkin, Ilya, Tampuu, Ardi, and Pungas, Taivo. Replicating the paper playing atari with deep reinforcement learning[mks.

LeCun, Yann A, Bottou, Léon, Orr, Genevieve B, and Müller, Klaus-Robert. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Martens, J. Deep learning via Hessian-free optimization. In *ICML'2010*, pp. 735–742, 2010.

Schaul, Tom, Antonoglou, Ioannis, and Silver, David. Unit tests for stochastic optimization. *arXiv preprint arXiv:1312.6055*, 2013.

Schraudolph, Nicol N. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

Sutskever, Ilya, Martens, James, Dahl, George, and Hinton, Geoffrey. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

Vinyals, Oriol and Povey, Daniel. Krylov subspace descent for deep learning. *arXiv preprint arXiv:1111.4259*, 2011.

Wright, Stephen J and Nocedal, Jorge. *Numerical optimization*, volume 2. Springer New York, 1999.

Zeiler, Matthew D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.