TOPOLOGY AND GEOMETRY OF DEEP RECTIFIED NETWORK OPTIMIZATION LANDSCAPES

C. Daniel Freeman *
Department of Physics
University of California at Berkeley
Berkeley, CA 94720, USA
daniel.freeman@berkeley.edu

Joan Bruna
Courant Institute of Mathematical Sciences
New York University
New York, NY ZIPGOHERE, USA
{robot,net}@wits.ac.za

Coauthor
Affiliation
Address
email

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 study conditions on the data distribution and model architecture that prevent the existence of bad local minima. Together with recent results that rigorously establish that no gradient descent can get stuck on saddle points, we conclude that gradient descent converges to a global optimum in deep rectified networks.

The conditioning of 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

*See our github page at PAGEGOHERE

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 (4). 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 $t \in (0,1)$. □.

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 $W_1, W_2, \dots, W_K$ 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 (5). Assume that $n_j \geq \min(n_1, n_K)$ for $j = 2 \dots K-1$ [TODO I think this is not necessary]. Then $\Omega_{F_o}(\lambda)$ 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 THE GREEDY ALGORITHM

For a pair of models with network parameters θ_i, θ_j, each with $F_e(\theta)$ below a threshold L_0, we aim to efficiently generate paths in the space of weights where the empirical loss along the path remains below the threshold. These paths are continuous curves belonging to $\Omega_F(\lambda)$–that is, the level sets of the loss function of interest.

We provide a greedy algorith, Dynamic String Sampling, which finds such a path below.

The algorithm recursively builds a string of models in the space of weights which continuously connect θ_i to θ_j. Models are added and trained until the pairwise linearly interpolated loss, i.e. $\max_{\mathrm{t}} \mathrm{F_e}(\mathrm{t}\theta_{\mathrm{i}} + (1-\mathrm{t})\theta_{\mathrm{j}})$ for $t \in (0, 1)$, is below the threshold, L_0, for every pair of neighboring models on the string. We provide a cartoon of the algorithm in Fig. 1.

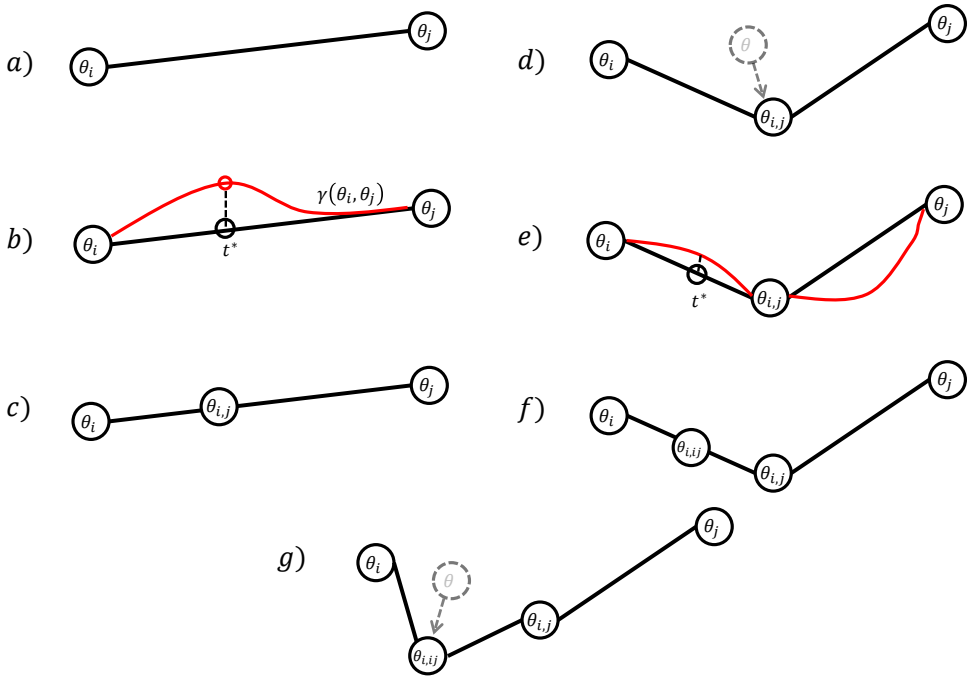

Figure 1: A cartoon of the algorithm. $a)$: The initial two models with approximately the same loss, L_0. $b)$: The interpolated loss curve, in red, and its global maximum, occuring at $t = t^*$. $c)$: The interpolated model $\Theta(\theta_i, \theta_j, t^*)$ is added and labeled $\theta_{i,j}$. $d)$: 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.

Algorithm 1 Greedy Dynamic String Sampling

1: $L_0 \leftarrow$ Threshold below which path will be found
2: $\Phi_1 \leftarrow$ randomly initialize θ_1, train $\Phi(x_i\ \theta_1)$ to L_0
3: $\Phi_2 \leftarrow$ randomly initialize θ_2, train $\Phi(x_i\ \theta_2)$ to L_0
4: BeadList $\leftarrow(\Phi_1, \Phi_2)$
5: Depth $\leftarrow 0$
6: **procedure** FINDCONNECTION(Φ_1, Φ_2)
7: $\quad t^* \leftarrow$ t such that $\left.\frac{d\gamma(\theta_1,\theta_2,t)}{dt}\right|_t = 0$ OR $t = 0.5$
8: $\quad \Phi_3 \leftarrow$ train $\Phi(x_i; t^*\theta_1 + (1-t^*)\theta_2)$ to L_0
9: $\quad$ BeadList $\leftarrow$ insert(Φ_3, after Φ_1, BeadList)
10: $\quad MaxError_1 \leftarrow max_t(F_e(t\theta_3 + (1-t)\theta_1))$
11: $\quad MaxError_2 \leftarrow max_t(F_e(t\theta_2 + (1-t)\theta_3))$
12: $\quad$ **if** $MaxError_1 > L_0$ **then return** FindConnection(Φ_1, Φ_3)
13: $\quad$ **if** $MaxError_2 > L_0$ **then return** FindConnection(Φ_3, Φ_2)
14: $\quad$ Depth $\leftarrow$ Depth+1

3.2 FAILURE CONDITIONS AND PRACTICALITIES

While the algorithm presented will faithfully certify two models are connected if the algorithm converges, it is worth emphasizing that the algorithm does 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. We comment more on diagnosing disconnections more carefully in section SYMMETRYDISCONNECT.

Further, if the $\mathbf{MaxError}$ exceeds L_0 for every new recursive branch as the algorithm progresses, the worst case runtime scales as $O(\exp(\mathbf{Depth}))$. Empirically, we find that the number of new models added at each depth does grow, but eventually saturates, and falls for a wide variety of models and architectures, so that the typical runtime is closer to $O(\text{poly}(\mathbf{Depth}))$—at least up until a critical value of L_0. We comment more on this in section NUMERICALDISCUSSION.

Finally, we find that training Φ_3 to αL_0 for $\alpha < 1$ in line 8 of the algorithm tends to aid convergence without noticeably impacting our numerics.

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.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

6 Constrained Dynamic String Sampling

While the algorithm presented in Sec. 3.1 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 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 3.2).