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

Hongkang Li  
Rensselaer Polytechnic Institute  
Troy, NY, USA  
lish5@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]–[3] to natural language processing [4]. 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 as 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 [5]–[7]. Nevertheless, it is well-known that training a one-hidden-layer neural network with only three nodes is NP-hard [8], 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 overparameterized networks has no spurious local minima [9]–[12], 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 [5], [13]–[15] or a Gaussian kernel processing from the Neural Tangent Kernel (NTK) view [6], [16]–[20].

In particular, for one-hidden-layer neural networks, [21]–[24] 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 [25] 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 [7], [26]–[30] with a few exceptions [31], [32]. Nevertheless, the neural network considered in [31], [32] 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 [5], [33] or cannot provide bounded generalization analysis [34]–[38].

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
mixture models are widely employed in plenty of applications, including data clustering [39–41], image segmentation [42], and few-shot learning [43]. 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 [44] and batch normalization [45] have been applied to accelerate the convergence rate and improve the generalization error. Although various works [46–48] 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, $\mathcal{Z}$ is a matrix, and $z$ is a vector. $Z$ is a set. $z_i$ denotes the $i$-th entry of $z$, and $Z_{i,j}$ denotes the $(i, j)$-th entry of $Z$. $[K]$ ($K > 0$) denotes the set including integers from 1 to $K$. $I_d \in \mathbb{R}^{d \times d}$ and $e_i$ represent the identity matrix in $\mathbb{R}^{d \times d}$ and the $i$-th standard basis vector, respectively. We use $\delta_i(Z)$ to denote the $i$-th largest singular value of $Z$. The matrix norm is defined as $\|Z\| = \delta_1(Z)$. The gradient and the Hessian of a function $f(W)$ are denoted by $\nabla f(W)$ and $\nabla^2 f(W)$, respectively.

Given a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and matrices $A \in \mathbb{R}^{n_1 \times d_1}$, $B \in \mathbb{R}^{n_2 \times d_2}$, $C \in \mathbb{R}^{n_3 \times d_3}$, the $(t_1, t_2, t_3)$-th entry of the tensor $T(A, B, C)$ is given by

$$
\sum_{i_1} \sum_{i_2} \sum_{i_3} T_{i_1, i_2, i_3} A_{i_1, i_1} B_{i_2, i_2} C_{i_3, i_3}.
$$

We follow the convention that $f(x) = O(g(x))$ (or $\Theta(g(x))$) 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 $(x, y) \in \mathbb{R}^d \times \{+1, -1\}$, where $\mathcal{X}$ is the Gaussian mixture distribution [49–51] such that

$$
x \sim \sum_{l=1}^{L} \lambda_l N(\mu_l, I_d),
$$

where $\lambda_l \in [0, 1]$ with $\sum_l \lambda_l = 1$, and $N$ denotes the multi-variate Gaussian distribution with mean $\mu_l \in \mathbb{R}^d$, and covariance $I_d$ for all $l \in [L]$. Let $M = (\mu_1, \mu_2, \ldots, \mu_L)$ and $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_L)$. 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 function:

$$
\phi(x) = \frac{1}{1 + \exp(-x)}.
$$

For any input $x \in \mathcal{X}$, the output of the teacher model is denoted as

$$
H(W^*; x) := \frac{1}{K} \sum_{j=1}^{K} \phi(w_j^T x),
$$

where $W^* = [w_1^*, \ldots, w_K^*] \in \mathbb{R}^{d \times K}$ is an unknown fixed weight matrix, and $w_j^* \in \mathbb{R}^d$ is the weight of the $j$-th neuron in the hidden layer. The corresponding output label $y$ satisfies

$$
P(y = 1|x) = H(W^*; x).
$$

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

$$
f_n(W) = \frac{1}{n} \sum_{i=1}^{n} \ell(W; x_i, y_i),
$$

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

$$
\ell(W; x, y) = -y \cdot \log(H(W; x)) - (1 - y) \cdot \log(1 - H(W; x)).
$$

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

$$
\min_{W \in \mathbb{R}^{d \times K}} f_n(W).
$$

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

1 The results can be generalized to any even activation function $\phi$ with bounded $\phi$, $\phi'$ and $\phi''$. Examples include tanh and erf.
III. ALGORITHM

We propose a gradient descent algorithm with tensor initialization method to solve (7). The method starts from an initialization \( W_0 \in \mathbb{R}^{d \times K} \) computed from the tensor initialization method (Subroutine 1) and then updates the iteration \( W_t \) using gradient descent with the step size \( \eta_0 \). The pseudo-code is summarized in Algorithm 1.

The tensor initialization algorithm is summarized in Subroutine 1. While existing tensor decomposition methods in [49] and [52] only apply for standard Gaussian distribution, we extend the methods to Gaussian mixture models. The intuition is similar to that in [7], where the directions of \( \{w_j^i\}_{j \in [K]} \) 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 \( Q_j \)’s are defined in Definition 1.

**Definition 1.** For \( j = 1, 2, 3 \), we define

\[
Q_j := E_{x \sim \sum_{l=1}^{L} \lambda_l N(\mu_l, \Sigma_l)} [(y \cdot (-1)^j p^{-1}(x) \nabla (j)p(x))], \tag{8}
\]

where \( p(x) \) is

\[
p(x) = \sum_{l=1}^{L} \lambda_l (2\pi)^{-d/2} \exp \left(-\frac{1}{2}(x - \mu_l)^\top (x - \mu_l)\right). \tag{9}
\]

**Algorithm 1 Gradient Descent with Tensor Initialization**

1. **Input:** Training data \( \{(x_i, y_i)\}_{i=1}^{n} \), the step size \( \eta_0 = (\sum_{l=1}^{L} \lambda_l (\|\mu_l\|_\infty + 1)^2)^{-1} \);
2. **Initialization:** \( W_0 \leftarrow \) Tensor initialization method via Subroutine 1;
3. for \( t = 0, 1, \ldots, T - 1 \) do
4. \( \hat{W}_{t+1} = W_t - \eta_0 \nabla f_n(W) \);
5. end for
6. **Output:** \( W_T \)

**Subroutine 1 Tensor Initialization Method**

1. **Input:** Partition \( \{(x_i, y_i)\}_{i=1}^{n} \) into three disjoint subsets \( \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3 \);
2. Compute \( Q_2 \) using \( \mathcal{D}_1 \). Estimate the subspace \( \hat{U} \) with respect to the largest \( K \) eigenvectors of \( Q_2 \);  
3. Compute \( \hat{R}_3 = Q_3(\hat{U}, \hat{U}, \hat{U}) \) from data set \( \mathcal{D}_2 \). Obtain \( \{\hat{v}_i\}_{i \in [K]} \) by decomposing \( \hat{R}_3 \);  
4. \( \hat{w}_j^i = \hat{U} \hat{v}_i \) for \( i \in [K] \);
5. Compute \( Q_1 \) from data set \( \mathcal{D}_3 \). Estimate the magnitude \( \hat{z} \) by solving the optimization problem

\[
\hat{z} = \arg \min_{\alpha \in \mathbb{R}^K} \frac{1}{2} \left\| Q_1 - \sum_{j=1}^{K} \alpha_j \hat{w}_j^* \right\|_2^2;
\]
6. **Return:** \( \hat{z}_j \hat{U} \hat{v}_j \) as the \( j \)-th column of \( W_0, j \in [K] \).

Subroutine 1 estimates the direction and magnitude of \( w_j^i, j \in [K] \) separately. The direction vector is defined as \( \hat{w}_j^i := w_j^i / \|w_j^i\| \), and the corresponding magnitude \( \|w_j^i\| \) is denoted as \( \hat{z}_j \). Line 2 estimates the subspace \( \hat{U} \) spanned by \( \{w_1^*, \ldots, w_K^*\} \) using \( \hat{Q}_2 \). Lines 3-4 estimate the direction vector \( \hat{w}_j^i \) by employing the KCL algorithm [53]. Line 5 estimates the magnitude \( \hat{z}_j \). Finally, the returned estimation of \( W^* \) is calculated as \( \hat{z}_j \hat{U} \hat{v}_j \).

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 \( \hat{W}_n \) near a permutation of the ground truth, denoted as \( W^* P^* \). The distance between \( \hat{W}_n \) and \( W^* P^* \) are upper bounded as a function of the number of samples in (12). Additionally, the required number of samples depends on \( B \), which is a function of input distributions in (2).

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

\[
n \geq n_{\text{sc}} := \text{poly}(c_0^{-1}, \kappa, K) B(\lambda, M) d \log^2 d \tag{10}
\]

for some \( c_0 \in (0, \frac{1}{2}) \) and positive value functions \( B(\lambda, M) \) and \( q(\lambda, M) \), with probability at least \( 1 - d^{-10} \), the iterates \( \{W_t\}_{t=1}^T \) returned by Algorithm 1 with step size \( \eta_0 = O\left(\left(\sum_{l=1}^{L} \lambda_l (\|\mu_l\|_\infty + 1)^2\right)^{-1}\right) \) converge linearly to a critical point \( \hat{W}_n \) with the rate of convergence \( v = 1 - K^{-2} q(\lambda, M) \), i.e.,

\[
\|W_t - \hat{W}_n\|_F \leq v^t \|W_0 - \hat{W}_n\|_F. \tag{11}
\]

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

\[
\|\hat{W}_n - W^* P^*\|_F \leq O\left(K^2 \cdot \sqrt{d \log n / n}\right). \tag{12}
\]

**Remark 1:** From (10), the required number of samples for successful estimation is a linear function of the input feature dimension \( d \) and \( B(\lambda, M) \), where \( B \) is a function of the input distribution parameters. Note that the degree of freedom of \( W^* \) is \( dK \), the sample complexity in (10) is nearly order-wise optimal in terms of \( d \). In addition, \( B(\lambda, M) \) is an increasing function of any entry of \( M \) (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 \( \hat{W}_n \) and the ground truth \( W^* P^* \) 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 \( \hat{W}_n \) linearly, and the rate of convergence is \( 1 - K^{-2} q(\lambda, M) \), denoted as \( \nu \). 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 $\lambda$ and $M$. Specifically, when the mean vectors $\{\mu_i\}_{i=1}^M$ in (2) increases, the sample complexity $n_{sc}$ increases as $B(\lambda, M)$ increases, and the convergence rate $v$ increases as $q(\lambda, M)$ decreases. Suppose some entry of $\mu_i$ go to infinity, $n_{sc}$ will go to infinity, and $v$ will converge to 1.

**Corollary 1.** Recall $M = (\mu_1, \cdots, \mu_L)$. Let $|\mu_{i(l)}|$ be the $i$-th entry of $\mu_i$. 

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

**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 $\hat{W} \in \mathbb{R}^{d \times K}$ are randomly generated such that each entry of $\hat{W}$ belongs to $N(0, 1)$. The corresponding training samples $\{x_i, y_i\}_{i=1}^n$ are randomly selected following (2) and (3) with the generated weights $\hat{W}$.

**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 $L = 2$, $\lambda_1 = \lambda_2 = \frac{1}{2}$, $\mu_1 = 0.5 \cdot 1$ and $\mu_2 = -\mu_1$. The number of neurons in the hidden layer is set as 3. For a given $\hat{W}$, we initialize the starting point $\lambda$ times randomly and denote $\hat{W}_n^{(m)}$ 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 $e_{\hat{W}}$ is less than $10^{-5}$, where $e_{\hat{W}}$ is the standard deviation of $(\hat{W}_n^{(1)}, \cdots, \hat{W}_n^{(L)})$.

In Fig. 1, we increase the feature dimension $d$ from 12 to 50 by 2 and vary the number of samples $n$ from $2 \times 10^3$ to $3 \times 10^4$. 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 $d = 10$, $\lambda_1 = 0.4$, $\lambda_2 = 0.6$, $\mu_1 = \mu \cdot 1$, $\mu_2 = 0$. 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 $n = 1 \times 10^4$, $\lambda_1 = \lambda_2 = 0.5$, $\mu_1 = -\mu_2 = C \cdot 1$ for a positive $C$. We vary $C$ and let $\hat{\mu} = \max_i |\hat{\mu}_i|$. Recall the relative error is defined as $\|W_i - \hat{W}_i\|_F \in (11)$. Fig. 3 shows the linear convergence of Algorithm 1 for different $\hat{\mu}$ and the impact of $|\hat{\mu}_l|_\infty$. As predicted in Theorem 1 when $\hat{\mu}$ 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 proven 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.
[46] 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.

[47] 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.

[48] 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.

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

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

[51] 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.

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

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