On the Quality of the Initial Basin in Overspecified Neural Networks

Itay Safran ITAY.SAFRAN@WEIZMANN.AC.IL
Ohad Shamir OHAD.SHAMIR@WEIZMANN.AC.IL
Weizmann Institute of Science, Israel

Abstract

Deep learning, in the form of artificial neural networks, has achieved remarkable practical success in recent years, for a variety of difficult machine learning applications. However, a theoretical explanation for this remains a major open problem, since training neural networks involves optimizing a highly non-convex objective function, and is known to be computationally hard in the worst case. In this work, we study the *geometric* structure of the associated non-convex objective function, in the context of ReLU networks and starting from a random initialization of the network parameters. We identify some conditions under which it becomes more favorable to optimization, in the sense of (i) High probability of initializing at a point from which there is a monotonically decreasing path to a global minimum; and (ii) High probability of initializing at a basin (suitably defined) with a small minimal objective value. A common theme in our results is that such properties are more likely to hold for larger ("overspecified") networks, which accords with some recent empirical and theoretical observations.

1. Introduction

Deep learning (in the form of multi-layered artificial neural networks) has been tremendously successful in recent years, and advanced the state of the art across a range of difficult machine learning applications. Inspired by the structure of biological nervous systems, these predictors are usually composed of several layers of simple computational units (or neurons), parameterized by a set of weights, which can collectively express highly complex functions. Given a dataset of labeled examples, these networks are generally trained by minimizing the average of some loss function over the data, using a local search procedure such as stochastic gradient descent.

Although the expressiveness and statistical performance of such networks is relatively well-understood, it is a major open problem to understand the computational tractability of training such networks. Although these networks are trained successfully in practice, most theoretical results are negative. For example, it is known that finding the weights that best fit a given training set, even for very small networks, is NP-hard (Blum & Rivest, 1992). Even if we relax the problem by allowing improper learning or assuming the data is generated by a network, the problem remains worst-case hard (see e.g. (Livni et al., 2014) for a discussion of this and related results). This theory-practice gap is a prime motivation for our work.

In this paper, we study the *geometric structure* of the objective function associated with training such networks, namely the average loss over the training data as a function of the network parameters. We focus on plain-vanilla, feed-forward networks which use the simple and popular ReLU activation function (see Sec. 2.1 for precise definitions), and losses convex in the network's predictions, for example the squared loss and cross-entropy loss. The structure of the resulting objective function is poorly understood. Not surprisingly, it is complex, highly non-convex, and local search procedures are by no means guaranteed to converge to a global minimum. Moreover, it is known that even if the network is composed of a single neuron, the function may have exponentially many local minima (Auer et al., 1996). Furthermore, as we discuss later in the paper, the construction can be done such that the vast majority of these local minima are sub-optimal. Nevertheless, our goal in this work is to understand whether, perhaps under some conditions, the function has some geometric properties which may make it more favorable to optimization.

Before continuing, we emphasize that our observations are purely geometric in nature, independent of any particular optimization procedure. Moreover, we make no claim that these properties necessarily imply that a practical local search procedure, such as stochastic gradient descent, will converge to a good solution (although proving such a result

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

could be an interesting direction for future work). Nevertheless, the properties we consider do seem indicative of the difficulty of the optimization problem, and we hope that our results can serve as a basis for further progress on this challenging research direction.

A recurring theme in our results is that such favorable properties can be shown to occur as the network size grows *larger*, perhaps larger than what would be needed to get good training error with unbounded computational power (hence the term *overspecified* networks). At first, this may seem counter-intuitive, as larger networks have more parameters, and training them involves apparently more complex optimization in a higher-dimensional space. However, higher dimensions also means more potential directions of descent, so perhaps the gradient descent procedures used in practice are more unlikely to get stuck in poor local minima and plateaus. Although difficult to formalize, this intuition accords with several recent empirical and theoretical evidence, which indicates that larger networks may indeed be easier to train (see (Livni et al., 2014) as well as (Choromanska et al., 2014; Dauphin et al., 2014; Bach, 2014)).

In the first part of our work (Sec. 3), we consider networks of arbitrary depth, where the weights are initialized at random using some standard initialization procedure. This corresponds to a random starting point in the parameter space. We then show that under some mild conditions on the loss function and the data set, as the network width increases, we are overwhelmingly likely to begin at a point from which there is a continuous, strictly monotonically decreasing path to a global minimum[1]. This means that although the objective function is non-convex, it is not "wildly" non-convex in the sense that the global minima are in isolated valleys which cannot be reached by descent procedures starting from random initialization. In other words, "crossing valleys" is not strictly necessary to reach a good solution (although again, we give no guarantee that this will happen for a specific algorithm such as stochastic gradient descent). We note that this accords well with recent empirical observations (Goodfellow & Vinyals, 2014), according to which the objective value of networks trained in practice indeed tends to decrease monotonically, as we move from the initialization point to the end point attained by the optimization algorithm. We also note that although we focus on plain-vanilla feed-forward networks, our analysis is potentially applicable to more general architectures, such as convolutional networks.

In the second part of our work (Sec. 4), we focus more specifically on two-layer networks with scalar-valued outputs. Although simpler than deeper networks, the associated optimization problem is still highly non-convex and exhibits similar worst-case computational difficulties. For such networks, we study a more fine-grained geometric property: We define a partition of the parameter space into convex regions (denoted here as basins), in each of which the objective function has a relatively simple, basin-like structure: Inside each such basin, every local minima of the objective function is global, all sublevel sets are connected, and in particular there is only a single connected set of minima, all global on that basin. We then consider the probability that a random initialization will land us at a basin with small minimal value. Specifically, we show that under various sets of conditions (such as low intrinsic data dimension, or a cluster structure), this event will occur with overwhelmingly high probability as the network size increases. As an interesting corollary, we show that the construction of (Auer et al., 1996), in which a single neuron network is overwhelmingly likely to be initialized at a bad basin, is actually surprisingly brittle to overspecification: If we replace the single neuron with a two-layer network comprised of just $\Omega(\log(d))$ neurons (d being the data dimension), and use the same dataset, then with overwhelming probability, we will initialize at a basin with a globally optimal minimal value.

As before, we emphasize that these results are purely geometric, and do not imply that an actual gradient descent procedure will necessarily attain such good objective values. Nevertheless, we do consider a property such as high probability of initializing in a good basin as indicative of the optimization difficulty of the problem.

We now turn to discuss some related work. Perhaps the result most similar to ours appears in (Livni et al., 2014), where it is shown that quite generally, if the number of neurons in the penultimate layer is larger than the data size, then global optima are ubiquitous, and "most" starting points will lead to a global optimum upon optimizing the weights of the last layer. Independently, (Haeffele & Vidal, 2015) also provided results of a similar flavor, where sufficiently large networks compared to the data size and dimension do not suffer from local minima issues. However, these results involve huge networks, which will almost invariably overfit, whereas the results in our paper generally apply to networks of more moderate size. Another relevant work is (Choromanska et al., 2014), which also investigates the objective function of ReLU networks. That work differs from ours by assuming data sampled from a standard Gaussian distribution, and considering asymptotically large networks with a certain type of random connectivity. This allows the authors to use tools from the theory of spin-glass models, and obtain interesting results on the asymptotic distribution of the objective values associated with critical points. Other results along similar lines appear in (Dauphin

[1]To be precise, we prove a more general result, which implies a monotonic path to any objective value smaller than that of the initial point, as long as some mild conditions are met. See Thm. 1 in Sec. 3 for a precise formulation.

et al., 2014). This is a worthy but rather different research direction than the one considered here, where we focus on theoretical investigation of non-asymptotic, finite-sized networks on fixed datasets, and consider different geometric properties of the objective function. Other works, such as (Arora et al., 2014; Andoni et al., 2014; Janzamin et al., 2015; Zhang et al., 2015) and some of the results in (Livni et al., 2014), study conditions under which certain types of neural networks can be efficiently learned. However, these either refer to networks quite different than standard ReLU networks, or focus on algorithms which avoid direct optimization of the objective function (often coupled with strong assumptions on the data distribution). In contrast, we focus on the geometry of the objective function, which is directly optimized by algorithms commonly used in practice. Finally, works such as (Bengio et al., 2005; Bach, 2014) study ways to convexify (or at least simplify) the optimization problem by re-parameterizing and lifting it to a higher dimensional space. Again, this involves changing the objective function rather than studying its properties.

2. Preliminaries and Notation

We use bold-faced letters to denote vectors, and capital letters to generally denote matrices. Given a natural number k, we let $[k]$ be shorthand for $\{1, \ldots, k\}$.

2.1. ReLU Neural Networks

We begin by giving a formal definition of the type of neural network studied in this work. A fully connected feedforward artificial neural network computes a function $\mathbb{R}^d \to \mathbb{R}^k$, and is composed of neurons connected according to a directed acyclic graph. Specifically, the neurons can be decomposed into layers, where the output of each neuron is connected to all neurons in the succeeding layer and them alone. We focus on ReLU networks, where each neuron computes a function of the form $\mathbf{x} \mapsto \left[\mathbf{w}^\top \mathbf{x} + b\right]_+$ where $\mathbf{w}$ is a weight vector and b is a bias term specific to that neuron, and $[\cdot]_+$ is the ReLU activation function $[z]_+ = \max\{0, z\}$.

For a vector $\mathbf{b} = (b_1, \ldots, b_n)$ and a matrix

$$W = \begin{pmatrix} \cdots & \mathbf{w}_1 & \cdots \\ & \vdots & \\ \cdots & \mathbf{w}_n & \cdots \end{pmatrix},$$

and letting $[W\mathbf{x} + \mathbf{b}]_+$ be a shorthand for $\left(\left[\mathbf{w}_1^\top \mathbf{x} + b_1\right]_+, \ldots, \left[\mathbf{w}_n^\top \mathbf{x} + b_n\right]_+\right)$, we can define a layer of n neurons as

$$\mathbf{x} \mapsto [W\mathbf{x} + \mathbf{b}]_+ .$$

Finally, by denoting the output of the i^{th} layer as O_i, we can define a network of arbitrary depth recursively by

$$O_{i+1} = [W_{i+1} O_i + \mathbf{b}_{i+1}]_+ ,$$

where $W_i, \mathbf{b}_i$ represent the matrix of weights and bias of the i^{th} layer, respectively. Following a standard convention for multi-layer networks, the final layer h is a purely linear function with no bias, i.e.

$$O_h = W_h \cdot O_{h-1}. \tag{1}$$

Define the *depth* of the network as the number of layers h, and denote the number of neurons n_i in the i^{th} layer as the *size* of the layer. We define the *width* of a network as $\max_{i \in [h]} n_i$.

We emphasize that in this paper, we focus on plain-vanilla networks, and in particular do not impose any constraints on the weights of each neuron (e.g. regularization, or having convolutional layers).

We define $\mathcal{W}$ to be the set of all network weights, which can be viewed as one long vector (its size of course depends on the size of the network considered). We will refer to the Euclidean space containing $\mathcal{W}$ as the *parameter space*.

Define the output of the network $N : \mathbb{R}^d \to \mathbb{R}^k$ over the set of weights $\mathcal{W}$ and an instance $\mathbf{x} \in \mathbb{R}^d$ by

$$N(\mathcal{W})(\mathbf{x}).$$

Note that depending on the dimension of W_h, the network's output can be either a scalar (e.g. for regression) or a vector (e.g. for the purpose of multiclass classification). An important property of the ReLU function, which we shall use later in the paper, is that it is positive-homogeneous: Namely, it satisfies for all $c \geq 0$ and $x \in \mathbb{R}$ that

$$[c \cdot x]_+ = c \cdot [x]_+ .$$

2.2. Objective Function

We use $S = (\mathbf{x}_t, \mathbf{y}_t)_{t=1}^m$ to denote the data on which we train the network, where $\mathbf{x}_t \in \mathbb{R}^d$ represents the t^{th} training instance and $\mathbf{y}_t \in \mathbb{R}^k$ represents the corresponding target output, and where m is used to denote the number of instances in the sample.

Throughout this work, we consider a loss function $\ell(\mathbf{y}, \mathbf{y}')$, where the first argument is the prediction and the second argument is the target value. We assume ℓ is convex in its first argument (e.g. the squared loss or the cross-entropy loss).

In its simplest form, training a neural network corresponds to finding a combination of weights which minimizes the average loss over the training data. More formally, we define the objective function as

$$L_S(N(\mathcal{W})) = \frac{1}{m} \sum_{t=1}^{m} \ell(N(\mathcal{W})(\mathbf{x}_t), \mathbf{y}_t).$$

We stress that even though ℓ is convex as a function of the network's prediction, $L_S(N(\mathcal{W}))$ is generally non-convex as a function of the network's weights. Also, we note that occasionally when the architecture is clear from context, we omit $N(\cdot)$ from the notation, and write simply $L_S(\mathcal{W})$.

2.3. Basins

In Sec. 4, we will we consider a partition of the parameter space into convex regions, in each of which the objective function $L_S(\mathcal{W})$ has a relatively simple basin-like form, and study the quality of the basin in which we initialize. In particular, we define a *basin* with respect to $L_S(\mathcal{W})$ as a closed and convex subset of the parameter space, on which $L_S(\mathcal{W})$ has connected sublevel sets, and where each local minimum is global. More formally, we have the following definition:

Definition 1. *(Basin) A closed and convex subset B of our parameter space is called a basin if the following conditions hold:*

- *B is connected, and for all $\alpha \in \mathbb{R}$, the set $B_{\leq}\alpha = \{\mathcal{W} \in B : L_S(\mathcal{W}) \leq \alpha\}$ is connected.*
- *If $\mathcal{W} \in B$ is a local minimum of L_S on B, then it is a global minimum of L_S on B.*

We define the basin value Bas(B) of a basin B as the minimal value[2] attained:

$$\text{Bas}(B) := \min_{\mathcal{W}\in B} L_S(\mathcal{W}).$$

Similarly, for a point $\mathcal{W}$ in the interior of a basin B we define its basin value as the value of the basin to which it belongs:

$$\text{Bas}(\mathcal{W}) := \text{Bas}(B).$$

In what follows, we consider basins with disjoint interiors, so the basin to which $\mathcal{W}$ belongs is always well-defined.

2.4. Initialization Scheme

As was mentioned in the introduction, we consider in this work questions such as the nature of the basin we initialize from, under some random initialization of the network weights. Rather than assuming a specific distribution, we will consider a general class of distributions which satisfy some mild independence and symmetry assumptions:

Assumption 1. *The initialization distribution of the network weights satisfies the following:*

- *The weights of every neuron are initialized independently.*
- *The vector of each neuron's weights (including bias) is drawn from a spherically symmetric distribution supported on non-zero vectors.*

This assumption is satisfied by most standard initialization schemes: For example, initializing the weights of each neuron independently from some standard multivariate Gaussian, up to some arbitrary scaling, or initializing each neuron uniformly from an origin-centered sphere of arbitrary radius. An important property of distributions satisfying Assumption 1 is that the signs of the weights of each neuron, viewed as a vector in $\mathbb{R}^n$, is uniformly distributed on $\{-1, +1\}^n$.

3. Networks of Any Depth: Path to Global Minima

In this section, we establish the existence of a continuous path in the parameter space of multilayer networks (of any depth), which is strictly monotonically decreasing in the objective value, and can reach an arbitrarily small objective value, including the global minimum. More specifically, we show in Thm. 1 that if the loss is convex in the network's predictions, and there exists *some* continuous path in the parameter space from the initial point $\mathcal{W}^{(0)}$ to a point with smaller objective value $\mathcal{W}^{(1)}$ (including possibly a global minimum, where the objective value along the path is not necessarily monotonic) which satisfies certain relatively mild conditions, then it is possible to find some other path from $\mathcal{W}^{(0)}$ to a point as good as $\mathcal{W}^{(1)}$, along which the objective value is strictly monotonically decreasing.

For the theorem to hold, we need to assume our starting point has a sufficiently large objective value. In Proposition 1 and Proposition 2, we prove that this will indeed occur with random initialization, with overwhelming probability. A different way to interpret this is that a significant probability mass of the surface of the objective function overlooks the global minimum. It should be noted that the path to the minimum might be difficult to find using local search procedures. Nevertheless, these results shed some light on the nature of the objective function, demonstrating that it is not "wildly" non-convex, in the sense that "crossing valleys" is not a must to reach a good solution, and accords with recent empirical evidence to this effect (Goodfellow & Vinyals, 2014).

For the results here, it would be convenient to re-write the objective function as $L(P(\mathcal{W}))$, where $\mathcal{W}$ is the vector of network parameters, $P(\mathcal{W})$ is an $m \times k$ matrix, which specifies the prediction for each of the m training points (the prediction can be scalar valued, i.e. $k = 1$, or vector-valued when $k > 1$), and L is the average loss over the training data. For example, for regression, a standard choice is the

[2]For simplicity, we will assume this minimal value is actually attained at some point in the parameter space. Otherwise, one can refer to an attainable value arbitrarily close to it.

squared loss, in which case

$$L(P(\mathcal{W})) = \frac{1}{m}\sum_{t=1}^{m}(N(\mathcal{W})(\mathbf{x}_t) - y_t)^2,$$

For classification, a standard choice in the context of neural networks is the cross-entropy loss coupled with a softmax activation function, which can be written as $\frac{1}{m}\sum_{t=1}^{m}\ell_t(N(\mathcal{W})(\mathbf{x}_t))$, where given a prediction vector $\mathbf{p}$ and letting j_t be an index of the correct class,

$$\ell_t(\mathbf{p}) = -\log\left(\frac{\exp(p_{j_t})}{\sum_j \exp(p_j)}\right).$$

Recall that although these losses are convex in the network's predictions, $L(P(\mathcal{W}))$ is still generally non-convex in the network parameters $\mathcal{W}$. Also, we remind that due to the last layer being linear, multiplying its parameters by some scalar c causes the output to change by c. Building on this simple observation, we have the following theorem.

Theorem 1. *Suppose $L : \mathbb{R}^{m\times k} \to \mathbb{R}$ is convex. Given a fully-connected network of any depth, with initialization point $\mathcal{W}^{(0)}$, suppose there exists a continuous path $\mathcal{W}^{(\lambda)}, \lambda \in [0,1]$ in the space of parameter vectors, starting from $\mathcal{W}^{(0)}$ and ending in another point $\mathcal{W}^{(1)}$ with strictly smaller objective value ($L(P(\mathcal{W}^{(1)})) < L(P(\mathcal{W}^{(0)}))$), which satisfies the following:*

1. *For some $\epsilon > 0$ and any $\lambda \in [0,1]$, there exists some $c_\lambda \geq 0$ such that $L(c_\lambda \cdot P(\mathcal{W}^{(\lambda)})) \geq L(P(\mathcal{W}^{(0)})) + \epsilon$.*
2. *The initial point $\mathcal{W}^{(0)}$ satisfies $L(P(\mathcal{W}^{(0)})) > L(\mathbf{0})$.*

Then there exists a continuous path $\tilde{\mathcal{W}}^{(\lambda)}, \lambda \in [0,1]$ from the initial point $\tilde{\mathcal{W}}^{(0)} = \mathcal{W}^{(0)}$, to some point $\tilde{\mathcal{W}}^{(1)}$ satisfying $L(P(\tilde{\mathcal{W}}^{(1)})) = L(P(\mathcal{W}^{(1)}))$, along which $L(P(\tilde{\mathcal{W}}^{(\lambda)}))$ is strictly monotonically decreasing in λ.

Intuitively, this result stems from the linear dependence of the network's output on the parameters of the last layer. Given the initial non-monotonic path $\mathcal{W}^{(\lambda)}$, we rescale the last layer's parameters at each $\mathcal{W}^{(\lambda)}$ by some positive factor $c^{(\lambda)}$ depending on λ (moving it closer or further from the origin), which changes its output and hence its objective value. We show it is possibly to do this rescaling, so that the rescaled path is continuous and has a monotonically decreasing objective value. In fact, although we focus here on ReLU networks, the theorem itself is quite general and holds even for networks with other activation functions. A formal proof and a more detailed intuition is provided in Subsection B.1.

The first condition in the theorem is satisfied by losses which get sufficiently large (as a function of the network predictions) sufficiently far away from the origin. In particular, it is generally satisfied by both the squared loss and the cross-entropy loss with softmax activations, assuming data points and initialization in general position[3]. The second condition requires the random initialization to be such that the initialized network has worse objective value than the all-zeros predictor. However, it can be shown to hold with probability close to $1/2$ (over the network's random initialization), for losses such as those discussed earlier:

Proposition 1. *If $L(\cdot)$ corresponds to the squared loss or cross-entropy loss with softmax activation, and the network parameters are initialized as described in Assumption 1, then $\mathbb{P}_{\mathcal{W}^{(0)}}\left[L(P(\mathcal{W}^{(0)})) > L(\mathbf{0})\right] \geq \frac{1}{2}\left(1 - 2^{-n_{h-1}}\right)$, where n_{h-1} is the number of neurons in the last layer before the output neurons.*

This proposition (whose proof appears in appendix B.2) is a straightforward corollary of the following result, which can be applied to other losses as well:

Proposition 2. *Suppose the network parameters $\mathcal{W}^{(0)}$ are initialized randomly as described in Assumption 1. Suppose furthermore that $L(\cdot)$ is such that*

$$\mathbb{P}\left[\, L(c\cdot P(\mathcal{W}^{(0)})) \text{ is strictly convex in } c \in [-1,1] \;\middle|\; P(\mathcal{W}^{(0)}) \neq \mathbf{0}\,\right] \geq r$$

for some $r > 0$ (where the probability is w.r.t. $\mathcal{W}^{(0)}$). Then $\mathbb{P}\left[L(P(\mathcal{W}^{(0)})) > L(\mathbf{0})\right] \geq \frac{r}{2}\left(1 - 2^{-n_{h-1}}\right)$.

Intuitively, the strict convexity property means that by initializing the neurons from a zero-mean distribution (such as a spherically symmetric one), we are likely to begin at a point with higher objective value than initializing at the mean of the distribution (corresponding to zero weights and zero predictions on all data points). A formal proof appears in Appendix B.3.

4. Two-layer ReLU Networks

We now turn to consider a more specific network architecture, namely two-layer networks with scalar output. While simpler than deeper architectures, two-layer networks still possess universal approximation capabilities (Cybenko, 1989), and encapsulate the challenge of optimizing a highly non-convex objective.

From this point onwards, we will consider for simplicity two-layer networks without bias (where $b = 0$ for all neu-

[3] For the squared loss, a sufficient condition is that for any λ, there is *some* data point on which the prediction of $N(\mathcal{W}^{(\lambda)})$ is non-zero. For the cross-entropy loss, a sufficient condition is that for any λ, there is *some* data point on which $N(\mathcal{W}^{(\lambda)})$ outputs an 'incorrect' prediction vector $\mathbf{p}$, in the sense that if i is the correct label, then $p_i \notin \arg\max_j p_j$.

rons, not just the output neuron), for the purpose of simplifying our analysis. This is justified, since one could simulate the additional bias term by incrementing the dimension of the data and mapping an instance in the dataset using $\mathbf{x} \mapsto (\mathbf{x}, 1) \in \mathbb{R}^{d+1}$, so that the last coordinate of the weight of a neuron will function as a bias term. Having such a fixed coordinate does not affect the validity of our results for two-layer nets.

We denote our network parameters by $(W, \mathbf{v})$ where the rows of the matrix $W \in \mathbb{R}^{n \times d}$ represent the weights of the first layer and $\mathbf{v} \in \mathbb{R}^n$ represents the weight of the output neuron, and denote a two-layer network of width n by $N_n(W, \mathbf{v}) : \mathbb{R}^d \to \mathbb{R}$. Our objective function with respect to two-layer networks is therefore given by

$$\begin{aligned} L_S(W, \mathbf{v}) &:= \frac{1}{m} \sum_{t=1}^{m} \ell\left(N_n(W, \mathbf{v})(\mathbf{x}_t), y_t\right) \\ &= \frac{1}{m} \sum_{t=1}^{m} \ell\left(\sum_{i=1}^{n} v_i \cdot \left[\langle \mathbf{w}_i, \mathbf{x}_t \rangle\right]_+, y_t\right), \end{aligned}$$

corresponding to the parameter space $\left\{(W, \mathbf{v}) : W \in \mathbb{R}^{n \times d}, \mathbf{v} \in \mathbb{R}^n\right\}$.

To say something interesting regarding two-layer nets, we partition our parameter space into regions, inside each the objective function takes a relatively simple form. Our partition relies on the observation that when considering the subset of our parameter space in which $\operatorname{sign}(\langle \mathbf{w}_i, \mathbf{x}_t \rangle)$, $\operatorname{sign}(v_i)$ are fixed for any neuron i and any sample instance t, the ReLU activation is then reduced to either the zero function or the identity on $\langle \mathbf{w}_i, \mathbf{x}_t \rangle$ for all $i \in [n], t \in [m]$, so the objective function takes the form $\frac{1}{m} \sum_{t=1}^{m} \ell\left(\sum_{i \in I_t} v_i \langle \mathbf{w}_i, \mathbf{x}_t \rangle, y_t\right)$ for some index sets $I_1, \ldots, I_m \subseteq [n]$. This function is not convex or even quasi-convex as a function of $(W, \mathbf{v})$. However, it does behave as a basin (as defined in Definition 1), and hence contain a single connected set of global minima, with no non-global minima. More formally, we have the following definition and lemma:

Definition 2. *(Basin Partition) For any $A \in \{-1, +1\}^{n \times d}$ and $\mathbf{b} \in \{-1, +1\}^n$, define $B_S^{A,\mathbf{b}}$ as the topological closure of a set of the form*

$$\begin{aligned} \{(W, v) \ &: \ \forall t \in [m], j \in [n], \\ &\operatorname{sign}(\langle \mathbf{w}_j, \mathbf{x}_t \rangle) = a_{j,t}, \operatorname{sign}(v_j) = b_j\}. \end{aligned}$$

We will ignore $B_S^{A,\mathbf{b}}$ corresponding to empty sets, since these are irrelevant to our analysis.

Lemma 1. *For any $A \in \{-1, +1\}^{n \times d}$, $\mathbf{b} \in \{-1, +1\}^n$ such that $B_S^{A,\mathbf{b}}$ is non-empty, $B_S^{A,\mathbf{b}}$ is a basin as defined in Definition 1.*

The reader is referred to Appendix A.1 for the proof of the lemma.

Note that Definition 2 refers to a partition of the parameter space into a finite number of convex polytopes. Recalling Assumption 1 on the initialization distribution (basically, that it is a Cartesian product of spherically-symmetric distributions), it is easy to verify that we will initialize in an interior of a basin with probability 1. Therefore, we may assume that we always initialize in some unique basin.

We now focus on understanding when are we likely to initialize at a basin with a low minimal value (which we refer to as the basin value). We stress that this is a purely geometric argument on the structure on the objective function. In particular, even though every local minimum in a basin is also global on the basin, it does not necessarily entail that an optimization algorithm such as stochastic gradient descent will necessarily converge to the basin's global minima (for example, it may drift to a different basin). However, we believe this type of geometric property is indicative of the optimization susceptibility of the objective function, and provides some useful insights on its structure.

We now turn to state a simple but key technical lemma, which will be used to prove the results presented later in this section. Moreover, this lemma also provides some insight into the geometry of the objective function for two-layer networks:

Lemma 2. *Let $N_n(W, \mathbf{v})$ denote a two-layer network of size n, and let*

$$(W, \mathbf{v}) = (\mathbf{w}_1, \ldots, \mathbf{w}_n, v_1, \ldots, v_n) \in \mathbb{R}^{nd+n}$$

be in the interior of some arbitrary basin. Then for any subset $I = (i_1, \ldots, i_k) \subseteq [n]$ we have

$$\mathit{Bas}(W, \mathbf{v}) \leq \mathit{Bas}(\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_k}, v_{i_1}, \ldots, v_{i_k}).$$

Where the right hand side is with respect to an architecture of size k.

This lemma captures in a way the power overspecification has in the context of two-layer networks: In terms of basin values, any initialization made using a network of width $n \geq k$ (i.e. with n neurons in the first layer) is at least as good as if we had used only a width k network. This is because in our definition of the basin partition, clamping the weights of any $n - k$ neurons to 0 still keeps us in the same basin, while only increasing the minimal value we can obtain using the k non-clamped neurons. Therefore, if we had only a k-width network to begin with, the corresponding basin value can only be larger. We refer the reader to Appendix A.2 for the proof of the lemma.

4.1. Bad Local Minima Results: Brittleness to Overspecification

The training objective function of neural network is known to be highly non-convex, even for simple networks. A classic and stark illustration of this was provided in (Auer et al., 1996) who showed that even for a network comprised of a single neuron (with certain types of non-ReLU activation functions, and with or without bias), the objective function can contain a huge number of local minima (exponentially many in the input dimension). In Appendix C, we provide an extension of this result by proving that with a similar construction, and for a neuron with ReLU activation, not only is the number of local minima very large, but the probability of initializing at a basin with good local minimum (using the natural analogue of the basin partition from Definition 2 for a single neuron) is exponentially small in the dimension.

The construction provided in (Auer et al., 1996) (as well as the one provided in Appendix C) relies on training sets S comprised of singleton instances $\mathbf{x}_t$, which are non-zero on a single coordinate. The objective function for a single ReLU neuron without bias can be written as $\sum_{t=1}^{m} \ell\left(\left[\langle \mathbf{w}, \mathbf{x}_t\rangle\right]_+, y_t\right)$, so if the $\mathbf{x}_t$'s are singletons, this can be written as a sum of functions, each depending only on a single coordinate of $\mathbf{w}$. The training examples are chosen so that along each coordinate, there are two basins and two distinct local minima, one over the positive values and one over the negative values, but only one of these minima is global. Under the initialization distribution considered, the probabilities of hitting the good basin along each coordinate are independent and strictly less than 1. Therefore, with overwhelming probability, we will "miss" the right basin on a constant portion of the coordinates, resulting in a basin value which is suboptimal by at least some constant.

It is natural to study what happens to such a hardness construction under overspecification, which here means replacing a single neuron by a two-layer network of some width $n > 1$, and training on the same dataset. Surprisingly, it turns out that in this case, the probability of reaching a suboptimal basin decays exponentially in n and becomes arbitrarily small already when $n = \Omega(\log(d))$. Intuitively, this is because for such constructions, for each coordinate it is enough that *one* of the n neurons in the first layer will have the corresponding weight initialized in the right basin. This will happen with overwhelming probability if n is moderately large. More formally, we have the following theorem:

Theorem 2. *For any n, let α denote the minimal objective value achievable with a width n two-layer network, with respect to a convex loss ℓ on a training set S where each $\mathbf{x}_t$ is a singleton. Then when initializing $(W, \mathbf{v}) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$ from a distribution satisfying Assumption 1, we have*

$$\mathbb{P}\left[Bas\left(W, \mathbf{v}\right) \le \alpha\right] \ge 1 - 2d\left(\frac{3}{4}\right)^n .$$

The reader is referred to Appendix B.4 for the full proof.

We note that α cannot be larger than the optimal value attained using a single neuron architecture. Also, we emphasize that the purpose of Thm. 2 is not to make a statement about neural networks for singleton datasets (which are not common in practice), but rather to demonstrate the brittleness of hardness constructions such as in (Auer et al., 1996) to overspecification, as more neurons are added to the first layer. This motivates us in further studying overspecification in the following subsections.

4.2. Data With Low Intrinsic Dimension

We now turn to provide a result, which demonstrates that for any dataset which is realizable using a two-layer network of a given width n (i.e. $L_S\left(N_n\left(W, \mathbf{v}\right)\right) = 0$ for some $(W, \mathbf{v})$), the probability of initializing from a basin containing a good minimum increases as we add more neurons to the first layer, corresponding to the idea of overspecification. We note that this result holds without significant additional assumptions, but on the flip side, the number of neurons required to guarantee a constant success probability increases exponentially with the intrinsic dimension of the data (rank (X), where X is the data matrix whose rows are $\mathbf{x}_1, \ldots, \mathbf{x}_m$), so a magnitude of $\Omega\left(n^{\text{rank}(X)}\right)$ neurons is required. Thus, the result is only meaningful when the intrinsic dimension and n are modest. In the next subsection, we provide results which require a more moderate amount of overspecification, under other assumptions.

To avoid making the result too complex, we will assume for simplicity that we use the squared loss $\ell(y, y') = (y - y')^2$ and that $\|\mathbf{x}_t\| \le 1$ for any training instance $\mathbf{x}_t$. However, an analogous result can be shown for any convex loss, with somewhat different dependencies on the parameters, and any other bound on the norms of the instances.

Theorem 3. *Assume each training instance $\mathbf{x}_t$ satisfies $\|\mathbf{x}_t\| \le 1$. Suppose that the training objective L_S refers to the average squared loss, and that $L_S\left(W^*, \mathbf{v}^*\right) = 0$ for some $(W^*, \mathbf{v}^*) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$ satisfying $|v_i^*| \cdot \|\mathbf{w}_i^*\| \le B \ \ \forall i \in [n]$, where B is some constant. For all $\epsilon > 0$, if*

$$\begin{aligned} p_\epsilon &= \frac{1}{2\pi\left(rank\left(X\right) - 1\right)}\left(\frac{\sqrt{\epsilon}}{nB}\sqrt{1 - \frac{\epsilon}{4n^2B^2}}\right)^{rank(X)-1} \\ &= \Omega\left(\left(\frac{\sqrt{\epsilon}}{nB}\right)^{rank(X)}\right), \end{aligned}$$

and we initialize a two-layer, width $c\lceil\frac{n}{p_\epsilon}\rceil$ network (for some $c \ge 2$), using a distribution satisfying Assumption

1, then

$$\mathbb{P}\left[Bas\left(W,\mathbf{v}\right)\le\epsilon\right]\ge 1-e^{-\frac{1}{4}cn}.$$

The proof idea is that with a large enough amount of overspecification, with high probability, there will be a subset of the neurons in the first layer for which the signs of their outputs on the data and the signs of their weights in the output neuron will resemble those of $(W^*,\mathbf{v}^*)$. Then, by using Lemma 2 we are able to argue that the initialization made in the remaining neurons does not degrade the value obtained in the aforementioned subset. We refer the reader to Appendix B.5 for the full proof.

4.3. Clustered or Full-rank Data

In this subsection, we will first show that when training on instances residing in high dimension d (specifically, when the dimension satisfies $m\le d$, where m is the number of training examples), we initialize at a good basin with high probability. Building on this result, we show that even when $m>d$, we still initialize at a good basin with high probability, as long as the data is clustered into $k\le d$ sufficiently small clusters.

Specifically, we begin by assuming that our data matrix X satisfies $\operatorname{rank}(X)=m$. We note that this immediately implies $m\le d$. This refers to data of very high intrinsic dimension, which is in a sense the opposite regime to the one considered in the previous subsection (where the data was assumed to have low intrinsic dimension). Even though this regime might be strongly prone to overfitting, this allows us to investigate the surface area of the objective function effectively, while also serving as a base for the clustered data scenario that we will be studying in Thm. 5.

We now state our formal result for such datasets, which implies that under the rank assumption, a two-layer network of size $\mathcal{O}(\log(m))$ is sufficient to initialize in a basin with a global minimum with overwhelming probability.

Theorem 4. *Assume rank $(X)=m$, and let the target outputs $y_1,\dots,y_m$ be arbitrary. For any n, let α be the minimal objective value achievable with a width n two-layer network. Then if $(W,\mathbf{v})\in\mathbb{R}^{n\times d}\times\mathbb{R}^n$ is initialized according to Assumption 1,*

$$\mathbb{P}\left[Bas\left(W,\mathbf{v}\right)\le\alpha\right]\ \ge\ 1-m\left(\frac{3}{4}\right)^n.$$

We refer the reader to Appendix B.6 for the full proof of the theorem.

As mentioned earlier, training on $m\le d$ examples, without imposing any regularization, is prone to overfitting. Thus, to say something meaningful in the $m>d$ regime, we will consider an extension of the previous result, where instead of having fewer data points than dimensions d, we assume that the training instances are composed of $k\le d$ relatively small clusters in general position. Intuitively, if the clusters are sufficiently small, the surface of the objective function will resemble that of having $k\le d$ data points, and will have a similar favorable structure.

We also point out that in a similar manner to as we did in Thm. 3, the theorem statement assumes that the objective function refers to the average squared loss over the data. However, the proof does not rely on special properties of this loss, and it is possible to generalize it to other convex losses (perhaps with a somewhat different resulting bound).

Theorem 5. *Consider the squared loss, and suppose our data is clustered into $k\le d$ clusters. Specifically, we assume there are cluster centers $\mathbf{c}_1,\dots,\mathbf{c}_k\in\mathbb{R}^d$ for which the training data $S=\{\mathbf{x}_t,y_t\}_{t=1}^m$ satisfies the following:*

- *$\exists\delta_1,\dots,\delta_k>0$ s.t. for all $\mathbf{x}_t$, there is a unique $j\in[k]$ such that $\|\mathbf{c}_j-\mathbf{x}_t\|\le\delta_j$.*
- *$\forall j\in[k]$ $\frac{\delta_j}{\|\mathbf{c}_j\|}\le 2\sin\left(\frac{\sqrt{2\pi}}{16d\sqrt{d}}\right)$ and $\forall j\in[k]$ $\|\mathbf{c}_j\|\ge c$ for some $c>0$.*
- *$\forall t\in[m]$ $\|\mathbf{x}_t\|\le B$ for some $B\in\mathbb{R}$.*
- *For some fixed γ, it holds that $|y_t-y_{t'}|\le\gamma\|\mathbf{x}_t-\mathbf{x}_{t'}\|_2$ for any $t,t'\in[m]$ such that $\mathbf{x}_t,\mathbf{x}_{t'}$ are in the same cluster.*

Let $\delta=\max_j\delta_j$. Denote as C the matrix which rows are $\mathbf{c}_1,\dots,\mathbf{c}_k$, and let $\sigma_{\max}\left(C^\top\right),\sigma_{\min}\left(C^\top\right)$ denote the largest and smallest singular values of $C^\top$ respectively. Let $\mathsf{c}(\mathbf{x}_t):\mathbb{R}^d\to\mathbb{R}^d$ denote the mapping of $\mathbf{x}_t$ to its nearest cluster center $\mathbf{c}_j$ (assumed to be unique), and finally, let $\hat{\mathbf{y}}=(\hat{y}_1,\dots,\hat{y}_k)\in\mathbb{R}^k$ denote the target values of arbitrary instances from each of the k clusters. Then if $(W,\mathbf{v})\in\mathbb{R}^{n\times d}\times\mathbb{R}^n$ is initialized from a distribution satisfying Assumption 1,

$$\mathbb{P}\left[Bas\left(W,\mathbf{v}\right)\le\mathcal{O}\left(\delta^2\right)\right]\ge 1-d\left(\frac{7}{8}\right)^n$$

Where the big $\mathcal{O}$ notation hides quadratic dependencies on $B,c^{-1},n,\sigma_{\min}^{-2}\left(C^\top\right),\sigma_{\max}\left(C^\top\right),\gamma,\|\hat{\mathbf{y}}\|_2$ (see the proof provided in Appendix B.7 for an explicit expression).

Note that δ measures how tight the clusters are, whereas $c,\sigma_{\max}\left(C^\top\right)$ and $\sigma_{\min}\left(C^\top\right)$ can be thought of as constants assuming the cluster centers are in general position. So, the theorem implies that for sufficiently tight clusters, with overwhelming probability, we will initialize from a basin containing a low-valued minimum, as long as the network size is $\Omega(\log(d))$.

Acknowledgements

This research is supported in part by an FP7 Marie Curie CIG grant, Israel Science Foundation grant 425/13, and the Intel ICRI-CI Institute. We thank Lukasz Kaiser for pointing out a bug (as well as the fix) in an earlier version of the paper.

References

Andoni, Alexandr, Panigrahy, Rina, Valiant, Gregory, and Zhang, Li. Learning polynomials with neural networks. In *ICML*, 2014.

Arora, Sanjeev, Bhaskara, Aditya, Ge, Rong, and Ma, Tengyu. Provable bounds for learning some deep representations. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 584–592, 2014.

Auer, Peter, Herbster, Mark, and Warmuth, Manfred K. Exponentially many local minima for single neurons. In *NIPS*, 1996.

Bach, Francis. Breaking the curse of dimensionality with convex neural networks. *arXiv preprint arXiv:1412.8690*, 2014.

Bengio, Yoshua, Roux, Nicolas L, Vincent, Pascal, Delalleau, Olivier, and Marcotte, Patrice. Convex neural networks. In *Advances in neural information processing systems*, pp. 123–130, 2005.

Blum, Avrim L and Rivest, Ronald L. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1): 117–127, 1992.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gérard Ben, and LeCun, Yann. The loss surface of multilayer networks. *arXiv preprint arXiv:1412.0233*, 2014.

Cybenko, George. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

Goodfellow, Ian J and Vinyals, Oriol. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Haeffele, Benjamin D and Vidal, René. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*, 2015.

Janzamin, Majid, Sedghi, Hanie, and Anandkumar, Anima. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *CoRR abs/1506.08473*, 2015.

Kumagai, Sadatoshi. An implicit function theorem: Comment. *Journal of Optimization Theory and Applications*, 31(2):285–288, 1980.

Leopardi, Paul. *Distributing points on the sphere: partitions, separation, quadrature and energy*. PhD thesis, University of New South Wales, 2007.

Li, Shengqiao. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011.

Livni, Roi, Shalev-Shwartz, Shai, and Shamir, Ohad. On the computational efficiency of training neural networks. In *NIPS*, pp. 855–863, 2014.

Zhang, Yuchen, Lee, Jason D, Wainwright, Martin J, and Jordan, Michael I. Learning halfspaces and neural networks with random initialization. *arXiv preprint arXiv:1511.07948*, 2015.