A Corrective View of Neural Networks: Representation, Memorization and Learning

Dheeraj Nagaraj

MIT

June 2020

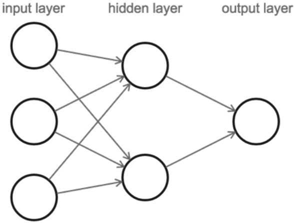
Joint work with Guy Bresler

Overview

Introduction

Neural Networks are universal approximators[1]. We introduce a mathematical tool to obtain

- Sharp bounds on the number of neurons required for representation

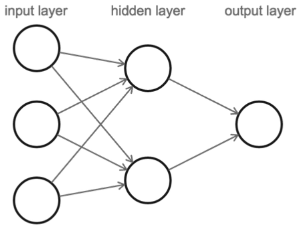


[1]Cybenko 1989.

Introduction

Neural Networks are universal approximators[1]. We introduce a mathematical tool to obtain

- Sharp bounds on the number of neurons required for representation
- State of the art memorization results

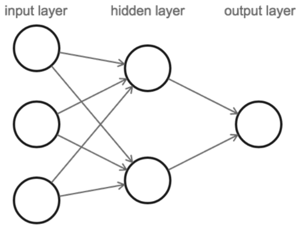


[1] Cybenko 1989.

Introduction

Neural Networks are universal approximators[1]. We introduce a mathematical tool to obtain

- Sharp bounds on the number of neurons required for representation
- State of the art memorization results
- Subpolynomial bounds on number of neurons required to learn low-degree polynomials via. SGD/GD



[1] Cybenko 1989.

Our Results - Memorization

- Neural networks can memorize (or interpolate) arbitrary labels quite easily

Our Results - Memorization

- Neural networks can memorize (or interpolate) arbitrary labels quite easily
- Long line of papers aims to understand memorization in over-parametrized networks via the study of SGD/GD

Our Results - Memorization

- Given n arbitrary points in $\mathcal{S}^{d-1}$ with arbitrary labels, how many neurons are necessary to memorize (or interpolate) ?

Our Results - Memorization

- Given n arbitrary points in $\mathcal{S}^{d-1}$ with arbitrary labels, how many neurons are necessary to memorize (or interpolate) ?
- minimum distance: θ, max error: ϵ (achieved via GD)
  Two-layer ReLU networks require $\tilde{O}(\frac{n}{\theta^4} \log \frac{1}{\epsilon})$ non-linear units

Our Results - Memorization

- Given n arbitrary points in $\mathcal{S}^{d-1}$ with arbitrary labels, how many neurons are necessary to memorize (or interpolate) ?
- minimum distance: θ, max error: ϵ (achieved via GD)
  Two-layer ReLU networks require $\tilde{O}(\frac{n}{\theta^4} \log \frac{1}{\epsilon})$ non-linear units
- Near optimal in n for two layer ReLU networks and first work to achieve this via. GD

Our Results - Memorization

Work	Assumption	Guarantee	Remarks
Allen-Zhu, Li, and Song 2018	Minimum distance θ	$O(\frac{n^{24}d}{\theta^8})$	
Du et al. 2019	Distinct points	$O(n^6)$	extra factors
Ji and Telgarsky 2019	NTK separability Minimum distance θ	$\log(n)$ $O(\frac{n^{24}}{\theta^8})$	
Oymak and Soltanolkotabi 2019	$d \leq n \leq cd^2$, data i.i.d unif$(\mathcal{S}^{d-1})$	$O(\frac{n^2}{d})$	w.h.p over data
Song and Yang 2019	Distinct points	$O(n^4)$	extra factors
Daniely 2019	$n = d^c$, i.i.d unif$(\mathcal{S}^{d-1})$	$\tilde{O}(n/d)$	w.h.p over data
Kawaguchi and Huang 2019	Minimum distance θ	$\tilde{O}(n)$	
Our Work	Minimum distance θ	$\tilde{O}(\frac{n}{\theta^4})$	

Table: Comparison of Guarantees for number of non-linear units

Our Results - Representation

- Approximate $f : \mathbb{R}^d \to \mathbb{R}$ over a ball of radius r, with respect to squared error

Our Results - Representation

- Approximate $f : \mathbb{R}^d \to \mathbb{R}$ over a ball of radius r, with respect to squared error
- Let F be fourier transform of f. Suppose:

$$\frac{1}{(2\pi)^d} \int (1 + \|\xi\|^{\Theta(ad)}) |F(\xi)| d\xi =: C_f$$

(roughly speaking, f has $\Theta(ad)$ bounded derivatives)

Our Results - Representation

- Approximate $f : \mathbb{R}^d \to \mathbb{R}$ over a ball of radius r, with respect to squared error
- Let F be fourier transform of f. Suppose:

$$\frac{1}{(2\pi)^d} \int (1 + \|\xi\|^{\Theta(ad)}) |F(\xi)| d\xi =: C_f$$

(roughly speaking, f has $\Theta(ad)$ bounded derivatives)

- There exists a two layer network with N non-linear units of ReLU and smoothed ReLU kind such that:

$$\mathrm{Error}(f; \hat{f}) \leq O\left(C(a, d) \frac{C_f^2}{N^a} \right).$$

Our Results - Representation

- Approximate $f : \mathbb{R}^d \to \mathbb{R}$ over a ball of radius r, with respect to squared error
- Let F be fourier transform of f. Suppose:

$$\frac{1}{(2\pi)^d} \int (1 + \|\xi\|^{\Theta(ad)})|F(\xi)| d\xi =: C_f$$

(roughly speaking, f has $\Theta(ad)$ bounded derivatives)

- There exists a two layer network with N non-linear units of ReLU and smoothed ReLU kind such that:

$$\mathrm{Error}(f; \hat{f}) \leq O\left(C(a, d)\frac{C_f^2}{N^a}\right).$$

- We can replace d with effective dimension $q \ll d$ when there is 'low-dimensional structure'. Ex: low-degree polynomials.

- For functions with $\Theta(ad)$ bounded derivatives,[2] previous results implement taylor series approximation

[2] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Our Results - Representation

- For functions with $\Theta(ad)$ bounded derivatives,[2] previous results implement taylor series approximation
- $O(\frac{1}{N^a})$ squared error - but complex deep networks with no known training results

[2] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Our Results - Representation

- For functions with $\Theta(ad)$ bounded derivatives,[2] previous results implement taylor series approximation
- $O(\frac{1}{N^a})$ squared error - but complex deep networks with no known training results
- Our results show that we can do the same with a two-layer network

[2] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Our Results - Learning Polynomials

- Consider degree q polynomials over d-dimensional input.

Our Results - Learning Polynomials

- Consider degree q polynomials over d-dimensional input.
- With suitable random feature sampling, we can learn this class of functions via GD upto error ϵ with:

$$O\left(C(q)d^{2q}\text{subpoly}(1/\epsilon)\right)$$

neurons of ReLU and smoothed ReLU kind.

Our Results - Learning Polynomials

- Consider degree q polynomials over d-dimensional input.
- With suitable random feature sampling, we can learn this class of functions via GD upto error ϵ with:

$$O\left(C(q)d^{2q}\mathrm{subpoly}(1/\epsilon)\right)$$

  neurons of ReLU and smoothed ReLU kind.
- First sub-polynomial learning bounds

Corrective Mechanism - The Main Idea

- Divide Neurons into multiple groups.

Corrective Mechanism - The Main Idea

- Divide Neurons into multiple groups.
- First group approximates the function under consideration.

Corrective Mechanism - The Main Idea

- Divide Neurons into multiple groups.
- First group approximates the function under consideration.
- Second group approximates the error produced by the first and corrects it.

Corrective Mechanism - The Main Idea

- Divide Neurons into multiple groups.
- First group approximates the function under consideration.
- Second group approximates the error produced by the first and corrects it.
- Third group approximates and corrects the error by the first two groups and so on.

Corrective Mechanism - The Main Idea

- Divide Neurons into multiple groups.
- First group approximates the function under consideration.
- Second group approximates the error produced by the first and corrects it.
- Third group approximates and corrects the error by the first two groups and so on.
- Under certain conditions, ‘a’ corrective steps give a rate of $1/N^a$.

Representation to Learning: Random Features Model

- 2 layer network (one linear, one non-linear):

$$\hat{f}(x) = \sum_{i=1}^{N} \kappa_i \mathrm{ReLU}(\langle w_i, x \rangle - T_i)$$

Representation to Learning: Random Features Model

- 2 layer network (one linear, one non-linear):

$$\hat{f}(x) = \sum_{i=1}^{N} \kappa_i \mathrm{ReLU}(\langle w_i, x \rangle - T_i)$$

- Draw w_i, T_i at random and optimize over κ_i.

Representation to Learning: Random Features Model

- 2 layer network (one linear, one non-linear):

$$\hat{f}(x) = \sum_{i=1}^{N} \kappa_i \mathrm{ReLU}(\langle w_i, x \rangle - T_i)$$

- Draw w_i, T_i at random and optimize over κ_i.
- Reduces non-convex optimization problem to a smooth convex optimization problem.

Representation to Learning: Random Features Model

- 2 layer network (one linear, one non-linear):

$$\hat{f}(x) = \sum_{i=1}^{N} \kappa_i \mathrm{ReLU}(\langle w_i, x \rangle - T_i)$$

- Draw w_i, T_i at random and optimize over κ_i.
- Reduces non-convex optimization problem to a smooth convex optimization problem.
- SGD for neural networks with a large number of neurons reduces to this approximately.

Representation to Learning: Random Features Model

- Pick w_i, T_i from some tractable distribution.

Representation to Learning: Random Features Model

- Pick w_i, T_i from some tractable distribution.
- Show via. probabilistic method that there exists κ_i^0 which achieves an error of at most ϵ.

Representation to Learning: Random Features Model

- Pick w_i, T_i from some tractable distribution.
- Show via. probabilistic method that there exists κ_i^0 which achieves an error of at most ϵ.
- The 'random features' optimization must give κ_i^* which can do better than κ_i^0 (error of at most 'ϵ')

Memorization - Proof

- Data : $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. Construct discrete Fourier transform:

$$F(\xi) = \sum_{j=1}^{n} y_j e^{i\langle \xi, x_j \rangle} .$$

Memorization - Proof

- Data : $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. Construct discrete Fourier transform:

$$F(\xi) = \sum_{j=1}^{n} y_j e^{i\langle \xi, x_j \rangle} .$$

- $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$, $\sigma \sim \frac{\sqrt{\log n}}{\theta}$, $\theta =$ min. distance.

Memorization - Proof

- Data : $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. Construct discrete Fourier transform:

$$F(\xi) = \sum_{j=1}^{n} y_j e^{i\langle \xi, x_j \rangle}\,.$$

- $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$, $\sigma \sim \frac{\sqrt{\log n}}{\theta}$, $\theta =$ min. distance.
- 'Inverse Fourier tranform':

$$\begin{aligned} y_j &\approx \mathbb{E} F(\xi) e^{-i\langle \xi, x_j \rangle} \\ &= \mathbb{E}|F(\xi)| \cos(\langle \xi, x_j \rangle + \psi(\xi)) \end{aligned} \tag{1}$$

Memorization - Proof

- 'Cosine Representation' : cos function as integrals of ReLU. Let $T \sim \text{unif}[-2,2]$, independent of ξ.

$$y_j \approx \mathbb{E}C(1+\tilde{O}(1/\theta^2))|F(\xi)|\eta(T,\xi)\mathrm{ReLU}\left(\frac{\langle \xi, x_j\rangle}{\omega_0} - T\right) \tag{2}$$

Memorization - Proof

- 'Cosine Representation' : cos function as integrals of ReLU. Let $T \sim \text{unif}[-2,2]$, independent of ξ.

$$y_j \approx \mathbb{E} C(1+\tilde{O}(1/\theta^2))|F(\xi)|\eta(T,\xi)\mathrm{ReLU}\left(\frac{\langle \xi, x_j\rangle}{\omega_0} - T\right) \qquad (2)$$

- Contruct empirical estimator: $(\xi_k, T_k) \sim \mathcal{N}(0,\sigma^2 I_d) \times \text{Unif}[-2,2]$ i.i.d:

$$\hat{y}_j^{(1)} = \frac{1}{N_0}\sum_{k=1}^{N_0} C(1+\tilde{O}(1/\theta^2))|F(\xi_k)|\eta(T_k,\xi_k)\mathrm{ReLU}\left(\frac{\langle \xi_k, x_j\rangle}{\omega_0} - T_k\right).$$

Memorization - Proof

- Contraction in ℓ^2 via Gaussian concentration:

$$\mathbb{E}\|\mathbf{y}-\hat{\mathbf{y}}^{(1)}\|_2^2 \leq \tilde{O}(\tfrac{n}{\theta^4 N_0})\|\mathbf{y}\|_2^2$$

Memorization - Proof

- Contraction in ℓ^2 via Gaussian concentration:

$$\mathbb{E}\|\mathbf{y} - \hat{\mathbf{y}}^{(1)}\|_2^2 \leq \tilde{O}(\tfrac{n}{\theta^4 N_0})\|\mathbf{y}\|_2^2$$

- Correction step: replace $\mathbf{y}$ with $\mathbf{y} - \hat{\mathbf{y}}^{(1)}$ and estimate it with $\hat{\mathbf{y}}^{(2)}$. We conclude:

$$\mathbb{E}\|\mathbf{y} - \hat{\mathbf{y}}^{(1)} - \hat{\mathbf{y}}^{(2)}\|_2^2 \leq \left[\tilde{O}(\tfrac{n}{\theta^4 N_0})\right]^2 \|\mathbf{y}\|_2^2 \,.$$

Memorization - Proof

- Contraction in ℓ^2 via Gaussian concentration:

$$\mathbb{E}\|\mathbf{y} - \hat{\mathbf{y}}^{(1)}\|_2^2 \leq \tilde{O}(\tfrac{n}{\theta^4 N_0})\|\mathbf{y}\|_2^2$$

- Correction step: replace $\mathbf{y}$ with $\mathbf{y} - \hat{\mathbf{y}}^{(1)}$ and estimate it with $\hat{\mathbf{y}}^{(2)}$. We conclude:

$$\mathbb{E}\|\mathbf{y} - \hat{\mathbf{y}}^{(1)} - \hat{\mathbf{y}}^{(2)}\|_2^2 \leq \left[\tilde{O}(\tfrac{n}{\theta^4 N_0})\right]^2 \|\mathbf{y}\|_2^2 \, .$$

- Continue $l = O(\log \frac{n}{\epsilon})$ times:

$$\mathbb{E}\|\mathbf{y} - \sum_{s=1}^{l} \hat{\mathbf{y}}^{(s)}\| \leq \left[\tilde{O}(\tfrac{n}{\theta^4 N_0})\right]^l \|\mathbf{y}\|_2^2 \leq \epsilon \, .$$

Memorization - Proof

- Contraction in ℓ^2 via Gaussian concentration:

$$\mathbb{E}\|\mathbf{y} - \hat{\mathbf{y}}^{(1)}\|_2^2 \leq \tilde{O}(\tfrac{n}{\theta^4 N_0})\|\mathbf{y}\|_2^2$$

- Correction step: replace $\mathbf{y}$ with $\mathbf{y} - \hat{\mathbf{y}}^{(1)}$ and estimate it with $\hat{\mathbf{y}}^{(2)}$. We conclude:

$$\mathbb{E}\|\mathbf{y} - \hat{\mathbf{y}}^{(1)} - \hat{\mathbf{y}}^{(2)}\|_2^2 \leq \left[\tilde{O}(\tfrac{n}{\theta^4 N_0})\right]^2 \|\mathbf{y}\|_2^2 \,.$$

- Continue $l = O(\log \frac{n}{\epsilon})$ times:

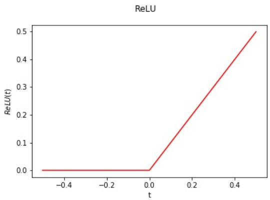
$$\mathbb{E}\|\mathbf{y} - \sum_{s=1}^{l} \hat{\mathbf{y}}^{(s)}\| \leq \left[\tilde{O}(\tfrac{n}{\theta^4 N_0})\right]^l \|\mathbf{y}\|_2^2 \leq \epsilon \,.$$

- We conclude that memorization requires $\tilde{O}(\frac{n}{\theta^4} \log \frac{1}{\epsilon})$ activation functions.

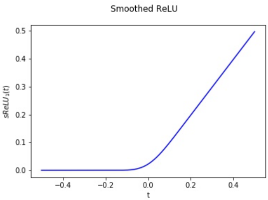
Representation Theorems

- Similar procedure as Memorization.

- Similar procedure as Memorization.
- Uses a mixture of ReLU and smoothed ReLU (SReLU_k) activation functions. SReLU_k are same as ReLU outside a neighborhood of 0 and are $2k$ times continuously differentiable.



(a) ReLU



(b) SReLU_1

Figure: Illustrating ReLU and SReLU activation functions.

Representation Theorems

- Approximate target function f by $\hat{f}^{(1)}$ (two layer SReLU_k network) with N activation functions.

[3] Barron 1993.

- Approximate target function f by $\hat{f}^{(1)}$ (two layer SReLU_k network) with N activation functions.
- $$(f - \hat{f}^{(1)})^2 \le \frac{C_f^2}{N}\,.$$

  Where C_f is a norm on the Fourier transform of f.[3]

[3] Barron 1993.

Representation Theorems

- Approximate target function f by $\hat{f}^{(1)}$ (two layer SReLU_k network) with N activation functions.
- $$(f - \hat{f}^{(1)})^2 \leq \frac{C_f^2}{N}\,.$$

  Where C_f is a norm on the Fourier transform of f.[3]
- Fourier transform of $\hat{f}^{(1)}$ is an unbiased estimator for the Fourier tranform of f. Therefore, (roughly)

$$C_{f-\hat{f}^{(1)}} \leq C\frac{C_f}{\sqrt{N}}\,.$$

[3] Barron 1993.

Representation Theorems

- Let $f^{\text{rem}} := f - \hat{f}^{(1)}$. We can approximate f^{rem} by $\hat{f}^{(2)}$ with N non-linear units such that:

$$(f^{\text{rem}} - \hat{f}^{(2)})^2 \leq \frac{C^2_{f^{\text{rem}}}}{N} \leq \frac{C^2_f}{N^2} .$$

- Let $f^{\text{rem}} := f - \hat{f}^{(1)}$. We can approximate f^{rem} by $\hat{f}^{(2)}$ with N non-linear units such that:

$$(f^{\text{rem}} - \hat{f}^{(2)})^2 \leq \frac{C^2_{f^{\text{rem}}}}{N} \leq \frac{C^2_f}{N^2}.$$

- Therefore, $\hat{f}^{(1)} + \hat{f}^{(2)}$ approximates f up to an error of $\frac{1}{N^2}$. We can continue this 'a' times to get rates of $\frac{1}{N^a}$.

Representation Theorems

- Let $f^{\text{rem}} := f - \hat{f}^{(1)}$. We can approximate f^{rem} by $\hat{f}^{(2)}$ with N non-linear units such that:

$$(f^{\text{rem}} - \hat{f}^{(2)})^2 \leq \frac{C^2_{f^{\text{rem}}}}{N} \leq \frac{C^2_f}{N^2}\,.$$

- Therefore, $\hat{f}^{(1)} + \hat{f}^{(2)}$ approximates f up to an error of $\frac{1}{N^2}$. We can continue this 'a' times to get rates of $\frac{1}{N^a}$.
- After each corrective step, the remainder function becomes less and less smooth till further approximation is impossible (depending on how smooth the original function is).

- Consider $f(x) = \sum_V J_V p_V(x)$ - degree q multinomial over $\mathbb{R}^d$.

[4] Andoni et al. 2014; Yehudai and Shamir 2019.

[5] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Application : Learning Low-degree Polynomials

- Consider $f(x) = \sum_V J_V p_V(x)$ - degree q multinomial over $\mathbb{R}^d$.
- **State of the art learning result via GD:**[4] $\Omega(d^{2q}\mathrm{poly}(1/\epsilon))$

[4] Andoni et al. 2014; Yehudai and Shamir 2019.

[5] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Application : Learning Low-degree Polynomials

- Consider $f(x) = \sum_V J_V p_V(x)$ - degree q multinomial over $\mathbb{R}^d$.
- **State of the art learning result via GD:**[4] $\Omega(d^{2q}\text{poly}(1/\epsilon))$
- **Purely representation results with complex deep networks:**[5] $O(d^q\text{polylog}(\frac{1}{\epsilon}))$. (No learning guarantees)

[4] Andoni et al. 2014; Yehudai and Shamir 2019.

[5] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Application : Learning Low-degree Polynomials

- Consider $f(x) = \sum_V J_V p_V(x)$ - degree q multinomial over $\mathbb{R}^d$.
- **State of the art learning result via GD:**[4] $\Omega(d^{2q}\text{poly}(1/\epsilon))$
- **Purely representation results with complex deep networks:**[5] $O(d^q \text{polylog}(\frac{1}{\epsilon}))$. (No learning guarantees)
- **Our results for learning:** $O(d^{q(1+\delta)}\text{subpoly}_q(1/\epsilon))$ ($\delta \to 0$ as $\epsilon \to 0$). Gives us the first sub-polynomial learning guarantees.

[4] Andoni et al. 2014; Yehudai and Shamir 2019.

[5] Liang and Srikant 2016; Safran and Shamir 2017; Yarotsky 2017.

Application: Learning Low-degree Polynomials

- Learning results are an application of the representation results under random features regime.

Application: Learning Low-degree Polynomials

- Learning results are an application of the representation results under random features regime.
- $f(x)$ effective dimension $q << d$. It is infinitely differentiable - so we can achieve rates of $\frac{C(a,q)}{N^a}$ for arbitrary $a \in \mathbb{N}$.

Application: Learning Low-degree Polynomials

- Learning results are an application of the representation results under random features regime.
- $f(x)$ effective dimension $q << d$. It is infinitely differentiable - so we can achieve rates of $\frac{C(a,q)}{N^a}$ for arbitrary $a \in \mathbb{N}$.
- Sample ω_i and T_i from a tractable distribution, there exist coefficients b_i such that the random neural network

$$\hat{f}(x; \mathbf{b}) = \sum_{i=1}^{N} b_i \mathrm{SReLU}_{j_i}\left(\frac{\langle \omega_i, x \rangle}{\sqrt{q}} - T_i\right)$$

approximates f up to a squared error of $C(a,q)\frac{d^{q(a+1)}}{N^a}$ in expectation

- Whenever $N \geq C(a, q)d^{q\frac{a+1}{a}}(\epsilon\delta)^{-\frac{1}{a}}$, with probability atleast $1-\delta$ (over the randomness in the weights), we can pick coefficients $b_{i,j,V}$ so that squared error is at most ϵ.

Learning Low-degree Polynomials

- Whenever $N \geq C(a, q)d^{q\frac{a+1}{a}}(\epsilon\delta)^{-\frac{1}{a}}$, with probability atleast $1 - \delta$ (over the randomness in the weights), we can pick coefficients $b_{i,j,V}$ so that squared error is at most ϵ.
- Let $a \to \infty$ slowly enough as $\epsilon \to 0$. This gives us subpolynomial bounds.

Thank You