Learning and generalization of one-hidden-layer neural networks, going beyond standard Gaussian data

Hongkang Li
Rensselaer Polytechnic Institute
Troy, NY, USA
lih35@rpi.edu

Shuai Zhang
Rensselaer Polytechnic Institute
Troy, NY, USA
zhangs21@rpi.edu

Meng Wang
Rensselaer Polytechnic Institute
Troy, NY, USA
wangm7@rpi.edu

***Abstract*—This paper analyzes the convergence and generalization of training a one-hidden-layer neural network when the input features follow the Gaussian mixture model consisting of a finite number of Gaussian distributions. Assuming the labels are generated from a teacher model with an unknown ground truth weight, the learning problem is to estimate the underlying teacher model by minimizing a non-convex risk function over a student neural network. With a finite number of training samples, referred to the sample complexity, the iterations are proved to converge linearly to a critical point with guaranteed generalization error. In addition, for the first time, this paper characterizes the impact of the input distributions on the sample complexity and the learning rate.**

***Index Terms*—Gaussian mixture model, convergence, generalization, sample complexity, neural networks**

I. Introduction

The recent success of machine learning mainly benefits from the developments of neural networks. It is surprising to witness the superior empirical performance in various applications ranging from imaging processing [1], [2] to natural language processing [3]. However, the lack of theoretical generalization guarantee has become a rising concern with the widespread utilization of neural networks. That is, the learned model from a finite number of training samples has guaranteed test error on the unseen data. From the perspective of optimization, achieving guaranteed generalization is to obtain both a small training error and a bounded generalization gap (difference between the training and test errors), which are referred to convergence and generalization analysis. Despite the high test accuracy in numerical experiments, the convergence and generalization analysis are limited, and significant breakthroughs focus on shallow neural networks [4]–[6]. Nevertheless, it is well-known that training a one-hidden-layer neural network with only three nodes is NP-hard [7], and various assumptions are imposed to ensure feasible analysis.

One representative line of works studies the overparameterized neural networks, where the number of trainable parameters is far larger than the number of training samples. In particular, the optimization problem of training on overparameterized networks has no spurious local minima [8]–[11], and thus the random initialization can achieve bounded training errors. However, such learned models may suffer from overfitting and cannot provide a generalization guarantee without additional assumptions. With the assumptions of infinite width of neurons, training a neural network via gradient descent can be approximated by a differential equation from the Mean Field (MF) view [4], [12]–[14] or a Gaussian kernel processing from the Neural Tangent Kernel (NTK) view [5], [15]–[18]. In particular, for one-hidden-layer neural networks, [19]–[22] provide bounded generalization errors for some target functions with nice properties, i.e., even or infinitely smooth functions. Nevertheless, both MF and NTK frameworks require a significantly larger number of neurons than that in practical cases.

For training neural networks with a fixed number of neurons, the landscapes of the objective functions are highly non-convex. They can contain intractably many spurious local minima [23] with multiple neurons. This line of work follows the "teacher-student" setup, where the training data is generated by a teacher neural network. The learning is performed on a student network by minimizing the empirical risk functions of the training data. To achieve the optimal with guaranteed generalization, most of the theoretical works with a fixed number of neurons assume the input belongs to standard Gaussian distributions (zero mean and unit variance) for feasible analysis [6], [24]–[28] with a few exceptions [29], [30]. Nevertheless, the neural network considered in [29], [30] only contains a single neuron in the hidden layer, and the objective function has a large benign landscape near the desired point such that random initialization succeeds with constant probability regardless of the non-convexity. Other works considering non-Gaussian input features either require an infinitely large number of neurons [4], [31] or cannot provide bounded generalization analysis [32]–[36].

Overall, the generalization analysis of neural networks beyond the standard Gaussian input is less investigated and has not received enough attention. On the one hand, Gaussian

This work was supported in part by AFOSR FA9550-20-1-0122, ARO W911NF-21-1-0255, NSF 1932196, and the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), part of the IBM AI Horizons Network (http://ibm.biz/AIHorizons).

mixture models are widely employed in plenty of applications, including data clustering [37]–[39], image segmentation [40], and few-shot learning [41]. On the other hand, the study of Gaussian mixture models casts an insight into understanding how the mean and variance of the input features affect the performance. In practice, the input distribution will affect the learning performance, and several data pre-processing techniques such as data whitened [42] and batch normalization [43] have been applied to accelerate the convergence rate and improve the generalization error. Although various works [44]–[46] have studied the enormous success of batch normalization from different scopes and provided different explanations, none of them have studied from the perspective of generalization analysis.

Contributions: This paper provides a theoretical analysis of learning one-hidden-layer neural networks when the input distribution follows a Gaussian mixture model containing an arbitrary number of Gaussian distributions with arbitrary mean. Specific contributions include:

1. **Proposition of a gradient descent algorithm and tensor initialization with Gaussian mixture inputs:** This is the first paper to propose a gradient descent with tensor initialization method for Gaussian mixture inputs.

2. **Guaranteed generalization error with Gaussian mixture inputs for binary classification problems:** For the first time, we give a guaranteed generalization error for binary classification problems when the inputs follow Gaussian mixture model.

3. **First sample complexity analysis depending on the parameters of the input distribution:** Our theorem provides the first explicit characterization of the required number of samples to learn the neural network.

Notations: Vectors are in bold lowercase, and matrices and tensors are in bold uppercase. Scalars are in normal fonts. Sets are in calligraphic fonts. For example, $\boldsymbol{Z}$ is a matrix, and $\boldsymbol{z}$ is a vector. $\mathcal{Z}$ is a set. z_i denotes the i-th entry of $\boldsymbol{z}$, and $Z_{i,j}$ denotes the (i,j)-th entry of $\boldsymbol{Z}$. $[K]$ $(K>0)$ denotes the set including integers from 1 to K. $\boldsymbol{I}_d \in \mathbb{R}^{d\times d}$ and $\boldsymbol{e}_i$ represent the identity matrix in $\mathbb{R}^{d\times d}$ and the i-th standard basis vector, respectively. We use $\delta_i(\boldsymbol{Z})$ to denote the i-th largest singular value of $\boldsymbol{Z}$. The matrix norm is defined as $\|\boldsymbol{Z}\| = \delta_1(\boldsymbol{Z})$. The gradient and the Hessian of a function $f(\boldsymbol{W})$ are denoted by $\nabla f(\boldsymbol{W})$ and $\nabla^2 f(\boldsymbol{W})$, respectively.

Given a tensor $\boldsymbol{T} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and matrices $\boldsymbol{A} \in \mathbb{R}^{n_1\times d_1}$, $\boldsymbol{B} \in \mathbb{R}^{n_2\times d_2}$, $\boldsymbol{C} \in \mathbb{R}^{n_3\times d_3}$, the (i_1,i_2,i_3)-th entry of the tensor $\boldsymbol{T}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$ is given by

$$\sum_{i_1'}^{n_1}\sum_{i_2'}^{n_2}\sum_{i_3'}^{n_3} \boldsymbol{T}_{i_1',i_2',i_3'}\boldsymbol{A}_{i_1',i_1}\boldsymbol{B}_{i_2',i_2}\boldsymbol{C}_{i_3',i_3}. \tag{1}$$

We follow the convention that $f(x) = O(g(x))$ (or $\Omega(g(x))$, $\Theta(g(x))$) means that $f(x)$ is at most, at least, or in the order of $g(x)$, respectively.

II. Problem Formulation

In this paper, we consider the distribution $(\mathcal{X},\mathcal{Y})$ over $(\boldsymbol{x},y) \in \mathbb{R}^d \times \{+1,-1\}$, where $\mathcal{X}$ is the Gaussian mixture distribution [47]–[49] such that

$$\boldsymbol{x} \sim \sum_{l=1}^{L} \lambda_l \mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{I}_d), \tag{2}$$

where $\lambda_l \in [0,1]$ with $\sum_l \lambda_l = 1$, and $\mathcal{N}$ denotes the multi-variate Gaussian distribution with mean $\boldsymbol{\mu}_l \in \mathbb{R}^d$, and covariance $\boldsymbol{I}_d$ for all $l \in [L]$. Let $\boldsymbol{M} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_L)$ and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_L)$. The teacher model in this paper is a fully connected neural network consisting of an input layer, a hidden layer, followed by a pooling layer. The number of neurons in the hidden layer is denoted as K. All neurons are equipped with Sigmoid activation functions[1] $\phi(x) = \frac{1}{1+\exp(-x)}$. For any input $\boldsymbol{x} \in \mathcal{X}$, the output of the teacher model is denoted as

$$H(\boldsymbol{W}^*;\boldsymbol{x}) := \frac{1}{K}\sum_{j=1}^{K} \phi({\boldsymbol{w}_j^*}^\top \boldsymbol{x}), \tag{3}$$

where $\boldsymbol{W}^* = [\boldsymbol{w}_1^*, ..., \boldsymbol{w}_K^*] \in \mathbb{R}^{d\times K}$ is an unknown fixed weight matrix, and $\boldsymbol{w}_j^* \in \mathbb{R}^d$ is the weight of the j-th neuron in the hidden layer. The corresponding output label y satisfies

$$\mathbb{P}(y=1|\boldsymbol{x}) = H(\boldsymbol{W}^*;\boldsymbol{x}). \tag{4}$$

The training process is over a student neural network with the same architecture as the teacher model, and the trainable parameters are denoted as $\boldsymbol{W} \in \mathbb{R}^{d\times K}$. Given n pairs of training samples $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ from $(\mathcal{X},\mathcal{Y})$, the empirical risk function is

$$f_n(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{W};\boldsymbol{x}_i,y_i), \tag{5}$$

and $\ell(\boldsymbol{W};\cdot,\cdot)$ is the cross-entropy loss function as

$$\begin{aligned}\ell(\boldsymbol{W};\boldsymbol{x},y) = &-y\cdot\log(H(\boldsymbol{W};\boldsymbol{x}))\\ &-(1-y)\cdot\log(1-H(\boldsymbol{W};\boldsymbol{x})).\end{aligned} \tag{6}$$

The neural network training problem is to minimize the following non-convex objective function:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{d\times K}} : \quad f_n(\boldsymbol{W}). \tag{7}$$

$\boldsymbol{W}^*$ may not be a global optimal solution to (7) because y is the quantization of $H(\boldsymbol{W}^*;\boldsymbol{x})$. However, $\boldsymbol{W}^*$ minimizes the expectation of (7) over $(\mathcal{X},\mathcal{Y})$. Note that for any permutation matrix $\boldsymbol{P} \in \mathbb{R}^{K\times K}$, we have $H(\boldsymbol{W};\boldsymbol{x}) = H(\boldsymbol{W}\boldsymbol{P};\boldsymbol{x})$, and $f_n(\boldsymbol{W}\boldsymbol{P}) = f_n(\boldsymbol{W})$. Hence, the estimation is considered successful if one finds weights $\boldsymbol{W}$ close to any column permutation of $\boldsymbol{W}^*$.

III. Algorithm

We propose a gradient descent algorithm with tensor initialization method to solve (7). The method starts from an initialization $\boldsymbol{W}_0 \in \mathbb{R}^{d\times K}$ computed from the tensor initialization method (Subroutine 1) and then updates the iteration

[1]The results can be generalized to any even activation function ϕ with bounded ϕ, ϕ' and ϕ''. Examples include tanh and erf.

$\boldsymbol{W}_t$ using gradient descent with the step size η_0. The pseudo-code is summarized in Algorithm 1.

The tensor initialization algorithm is summarized in Subroutine 1. While existing tensor decomposition methods in [50] and [6] only apply for standard Gaussian distribution, we extend the methods to Gaussian mixture models. The intuition is similar to that in [6], where the directions of $\{\boldsymbol{w}_j^*\}_{j\in[K]}$ are first estimated through decomposing the high-order tensor, and then the corresponding magnitudes are estimated through solving a linear regression problem. However, the high-order tensors are defined in a fairly different way because of the difference between input distributions. Formally, the high order tensor $\boldsymbol{Q}_j$'s are defined in Definition 1.

Definition 1. *For $j = 1, 2, 3$, we define*

$$\boldsymbol{Q}_j := \mathbb{E}_{\boldsymbol{x}\sim\sum_{l=1}^{L}\lambda_l\mathcal{N}(\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)}[y\cdot(-1)^j p^{-1}(\boldsymbol{x})\nabla^{(j)}p(\boldsymbol{x})], \quad (8)$$

where $p(\boldsymbol{x})$ is

$$p(\boldsymbol{x}) = \sum_{l=1}^{L}\lambda_l(2\pi)^{-\frac{d}{2}}\exp\big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_l)^\top(\boldsymbol{x}-\boldsymbol{\mu}_l)\big). \quad (9)$$

Algorithm 1 Gradient Descent with Tensor Initialization

1: **Input:** Training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the step size $\eta_0 = \big(\sum_{l=1}^{L}\lambda_l(\|\boldsymbol{\mu}_l\|_\infty+1)^2\big)^{-1}$;
2: **Initialization:** $\boldsymbol{W}_0 \leftarrow$ Tensor initialization method via Subroutine 1;
3: **for** $t = 0, 1, \cdots, T-1$ **do**
3: $\quad\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta_0\nabla f_n(\boldsymbol{W})$
4: **end for**
5: **Output:** $\boldsymbol{W}_T$

Subroutine 1 Tensor Initialization Method

1: **Input:** Partition $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ into three disjoint subsets $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$;
2: Compute $\widehat{\boldsymbol{Q}}_2$ using $\mathcal{D}_1$. Estimate the subspace $\widehat{\boldsymbol{U}}$ with respect to the largest K eigenvectors of $\widehat{\boldsymbol{Q}}_2$;
3: Compute $\widehat{\boldsymbol{R}}_3 = \widehat{\boldsymbol{Q}}_3(\widehat{\boldsymbol{U}}, \widehat{\boldsymbol{U}}, \widehat{\boldsymbol{U}})$ from data set $\mathcal{D}_2$. Obtain $\{\hat{\boldsymbol{v}}_i\}_{i\in[K]}$ by decomposing $\widehat{R}_3$;
4: $\bar{\boldsymbol{w}}_i^* = \widehat{\boldsymbol{U}}\hat{\boldsymbol{v}}_i$ for $i \in [K]$;
5: Compute $\widehat{\boldsymbol{Q}}_1$ from data set $\mathcal{D}_3$. Estimate the magnitude $\widehat{\boldsymbol{z}}$ by solving the optimization problem

$$\widehat{\boldsymbol{z}} = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^K}\frac{1}{2}\|\widehat{\boldsymbol{Q}}_1 - \sum_{j=1}^{K}\alpha_j\bar{\boldsymbol{w}}_j^*\|^2;$$

6: **Return:** $\hat{z}_j\widehat{\boldsymbol{U}}\hat{\boldsymbol{v}}_j$ as the jth column of $\boldsymbol{W}_0$, $j \in [K]$.

Subroutine 1 estimates the direction and magnitude of $\boldsymbol{w}_j^*, j \in [K]$ separately. The direction vector is defined as $\bar{\boldsymbol{w}}_j^* := \boldsymbol{w}_j^*/\|\boldsymbol{w}_j^*\|$, and the corresponding magnitude $\|\boldsymbol{w}_j^*\|$ is denoted as z_j. Line 2 estimates the subspace $\widehat{\boldsymbol{U}}$ spanned by $\{\boldsymbol{w}_1^*, \cdots, \boldsymbol{w}_K^*\}$ using $\widehat{\boldsymbol{Q}}_2$. Lines 3-4 estimate the direction vector $\bar{\boldsymbol{w}}_j^*$ by employing the KCL algorithm [51]. Line 5 estimates the magnitude z_j. Finally, the returned estimation of $\boldsymbol{W}^*$ is calculated as $\hat{z}_j\widehat{\boldsymbol{U}}\hat{\boldsymbol{v}}_j$.

IV. Theoretical results

Theorem 1 indicates that with sufficient number of samples as in (10), the iterates returned by proposed Algorithm 1 converge linearly to a critical point $\widehat{\boldsymbol{W}}_n$ near a permutation of the ground truth, denoted as $\boldsymbol{W}^*\boldsymbol{P}^*$. The distance between $\widehat{\boldsymbol{W}}_n$ and $\boldsymbol{W}^*\boldsymbol{P}^*$ are upper bounded as a function of the number of samples in (12). Additionally, the required number of samples depends on $\mathcal{B}$, which is a function of input distributions in (2).

Theorem 1. *Given the samples from $(\mathcal{X}, \mathcal{Y})$ with size n satisfying*

$$n \geq n_{sc} := poly(\epsilon_0^{-1}, \kappa, K)\mathcal{B}(\boldsymbol{\lambda}, \boldsymbol{M})d\log^2 d \quad (10)$$

for some $\epsilon_0 \in (0, \frac{1}{4})$ and positive value functions $\mathcal{B}(\boldsymbol{\lambda}, \boldsymbol{M})$ and $q(\boldsymbol{\lambda}, \boldsymbol{M})$, with probability at least $1 - d^{-10}$, the iterates $\{\boldsymbol{W}_t\}_{t=1}^T$ returned by Algorithm 1 with step size $\eta_0 = O\Big(\big(\sum_{l=1}^{L}\lambda_l(\|\tilde{\boldsymbol{\mu}}_l\|_\infty+1)^2\big)^{-1}\Big)$ converge linearly to a critical point $\widehat{\boldsymbol{W}}_n$ with the rate of convergence $v = 1-K^{-2}q(\boldsymbol{\lambda}, \boldsymbol{M})$, i.e.,

$$||\boldsymbol{W}_t - \widehat{\boldsymbol{W}}_n||_F \leq v^t||\boldsymbol{W}_0 - \widehat{\boldsymbol{W}}_n||_F. \quad (11)$$

Moreover, there exists a permutation matrix $\boldsymbol{P}^*$ such that the distance between $\boldsymbol{W}^*\boldsymbol{P}^*$ and $\widehat{\boldsymbol{W}}_n$ is bounded by

$$||\widehat{\boldsymbol{W}}_n - \boldsymbol{W}^*\boldsymbol{P}^*||_F \leq O\Big(K^{\frac{5}{2}}\cdot\sqrt{d\log n/n}\Big). \quad (12)$$

Remark 1: From (10), the required number of samples for successful estimation is a linear function of the input feature dimension d and $\mathcal{B}(\boldsymbol{\lambda}, \boldsymbol{M})$, where $\mathcal{B}$ is a function of the input distribution parameters. Note that the degree of freedom of $\boldsymbol{W}^*$ is dK, the sample complexity in (10) is nearly order-wise optimal in terms of d. In addition, $\mathcal{B}(\boldsymbol{\lambda}, \boldsymbol{M})$ is an increasing function of any entry of $\boldsymbol{M}$ (from Corollary 1 below), indicating that the sample complexity increases as the mean of any Gaussian component increases.

Remark 2: From (11), the distance between the convergent point $\widehat{\boldsymbol{W}}_n$ and the ground truth $\boldsymbol{W}^*\boldsymbol{P}^*$ is in the order of $\sqrt{d\log n/n}$. When the number of samples increases, the convergent point moves closer to the ground truth, implying a smaller generalization error.

Remark 3: From (12), we can see that the iterations converge to $\widehat{\boldsymbol{W}}_n$ linearly, and the rate of convergence is $1 - K^{-2}q(\boldsymbol{\lambda}, \boldsymbol{M})$, denoted as ν. When the number of neurons decreases or the input distribution changes such that q increases, the rate of convergence decreases, indicating a faster convergence.

Corollary 1 states the impact of the distribution parameters λ and $\boldsymbol{M}$. Specifically, when the mean vectors $\{\boldsymbol{\mu}_l\}_{l=1}^M$ in (2) increases, the sample complexity n_{sc} increases as $\mathcal{B}(\boldsymbol{\lambda}, \boldsymbol{M})$ increases, and the convergence rate v increases as $q(\boldsymbol{\lambda}, \boldsymbol{M})$ decreases. Suppose some entry of $\boldsymbol{\mu}_l$ go to infinity, n_{sc} will go to infinity, and ν will converge to 1.

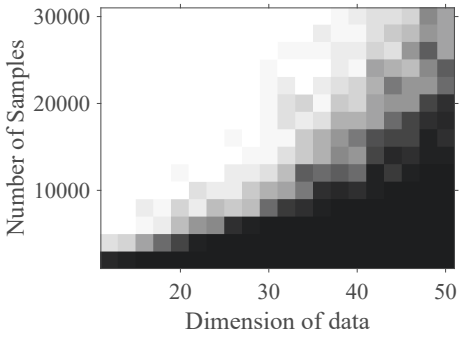

Fig. 1. The sample complexity against the feature dimension d

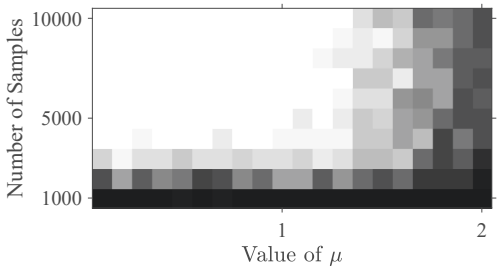

Fig. 2. The sample complexity when one mean changes

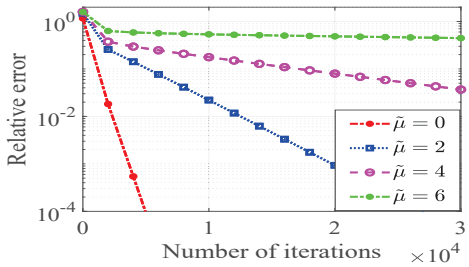

Fig. 3. The convergence rate with different $\tilde{\mu}$

Corollary 1. *Recall* $\boldsymbol{M} = (\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_L)$. *Let* $|\boldsymbol{\mu}_{l(i)}|$ *be the* i*-th entry of* $\boldsymbol{\mu}_l$*,*

1) $\mathcal{B}(\boldsymbol{\lambda}, \boldsymbol{M})$ *is an increasing function of* $|\boldsymbol{\mu}_{l(i)}|$.
2) $q(\boldsymbol{\lambda}, \boldsymbol{M})$ *is a decreasing function of* $|\boldsymbol{\mu}_{l(i)}|$.

V. Simulation

Here we present our results for numerical experiments. All simulations are implemented in MATLAB 2021b on a workstation with 3.40GHz Intel Core i7. The weights of the teacher model $\boldsymbol{W}^* \in \mathbb{R}^{d\times K}$ are randomly generated such that each entry of $\boldsymbol{W}^*$ belongs to $\mathcal{N}(0,1)$. The corresponding training samples $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ are randomly selected following (2) and (3) with the generated weights $\boldsymbol{W}^*$.

A. Sample complexity

We first show how the number of required samples is related to the dimension of data. The parameters of input distribution are selected as $L = 2$, $\lambda_1 = \lambda_2 = \frac{1}{2}$, $\boldsymbol{\mu}_1 = 0.5 * \mathbf{1}$ and $\boldsymbol{\mu}_2 = -\boldsymbol{\mu}_1$. The number of neurons in the hidden layer is set as 3. For a given $\boldsymbol{W}^*$, we initialize the starting point M times randomly and denote $\widehat{\boldsymbol{W}}_n^{(m)}$ as the output of Algorithm 1 with the m-th initialization, and $M = 20$. An experiment is viewed as a success if the average estimation error $e_{\boldsymbol{W}^*}$ is less than 10^{-3}, where $e_{\boldsymbol{W}^*}$ is the standard deviation of $(\widehat{\boldsymbol{W}}_n^{(1)}, \cdots, \widehat{\boldsymbol{W}}_n^{(L)})$.

In Fig. 1, we increase the feature dimension d from 12 to 50 by 2 and vary the number of samples n from 2×10^3 to 3×10^4. We test 20 independent experiments for each pair of d and n and then show the average success rate using grey blocks, where black ones mean rate 0 and white ones mean rate 1. In Fig. 1, the boundary line of black and white parts is almost straight, indicating an approximate linearity between n and d, which verifies our result in (10).

We then study the impact on the sample complexity when the mean value in the Gaussian mixture model changes. Set $d = 10$, $\lambda_1 = 0.4$, $\lambda_2 = 0.6$, $\boldsymbol{\mu}_1 = \mu \cdot \mathbf{1}$, $\boldsymbol{\mu}_2 = \mathbf{0}$. Fig. 2 shows that when the mean increases from 0 to 2, the sample complexity increases, matching our theoretical analyses in Section IV.

B. Convergence analysis

We next fix $d = 5$ and evaluate the convergence rate of Algorithm 1. We set $n = 1\times 10^4$, $\lambda_1 = \lambda_2 = 0.5$, $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_2 = C\cdot\mathbf{1}$ for a positive C. We vary C and let $\tilde{\mu} = \max_l \|\tilde{\boldsymbol{\mu}}_l\|_\infty$. Recall the relative error is defined as $\|\boldsymbol{W}_t - \widehat{\boldsymbol{W}}_n\|_F$ in (11). Fig. 3 shows the linear convergence of Algorithm 1 for different $\tilde{\mu}$ and the impact of $\|\tilde{\boldsymbol{\mu}}_l\|_\infty$. As predicted in Theorem 1, when $\tilde{\mu}$ increases, gradient descent converges slower.

VI. Conclusion

In this paper, we study the problem of learning a fully connected neural network when the input features belong to the Gaussian mixture model from the theoretical perspective. We propose a gradient descent algorithm with tensor initialization, and the iterates are proved to converge linearly to a critical point with guaranteed generalization. Additionally, we establish the sample complexity for successful recovery, and the sample complexity is proved to be dependent on the parameters of the input distribution. Possible future direction is to analyze the influence of variance on the learning performance.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[4] S. Mei, A. Montanari, and P.-M. Nguyen, "A mean field view of the landscape of two-layer neural networks," *Proceedings of the National Academy of Sciences*, vol. 115, no. 33, pp. E7665–E7671, 2018.

[5] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in neural information processing systems*, 2018, pp. 8571–8580.

[6] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 4140–4149. [Online]. Available: https://arxiv.org/pdf/1706.03175.pdf

[7] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," *Neural Networks*, vol. 5, no. 1, pp. 117–127, 1992.

[8] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in neural information processing systems*, 2014, pp. 855–863.

[9] Y. Li and Y. Yuan, "Convergence analysis of two-layer neural networks with ReLU activation," in *Advances in Neural Information Processing Systems*, 2017, pp. 597–607.

[10] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[11] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 742–769, 2018.

[12] L. Chizat and F. Bach, "On the global convergence of gradient descent for over-parameterized models using optimal transport," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, 2018, p. 3040–3050.

[13] C. Fang, Y. Gu, W. Zhang, and T. Zhang, "Convex formulation of overparameterized deep neural networks," *arXiv:1911.07626*, 2019.

[14] P.-M. Nguyen, "Mean field limit of the learning dynamics of multilayer neural networks," 2019.

[15] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 242–252.

[16] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eK3i09YQ

[17] D. Zou, Y. Cao, D. Zhou, and Q. Gu, "Gradient descent optimizes over-parameterized deep relu networks," *Machine Learning*, vol. 109, no. 3, pp. 467–492, 2020.

[18] D. Zou and Q. Gu, "An improved analysis of training over-parameterized deep neural networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, 2019.

[19] S. Mei, T. Misiakiewicz, and A. Montanari, "Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit," in *Proceedings of the Thirty-Second Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, A. Beygelzimer and D. Hsu, Eds., vol. 99. Phoenix, USA: PMLR, 25–28 Jun 2019, pp. 2388–2464.

[20] F. Wang, K. Li, C. Liu, Z. Mi, M. Shafie-Khah, and J. P. Catalão, "Synchronous pattern matching principle-based residential demand response baseline estimation: Mechanism analysis and approach description," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6972–6985, 2018.

[21] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in over-parameterized neural networks, going beyond two layers," in *Advances in neural information processing systems*, 2019, pp. 6155–6166.

[22] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," in *36th International Conference on Machine Learning, ICML 2019*. International Machine Learning Society (IMLS), 2019, pp. 477–502.

[23] I. Safran and O. Shamir, "Spurious local minima are common in two-layer relu neural networks," in *International Conference on Machine Learning*, 2018, pp. 4430–4438.

[24] S. Zhang, M. Wang, J. Xiong, S. Liu, and P.-Y. Chen, "Improved linear convergence of training cnns with generalizability guarantees: A one-hidden-layer case," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[25] S. Zhang, M. Wang, S. Liu, P.-Y. Chen, and J. Xiong, "Guaranteed convergence of training convolutional neural networks via accelerated gradient descent," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, 2020. [Online]. Available: doi: 10.1109/CISS48834.2020.1570627111

[26] S. Zhang, M. Wang, S. Liu, P.-Y. Chen, and J. Xiong, "Fast learning of graph neural networks with guaranteed generalizability:one-hidden-layer case," in *2020 International Conference on Machine Learning (ICML)*, 2020.

[27] S. Zhang, M. Wang, S. Liu, P.-Y. Chen, and J. Xiong, "Why lottery ticket wins? a theoretical perspective of sample complexity on pruned neural networks," in *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[28] H. Fu, Y. Chi, and Y. Liang, "Guaranteed recovery of one-hidden-layer neural networks via cross entropy," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3225–3235, 2020.

[29] S. S. Du, J. D. Lee, and Y. Tian, "When is a convolutional filter easy to learn?" in *International Conference on Learning Representations*, 2018.

[30] S. Mei, Y. Bai, and A. Montanari, "The landscape of empirical risk for non-convex losses," *Ann. Statist.*, vol. 46, no. 6A, pp. 2747–2774, 2018.

[31] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 8157–8166.

[32] Y. Yoshida and M. Okada, "Data-dependence of plateau phenomenon in learning with neural network — statistical mechanical analysis," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 1722–1730.

[33] S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová, "Modelling the influence of data structure on learning in neural networks: the hidden manifold model," *arXiv preprint arXiv: 1909.11500*, 2019.

[34] S. Goldt, G. Reeves, M. Mézard, F. Krzakala, and L. Zdeborová, "The gaussian equivalence of generative models for learning with two-layer neural networks," 2020.

[35] F. Mignacco, F. Krzakala, P. Urbani, and L. Zdeborová, "Dynamical mean-field theory for stochastic gradient descent in gaussian mixture classification," *Arxiv preprint Arxiv: 2006.06098*, 2020.

[36] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "When do neural networks outperform kernel methods?" *ArXiv preprint arXiv: 2006.13409*, 2020.

[37] S. Dasgupta, "Learning mixtures of gaussians," in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 1999, pp. 634–644.

[38] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 381–396, 2002.

[39] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[40] H. Permuter, J. Francos, and I. Jermyn, "A study of gaussian mixture models of color and texture features for image classification and segmentation," *Pattern Recognition*, vol. 39, no. 4, pp. 695–706, 2006.

[41] S. Yang, L. Liu, and M. Xu, "Free lunch for few-shot learning: Distribution calibration," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=JWOiYxMG92s

[42] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 9–50.

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[44] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Advances in Neural Information Processing Systems*, 2018, pp. 7694–7705.

[45] E. Chai, M. Pilanci, and B. Murmann, "Separating the effects of batch normalization on cnn training speed and stability using classical adaptive filter theory," *arXiv preprint arXiv:2002.10674*, 2020.

[46] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" in *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[47] K. Pearson, "Contributions to the mathematical theory of evolution," *Philosophical Transactions of the Royal Society of London. A*, vol. 185, pp. 71–110, 1894.

[48] D. M. Titterington, A. F. Smith, and U. E. Makov, *Statistical analysis of finite mixture distributions*. Wiley, 1985.

[49] D. Hsu and S. M. Kakade, "Learning mixtures of spherical gaussians: moment methods and spectral decompositions," in *Proceedings of the*

4th conference on Innovations in Theoretical Computer Science, 2013, pp. 11–20.

[50] M. Janzamin, H. Sedghi, and A. Anandkumar, "Score function features for discriminative learning: Matrix and tensor framework," *arXiv preprint arXiv:1412.2863*, 2014.

[51] V. Kuleshov, A. Chaganty, and P. Liang, "Tensor factorization via matrix factorization," in *Artificial Intelligence and Statistics*, 2015, pp. 507–516.