Topology and Geometry of Deep Rectified Network Optimization Landscapes

C. Daniel Freeman and Joan Bruna
Department of Statistics, University of California, Berkeley, CA 94720, USA
Courant Institute of Mathematical Sciences, New York University, New York USA

September 16, 2016

Abstract

The loss surface of deep neural networks has recently attracted interest in the optimization and machine learning communities as a prime example of high-dimensional non-convex problem. Some insights were recently gained using spin glass models, but at the expense of strongly simplifying the nonlinear nature of the model

In this work, we do not make any such assumption and prove that deep rectified networks do not have bad local minima. Together with recent results that rigorously establish that no gradient descent can get stock on saddle points, we conclude that gradient descent converges to a global optimum in deep rectified networks.

The conditioning of such gradient descent is the next challenge we address. We study this question by estimating the geometry of level sets, and we introduce an algorithm to estimate the regularity of such sets on large-scale networks. Our empirical results suggest that these sets become exponentially more curvy as the energy level decays, in accordance to what is observed in practice.

Contents

1 Introduction

- Context of the problem
- Related work: Spin glass, recent result.
- Gradient Descent converges to minimizers (Jordan Recht et al).
- Main result on connectedness of level sets.
- Geometry of the level sets. Algorithm to estimate the geodesics along level sets. Measure of curvature of these sets.

2 Topology of Level Sets

Let P be a probability measure on a product space $\mathcal{X} \times \mathcal{Y}$, where we assume $\mathcal{X}$ and $\mathcal{Y}$ are Euclidean vector spaces for simplicity. Let $\{(x_i, y_i)\}_i$ be an iid sample of size L drawn from P defining the training set. We consider the classic empirical risk minimization of the form

$$F_e(\theta) = \frac{1}{L} \sum_{l=1}^{L} \|\Phi(x_i; \theta) - y_i\|^2 \ , \tag{1}$$

where $\Phi(x; \theta)$ encapsulates the feature representation that uses parameters $\theta \in \mathbb{R}^S$. In a deep neural network, this parameter contains the weights and biases used in all layers. For convenience, in our analysis we will also use the oracle risk minimization:

$$F_o(\theta) = \mathbb{E}_{(X,Y)\sim P} \|\Phi(X; \theta) - Y\|^2 \ , \tag{2}$$

We define the level set of $F(\theta)$ as

$$\Omega_F(\lambda) = \{\theta \in \mathbb{R}^S \ ; \ F(\theta) \leq \lambda\} \ . \tag{3}$$

The first question we study is the structure of critical points of $F_e(\theta)$ and $F_o(\theta)$ when Φ is a multilayer neural network. In particular, we are interested to know whether F_e has local minima which are not global minima. This question is answered by knowing whether $\Omega_F(\lambda)$ is connected at each energy level λ:

Proposition 2.1. *If $\Omega_F(\lambda)$ is connected for all λ then every local minima of $F(\theta)$ is a global minima.*

Proof: Suppose that θ_1 is a local minima and θ_2 is a global minima, but $F(\theta_1) > F(\theta_2)$. If $\lambda = F(\theta_1)$, then clearly θ_1 and θ_2 both belong to $\Omega_F(\lambda)$. Suppose now that $\Omega_F(\lambda)$ is connected. Then we could find a smooth (i.e. continuous and differentiable) path $\gamma(t)$ with $\gamma(0) = \theta_1$, $\gamma(1) = \theta_2$ and $F(\gamma(t)) \leq \lambda = F(\theta_1)$. In particular, as $t \to 0$, we have

$$\begin{aligned} F(\gamma(t)) &= F(\theta_1) + t\langle \nabla F(\theta_1), \dot{\gamma}(0)\rangle + \frac{t^2}{2}\left(\dot{\gamma}(0)^T HF(\theta_1)\dot{\gamma}(0) + \langle \nabla F(\theta_1), \ddot{\gamma}(0)\rangle\right) + o(t^2) \\ &= F(\theta_1) + \frac{t^2}{2}\dot{\gamma}(0)^T HF(\theta_1)\dot{\gamma}(0) + o(t^2) \ , \end{aligned}$$

which shows that $F(\gamma(t)) \leq F(\theta_1)$ for all t is incompatible with $H(\theta_1) \succeq 0$. □

2.1 The Linear Case

A particularly simple but insightful case is when F is a multilayer network defined by

$$\Phi(x; \theta) = W_K \dots W_1 x \ , \ \theta = (W_1, \dots, W_K) \ . \tag{4}$$

This model defines a non-convex (and non-concave) loss $F_e(\theta)$ which has been recently studied in [**?**] concurrently with our work. We provide here an alternative proof that in that case, there are no poor local minima.

We have the following result.

Proposition 2.2. *Let $W_1, W_2, \dots, W_K$ be weight matrices of sizes $n_k \times n_{k+1}$, $k < K$, and let $F_e(\theta)$, $F_o(\theta)$ denote the risk minimizations using Φ as in (**??**). Assume that $n_j \geq \min(n_1, n_K)$ for $j = 2 \dots K-1$ [TODO I think this is not necessary]. Then $\Omega_{F_e}(\lambda)$ is connected for all λ, as well as F_o.*

Proof: We proceed by induction over the number of layers K. For $K = 1$, the loss $F(\theta)$ is convex. Let θ_1, θ_2 be two arbitrary points in a level set Ω_λ. Thus $L(\theta_1) \leq \lambda$ and $L(\theta_2) \leq \lambda$. We have

$$L(t\theta_1 + (1-t)\theta_2) \leq tL(\theta_1) + (1-t)L(\theta_2) \leq \lambda \ ,$$

and thus a linear path is sufficient in that case to connect θ_1 and θ_2.

Suppose the result is true for $K-1$. Let $\theta_1 = (W_1^1, \dots, W_K^1)$ and $\theta_2 = (W_1^2, \dots, W_K^2)$ with $L(\theta_1) \leq \lambda$, $L(\theta_2) \leq \lambda$. For each $W_1, \dots, W_K$, we denote $\tilde{W}_j = W_j$ for $j < K-1$ and $\tilde{W}_{K-1} = W_K W_{K-1}$. By induction hypothesis, the loss expressed in terms of $\tilde{\theta} = (\tilde{W}_1, \dots, \tilde{W}_{K-1})$ is connected between $\tilde{\theta}_1$ and $\tilde{\theta}_2$. Let $\tilde{W}_{K-1}(t)$ the corresponding path projected in the last layer. We just need to produce a path in the variables $W_{K-1}(t)$, $W_K(t)$ such that (i) $W_{K-1}(0) = W_{K-1}^1$, $W_{K-1}(1) = W_{K-1}^2$, (ii) $W_K(0) = W_K^1$, $W_K(1) = W_K^2$, and (iii) $W_K(t)W_{K-1}(t) = \tilde{W}_{K-1}(t)$ for $t \in (0,1)$. We construct it as follows. Let

$$W_K(t) = tW_K^2 + (1-t)W_K^1 + t(1-t)V \ ,$$

$$W_{K-1}(t) = W_K(t)^\dagger \tilde{W}_{K-1}(t) \ ,$$

where $W_K(t)^\dagger = (W_K(t)^T W_K(t))^{-1} W_K(t)^T$ denotes the pseudoinverse and V is a $n_{K-1} \times n_K$ matrix drawn from a iid distribution. Conditions (i) and (ii) are immediate from the definition, and condition (iii) results from the fact that

$$W_K(t) W_K(t)^\dagger = \mathbf{I}_{N_K} \ ,$$

since $W_K(t)$ has full rank for all $t \in (0,1)$. $\square$.

2.2 Half-Rectified Nonlinear Case

We now study the setting given by

$$\Phi(x;\theta) = W_K \rho W_{K-1} \rho \dots \rho W_1 x \ , \ \theta = (W_1, \dots, W_K) \ , \tag{5}$$

where $\rho(z) = \max(0, z)$. The biases can be implemented by replacing the input vector x with $\overline{x} = (x, 1)$ and by rebranding each parameter matrix as

$$\overline{W}_i = \left(\begin{array}{c|c} W_i & b_i \\ \hline 0 & 1 \end{array} \right) ,$$

where b_i contains the biases for each layer. For simplicity, we continue to use W_i and x in the following.

We start with a characterization of the oracle loss.

Theorem 2.3. *Let $W_1, W_2, \ldots, W_K$ be weight matrices of sizes $n_k \times n_{k+1}$, $k < K$, and let $F_e(\theta)$, $F_o(\theta)$ denote the risk minimizations using Φ as in (**??**). Assume that $n_j \geq \min(n_1, n_K)$ for $j = 2 \ldots K-1$ [TODO I think this is not necessary]. Then $\Omega_{F_o}(\lambda)$ is connected for all λ.*

Proof: We will again prove the result by induction over the depth K. Suppose first that $K = 2$. The oracle risk is

$$F_o(W_1, W_2) = \mathbb{E}\left(\|W_2 \rho W_1 X - Y\|^2\right) .$$

If we denote $X_{W_1} = \rho W_1 X$, let us verify that $F_o(W_1, W_2)$ only depends upon the correlation operator of X_{W_1} and its cross-correlation to Y. Indeed, we have

$$\begin{aligned} \mathbb{E}\left(\|W_2 \rho W_1 X - Y\|^2\right) &= \mathbb{E}\left(\|W_2 X_{W_1} - Y\|^2\right) \\ &= W_2 \Sigma_{W_1} W_2^T + \Sigma_Y - 2Tr\left(W_2 \Sigma_{W_1, Y}\right) , \end{aligned}$$

where $\Sigma_{W_1} = \mathbb{E}\left(X_{W_1} X_{W_1}^T\right)$ and $\Sigma_{W_1,Y} = \mathbb{E}\left(X_{W_1} Y^T\right)$. Let us see that when $\rho(z)$ is the half-rectification the covariance structure of X_{W_1} can be easily related to the original distribution. Indeed, we have the following

Lemma 2.4. *Let $Z = \rho W X$ with $\rho(z) = \max(0, z)$. Then*

$$\Sigma_Z = \tilde{W}^T \Sigma_X \tilde{W} \tag{6}$$

□

[TODO F_e case].

3 Geometry of Level Sets

3.1 Definitions

For a model with network parameters θ_i, and a learning problem with sample space X, the fundamental object of study is the loss function, $L(X, \theta_i)$. In practice, one only has access to an estimate of the loss function over some restricted subset, χ_i, of the sample space: $E(L(X, \theta_i), \chi_i)$. Unless otherwise stated, the loss functions computed throughout are assumed to be on a restricted test set not used during training.

A key ingredient of the algorithm is the use of an *interpolated model.* For two given models, θ_1 and θ_2, we defined the interpolated model with parameter t as follows:

$$\Theta(\theta_1, \theta_2, t) := \theta_1(1-t) + \theta_2 t \tag{7}$$

Thus, the interpolated model parameters—i.e., weights and biases—are simply linearly interpolated between two given models.

Additionally, the algorithm requires an estimate of the interpolated loss curve:

$$\gamma(\theta_1, \theta_2) := L(X, \Theta(\theta_1, \theta_2, t)), t \in [0, 1] \tag{8}$$

or, an estimate of the loss on those models which are linear interpolations sitting between θ_1 and θ_2. More specifically, we seek efficient estimates of the location of the maxima, $t^* := \frac{d\gamma(\theta_1,\theta_2,t)}{dt}\Big|_{t^*} = 0, \frac{d^2\gamma(\theta_1,\theta_2,t)}{dt^2}\Big|_{t^*} < 0$. While in principle, the interpolated loss curve could have rich structure, in practice it is generally fairly smooth, thus straightforward hill climbing algorithms can efficiently locate these points.

Finally, for a pair of models (θ_1, θ_2), it will be convenient to define the maximum interpolated error:

$$\Gamma(\theta_1, \theta_2) := \min_{\Theta^*(\theta_1,\theta_2)} \max \mathrm{L}(\mathrm{X}, \theta_\mathrm{i})\Big|_{\theta_\mathrm{i} \in \Theta^*} \tag{9}$$

where $\Theta^*(\theta_1, \theta_2)$ is *any* continuous path in the space of weights connecting θ_1 and θ_2. Thus, $\Gamma(\theta_1, \theta_2)$ represents the minimum possible maximum loss achieved by those paths in the space of weights connecting θ_1 and θ_2. More intuitively, if $\Gamma(\theta_1, \theta_2) \leq \max(\mathrm{L}(\mathrm{X}, \theta_1), \mathrm{L}(\mathrm{X}, \theta_2))$, then the models are "connected"—there exists a continuous path in the space of models with total loss never exceeding the maximum loss achieved by θ_1 or θ_2.

3.2 The Greedy Algorithm

1. Train two models θ_i and θ_j to a threshold loss value, L_0.

2. Determine the location of the global maxima, t^*, on the interpolated loss curve $\gamma(\theta_i, \theta_j)$.

3. Perform gradient descent on the interpolated model $\Theta(\theta_i, \theta_j, t^*) := \theta_{i,j}$ until it is below αL_0 for some $\alpha \in [0, 1]$.

4. Calculate the maxima of the interpolated losses $\gamma(\theta_i, \theta_{i,j})$ and $\gamma(\theta_{i,j}, \theta_j)$. If these maxima are below L_0, then stop recursing on this branch and proceed to remaining branches(see 5). If not, proceed to step 5.

5. For those pairs, θ_a, θ_b from step 4 for which the maxima exceeds L_0, start a new branch by returning to step 2 and making the replacement $i- > a$ and $j- > b$. If depth exceeds d, stop (see below).

We provide a cartoon of the algorithm in Fig. **??**. If the algorithm succeeds, then the output of the algorithm is a sequence of models, θ_i such that the pairwise interpolated loss curve between each in a sequence will be less than the threshold L_0. Thus, the algorithm outputs a continuous path in parameter space connecting the original two models such that everywhere along the path, the total loss is less than or equal to the loss of the original models.

As written, if a path does *not* exist, then the algorithm will clearly not converge. Thus, on top of the parameter α, discussed below, the algorithm has an additional free parameter in the *depth* chosen to explore. For convenience, we define the string of models produced by the algorithm at depth d with parameter α to be the *interpolated string*, $S(\theta_1, \theta_2, \alpha, d)$. These are precisely those models recursively generated by the algorithm in step 3. Further, these models are naturally ordered along a path, starting from θ_1 and terminating on θ_2, as indicated in Fig. **??**.

Finally, to use this as a tool to diagnose convexity, we define the *maximum interpolated error at depth d and tolerance* α:

$$\tilde{\Gamma}(\theta_1, \theta_2, d, \alpha) := \max_{i,\ j \text{ neighbors in } \mathrm{S}(\theta_1, \theta_2, \alpha, \mathrm{d})} \gamma(\theta_\mathrm{i}, \theta_\mathrm{j}, \mathrm{t}) \tag{10}$$

where by "neighbors in $S(\theta_1, \theta_2, \alpha, d)$", we only mean that the models are immediately adjacent on the interpolating string. This quantity upper bounds the true maximum interpolated error, i.e. (**??**).

In summary: the algorithm recursively produces and trains new models lying on a continuous path in the space of model parameters, i.e. a string. Training via gradient descent biases the path towards valleys on the loss surface, thus encouraging the loss along this path to be low. In practice, the parameter α is chosen to be less than 1 to aid convergence. We provide numerical and theoretical evidence for this choice in section SECTIONGOHERE.

3.3 Constrained Dynamic String Sampling

While the algorithm presented in Sec. **??** is fast for sufficiently smooth families of loss surfaces with few saddle points, here we present a slightly modified version which, while slower, provides more control over the convergence of the string. Instead of training intermediate models via full SGD to a desired accuracy, intermediate models will be subject to a constraint that ensures they are "close" to the neighboring models on the string. Specifically, intermediate models will be constrained to the unique hyperplane in

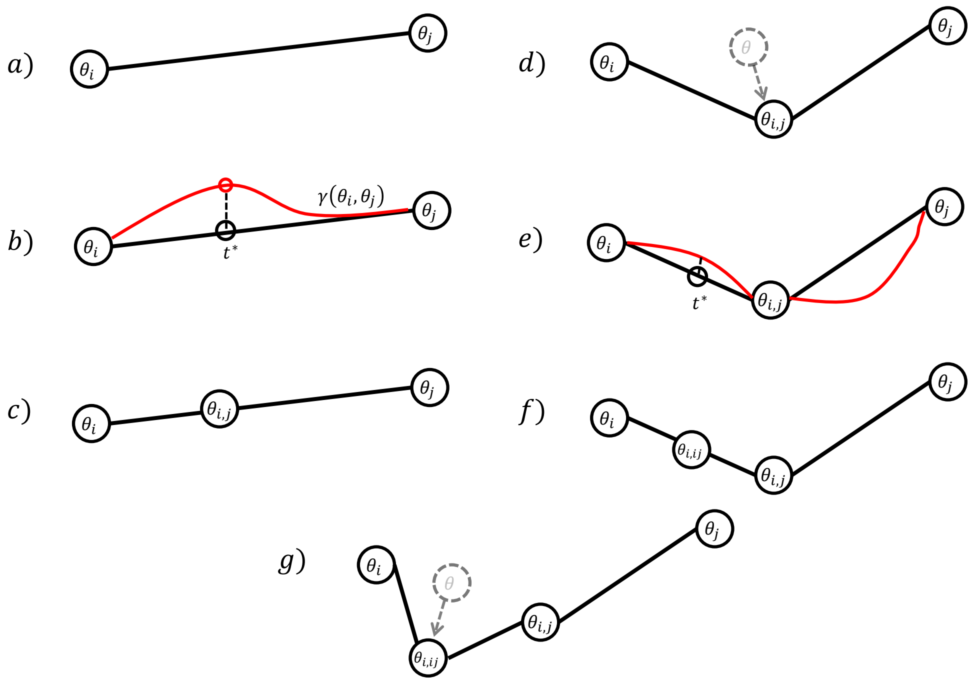

Figure 1: A cartoon of the algorithm. $a)$: The initial two models with approximately the same loss, L_0. $b)$: The interpolated loss curve, in red, and its global maximum, occuring at $t = t^*$. $c)$: The interpolated model $\Theta(\theta_i, \theta_j, t^*)$ is added and labeled $\theta_{i,j}$. $d)$: Stochastic gradient descent is performed on the interpolated model until its loss is below αL_0. $e)$: New interpolated loss curves are calculated between the models, pairwise on a chain. $f)$: As in step $c)$, a new model is inserted at the maxima of the interpolated loss curve between θ_i and $\theta_{i,j}$. $g)$: As in step $d)$, gradient descent is performed until the model has low enough loss.

weightspace equidistant from its two neighbors. This is similar to a sort of "L_1 regularization" where the loss function for a given model on the string, θ_i, has an additional term $\tilde{L}(\theta) = L(\theta) + \zeta(\|\theta_{i-1} - \theta_i\| + \|\theta_{i+1} + \theta_i\|)$. The strength of the ζ regularization term controls the "springy-ness" of the weightstring. note: make this more precise, the hyperplane constraint is stronger than the L_1 constraint...L_1 only keeps the model in a ball close to the midpoint between the models.

Because adapting DSS to use this constraint is straightforward, here we will describe an alternative "breadth-first" approach wherein models are trained in parallel until convergence. This alternative approach has the advantage that it will indicate a disconnection between two models "sooner" insofar as it will be clear two models cannot be connected once the loss on either of the two initial models, θ_1 or θ_2, is less than $\Gamma(\theta_1, \theta_2)$. The precise geometry of the loss surface will dictate which approach to use in practice.

Given two random models σ_i and σ_j where $|\sigma_i - \sigma_j| < \kappa$, we aim to follow the evolution of the family of models connecting σ_i to σ_j. Intuitively, almost every continuous path in the space of random models connecting σ_i to σ_j has, on average, the same (high) loss. For simplicity, we choose to initialize the string to the linear segment interpolating between these two models. If this entire segment is evolved via gradient descent, the segment will either evolve into a string which is entirely contained in a basin of the loss surface, or some number of points will become fixed at a higher loss. These fixed points are difficult to detect directly, but will be indirectly detected by the persistence of a large interpolated loss between two adjacent models on the string.

The algorithm proceeds as follows:

(0.) Initialize model string to have two models, σ_i and σ_j.

1. Begin training all models to the desired loss, keeping the instantaneous loss of all models being trained approximately constant..

2. If the pairwise interpolated loss $\gamma(\sigma_n, \sigma_{n+1})$ exceeds a tolerance α_1, insert a new model at the maximum of the interpolated loss between these two models. For simplicity, this tolerance is chosen to be $(1 + \alpha_1^*)$ times the instantaneous loss of all other models on the string.

3. Repeat steps (1) and (2) until all models (and interpolated errors) are below a threshold loss L_0, or until a chosen failure condition (see **??**).

3.4 Failure Conditions

While the algorithms presented will faithfully certify two models are connected if the algorithm converges, it is worth reemphasizing that they do not guarantee that two models are disconnected if the algorithm fails to converge. In general, the problem of determining if two models are connected can be made arbitrarily difficult by choice of a particularly pathological geometry for the loss function, so we are constrained to heuristic arguments for determining when to stop running the algorithm.

Thankfully, in practice, loss function geometries for problems of interest are not intractably difficult to explore.

4 Numerical Experiments

For our numerical experiments, we aimed to extract qualitative features of both small, toy networks, as well as of larger workhorse networks suitable for use on real world tasks (e.g. MNIST). At its core, the maximum interpolated error (i.e., (**??**)) is a measure of problem nonconvexity—or, more precisely, of the nonconvexity of the loss surface of a given architecture on a particular learning problem.

4.1 Polynomial Regression

Polynomial function regression is a task for which small neural networks can achieve extremely high accuracy. For our numerical experiments, we studied a 1-4-4-1 fully connected multilayer perceptron style architecture with RELU activation and RMSProp optimization. For ease-of-analysis, we restricted the family of polynomials to be strictly contained in the interval $x \in [0, 1]$ and $f(x) \in [0, 1]$.

Discussion of different Loss functions

etc.

4.2 Convolutional Neural Networks

5 Discussion

- Future: Generalization Error Question.