On the saddle point problem for non-convex optimization

Razvan Pascanu
Université de Montréal
r.pascanu@gmail.com

Yann N. Dauphin
Université de Montréal
dauphiya@iro.umontreal.ca

Surya Ganguli
Stanford University
sganguli@standford.edu

Yoshua Bengio
Université de Montréal
CIFAR Fellow
yoshua.bengio@umontreal.ca

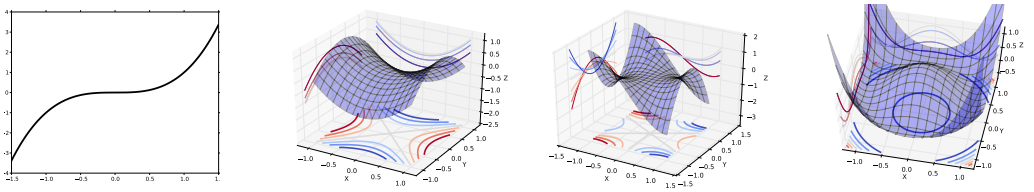
Abstract

A central challenge to many fields of science and engineering involves minimizing non-convex error functions over continuous, high dimensional spaces. Gradient descent or quasi-Newton methods are almost ubiquitously used to perform such minimizations, and it is often thought that a main source of difficulty for the ability of these local methods to find the global minimum is the proliferation of local minima with much higher error than the global minimum. Here we argue, based on results from statistical physics, random matrix theory, and neural network theory, that a deeper and more profound difficulty originates from the proliferation of saddle points, not local minima, especially in high dimensional problems of practical interest. Such saddle points are surrounded by high error plateaus that can dramatically slow down learning, and give the illusory impression of the existence of a local minimum. Motivated by these arguments, we propose a new algorithm, the saddle-free Newton method, that can rapidly escape high dimensional saddle points, unlike gradient descent and quasi-Newton methods. We apply this algorithm to deep neural network training, and provide preliminary numerical evidence for its superior performance.

1 Introduction

It is often the case that our geometric intuition, derived from our experience within a low dimensional physical world, is woefully inadequate for thinking about the geometry of typical error surfaces in high-dimensional spaces. Consider, for example, minimizing a typical, randomly chosen error function of a single scalar variable. More precisely, the error function could be a single draw of a Gaussian process (Rasmussen and Williams, 2005). Such a random error function would, with high probability over the choice of function, have many local minima (maxima), in which the gradient vanishes and the second derivative is negative (positive). However, it is highly unlikely to have a saddle point (see Figure 1 (a)), in which the gradient *and* the second derivative vanish. Indeed, such saddle points, being a degenerate condition, would occur with probability zero. Similarly, typical, random error functions on higher dimensional spaces of N variables are likely to have many local minima for very small N. However, as we review below, as the dimensionality N increases, local minima with high error relative to the global minimum occur with a probability that is exponentially small in N.

In general, consider an error function $f(\theta)$ where θ is an N dimensional continuous variable. A critical point is by definition a point θ where the gradient of $f(\theta)$ vanishes. All critical points of



(a) (b) (c) (d)

Figure 1: Illustrations of three different types of saddle points (a-c) plus a gutter structure (d). Note that for the gutter structure, any point from the circle $x^2 + y^2 = 1$ is a minimum. The shape of the function is that of the bottom of a bottle of wine. This means that the minimum is a "ring" instead of a single point. The Hessian is singular at any of these points. (c) shows a Monkey saddle where you have both a min-max structure as in (b) but also a 0 eigenvalue, which results, along some direction, in a shape similar to (a).

$f(\theta)$ can be further characterized by the curvature of the function in its vicinity, as described by the eigenvalues of the Hessian. Note that the Hessian is symmetric and hence the eigenvalues are real numbers. The following are the four possible scenarios:

- If all eigenvalues are non-zero and positive, then the critical point is a local minimum.
- If all eigenvalues are non-zero and negative, then the critical point is a local maximum.
- If the eigenvalues are non-zero and we have both positive and negative eigenvalues, then the critical point is a saddle point with a *min-max* structure (see Figure 1 (b)). That is, if we restrict the function f to the subspace spanned by the eigenvectors corresponding to positive (negative) eigenvalues, then the saddle point is a maximum (minimum) of this restriction.
- If the Hessian matrix is singular, then the *degenerate* critical point can be a saddle point, as it is, for example, for $\theta^3, \theta \in \mathbb{R}$ or for the monkey saddle (Figure 1 (a) and (c)). If it is a saddle, then, if we restrict θ to only change along the direction of singularity, the restricted function does not exhibit a minimum nor a maximum; it exhibits, to second order, a plateau. When moving from one side to other of the plateau, the eigenvalue corresponding to this picked direction generically changes sign, being exactly zero at the critical point. Note that an eigenvalue of zero can also indicate the presence of a gutter structure, a degenerate minimum, maximum or saddle, where a set of connected points are all minimum, maximum or saddle structures of the same shape and error. In Figure 1 (d) it is shaped as a circle. The error function looks like the bottom of a wine bottle, where all points along this circle are minimum of equal value.

A plateau is an almost flat region in some direction. This structure is given by having the eigenvalues (which describe the curvature) corresponding to the directions of the plateau be *close to 0*, but *not exactly 0*. Or, additionally, by having a large discrepancy between the norm of the eigenvalues. This large difference would make the direction of "relative" small eigenvalues look like flat compared to the direction of large eigenvalues.

Saddle points are unstable under gradient descent dynamics on the error surface, as the dynamics is repelled away from the saddle by directions of negative curvature. However, this repulsion can occur slowly due to plateaus of small negative eigenvalues. A similar slow-down can occur for local minima with small positive eigenvalues. Second order methods were designed to rapidly descend local minima by rescaling gradient steps by the inverse eigenvalues. However, the Newton method does not treat saddle points appropriately; as argued below, saddle-points instead become attractive under the Newton dynamics.

Thus given the proliferation of saddle points, not local minima, in high dimensional problems, the entire theoretical justification for quasi-Newton methods, i.e. the ability to rapidly descend to the bottom of a convex local minimum, becomes less relevant in high dimensional nonconvex optimization. In this work, we propose an algorithm whose theoretical justification is motivated by ensuring rapid escape from saddle points. This algorithm leverages second order-curvature information in

a fundamentally different way than quasi-Newton methods, and also, in preliminary results, outperforms them in some high dimensional problems.

1.1 The prevalence of saddle points

Here we review arguments from disparate literatures suggesting that saddle points, not local minima, provide a fundamental impediment to rapid high dimensional non-convex optimization. One line of evidence comes from statistical physics, where the nature of critical points of random Gaussian error functions on high dimensional continuous domains is studied Bray and Dean (2007); Fyodorov and Williams (2007) using replica theory, a theoretical technique for analyzing high dimensional systems with *quenched disorder* like spin glasses (see Parisi (2007) for a recent review). In particular, Bray and Dean (2007) counted the typical number of critical points of a random function in a finite cubic volume in N dimensional space within a range of error ϵ and index α. By definition the index α is the fraction of negative eigenvalues of the Hessian at the critical point. Of course every such function has a unique global minimum at $\epsilon = \epsilon_{\text{min}}$ and $\alpha = 0$ and a global maximum at $\epsilon = \epsilon_{\text{max}}$ and $\alpha = 1$, where ϵ_{min} and ϵ_{max} do not depend strongly on the detailed realization of the random function due to concentration of measure. In Bray and Dean (2007), the authors found that any such function, over a large enough volume, has exponentially many critical points, but in the $\epsilon - \alpha$ plane all the critical points are overwhelmingly likely to be located on a monotonically increasing curve $\epsilon^*(\alpha)$ that rises from ϵ_{min} to ϵ_{max} as α ranges from 0 to 1. Indeed the probability that a critical point lies an $O(1)$ distance off this curve, both over the choice of a critical point for a given function and over the choice of random function from the Gaussian ensemble, is exponentially small in the dimensionality N, for large N.

Intuitively, these theoretical results indicate that critical points at any intermediate error ϵ above the global minimum ϵ_{min} are exponentially likely to be saddle points, with the fraction of negative eigenvalues α of the Hessian monotonically increasing with ϵ. Furthermore, any critical point with a very small fraction of negative eigenvalues is exponentially likely to occur at low error ϵ close to ϵ_{min}. In particular, any local minimum with $\alpha = 0$ will have an error exceedingly close to that of the global minimum. Thus the optimization landscape of a random Gaussian error function has no local minima with high error, but is instead riddled with exponentially many saddle points at errors far above that of the global minimum error.

These results can be further understood via random matrix theory. Indeed, for a random error function in N dimensions, the Hessian at a critical point at error ϵ can be thought of as an N by N random symmetric matrix whose eigenvalue distribution depends on ϵ. Bray and Dean (2007) found that the entire eigenvalue distribution of the Hessian took the form of Wigner's famous semi-circular law (Wigner, 1958), but shifted by an amount determined by ϵ. Indeed, a completely unconstrained random symmetric matrix has a symmetric eigenvalue density on the real axis shaped like a semicircle with both mode and mean at 0. Thus any eigenvalue is positive or negative with probability $1/2$. The eigenvalue distribution of the Hessian of the critical point at error ϵ is a shifted semicircle, where the shift ensures that the fraction of negative eigenvalues α is given exactly by the solution to $\epsilon = \epsilon^*(\alpha)$. When the error $\epsilon = \epsilon_{\text{min}}$, the semicircular distribution of the Hessian is shifted so far to the right that all eigenvalues are positive, corresponding to the global minimum. As the error ϵ of the critical point increases, the semi-circular eigenvalue distribution shifts to the left, and the fraction of negative eigenvalues α increases. At intermediate error ϵ, half way between ϵ_{min} and ϵ_{max}, the semicircular distribution of eigenvalues has its mode at 0. This implies that the highest density of eigenvalues occurs near 0, and so a typical critical point at intermediate error not only has many negative curvature directions, but also many approximate plateau directions, in which a finite fraction of eigenvalues of the Hessian lie near 0. The existence of these approximate plateau directions, in addition to the negative directions, would of course have significant implications for high dimensional non-convex optimization, in particular, dramatically slowing down gradient descent dynamics. Moreover, Fyodorov and Williams (2007) give a review of work in which qualitatively similar results are derived for random error functions superimposed on a quadratic error surface.

Before continuing, we note that the random matrix perspective concisely and intuitively crystalizes the striking difference between the geometry of low and high dimensional error surfaces. For $N = 1$, the Hessian of a random function is a single random number, and so with overwhelming probability it will be positive or negative; the event that it is 0 is a set of measure zero. This reflects the intuition that saddle points in 1 dimension are extremely unlikely, while maxima and minima always occur.

Alternatively, an unconstrained random Gaussian matrix in N dimensions has a probability $O(e^{-N})$ that all of its eigenvalues are positive. This fact reflects that local minima with error far higher than the global minima are exponentially unlikely. The Hessian at a critical point with error very close to the global minimum is not a fully unconstrained random Gaussian matrix; the fact that the error is so small, shifts its eigenvalue distribution to the right, so that more eigenvalues are positive Bray and Dean (2007); Fyodorov and Williams (2007).

Thus the above work indicates that for typical, generic functions chosen from a random Gaussian ensemble of functions, local minima with high error are exponentially rare in the dimensionality of the problem, but saddle points with many negative and approximate plateau directions are exponentially likely at high error. However, are the error landscapes of practical problems of interest somehow not reflective of generic error landscapes, and therefore not riddled with similar saddle points? Although our proposed algorithm described below should be generically useful for a wide variety of problems, given that our proximal motivation is ultimately training deep neuronal networks, we review evidence from that literature that saddle points also play a prominent role in the learning dynamics.

In Baldi and Hornik (1989) the error surface of a single hidden layer MLP with linear units is analysed. The number of hidden units is assumed to be less than the number of inputs units. Such an error surface shows only saddle-points and *no* local minimum or local maximum. This result is qualitatively consistent with the observation made by Bray and Dean (2007). In fact, as long as we do not *get stuck* in the plateaus surrounding these saddle points, for such a model we are guaranteed to obtain the global minimum of the error. Indeed Saxe *et al.* (2014), analyzed the dynamics of learning in the presence of these saddle points, and showed that they arise due to scaling symmetries in the weight space of deep linear feedforward models. These scaling symmetries enabled Saxe *et al.* (2014) to find new exact solutions to the nonlinear dynamics of learning in deep linear networks. These learning dynamics exhibit plateaus of high error followed by abrupt transitions to better performance, and they qualitatively recapitulate aspects of the hierarchical development of semantic concepts in infants (Saxe *et al.*, 2013).

In Saad and Solla (1995) the dynamics of stochastic gradient descent are analyzed for soft committee machines. The paper explores how well a student model can learn to imitate a teacher model which was randomly sampled. An important observation of this work is showing that learning goes through an initial phase of *being trapped in the symmetric subspace*. In other words, due to symmetries in the randomly initialized weights, the network has to traverse one or more plateaus that are caused by units with similar behaviour. Rattray *et al.* (1998); Inoue *et al.* (2003) provides further analysis, stating that the initial phase of learning is plagued with saddle point structures caused by symmetries in the weights. Intuitively, the escape from these saddle points corresponds to weight differentiation of the afferent synapses onto hidden units. Being trapped along the symmetric subspace corresponds to pairs of hidden units computing the same function on the input distribution. Exiting the symmetric subspace corresponds to hidden units learning to become different from each other, thereby specializing and learning the internal representations of the teacher neural network. Interestingly, the error function in the vicinity of the symmetric subspace has a saddle point structure, and signals that hidden unit differentiation will lower error by following directions of negative curvature. Thus first order gradient descent dynamics yields a plateau in the error because it is difficult for such dynamics to rapidly escape from the saddle point in the vicinity of the symmetric subspace by specializing hidden units.

Mizutani and Dreyfus (2010) looks at the effect of negative curvature for learning and implicitly at the effect of saddle point structures in the error surface. Their findings are similar. A proof is given where the error surface of a single layer MLP is shown to have saddle points (where the Hessian matrix is indefinite).

2 Optimization algorithms near saddle points

The above work suggests that both in typical randomly chosen high dimensional error surfaces, and neural network training error surfaces, a proliferation of saddle points with error much higher than the local minimum constitute the predominant obstruction to rapid non convex optimization. We now provide a theoretically justified algorithm to deal with this problem. We will focus on nondegenerate saddle points, namely those for which the Hessian is not exactly singular. These critical

points can be locally analyzed based on a unique reparametrization of the function as described by Morse's lemma (see chapter 7.3, Theorem 7.16 in Callahan (2010)).

This reparametrization is given by taking a Taylor expansion of the function f around the critical point. If we assume that the Hessian is not singular, then there is a neighbourhood around this critical point where this approximation is reliable and, since the first order derivatives vanish, the Taylor expansion is given by:

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

Let us denote by $\mathbf{e}_{[1]}, \ldots, \mathbf{e}_{[n_\theta]}$ the eigenvectors of the Hessian $\mathbf{H}$ and by $\lambda_{[1]}, \ldots, \lambda_{[n_\theta]}$ the corresponding eigenvalues. We can now make a change of coordinates into the space span by these eigenvectors:

$$\Delta\mathbf{v} = \frac{1}{2} \begin{bmatrix} \mathbf{e}_{[1]}{}^T \\ \cdots \\ \mathbf{e}_{[n_\theta]}{}^T \end{bmatrix} \Delta\theta \tag{2}$$

$$f(\theta^* + \Delta\theta) = f(\theta^*) + \frac{1}{2} \sum_{i=1}^{n_\theta} \lambda_{[i]} (\mathbf{e}_{[i]}{}^T \Delta\theta)^2 = f(\theta^*) + \sum_{i=1}^{n_\theta} \lambda_{[i]} \Delta\mathbf{v}_i^2 \tag{3}$$

For *gradient descent* we can see that, as expected, the gradient points in the right direction close to a saddle point. Namely, if some eigenvalue $\lambda_{[i]}$ is positive, then we move towards θ^* in that direction because the restriction of f to the corresponding eigenvector is $f(\theta^*) + \lambda_{[i]} \Delta\mathbf{v}_i^2$, which has a minimum when $\mathbf{v}_i = 0$. On the other hand, if the eigenvalue is negative, then the gradient descent will move away from θ^* which is a maximum when restricting the loss function to the corresponding eigenvector of said eigenvalue.

The downside of gradient descent is not the direction, but the *size* of the step along each eigenvector. The step we will take, for any direction $\mathbf{e}_{[i]}$, is given by $-2\lambda_{[i]}\Delta\mathbf{v}_i$. Because the gradients are proportional to the corresponding eigenvalues of the Hessian, the eigenvalue dictates how fast we move in each direction. Note that also, to avoid divergence, the learning rate has to be at most $1/|\lambda_{[max]}|$. Therefore, if there is a large discrepancy between eigenvalues, then gradient descent will have to take very small steps in some directions. This means that it might takes a very long time to move away form the critical point, if the critical point is a saddle point, or to the critical point if it is a minimum.

The *Newton method* solves the slowness problem by properly rescaling the gradients in each direction with the inverse of the corresponding eigenvalue. The step we take now is given by $-\Delta\mathbf{v}_i$. However, this approach can result in moving in the wrong direction. Specifically, if an eigenvalue is negative, then by multiplying with its inverse, the Newton method would change the sign of the gradient along this eigenvector. Instead of taking the step away from θ^* in the direction of negative curvature (where θ^* is a maximum), Newton method moves towards θ^*. This effectively makes θ^* an *attractor* for the dynamics of the Newton method, making this algorithm converge to this unstable critical point. Therefore, while gradient descent might still escape saddle points in finite time, Newton method can not and it will converge to said saddle point.

A *trust region* approach is a practical implementation of second order methods for non-convex problems, where the Hessian is damped to remove negative curvature. Damping the Hessian by adding the identity matrix times some constant α is equivalent to adding α to each of the eigenvalues of the Hessian. That is, we now rescale the gradients by multiplying them with $1/\lambda_{[i]}+\alpha$, resulting in the step $-\left(\lambda_{[i]}/\lambda_{[i]}+\alpha\right)\Delta\mathbf{v}_i$. In particular, to deal with negative curvature, one has to increase the damping coefficient α enough such that even for the most negative eigenvalue $\lambda_{[min]}$ we have $\lambda_{[min]} + \alpha > 0$. This ensures moving in a descent direction. The downside is again the step size in each direction. Adding a fixed constant to each eigenvalue makes the ratio $\lambda_{[i]}/\lambda_{[i]}+\alpha$ far from 1 for most eigenvalues, especially when we have a large discrepancy between them.

Beside damping, another approach of dealing with negative curvature for second order methods is to ignore them. This can be done regardless of the approximation of the Newton method used, for

example as either a truncated Newton method or a BFGS approximation (see Nocedal and Wright (2006) chapters 4 and 7). By ignoring direction of negative curvature, we will not be able to escape saddle points, as there is no direction in which we move away from θ^*. Damping and ignoring the directions of negative curvature are the main existing approaches to deal with negative curvature.

Natural gradient descent is a first order method that relies on the curvature of the parameter manifold. That is, we take a step that induces a constant change in the behaviour of the model as measured by the KL-divergence between the model before taking the step and after. For example, the recently proposed Hessian-Free Optimization (Martens, 2010) was shown to be a variant of natural gradient descent (Pascanu and Bengio, 2014). The algorithm ends up doing an update similar to the Newton method, just that instead of inverting the Hessian we invert Fisher Information Matrix, $\mathbf{F}$, which is positive definite by construction. It is argued in Rattray *et al.* (1998); Inoue *et al.* (2003) that natural gradient descent can address certain saddle point structures effectively. Specifically, it can resolve those saddle points arising from having units behaving very similar. In Mizutani and Dreyfus (2010), however, it is argued that natural gradient descent does also suffer when negative curvature is present. One particular known issue is the over-realizable regime, when the model is over complete. In this situations, while there exists a stationary solution θ^*, the Fisher matrix around θ^* is rank deficient. Numerically, this means that the Gauss Newton direction can be (close to) orthogonal to the gradient at some distant point from θ^* (Mizutani and Dreyfus, 2010). A line search in this direction would fail and lead to the algorithm converging to some non-stationary point. Another weakness of natural gradient is that the *residual* $\mathbf{S}$ defined as the difference between the Hessian and the Fisher Information Matrix can be large close to certain saddle points that exhibit strong negative curvature. This means that the landscape close to these critical points can be dominated by $\mathbf{S}$, meaning that the rescaling provided by $\mathbf{F}^{-1}$ is not optimal in all directions as it does not match the eigenvalues of the Hessian.

The same is true for TONGA Le Roux *et al.* (2007), an algorithm similar to natural gradient descent. TONGA relies on the covariance of the gradients for the rescaling factor. As these gradients vanish close to a critical point, their covariance will indicate that one needs to take much larger steps then needed close to critical points.

3 Generalized trust region methods

We will refer to a straight forward extension of trust region methods as *generalized trust region methods*. The extension involves two simple changes of the method. The first one is that we allow to take a first order Taylor expansion of the function to minimize instead of always relying on a second order Taylor expansion as typically done in trust region methods.

The second change is that we replace the constraint on the norm of the step $\Delta\theta$ by a constraint on the distance between θ and $\theta + \Delta\theta$. The distance measure is also not specified and can be specific to the instance of generalized trust region method used. If we define $\mathcal{T}_k(f, \theta, \Delta\theta)$ to indicate a k-th order Taylor series expansion of f around θ evaluated at $\theta + \Delta\theta$, then we can summarize a generalized trust region as:

$$\begin{aligned} \Delta\theta = \arg\min_{\Delta\theta} \mathcal{T}_k\{f, \theta, \Delta\theta\} \quad \text{with } k \in \{1, 2\} \\ \text{s. t. } d(\theta, \theta + \Delta\theta) \leq \Delta \end{aligned} \tag{4}$$

4 Addressing the saddle point problem

In order to address the saddle point problem, we will look for a solution within the family of generalized trust region methods. We know that using the Hessian as a rescaling factor can result in a non-descent direction because of the negative curvature. The analysis above also suggest that correcting negative curvature by an additive term results in a suboptimal step, therefore we want the resulting step from this trust region method to not be a function of the Hessian. We therefore use a first order Taylor expansion of the loss. This means that the curvature information has to come from the constraint by picking a suitable distance measure d.

4.1 Limit the influence of second order terms – saddle-free Newton Method

The analysis carried out for different optimization techniques states that, close to nondegenerate critical points, what we want to do is to rescale the gradient in each direction $\mathbf{e}_{[i]}$ by $1/|\lambda_{[i]}|$. This achieves the same optimal rescaling as the Newton method, while preserving the sign of the gradient and therefore avoids making saddle point attractors of the dynamics of learning. The idea of taking the absolute value of the eigenvalues of the Hessian was suggested before. See, for example, in Nocedal and Wright (2006, chapter 3.4) or in Murray (2010, chapter 4.1). However, we *are not aware* of any proper justification of this algorithm or even a proper detailed exploration (empirical or otherwise) of this idea.

The problem is that one can not simply replace $\mathbf{H}$ by $|\mathbf{H}|$, where $|\mathbf{H}|$ is the matrix obtained by taking the absolute value of each eigenvalue of $\mathbf{H}$, without proper justification. For example, one obvious question is: are we still optimizing the same function? While we might be able to argue that we do the right thing close to critical points, can we do the same far away from these critical points? In what follows we will provide such a justification.

Let us consider the function we want to minimize f by employing a generalized trust region method that works on a first order approximation of f and enforces some constraint on the step taken based on some distance measure d between θ and $\theta + \Delta\theta$. Since the minimum of the first order approximation of f is at infinity, we know that within this generalized trust region approach we will always jump to the border of the trust region.

So the proper question to ask is how far from θ can we trust our first order approximation of f. One measure of this trustfulness is given by how much the second order term of the Taylor expansion of f influences the value of the function at some point $\theta + \Delta\theta$. That is we want the following constraint to hold:

$$d(\theta, \theta + \Delta\theta) = \left| f(\theta) + \nabla f \Delta\theta + \frac{1}{2}\Delta\theta^T \mathbf{H} \Delta\theta - f(\theta) - \nabla f \Delta\theta \right| = \frac{1}{2}\left|\Delta\theta^T \mathbf{H} \Delta\theta\right| \leq \Delta \qquad (5)$$

where ∇f is the partial derivative of f with respect to θ and $\Delta \in \mathbb{R}$ is some some small value that indicates how much discrepancy we are willing to accept between our first order approximation of f and the second order approximation of f.Note that the distance measure d takes into account the curvature of the function.

Equation (5) is also not easy to solve for $\Delta\theta$ in more than one dimension. If we resolve the absolute value by taking the square of the distance we get a function that is quartic in $\Delta\theta$ (the term is raised to the power 4). We address this problem by relying on the following Lemma.

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

Proof. Let $\mathbf{e}_{[1]}, \ldots \mathbf{e}_{[n]}$ be the different eigenvectors of $\mathbf{A}$ and $\lambda_{[1]}, \ldots \lambda_{[n]}$ the corresponding eigenvalues. We now re-write the identity by expressing the vector $\mathbf{x}$ in terms of these eigenvalues:

$$\begin{aligned}
|\mathbf{x}^T \mathbf{A} \mathbf{x}| &= \left| \sum_i (\mathbf{x}^T \mathbf{e}_{[i]}) {\mathbf{e}_{[i]}}^T \mathbf{A} \mathbf{x} \right| \\
&= \left| \sum_i (\mathbf{x}^T \mathbf{e}_{[i]}) \lambda_{[i]} ({\mathbf{e}_{[i]}}^T \mathbf{x}) \right| \\
&= \left| \sum_i \lambda_{[i]} (\mathbf{x}^T \mathbf{e}_{[i]})^2 \right|
\end{aligned}$$

We can now use the triangle inequality $|\sum_i x_i| \leq \sum_i |x_i|$ and get that

$$
\begin{aligned}
|\mathbf{x}^T \mathbf{A} \mathbf{x}| &\leq \sum_i |(\mathbf{x}^T \mathbf{e}_{[i]})^2 \lambda_{[i]}| \\
&= \sum_i (\mathbf{x}^T \mathbf{e}_{[i]}) |\lambda_{[i]}| ({\mathbf{e}_{[i]}}^T \mathbf{x}) \\
&= \mathbf{x}^T |\mathbf{A}| \mathbf{x}
\end{aligned}
$$

□

Instead of using the originally proposed distance measure, based on lemma 1, we will approximate the distance by its upper bound given by $\Delta\theta|\mathbf{H}|\Delta\theta$, resulting in the following generalized trust region method:

$$
\begin{aligned}
\Delta\theta = \arg\min_{\Delta\theta} & f(\theta) + \nabla f \Delta\theta \\
\text{s. t. } & \Delta\theta^T |\mathbf{H}| \Delta\theta \leq \Delta
\end{aligned}
\tag{6}
$$

Note that as discussed before, the inequality constraint can be turned into a equality one as the first order approximation of f has a minimum at infinity and we always jump to the border of the trust region. Similar to Pascanu and Bengio (2014), we can use Lagrange multipliers to get the solution of this constraint optimization. This gives (up to a scalar that we fold into the learning rate) a step of the form:

$$
\Delta\theta = -\nabla f |\mathbf{H}|^{-1} \tag{7}
$$

The algorithm is a trust region method that uses the curvature of the function to define the shape of the trust region. It allows to move further in directions of low curvature and enforces to move less in direction of high curvature. If the Hessian is positive definite the method behaves identically to the Newton method. Close to a nondegenerate critical points, it takes the optimum step, by scaling based on the eigenvalues of the Hessian which describe the geometry of the surface locally, while moving away from the critical point in the direction of negative curvature.

5 Experimental results – empirical evidence for the saddle point problem

In this section we run experiments on a scaled down version of MNIST, where each input image is rescaled to be of size 10×10. This rescaling allows us to construct models that are small enough such that we can implement the exact Newton method and saddle-free Newton method, without relying on any kind of approximations.

As a baseline we also consider minibatch stochastic gradient descent, the de facto optimization algorithm for such models. We additionally use momentum to improve the convergence speed of this algorithm. The hyper-parameters of minibatch SGD – the learning rate, minibatch size and the momentum constant – are chosen using random search (Bergstra and Bengio, 2012). We pick the best configurations from approximately 80 different choices. The learning rate and momentum constant are sampled on a logarithmic scale, while the minibatch size is sampled from the set $\{1, 10, 100\}$. The best performing hyper-parameter values for SGD are provided in Table 1.

Damped Newton method is a trust region method where we damp the Hessian $\mathbf{H}$ by adding the identity times the damping factor. No approximations are used in computing the Hessian or its inverse (beside numerical inaccuracy due to machine precision). For the saddle-free Newton we also damp the matrix $|\mathbf{H}|$, obtained by taking the absolute value of the eigenvalues of the Hessian. At each step, for both methods, we pick the best damping coefficient among the following values: $\{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We do not perform an additional line search for the step size, but rather consider a fixed learning rate of 1. Note that by searching over the damping coefficient we implicitly search for the optimum step size as well. These two methods are run in batch mode.

The results of these experiments are visualized in Figure 2. Figure 2 (a) looks at the minimum error reached by different algorithms as a function of the model size. The plot provides empirical

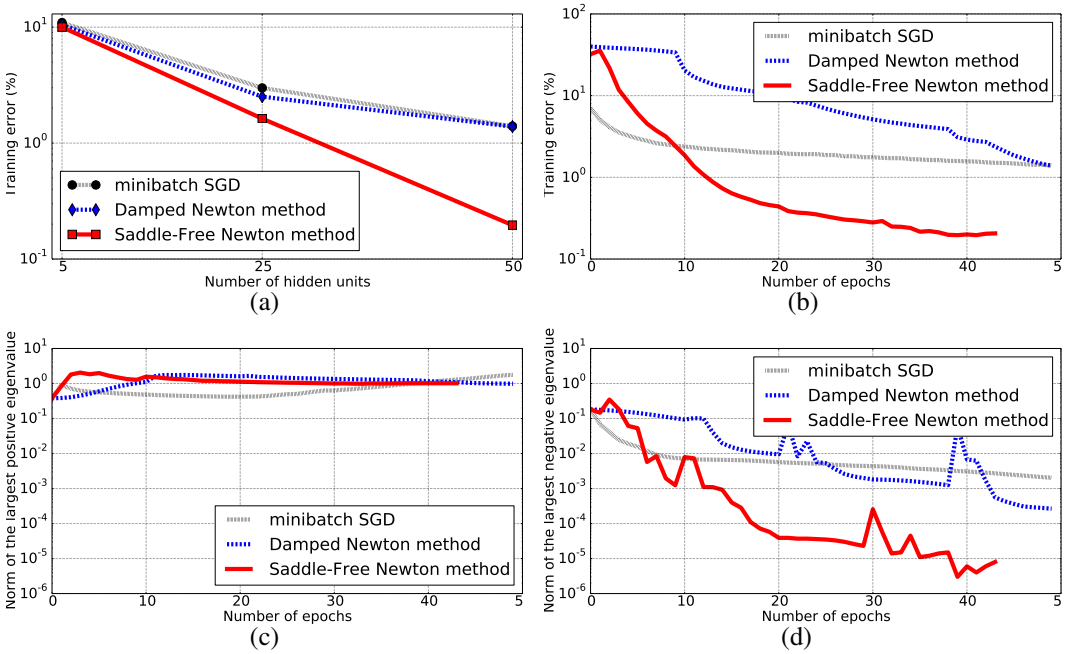


Figure 2: Empirical evaluation of different optimization algorithms for a single layer MLP trained on the rescaled MNIST dataset. In (a) we look at the minimum error obtained by the different algorithms considered as a function of the model size. (b) shows the optimum training curve for the three algorithms. The error is plotted as a function of the number of epochs. (c) looks at the evolution of the norm of the largest positive eigenvalue of the Hessian and (d) at the norm of the largest negative eigenvalue of the Hessian.

Model size	learning rate	momentum constant	minibatch size
5 units	0.074	0.031	10
25 units	0.040	0.017	10
50 units	0.015	0.254	1

Table 1: Best performing hyper-parameters for stochastic gradient descent.

evidence that, as the dimensionality of the problem increases, the number of saddle points also increases (exponentially so). We argue that for the larger model (50 hidden units), the likelihood of an algorithms such as SGD or Newton method to stop near a saddle point becomes higher (as the number of saddle points is much larger) and therefore we should see these algorithms perform worse in this regime. The plot confirms our intuition. We see that for the 50 hidden units case, saddle-free outperforms the other two methods considerably.

Figure 2 (b) depicts the training error versus the number of epochs that the model already went through. This plot suggest that saddle-free behaves well not only near a critical point but also far from them, taking a reasonable large steps.

In Figure 2 (c) we look at the norm of the largest positive eigenvalue of the Hessian as a function of the number of training epochs for different optimization algorithms. Figure 2 (d) looks in a similar fashion at the largest negative eigenvalues of the Hessian. Both these quantities are approximated using the Power method. The plot clearly shows that initially there is a direction of negative curvature (and therefore we are bound to go near saddle points). The norm of the largest negative eigenvalue is close to that of the largest positive eigenvalue. As learning progresses, the norm of the negative eigenvalue decreases as it is predicted by the theory of random matrices Bray and Dean (2007) (think of the semi-circular distribution being shifted to the right). For stochastic gradient descent and Damped Newton method, however, even at convergence we still have a reasonably large negative eigenvalue, suggesting that we have actually "converged" to a saddle point rather than a lo-

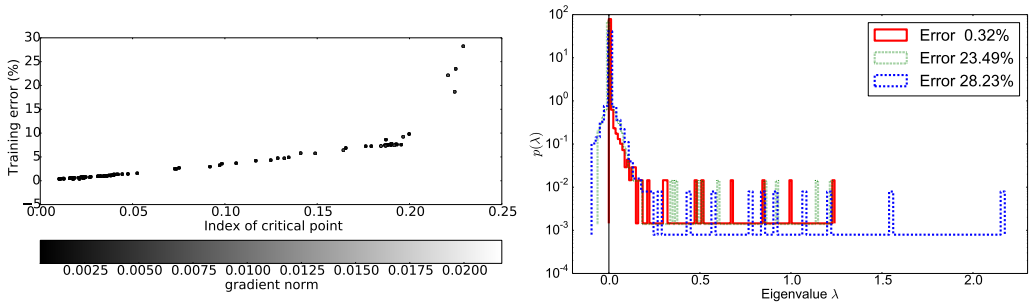


Figure 3: The plot on the left depict the training error versus the index (fraction of negative eigenvalues) for different critical points found nearby the path taken by different runs of the saddle-free Newton method. The critical points are discovered using the Newton method. Note that the gray level of each point is given by the norm of the gradient where the Hessian was measured. It can be regarded as a measure of noise (how far from the actual critical point we have actually converged). The plot on the right shows the distribution of eigenvalues of the Hessian for three different critical points selected based on their error. Note that the y-axis is on a log scale.

cal minimum. For saddle-free Newton method the value of the most negative eigenvalue decreases considerably, suggesting that we are more likely to have converged to an actual local minimum.

Figure 3 is an empirical evaluation of whether the properties predicted by Bray and Dean (2007) for random Gaussian error functions hold for neural networks.

To obtain this plot we used the Newton method to discover nearby critical points along the path taken by the saddle-free Newton algorithm. We consider 20 different runs of the saddle-free algorithm, each using a different random seed. We then run 200 jobs. The first 100 jobs are looking for critical points near the value of the parameters obtained after some random number of epochs (between 0 and 20) of a randomly selected run (among the 20 different runs) of saddle-free Newton method. To this starting position uniform noise is added of small amplitude (the amplitude is randomly picked between the different values $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ The last 100 jobs look for critical points near uniformally sampled weights (the range of the weights is given by the unit cube). The task (dataset and model) is the same as the one used previously.

In Figure 3, the plot on the left shows the index of the critical point (fraction of negative eigenvalues of the Hessian at the critical point) versus the training error that it obtains. This plot shows that all critical points, projected in this plane, align in a monotonically decreasing curve, as predicted by the theoretical results on random Gaussian error functions(Bray and Dean, 2007). This provides evidence that most critical points with corresponding to large error has to be, with high probability, a saddle point, and not a local minima.

The plot on the right looks at the distribution of the eigenvalues of the Hessian at 3 different critical points picked according to the error that they realise. Note that the plot is on a log scale on the y-axis. These distributions *do not follow* the semi-circular rule, as predicted by the theory of random matrices. This is probably caused by the structure of the neural network (and of the task). However, the generic observation of (Bray and Dean, 2007), that as the error decreases the distribution shifts to the right seems to hold, with the exception of the peak that we have around 0. The fact that we have a large number of eigenvalues at 0, and a few eigenvalues that are sufficiently large suggests that any of these saddle-points are surrounded by plateaus, in which the different algorithms might end up taking a suboptimal step.

6 Conclusion

In the introduction of this work we provided a thorough literature review of works that argue for the prevalence of saddle points in large scale non-convex problems or how learning addresses negative curvature. We tried to expand this collection of results by providing an intuitive analysis of how different existing optimization techniques behave near such structures. Our analysis clearly shows

that while some algorithms might not be "stuck" in the plateau surrounding the saddle point they do take a suboptimal step.

The analysis also suggests what would be the optimal step. We extend this observation, that was done prior to this work, to a proper optimization algorithm by relying on the framework of generalized trust region methods. Within this framework, at each step, we optimize a first order Taylor approximation of our function, constraint to a region within which this approximation is reliable. The size of this region (in each direction) is determined by how different the first order approximation of the function is from the second order approximation of the function.

From this we derive an algorithm that we call saddle-free Newton method, that looks similarly to the Newton method, just that the matrix that we need to invert is obtained from the Hessian matrix by taking the absolute value of all eigenvalues. We show empirically that our claims hold on a small model trained on a scaled-down version of MNIST, where images are scaled to be 10×10 pixels.

As future work we are interested in mainly two directions. The first direction is to provide a pipeline for saddle-free Newton method that allows to scale the algorithm to high dimensional problems, where we can not afford to compute the entire Hessian matrix. The second direction is to further extend the theoretical analysis of critical points by specifically looking at neural network models.

Acknowledgments

The authors would like to acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. We would also like to thank the developers of Theano (Bergstra *et al.*, 2010; Bastien *et al.*, 2012). Razvan Pascanu is supported by a Google DeepMind Fellowship.

References

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, **2**(1), 53–58.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, **13**, 281–305.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Bray, A. J. and Dean, D. S. (2007). Statistics of critical points of gaussian fields on large-dimensional spaces. *Physics Review Letter*, **98**, 150201.

Callahan, J. (2010). *Advanced Calculus: A Geometric View*. Undergraduate Texts in Mathematics. Springer.

Fyodorov, Y. V. and Williams, I. (2007). Replica symmetry breaking condition exposed by random matrix calculation of landscape complexity. *Journal of Statistical Physics*, **129**(5-6), 1081–1116.

Inoue, M., Park, H., and Okada, M. (2003). On-line learning theory of soft committee machines with correlated hidden units steepest gradient descent and natural gradient descent. *Journal of the Physical Society of Japan*, **72**(4), 805–810.

Le Roux, N., Manzagol, P.-A., and Bengio, Y. (2007). Topmoumoute online natural gradient algorithm. *Advances in Neural Information Processing Systems*.

Martens, J. (2010). Deep learning via hessian-free optimization. In *International Conference in Machine Learning*, pages 735–742.

Mizutani, E. and Dreyfus, S. (2010). An analysis on negative curvature induced by singularity in multi-layer neural-network learning. In *Advances in Neural Information Processing Systems*, pages 1669–1677.

Murray, W. (2010). Newton-type methods. Technical report, Department of Management Science and Engineering, Stanford University.

Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer.

Parisi, G. (2007). Mean field theory of spin glasses: statistics and dynamics. Technical Report Arxiv 0706.0094.

Pascanu, R. and Bengio, Y. (2014). Revisiting natural gradient for deep networks. In *International Conference on Learning Representations*.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Rattray, M., Saad, D., and Amari, S. I. (1998). Natural Gradient Descent for On-Line Learning. *Physical Review Letters*, **81**(24), 5461–5464.

Saad, D. and Solla, S. A. (1995). On-line learning in soft committee machines. *Physical Review E*, **52**, 4225–4243.

Saxe, A., McClelland, J., and Ganguli, S. (2013). Learning hierarchical category structure in deep neural networks. *Proceedings of the 35th annual meeting of the Cognitive Science Society*, pages 1271–1276.

Saxe, A., McClelland, J., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *International Conference on Learning Representations*.

Wigner, E. P. (1958). On the distribution of the roots of certain symmetric matrices. *The Annals of Mathematics*, **67**(2), 325–327.