Learning Network Parameters in the ReLU Model

Arya Mazumdar
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003, USA
arya@cs.umass.edu

Ankit Singh Rawat
Google Research
New York, NY 10011, USA
ankitsrawat@google.com

Abstract

Rectified linear units, or ReLUs, have become a preferred activation function for artificial neural networks. In this paper we consider the problem of learning a generative model in the presence of nonlinearity (modeled by the ReLU functions). Given a set of signal vectors $\mathbf{y}^i \in \mathbb{R}^d, i = 1, 2, \ldots, n$, we aim to learn the network parameters, i.e., the $d \times k$ matrix A, under the model $\mathbf{y}^i = \mathrm{ReLU}(A\mathbf{c}^i + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^d$ is a random bias vector, and $\mathbf{c}^i \in \mathbb{R}^k$ are arbitrary unknown latent vectors. We show that it is possible to recover the column space of A within an error of $O(d)$ (in Frobenius norm) under certain conditions on the distribution of $\mathbf{b}$.

1 Introduction

Rectified Linear Unit (ReLU) is a basic nonlinear function defined to be $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}_+ \cup \{0\}$ as $\mathrm{ReLU}(x) \equiv \max(0, x)$. For any matrix X, $\mathrm{ReLU}(X)$ denotes the matrix obtained by applying the ReLU function on each of the coordinates of the matrix X. ReLUs are building blocks of many nonlinear data-fitting problems based on deep neural networks (see, e.g., [20] for a good exposition). In particular, [7] showed that supervised training of very deep neural networks is much faster if the hidden layers are composed of ReLUs.

Let $\mathcal{Y} \subset \mathbb{R}^d$ be a collection of signal vectors that are of interest to us. Depending on the application at hand, the signal vectors, i.e., the constituents of $\mathcal{Y}$, may range from images, speech signals, network access patterns to user-item rating vectors and so on. We assume that the signal vectors satisfy a generative model, where each signal vector can be approximated by a map $g : \mathbb{R}^k \to \mathbb{R}^d$ from the latent space to the ambient space, i.e., for each $\mathbf{y} \in \mathcal{Y}$,

$$\mathbf{y} \approx g(\mathbf{c}) \text{ for some } \mathbf{c} \in \mathbb{R}^k. \tag{1}$$

In this paper we consider the following specific model (*single layer* ReLU-network), with the weight (*generator*) matrix $A \in \mathbb{R}^{d \times k}$ and bias $\mathbf{b} \in \mathbb{R}^d$:

$$\mathbf{y} \approx \mathrm{ReLU}(A\mathbf{c} + \mathbf{b}). \tag{2}$$

The generative model in (2) raises multiple interesting questions that play fundamental role in understanding the underlying data and designing systems and algorithms for information processing. Here, we consider the following network parameter learning problem under the specific generative model of (2).

Learning the network parameters: Given the n observations $\{\mathbf{y}^i\}_{i \in [n]} \subset \mathbb{R}^d$ from the model (cf. (2)), recover the parameters of the model, i.e., $A \in \mathbb{R}^{d \times k}$ such that

$$\mathbf{y}^i = \mathrm{ReLU}(A\mathbf{c}^i + \mathbf{b}) \; \forall \; i \in [n], \tag{3}$$

with latent vectors $\{\mathbf{c}^i\}_{i \in [n]} \subset \mathbb{R}^k$. We assume that the bias vector $\mathbf{b}$ is a random vector comprising of i.i.d. coordinates with each coordinate distributed according to the probability density function

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

$p(\cdot)$. This question is closely related to the dictionary-learning problem [16]. We also note that this question is different from the usual task of training a model (such as, [11]), in which case the set $\{\mathbf{c}^i\}_{i\in[n]}$ is also known (and possibly chosen accordingly) in addition to $\{\mathbf{y}^i\}_{i\in[n]}$.

Related works. There have been a recent surge of interest in learning ReLUs, and the above question is of basic interest even for a single-layer network (i.e., nonlinearity comprising of a single ReLU function). It is conceivable that understanding the behavior of a single-layer network would allow one to use some iterative peeling off technique to develop a theory for the generative models comprising of multiple layers.

To the best of our knowledge, the network parameter learning problem, even for single-layer networks has not been studied as such, i.e., theoretical guarantees do not exist. Only in a very recent paper [22] the unsupervised problem was studied when the latent vectors $\{\mathbf{c}^i\}_{i\in[n]}$ are random Gaussian. The principled approaches to solve this unsupervised problem in practice reduce this to the 'training' problem, such as the autoencoders [10] that learn features by extensive end-to-end training of encoder-decoder pairs; or use the recently popular generative adversarial networks (GAN) [9] that utilize a discriminator network to tune the generative network. The method that we are going to propose here can be seen as an alternative to using GANs for this purpose, and can be seen as an isolated 'decoder' learning of the autoencoder. Note that the problem bears some similarity with matrix completion problems, a fact we greatly exploit. In matrix completion, a matrix M is visible only partially, and the task is to recover the unknown entries by exploiting some prior knowledge about M. In the case of (3), we are more likely to observe the positive entries of the matrix M, which, unlike a majority of matrix completion literature, creates the dependence between M and the sampling procedure.

Main result and technique. We define the $d \times n$ observation matrix $Y = [\mathbf{y}^1\ \mathbf{y}^2\ \cdots\ \mathbf{y}^n]$. Similarly, we define the $k \times n$ coefficient matrix $C = [\mathbf{c}^1\ \mathbf{c}^2\ \cdots\ \mathbf{c}^n]$. With this notion, we can concisely represent the n observation vectors as

$$Y = \mathrm{ReLU}\left(AC + \mathbf{b} \otimes \mathbf{1}^T\right) = \mathrm{ReLU}\left(M + \mathbf{b} \otimes \mathbf{1}^T\right), \tag{4}$$

where $M = AC$ and $\mathbf{1} \in \mathbb{R}^n$ denotes the all-ones vector and $\otimes$ denotes the Kronecker product. Recall our assumption that $\mathbf{b}$ is a random vector comprising of i.i.d. coordinates with each coordinate distributed according to the probability density function $p(\cdot)$.

Note that this model ensures that the bias corresponding to each coordinate is random, but does not change over different signal vectors. We employ a natural approach to learn the underlying weight matrix A from the observation matrix Y. As the network maps a lower dimensional latent vector $\mathbf{c} \in \mathbb{R}^k$ to obtain a signal vector $\mathbf{y} = \mathrm{ReLU}(A\mathbf{c} + \mathbf{b})$ in dimension $d > k$, the matrix $M = AC$ (cf. (4)) is a low-rank matrix as long as $k < \min\{d, n\}$. In our quest of recovering the weight matrix A, we first focus on estimating the matrix M, when given access to Y. This task can be viewed as estimating a *low-rank* matrix from its partial (randomized) observations.

One of the main challenges that we face here is that while an entry of the matrix Y is a random variable (since $\mathbf{b}$ is a random bias), whether that is being observed or being cut-off by the ReLU function (for being negative) depends on the value of the entry itself. In general matrix completion literature, the entries of the matrix being observed are sampled independent of the underlying matrix itself (see, e.g., [3, 12, 4] and references therein). For this reason, we cannot use most of these results off-the-shelf. However, similar predicament is (partially) present in [5], where entries are quantized while being observed. This motivates us to employ a maximum-likelihood method inspired by [5].

That said, our observation model differs from [5] in a critical way: in our case the bias vector, while random, does not change over observations. This translates to less freedom during the transformation of the original matrix to the observed matrix, leading to dependence among the elements in a row. Furthermore, the analysis becomes notably different since the positive observations are not quantized. Furthermore, although our formulation is close to [5], because of the aforementioned differences in the observation models, we get much stronger guarantee on the recovery of the matrix. Indeed, our results are comparable to analogous results of [21, 13] that also study the quantized matrix completion problem.

We show that our method guarantees the recovery of the matrix AC from Y with an error in Frobenius norm at most $O(\sqrt{d})$ with high probability (see Theorem 1 for the formal statement). Then leveraging the well known results on matrix perturbation [23], it is possible to also recover the column space of A with a similar guarantee.

Extension to multi-layer networks. Our aim is to use a 'peeling' technique to extend our results to multi-layer networks. While rigorous theoretical guarantees for the network parameter learning problem do not extend to multi-layer networks as of now, we can still use the peel-off decoding for this case. Note that, even for the 'training' problem of such network, which is a less demanding task than our problem, no rigorous guarantee exists beyond only two-layers. For example, the state-of-the art theoretical results for training such as in [11], hold only for one layer of nonlinearity (the second layer is assumed to have a linear activation function in [11]). In fact most recent theoretical guarantees in this domain are restricted to one or two layer networks e.g., [25, 15, 6, 8, 18, 24]. Given this, obtaining provable guarantees for multi-layer case of network parameter learning problem seems to be a challenging future work.

2 Learning parameters in a single-layer ReLU-network

We now focus on the task of estimating M from the observation matrix Y (cf. (4)). For $i \in [d]$, we define $\mathcal{N}_Y(i) \subseteq [n]$ as the set of positive coordinates of the i-th row of the matrix Y, i.e.,

$$\mathcal{N}_Y(i) = \{j \in [n] : Y_{i,j} > 0\} \text{ and } N_{Y,i} = |\mathcal{N}_Y(i)|.$$

Note that, for $i \in [d]$, the original matrix M needs to satisfy

$$M_{i,j} + b_i = Y_{i,j} \text{ for } j \in \mathcal{N}_Y(i) \tag{5}$$

where $M_{i,j}$ denotes the (i,j)-th entry of M, and

$$M_{i,j} + b_i < 0 \text{ for } j \in \overline{\mathcal{N}}_Y(i) := [n] \backslash \mathcal{N}_Y(i). \tag{6}$$

For $i \in [d]$ and $j \in [n]$, let $M_{i,(j)}$ denote the j-th largest element of the i-th row of M, i.e., for $i \in [d]$, $M_{i,(1)} \geq M_{i,(2)} \geq \cdots \geq M_{i,(n)}$. It is straightforward to verify from (5) that $\mathcal{N}_Y(i)$ denotes the indices of $N_{Y,i}$ largest entries of the i-th row of M. Furthermore, with $N_{Y,i} \in [n]$, we have $b_i = Y_{i,(1)} - M_{i,(1)} = \cdots = Y_{i,(N_{Y,i})} - M_{i,(N_{Y,i})}$. Similarly, it follows from (6) that whenever we have $N_{Y,i} = 0$, then b_i belongs to the interval $(-\infty, -\max_{j\in[n]} M_{i,j}) = (-\infty, -M_{i,(1)})$. Based on these observation, we define the set of matrices $\mathcal{X}_{Y,\nu,\gamma} \subset \mathbb{R}^{d\times n}$ as

$$\mathcal{X}_{Y,\nu,\gamma} = \big\{X : \|X\|_\infty \leq \gamma;\ Y_{i,(1)} - X_{i,(2)} = \cdots = Y_{i,(N_{Y,i})} - X_{i,(N_{Y,i})};\ \text{and}\ X_{i,(N_{Y,i})} \geq \max_{j\in\overline{\mathcal{N}}_Y(i)} X_{i,j} + \nu\ \forall i \in [d]\big\}. \tag{7}$$

Recall that, $p : \mathbb{R} \to \mathbb{R}$ denotes the probability density function of each bias RV B. We use the notation $F(x_1, x_2) = \mathbb{P}(-x_1 \leq B \leq -x_2)$ and $X_i^* = \max_{j\in[n]} X_{i,j}$. Thus, the (normalized) log-likelihood of observing Y given that X is the original matrix takes the following form [17].

$$\overline{\mathcal{L}}_Y(X) = \sum_{i\in[d]} \Big(\mathbb{1}_{\{N_{Y,i}=0\}} \cdot \log \frac{F(\infty, X_i^*)}{F(\infty, 0)} + \sum_{s=1}^{n} \mathbb{1}_{\{N_{Y,i}=s\}} \cdot \log \frac{p(Y_{i,(s)} - X_{i,(s)})}{p(Y_{i,(s)})} \Big). \tag{8}$$

In order to recover the matrix M from the observation matrix Y, we employ the following program.

$$\underset{X\in\mathbb{R}^{d\times n}}{\text{maximize}} \quad \overline{\mathcal{L}}_Y(X) \text{ subject to } X \in \mathcal{X}_{Y,\nu,\gamma}. \tag{9}$$

Define $\omega_{p,\gamma,\nu}$ to be such that $F(x,y) \geq \omega_{p,\gamma,\nu}$ for all $x, y \in [-\gamma, \gamma]$ with $|x - y| > \nu$. In what follows, we simply refer this quantity as ω_p given that γ and ν are clear from context. Further define the following flatness and Lipschitz parameters associated with a function $f : \mathbb{R} \to \mathbb{R}$:

$$\beta_\alpha(f) := \inf_{|x|\leq\alpha} \frac{|f'(x)|^2}{4|f(x)|}, \tag{10}$$

$$L_\alpha(f) := \max\Big\{ \sup_{|x|\leq\alpha} \frac{f(x)}{\int_{-\infty}^{x} f(y)dy}, \sup_{|x|\leq\alpha} \frac{f'(x)}{f(x)} \Big\}. \tag{11}$$

The following result characterizes the performance of the program proposed in (9).

Theorem 1. *Assume that $\|M\|_\infty \leq \gamma$ and the observation matrix Y is related to M according to* (4)*. Let $\widehat{M}$ be the solution of the program specified in* (9)*, and the bias density function $p(x)$ is differentiable with bounded derivative. Then, the following holds with probability at least $1 - \frac{1}{d+n}$.*

$$\|M - \widehat{M}\|_F^2 \leq C_0 L_\gamma(p) \cdot \frac{\gamma d}{\beta_\gamma(p)\omega_p}, \tag{12}$$

where, C_0 is a constant. The quantities $\beta_\gamma(p)$ and $L_\gamma(p)$ depend on the distribution of the bias and are defined in (10) *and* (11)*, respectively.*

The full proof of Theorem 1 is omitted due to page limit. In the first step of the proof, we show that given the observation matrix Y, for any $X \in \mathcal{X}_{Y,\nu,\gamma}$ (cf. (7)), we have

$$\mathbb{E}\left[\overline{\mathcal{L}}_Y(M) - \overline{\mathcal{L}}_Y(X)\right] \geq \beta_\gamma(p)\omega_p \cdot \|M - X\|_F^2. \tag{13}$$

This fact follows from the regularity assumptions on the probability density p and a sequence of inequalities similar to ones that relate KL divergence and the Hellinger divergence. In the next step, we show that,

$$\mathbb{E}\left[\overline{\mathcal{L}}_Y(M) - \overline{\mathcal{L}}_Y(\widehat{M})\right] \leq 2 \sup_{X \in \mathcal{X}} \left|\overline{\mathcal{L}}_Y(X) - \mathbb{E}\left[\overline{\mathcal{L}}_Y(X)\right]\right|. \tag{14}$$

To upper bound the right-hand side we resort to standard techniques to bound the supremum of an empirical process such as the symmetrization and the contraction principle [14]. The detail can be found in the full version [17].

Recovering the network parameters. Let us denote the recovered matrix $\widehat{M}$ as $\widehat{M} = M + E$, where E is the perturbation matrix that has bounded Frobenius norm (cf. (12)). Now the task of recovering the parameters of the single-layer ReLU-network is equivalent to solving for A given $\widehat{M} = M + E = AC + E$. Note that, even without the perturbation E we could only hope to recover the column space of A and not the exact matrix A.

Let U_k and $\widehat{U}_k$ be the top k left singular vectors of M and $\widehat{M}$, respectively. Let σ_k, the smallest non-zero singular value of M, is at least $\delta > 0$. Then, it follows from standard results from matrix-perturbation theory (cf. [23]) that there exists an orthogonal matrix $O \in \mathbb{R}^{k \times k}$ such that

$$\|U_k - \widehat{U}_k O\|_F \leq \frac{2^{3/2}(2\sigma_1 + \|E\|) \cdot \min\{\sqrt{k}\|E\|, \|E\|_F\}}{\delta^2} \leq \frac{2^{3/2}(2\sigma_1 + \|E\|_F) \cdot \|E\|_F}{\delta^2},$$

which is a guarantee that the column space of A is recovered by SVD within an error of $O(d)$ in Frobenius norm by the column space of $\widehat{U}_k$. Note that σ_1 is the largest singular value of M.

Future directions: Extension to multi-layer networks. To learn a multi-layer ReLU network, we propose a 'peeling' decoder as defined below. For better understanding, let us consider a two-layer model as follows:

$$Y = \mathrm{ReLU}\Big(A_2 \mathrm{ReLU}\left(A_1 C + \mathbf{b}_1 \otimes \mathbf{1}^T\right) + \mathbf{b}_2\Big) = \mathrm{ReLU}\left(M_2 + \mathbf{b} \otimes \mathbf{1}^T\right), \tag{15}$$

where $M_2 = A_2 \mathrm{ReLU}\left(A_1 C + \mathbf{b}_1 \otimes \mathbf{1}^T\right)$. Our 'peeling' decoder will approach this layer-by-layer. First we use the likelihood-based matrix completion technique outlined in (9) to recover M_2 as in the case of a one-layer network. Note that we will not be able to recover M_2 exactly, as Theorem 1 guarantees recovery only up to a Frobenius norm error. This creates a hurdle to recover $A_1 C$, since our current method does not handle dense bounded norm noise while learning network parameters. Also it will not be realistic to assume a probabilistic model for this bounded noise, since in the peeling off process the noise is coming from decoding of the previous layer. Under certain coherence conditions on the matrix to be learned, it is possible to handle such situations (see e.g., [2, 19, 1]). Note that, such coherence condition must be satisfied by the parameters of every layer of the network. An amenable but practical algorithm that is resilient to dense noise, under reasonable assumption on the network parameters, is an immediate area of interest.

Even when this first hurdle can be crossed, we still have a task of factorization of M_2 to find A_2 and $\mathrm{ReLU}\left(A_1 C + \mathbf{b}_1 \otimes \mathbf{1}^T\right)$, and in general we cannot find a unique factorization. Here as well, with additional reasonable assumptions the factorization can be made unique. Note that this might be simpler that the factorization step of the one-layer network, since there is already a lot of structure in the latter matrix (such as nonnegativity due to being output of a ReLU and about $\sim 50\%$ sparsity).

Provided this can be made to work analytically for two layers, there should not be any theoretical issue left to extend the process to multiple layers.

References

[1] M. F. Balcan and H. Zhang. Noise-tolerant life-long matrix completion via adaptive sampling. In *Advances in Neural Information Processing Systems*, pages 2955–2963. 2016.

[2] E. J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, June 2010.

[3] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, Apr 2009.

[4] S. Chatterjee. Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43(1):177–214, 02 2015.

[5] M. A. Davenport, Y. Plan, E. van den Berg, and M. Wootters. 1-bit matrix completion. *Information and Inference: A Journal of the IMA*, 3(3):189–223, July 2014.

[6] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. In *International Conference on Learning Representations*, 2018.

[7] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[8] Surbhi Goel, Adam Klivans, and Raghu Meka. Learning one convolutional layer with overlapping patches. *arXiv preprint arXiv:1802.02547*, 2018.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680. 2014.

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] M. Janzamin, H. Sedghi, and A. Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

[12] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, June 2010.

[13] J. Lafond, O. Klopp, E. Moulines, and J. Salmon. Probabilistic low-rank matrix completion on finite alphabets. In *Advances in Neural Information Processing Systems*, pages 1727–1735, 2014.

[14] M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and processes*. Springer Science & Business Media, 2013.

[15] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, pages 597–607, 2017.

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[17] A. Mazumdar and A. S. Rawat. Representation learning and recovery in the ReLU model. *arXiv preprint arXiv:1803.04304*, 2018.

[18] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. *arXiv preprint arXiv:1804.06561*, 2018.

[19] S. Negahban and M. J. Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *Journal of Machine Learning Research*, 13(1):1665–1697, May 2012.

[20] M. Soltanolkotabi. Learning relus via gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2004–2014. 2017.

[21] A. Soni, S. Jain, J. Haupt, and S. Gonella. Noisy matrix completion under sparse factor models. *IEEE Transactions on Information Theory*, 62(6):3636–3661, 2016.

[22] Shanshan Wu, Alexandros Dimakis, and Sujay Sanghavi. Learning distributions generated by one-layer relu networks. *arXiv preprint arXiv:1909.01812*, 2019.

[23] Y. Yu, T. Wang, and R. J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

[24] Xiao Zhang, Yaodong Yu, Lingxiao Wang, and Quanquan Gu. Learning one-hidden-layer relu networks via gradient descent. *arXiv preprint arXiv:1806.07808*, 2018.

[25] Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 4140–4149. PMLR, 2017.