Complexity of Deep Neural Networks from the Perspective of Functional Equivalence

The Hong Kong Polytechnic University
香港理工大學
Department of Applied Mathematics
應 用 數 學 系

Guohao Shen

Xuzhou, Jiangsu

April 20, 2024

Outline

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - tend to be easier to train [9, 1, 7].
  - exhibit better generalization [5, 28, 29].
  - did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - Complexity [3, 22, 4]
  - Approximation power [36, 18, 39]
  - Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - tend to be easier to train [9, 1, 7].
  - exhibit better generalization [5, 28, 29].
  - did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - Complexity [3, 22, 4]
  - Approximation power [36, 18, 39]
  - Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Motivations

- Neural networks have shown remarkable success particularly large models
  - ▷ Challenges: explain the generalization of overparameterized models [24, 26, 30].
- Overparameterized networks appeared to contradict common sense:
  - ▷ tend to be easier to train [9, 1, 7].
  - ▷ exhibit better generalization [5, 28, 29].
  - ▷ did not tend to overfit [37].
- Current theoretical understanding on the generalization:
  - ▷ Complexity [3, 22, 4]
  - ▷ Approximation power [36, 18, 39]
  - ▷ Optimization [11]

Existing Generalization Theory

- Given data dist. Z, loss L and a hypothesis f, define the risk $\mathcal{R}$, target f^* by

$$\mathcal{R}(f) := \mathbb{E}[L(f, Z)] \qquad \text{and} \qquad f^* := \arg \min_{f \text{ measurable}} \mathcal{R}(f).$$

- Limited computational capability, only search the minimizer over a hypothesis space $\mathcal{F}_n$ (e.g., deep neural networks).

$$f_n^* = \arg \min_{f \in \mathcal{F}_n} \mathcal{R}(f).$$

- Unknown dist. of Z, only have a sample $\{Z_i\}_{i=1}^n$.
$\Rightarrow$ Minimize empirical risk $\mathcal{R}_n(\cdot)$, get empirical risk minimizer $\hat{f}_n$ (ERM)

$$\mathcal{R}_n(f) := \sum_{i=1}^n L(f, Z_i)/n, \qquad \hat{f}_n \in \arg \min_{f \in \mathcal{F}_n} \mathcal{R}_n(f).$$

- No closed-form solution for $\hat{f}_n$.
$\Rightarrow$ Use optimization algorithm, get $\hat{f}_{n,opt}$ in practice.

Existing Generalization Theory

- Given data dist. Z, loss L and a hypothesis f, define the risk $\mathcal{R}$, target f^* by

$$\mathcal{R}(f) := \mathbb{E}[L(f, Z)] \qquad \text{and} \qquad f^* := \arg \min_{f \text{ measurable}} \mathcal{R}(f).$$

- Limited computational capability, only search the minimizer over a hypothesis space $\mathcal{F}_n$ (e.g., deep neural networks).

$$f_n^* = \arg \min_{f \in \mathcal{F}_n} \mathcal{R}(f).$$

- Unknown dist. of Z, only have a sample $\{Z_i\}_{i=1}^n$.
  $\Rightarrow$ Minimize empirical risk $\mathcal{R}_n(\cdot)$, get empirical risk minimizer $\hat{f}_n$ (ERM)

$$\mathcal{R}_n(f) := \sum_{i=1}^n L(f, Z_i)/n, \qquad \hat{f}_n \in \arg \min_{f \in \mathcal{F}_n} \mathcal{R}_n(f).$$

- No closed-form solution for $\hat{f}_n$.
  $\Rightarrow$ Use optimization algorithm, get $\hat{f}_{n,opt}$ in practice.

Existing Generalization Theory

- Given data dist. Z, loss L and a hypothesis f, define the risk $\mathcal{R}$, target f^* by

$$\mathcal{R}(f) := \mathbb{E}[L(f, Z)] \qquad \text{and} \qquad f^* := \arg \min_{f \text{ measurable}} \mathcal{R}(f).$$

- Limited computational capability, only search the minimizer over a hypothesis space $\mathcal{F}_n$ (e.g., deep neural networks).

$$f_n^* = \arg \min_{f \in \mathcal{F}_n} \mathcal{R}(f).$$

- Unknown dist. of Z, only have a sample $\{Z_i\}_{i=1}^n$.
  $\Rightarrow$ Minimize empirical risk $\mathcal{R}_n(\cdot)$, get empirical risk minimizer $\hat{f}_n$ (ERM)

$$\mathcal{R}_n(f) := \sum_{i=1}^n L(f, Z_i)/n, \qquad \hat{f}_n \in \arg \min_{f \in \mathcal{F}_n} \mathcal{R}_n(f).$$

- No closed-form solution for $\hat{f}_n$.
  $\Rightarrow$ Use optimization algorithm, get $\hat{f}_{n,opt}$ in practice.

Existing Generalization Theory

- Given data dist. Z, loss L and a hypothesis f, define the risk $\mathcal{R}$, target f^* by

$$\mathcal{R}(f) := \mathbb{E}[L(f, Z)] \qquad \text{and} \qquad f^* := \arg \min_{f \text{ measurable}} \mathcal{R}(f).$$

- Limited computational capability, only search the minimizer over a hypothesis space $\mathcal{F}_n$ (e.g., deep neural networks).

$$f_n^* = \arg \min_{f \in \mathcal{F}_n} \mathcal{R}(f).$$

- Unknown dist. of Z, only have a sample $\{Z_i\}_{i=1}^n$.
  $\Rightarrow$ Minimize empirical risk $\mathcal{R}_n(\cdot)$, get empirical risk minimizer $\hat{f}_n$ (ERM)

$$\mathcal{R}_n(f) := \sum_{i=1}^n L(f, Z_i)/n, \qquad \hat{f}_n \in \arg \min_{f \in \mathcal{F}_n} \mathcal{R}_n(f).$$

- No closed-form solution for $\hat{f}_n$.
  $\Rightarrow$ Use optimization algorithm, get $\hat{f}_{n,opt}$ in practice.

Generalization Error

- Generalization Error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(f^*)$
- Generalization Error Bound:

$$
\begin{aligned}
&\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(f^*) \\
=&\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n) + \mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) + \mathcal{R}(f_n^*) - \mathcal{R}(f^*) \\
\leq&|\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)| + |\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)| + |\mathcal{R}(f_n^*) - \mathcal{R}(f^*)|
\end{aligned}
$$

-
  - ▷ Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$
  - ▷ Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$
  - ▷ Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$.

- Generalization Error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(f^*)$
- Generalization Error Bound:

$$
\begin{aligned}
&\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(f^*) \\
=&\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n) + \mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) + \mathcal{R}(f_n^*) - \mathcal{R}(f^*) \\
\leq&|\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)| + |\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)| + |\mathcal{R}(f_n^*) - \mathcal{R}(f^*)|
\end{aligned}
$$

-
  - Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$
  - Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$
  - Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$.

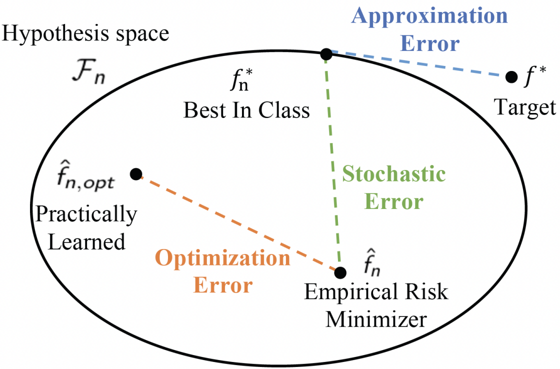
Generalization Error

- Generalization Error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(f^*)$
- Generalization Error Bound:

$$
\begin{aligned}
&\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(f^*) \\
=&\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n) + \mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) + \mathcal{R}(f_n^*) - \mathcal{R}(f^*) \\
\leq&|\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)| + |\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)| + |\mathcal{R}(f_n^*) - \mathcal{R}(f^*)|
\end{aligned}
$$

-
  - ▷ Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$
  - ▷ Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$
  - ▷ Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$.

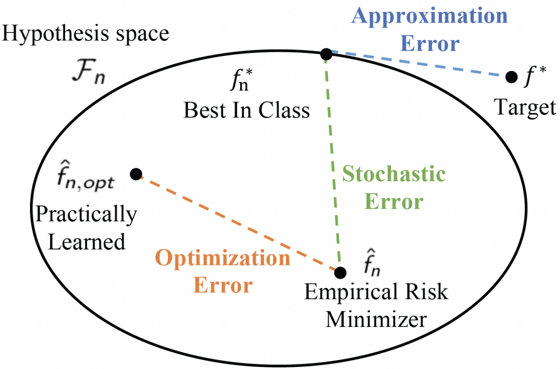
Generalization Error Decomposition



Contributions in this work:

- Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$. (Improved in this work ✓)
- Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$ (Improved in this work ✓)
- Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$. (Improved in this work ?)

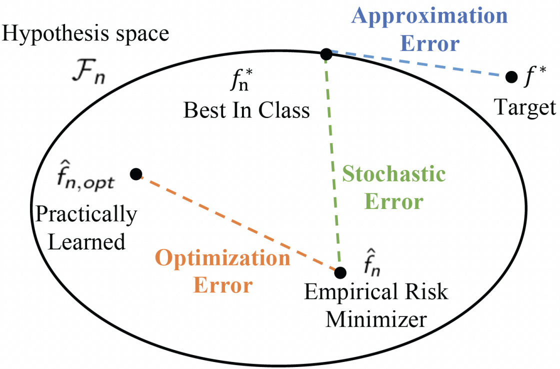
Generalization Error Decomposition



Contributions in this work:

- ▷ Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$. (Improved in this work ✓)
- ▷ Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$ (Improved in this work ✓)
- ▷ Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$. (Improved in this work ?)

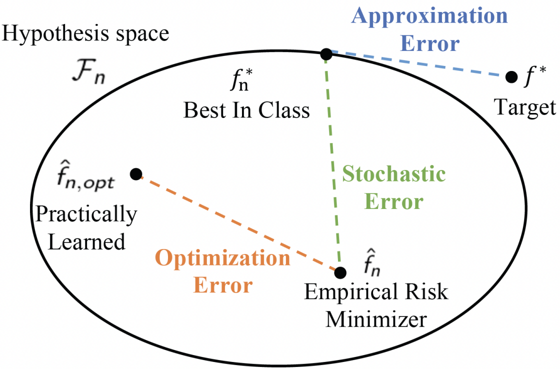
Generalization Error Decomposition



Contributions in this work:

- ▷ Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$. (Improved in this work ✓)
- ▷ Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$ (Improved in this work ✓)
- ▷ Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$. (Improved in this work ?)

Generalization Error Decomposition



Contributions in this work:

- Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*)$. (Improved in this work ✓)
- Optimization error: $\mathcal{R}(\hat{f}_{n,opt}) - \mathcal{R}(\hat{f}_n)$ (Improved in this work ✓)
- Approximation error: $\mathcal{R}(f_n^*) - \mathcal{R}(f^*)$. (Improved in this work ?)

Outline

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \tag{1}$$

  where
  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \leq \mathcal{B}, \|\mathrm{b_i}\|_\infty \leq \mathcal{B}\}.$
  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, \ldots, d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \tag{1}$$

  where

  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i=0}}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \leq \mathcal{B}, \|\mathrm{b_i}\|_\infty \leq \mathcal{B}\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, \ldots, d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \tag{1}$$

  where

  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \le \mathcal{B}, \|\mathrm{b_i}\|_\infty \le \mathcal{B}\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, \ldots, d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \quad (1)$$

  where

  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \leq \mathcal{B}, \|\mathrm{b_i}\|_\infty \leq \mathcal{B}\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, \ldots, d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \tag{1}$$

  where

  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \leq \mathcal{B}, \|\mathrm{b_i}\|_\infty \leq \mathcal{B}\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, ..., d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \tag{1}$$

  where

  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \leq \mathcal{B}, \|\mathrm{b_i}\|_\infty \leq \mathcal{B}\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, ..., d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

Feedforward neural networks

- A feedforward network $f(\cdot;\theta)$ with L hidden layers

$$f(x;\theta) = \mathcal{A}_{L+1} \circ \sigma_L \circ \mathcal{A}_L \circ \cdots \circ \sigma_2 \circ \mathcal{A}_2 \circ \sigma_1 \circ \mathcal{A}_1(x). \qquad (1)$$

  where

  1. $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$ collects parameters including weight matrices and bias vectors.
  2. $\mathcal{A}_l(x) = W^{(l)}x + b^{(l)}$ is the i-th linear transformation with $x \in \mathbb{R}^{d_l}$ where $W^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ is the weight matrix and $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the bias vector.
  3. σ_l is an element-wise activation function, e.g., ReLU, Sigmoid, Tanh.

- The Class of networks
  $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{with } \|\mathrm{W_i}\|_\infty \le \mathcal{B}, \|\mathrm{b_i}\|_\infty \le \mathcal{B}\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{d_1, ..., d_L\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{L}\{d_{i+1} \times (d_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{L} d_i$

An example of Feedforward Neural Network

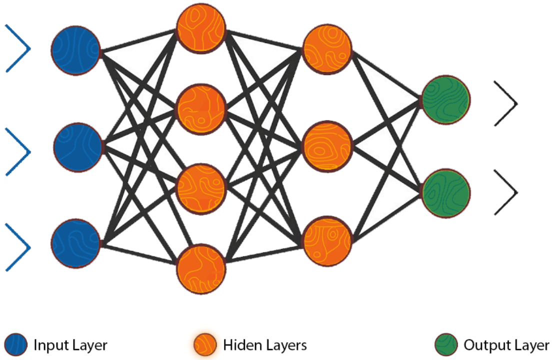


Figure: A feedforward neural network with width $\mathcal{W} = 4$, depth $\mathcal{D} = 2$, size $\mathcal{S} = 39$, number of neurons $\mathcal{U} = 7$ and $(d_0, d_1, d_2, d_3) = (3, 4, 3, 2)$.

Functional Equivalence

- The parameterization of a network is redundant.
- Different parameters produce identical function outputs.
  - Non-identifiability of weight matrices or bias vectors
  - Non-identifiability of activation functions

Functional Equivalence

- The parameterization of a network is redundant.
- Different parameters produce identical function outputs.
  - ▷ Non-identifiability of weight matrices or bias vectors
  - ▷ Non-identifiability of activation functions

Functional Equivalence

- The parameterization of a network is redundant.
- Different parameters produce identical function outputs.
  - Non-identifiability of weight matrices or bias vectors
  - Non-identifiability of activation functions

Functional Equivalence

- The parameterization of a network is redundant.
- Different parameters produce identical function outputs.
  - ▷ Non-identifiability of weight matrices or bias vectors
  - ▷ Non-identifiability of activation functions

Functional Equivalence

- The parameterization of a network is redundant.
- Different parameters produce identical function outputs.
  - ▷ Non-identifiability of weight matrices or bias vectors
  - ▷ Non-identifiability of activation functions

Definition 1 (Functionally-Equivalent Neural Networks)

Two neural networks $f(x; \theta_1)$ and $f(x; \theta_2)$ are said to be functionally-equivalent on $\mathcal{X}$ if they produce the same input-output function for all possible inputs, i.e.,

$$f_1(x; \theta_1) = f_2(x; \theta_2) \quad \forall x \in \mathcal{X}, \tag{2}$$

where $\mathcal{X}$ is the input space and θ_1 and θ_2 denote the sets of parameters of the two networks, respectively.

Functionally Equivalent Networks via Scaling

Example (Scaling)

Consider two shallow neural networks parameterized by $\theta_1 = (W_1^{(1)}, b_1^{(1)}, W_1^{(2)}, b_1^{(2)})$ *and* $\theta_2 = (W_2^{(1)}, b_2^{(1)}, W_2^{(2)}, b_2^{(2)})$*, defined as:*

$$\begin{aligned} f(x;\theta_1) &= W_1^{(2)}\sigma(W_1^{(1)}x + b_1^{(1)}) + b_1^{(2)}, \\ f(x;\theta_2) &= W_2^{(2)}\sigma(W_2^{(1)}x + b_2^{(1)}) + b_2^{(2)} \end{aligned}$$

respectively, where $x \in \mathbb{R}^n$ *is the input to the network and* σ *satisfies* $\sigma(\lambda x) = \lambda\sigma(x)$ *for all* $x \in \mathbb{R}^n$ *and* $\lambda > 0$*. If there exists a scalar value* $\alpha > 0$ *such that:*

$$(W_2^{(1)}, b_2^{(1)}) = (\alpha W_1^{(1)}, \alpha b_1^{(1)}) \quad \text{and} \quad W_2^{(2)} = \frac{1}{\alpha} W_1^{(2)},$$

then $f(\cdot;\theta_1)$ *and* $f(\cdot;\theta_2)$ *are functionally equivalent.*

Scaling Invariance

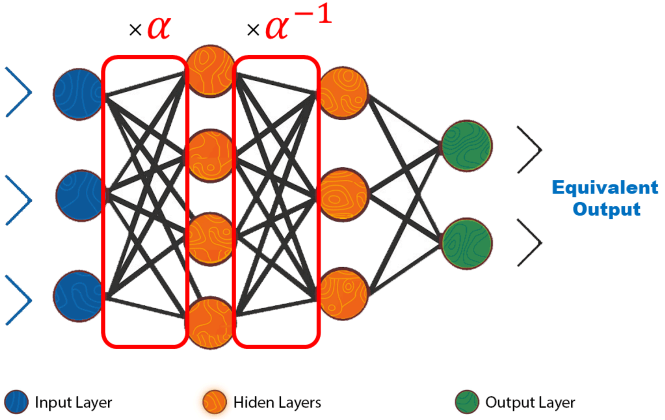


Figure: Functionally Equivalent Networks via Scaling.

- Applicable to ReLU, Leaky ReLU, and piecewise-linear activated networks.
- ReLU or Leaky ReLU: for all $x \in \mathbb{R}^n$ and $\lambda \geq 0$, $\sigma(\lambda x) = \lambda\sigma(x)$.
- Scaling invariance exists in deep networks across any two consecutive layers.

Scaling Invariance

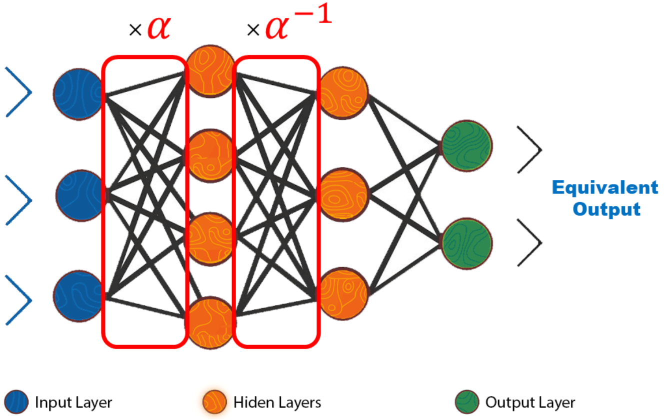


Figure: Functionally Equivalent Networks via Scaling.

- Applicable to ReLU, Leaky ReLU, and piecewise-linear activated networks.
- ReLU or Leaky ReLU: for all $x \in \mathbb{R}^n$ and $\lambda \geq 0$, $\sigma(\lambda x) = \lambda\sigma(x)$.
- Scaling invariance exists in deep networks across any two consecutive layers.

Scaling Invariance

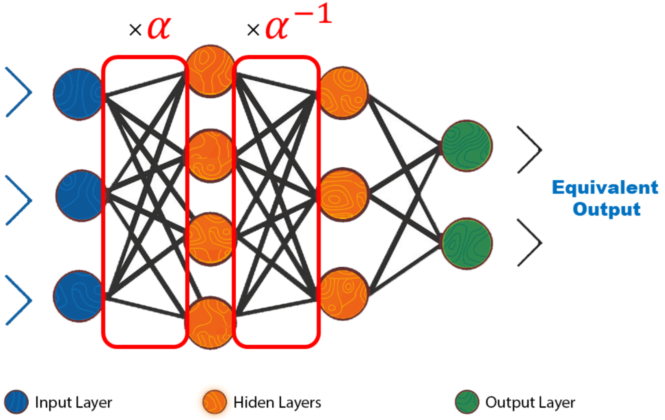


Figure: Functionally Equivalent Networks via Scaling.

- Applicable to ReLU, Leaky ReLU, and piecewise-linear activated networks.
- ReLU or Leaky ReLU: for all $x \in \mathbb{R}^n$ and $\lambda \geq 0$, $\sigma(\lambda x) = \lambda\sigma(x)$.
- Scaling invariance exists in deep networks across any two consecutive layers.

Functionally Equivalent Networks via Sign Flip

Example (Sign Flipping)

Consider two shallow neural networks $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ defined in Example 11 with σ being an odd function, that is $\sigma(-x) = -\sigma(x)$ for all $x \in \mathbb{R}^n$. If

$$(W_2^{(1)}, b_2^{(1)}) = (-W_1^{(1)}, -b_1^{(1)}) \quad \text{and} \quad W_2^{(2)} = -W_1^{(2)},$$

then $f(x;\theta_1)$ and $f(x;\theta_2)$ are functionally equivalent.

- Applicable to neural networks activated by Tanh, Sin and odd functions.
- Sigmoid is Sign-flip invariant up-to a constant 0.5 (constant is mitigated by using a bias[21]).
- Sign flip invariance generalizes to deep networks across any two consecutive layers.

Functionally Equivalent Networks via Sign Flip

Example (Sign Flipping)

Consider two shallow neural networks $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ defined in Example 11 with σ being an odd function, that is $\sigma(-x) = -\sigma(x)$ for all $x \in \mathbb{R}^n$. If

$$(W_2^{(1)}, b_2^{(1)}) = (-W_1^{(1)}, -b_1^{(1)}) \quad \text{and} \quad W_2^{(2)} = -W_1^{(2)},$$

then $f(x;\theta_1)$ and $f(x;\theta_2)$ are functionally equivalent.

- Applicable to neural networks activated by Tanh, Sin and odd functions.
- Sigmoid is Sign-flip invariant up-to a constant 0.5 (constant is mitigated by using a bias[21]).
- Sign flip invariance generalizes to deep networks across any two consecutive layers.

Functionally Equivalent Networks via Sign Flip

Example (Sign Flipping)

Consider two shallow neural networks $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ defined in Example 11 with σ being an odd function, that is $\sigma(-x) = -\sigma(x)$ for all $x \in \mathbb{R}^n$. If

$$(W_2^{(1)}, b_2^{(1)}) = (-W_1^{(1)}, -b_1^{(1)}) \quad \text{and} \quad W_2^{(2)} = -W_1^{(2)},$$

then $f(x;\theta_1)$ and $f(x;\theta_2)$ are functionally equivalent.

- Applicable to neural networks activated by Tanh, Sin and odd functions.
- Sigmoid is Sign-flip invariant up-to a constant 0.5 (constant is mitigated by using a bias[21]).
- Sign flip invariance generalizes to deep networks across any two consecutive layers.

Sign Flip Invariance

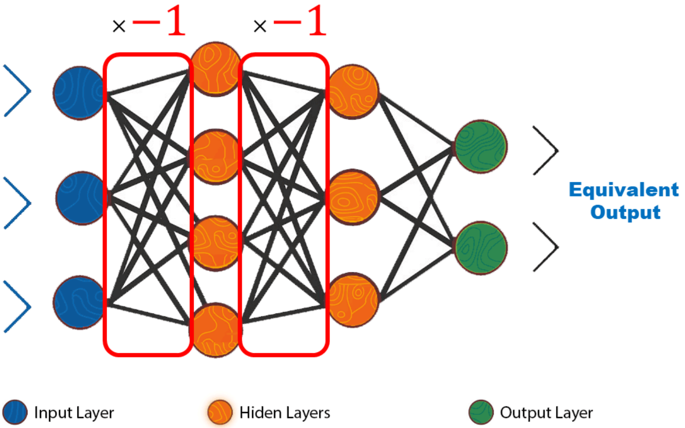


Figure: Functionally Equivalent Networks via Sign Flip.

Functionally Equivalent Networks via Permutation

Example (Permutation)

Consider two shallow neural networks $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ defined in Example 11 with σ being a general activation function. Let the dimension of the hidden layer of $f(x;\theta_1)$ and $f(x;\theta_2)$ be denoted by m. If there exists an $m \times m$ permutation matrix P such that

$$(PW_2^{(1)}, Pb_2^{(1)}) = (W_1^{(1)}, b_1^{(1)}) \quad \text{and} \quad W_2^{(2)}P = W_1^{(2)},$$

then $f(x;\theta_1)$ and $f(x;\theta_2)$ are functionally equivalent.

- Re-indexing neurons in a hidden layer and the corresponding rows of the weights matrix and bias vector will lead to a functionally equivalent network.
- Permutation invariance does not rely on any specific properties of activation functions, the most basic equivalence for networks.

Functionally Equivalent Networks via Permutation

Example (Permutation)

Consider two shallow neural networks $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ defined in Example 11 with σ being a general activation function. Let the dimension of the hidden layer of $f(x;\theta_1)$ and $f(x;\theta_2)$ be denoted by m. If there exists an $m \times m$ permutation matrix P such that

$$(PW_2^{(1)}, Pb_2^{(1)}) = (W_1^{(1)}, b_1^{(1)}) \quad \text{and} \quad W_2^{(2)}P = W_1^{(2)},$$

then $f(x;\theta_1)$ and $f(x;\theta_2)$ are functionally equivalent.

- Re-indexing neurons in a hidden layer and the corresponding rows of the weights matrix and bias vector will lead to a functionally equivalent network.
- Permutation invariance does not rely on any specific properties of activation functions, the most basic equivalence for networks.

Permutation Invariance

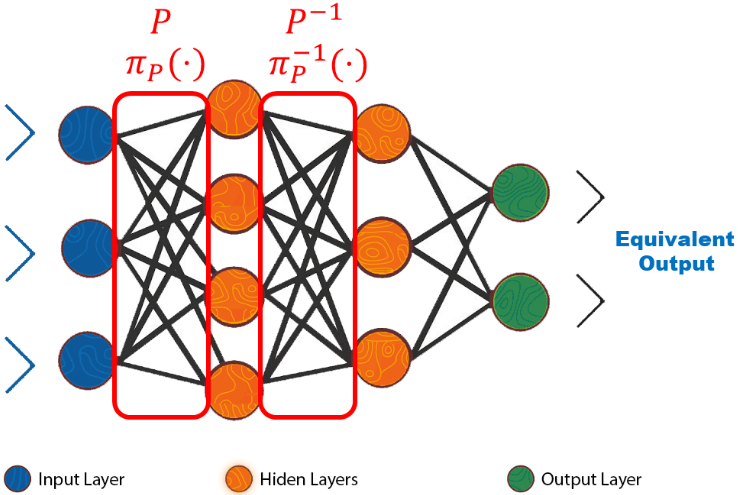


Figure: Functionally Equivalent Networks via Permutation.

Permutation Invariance for deep networks

Proposition (Permutation equivalence for deep networks)

Consider two neural networks $f(x;\theta_1)$ *and* $f(x;\theta_2)$ *with the same activations* $\sigma_1,\ldots,\sigma_L$ *and architecture*

$$f(x;\theta) = W^{(L+1)}\sigma_L(\cdots\sigma_1(W_1^{(1)}x + b_1^{(1)})\cdots) + b_1^{(L)}) + b^{(L+1)}$$

but parameterized by different parameters

$$\theta_j = (W_j^{(1)}, b_j^{(1)}, \ldots, W_j^{(L+1)}, b_j^{(L+1)}), \quad j = 1, 2$$

respectively, where $x \in \mathbb{R}^n$ *is the input to the network. Let* $P^\top$ *denote the transpose of matrix* P*. If there exists permutation matrices* $P_1,\ldots,P_L$ *such that*

$$\begin{aligned} W_1^{(1)} &= P_1 W_2^{(1)}, & b_1^{(1)} &= P_1 b_2^{(1)}, \\ W_1^{(l)} &= P_l W_2^{(l)} P_{l-1}^\top, & b_1^{(l)} &= P_l b_2^{(1)}, \quad l = 2,\ldots,L \\ W_1^{(L+1)} &= W_2^{(L+1)} P_L^\top, & b_1^{(L)} &= b_2^{(L)}, \end{aligned}$$

then $f(x;\theta_1)$ *and* $f(x;\theta_2)$ *are functionally equivalent.*

Activation function

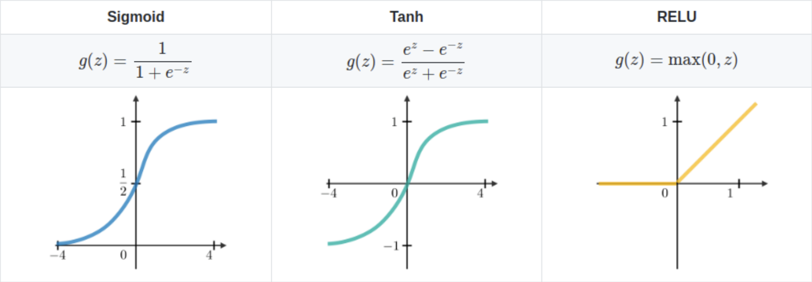


Table: Comparison among ReLU, Sigmoid and ReQU activation functions.

Activation	Formula	Sign flipping	Scaling	Permutation
Sigmoid	$[1+\exp(-x)]^{-1}$	✗	✗	✓
Tanh	$[1-\exp(-2x)]/[1+\exp(-2x)]$	✓	✗	✓
ReLU	$\max\{0, x\}$	✗	✓	✓
Leaky ReLU	$\max\{ax, x\}$ for $a > 0$	✗	✓	✓

Outline

Directions

- How redundant is the parameter space of neural network by equivalence? (Fundamental problem)
- What is the complexity of neural networks by considering equivalence? (Stochastic error)
- How does the symmetric geometry of parameter space help optimization? (Optimization error)

Directions

- How redundant is the parameter space of neural network by equivalence? (Fundamental problem)
- What is the complexity of neural networks by considering equivalence? (Stochastic error)
- How does the symmetric geometry of parameter space help optimization? (Optimization error)

Directions

- How redundant is the parameter space of neural network by equivalence? (Fundamental problem)
- What is the complexity of neural networks by considering equivalence? (Stochastic error)
- How does the symmetric geometry of parameter space help optimization? (Optimization error)

Effective Parameter Space: Shallow Networks

- Shallow neural network with a single hidden layer has universal approximation properties [6, 15]. Sufficient for many learning tasks [12, 14].
- Consider shallow networks with size $\mathcal{S} = (d_0 + 2) \times d_1 + 1$:

$$\mathcal{F}(1, d_0, d_1, B) = \{f(x;\theta) = W^{(2)}\sigma_1(W^{(1)}x + b^{(1)}) + b^{(2)} : \theta \in [-B, B]^{\mathcal{S}}\}.$$

- Permutation leads to equivalence classes of parameters yield the same realization. A canonical choice, of a set of representatives:

$$\Theta_0 := \{\theta \in [-B, B]^{\mathcal{S}} : b_1^{(1)} \geq b_2^{(1)} \geq \cdots \geq b_{d_1}^{(1)}\},$$

by restricting the bias $b^{(1)} = (b_1^{(1)}, \ldots, b_{d_1}^{(1)})^\top$ to have descending entries.

Effective Parameter Space: Shallow Networks

- Shallow neural network with a single hidden layer has universal approximation properties [6, 15]. Sufficient for many learning tasks [12, 14].

- Consider shallow networks with size $\mathcal{S} = (d_0 + 2) \times d_1 + 1$:

$$\mathcal{F}(1, d_0, d_1, B) = \{f(x;\theta) = W^{(2)}\sigma_1(W^{(1)}x + b^{(1)}) + b^{(2)} : \theta \in [-B, B]^{\mathcal{S}}\}.$$

- Permutation leads to equivalence classes of parameters yield the same realization. A canonical choice, of a set of representatives:

$$\Theta_0 := \{\theta \in [-B, B]^{\mathcal{S}} : b_1^{(1)} \geq b_2^{(1)} \geq \cdots \geq b_{d_1}^{(1)}\},$$

by restricting the bias $b^{(1)} = (b_1^{(1)}, \ldots, b_{d_1}^{(1)})^\top$ to have descending entries.

Effective Parameter Space: Shallow Networks

- Shallow neural network with a single hidden layer has universal approximation properties [6, 15]. Sufficient for many learning tasks [12, 14].
- Consider shallow networks with size $\mathcal{S} = (d_0 + 2) \times d_1 + 1$:

$$\mathcal{F}(1, d_0, d_1, B) = \{f(x;\theta) = W^{(2)}\sigma_1(W^{(1)}x + b^{(1)}) + b^{(2)} : \theta \in [-B, B]^{\mathcal{S}}\}.$$

- Permutation leads to equivalence classes of parameters yield the same realization. A canonical choice, of a set of representatives:

$$\Theta_0 := \{\theta \in [-B, B]^{\mathcal{S}} : b_1^{(1)} \geq b_2^{(1)} \geq \cdots \geq b_{d_1}^{(1)}\},$$

by restricting the bias $b^{(1)} = (b_1^{(1)}, \ldots, b_{d_1}^{(1)})^\top$ to have descending entries.

Effective Parameter Space

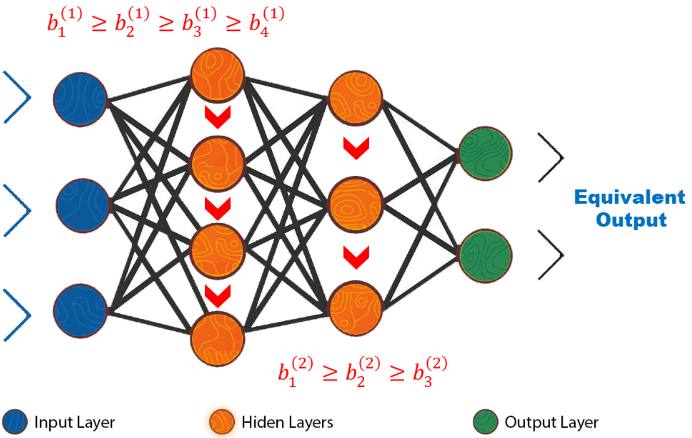


Figure: A canonical choice of representatives by restricting the bias vectors to have descending entries

Effective Parameter Space: Shallow Networks

- Θ_0 may not be the minimal set of representatives since there may be other symmetries.
- The set of representatives Θ_0 has two important properties
  - ▷ Neural networks $\{f(\cdot;\theta) : \theta \in \Theta_0\}$ parameterized by Θ_0 contains all the functions in $\{f(\cdot;\theta) : \theta \in \Theta\}$, i.e.,

$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$

  - ▷ The volume (in terms of Lebesgue measure) of the set of representatives Θ_0 is $(1/d_1!)$ times smaller that that of the parameter space Θ, i.e.,

$$\text{Volume}(\Theta) = d_1! \times \text{Volume}(\Theta_0).$$

Effective Parameter Space: Shallow Networks

- Θ_0 may not be the minimal set of representatives since there may be other symmetries.
- The set of representatives Θ_0 has two important properties
  - ▷ Neural networks $\{f(\cdot;\theta) : \theta \in \Theta_0\}$ parameterized by Θ_0 contains all the functions in $\{f(\cdot;\theta) : \theta \in \Theta\}$, i.e.,

$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$

  - ▷ The volume (in terms of Lebesgue measure) of the set of representatives Θ_0 is $(1/d_1!)$ times smaller that that of the parameter space Θ, i.e.,

$$\mathrm{Volume}(\Theta) = d_1! \times \mathrm{Volume}(\Theta_0).$$

Effective Parameter Space: Shallow Networks

- Θ_0 may not be the minimal set of representatives since there may be other symmetries.
- The set of representatives Θ_0 has two important properties
  - ▷ Neural networks $\{f(\cdot;\theta) : \theta \in \Theta_0\}$ parameterized by Θ_0 contains all the functions in $\{f(\cdot;\theta) : \theta \in \Theta\}$, i.e.,

$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$

  - ▷ The volume (in terms of Lebesgue measure) of the set of representatives Θ_0 is $(1/d_1!)$ times smaller that that of the parameter space Θ, i.e.,

$$\text{Volume}(\Theta) = d_1! \times \text{Volume}(\Theta_0).$$

Effective Parameter Space: Shallow Networks

- Θ_0 may not be the minimal set of representatives since there may be other symmetries.
- The set of representatives Θ_0 has two important properties
  - ▷ Neural networks $\{f(\cdot;\theta) : \theta \in \Theta_0\}$ parameterized by Θ_0 contains all the functions in $\{f(\cdot;\theta) : \theta \in \Theta\}$, i.e.,

$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$

  - ▷ The volume (in terms of Lebesgue measure) of the set of representatives Θ_0 is $(1/d_1!)$ times smaller that that of the parameter space Θ, i.e.,

$$\mathrm{Volume}(\Theta) = d_1! \times \mathrm{Volume}(\Theta_0).$$

Improved Covering Number Bound

Definition 2 (Covering Number)

Let $\mathcal{F} = f : \mathcal{X} \to \mathbb{R}$ be a class of functions. We define the supremum norm of $f \in \mathcal{F}$ as $\|f\|_\infty := \sup_{x \in \mathcal{X}} |f(x)|$. For a given $\epsilon > 0$, we define the covering number of $\mathcal{F}$ with radius ϵ under the norm $\|\cdot\|_\infty$ as the least cardinality of a subset $\mathcal{G} \subseteq \mathcal{F}$ satisfying

$$\sup_{f \in \mathcal{F}} \min_{g \in \mathcal{G}} \|f - g\|_\infty \leq \epsilon.$$

Denoted by $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$, the covering number measures the minimum number of functions in $\mathcal{F}$ needed to cover the set of functions within a distance of ϵ under the supremum norm.

- $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ provides a quantitative measure of the complexity of networks $\mathcal{F}$, with smaller values indicating simpler classes.
- Covering numbers, Rademacher complexity, VC dimension, and Pseudo dimension, are complexity measures in the analysis of learning theories.
- Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq \sqrt{\log[\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)]/n}$.

Improved Covering Number Bound

Definition 2 (Covering Number)

Let $\mathcal{F} = f : \mathcal{X} \to \mathbb{R}$ be a class of functions. We define the supremum norm of $f \in \mathcal{F}$ as $\|f\|_\infty := \sup_{x \in \mathcal{X}} |f(x)|$. For a given $\epsilon > 0$, we define the covering number of $\mathcal{F}$ with radius ϵ under the norm $\|\cdot\|_\infty$ as the least cardinality of a subset $\mathcal{G} \subseteq \mathcal{F}$ satisfying

$$\sup_{f \in \mathcal{F}} \min_{g \in \mathcal{G}} \|f - g\|_\infty \leq \epsilon.$$

Denoted by $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$, the covering number measures the minimum number of functions in $\mathcal{F}$ needed to cover the set of functions within a distance of ϵ under the supremum norm.

- $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ provides a quantitative measure of the complexity of networks $\mathcal{F}$, with smaller values indicating simpler classes.
- Covering numbers, Rademacher complexity, VC dimension, and Pseudo dimension, are complexity measures in the analysis of learning theories.
- Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq \sqrt{\log[\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)]/n}$.

Improved Covering Number Bound

Definition 2 (Covering Number)

Let $\mathcal{F} = f : \mathcal{X} \to \mathbb{R}$ be a class of functions. We define the supremum norm of $f \in \mathcal{F}$ as $\|f\|_\infty := \sup_{x \in \mathcal{X}} |f(x)|$. For a given $\epsilon > 0$, we define the covering number of $\mathcal{F}$ with radius ϵ under the norm $\|\cdot\|_\infty$ as the least cardinality of a subset $\mathcal{G} \subseteq \mathcal{F}$ satisfying

$$\sup_{f \in \mathcal{F}} \min_{g \in \mathcal{G}} \|f - g\|_\infty \leq \epsilon.$$

Denoted by $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$, the covering number measures the minimum number of functions in $\mathcal{F}$ needed to cover the set of functions within a distance of ϵ under the supremum norm.

- $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ provides a quantitative measure of the complexity of networks $\mathcal{F}$, with smaller values indicating simpler classes.
- Covering numbers, Rademacher complexity, VC dimension, and Pseudo dimension, are complexity measures in the analysis of learning theories.
- Stochastic error: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq \sqrt{\log[\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)]/n}$.

Improved Covering Number Bound

Theorem 3 (Covering number of shallow neural networks)

Consider the class of shallow neural networks $\mathcal{F} := \mathcal{F}(1, d_0, d_1, B)$ parameterized by $\theta \in \Theta = [-B, B]^{\mathcal{S}}$. Suppose the radius of the domain $\mathcal{X}$ of $f \in \mathcal{F}$ is bounded by some $B_x > 0$, and the activation σ_1 is continuous. Then for any $\epsilon > 0$, the covering number

$$\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty) \leq (16B^2(B_x + 1)\sqrt{d_0}d_1/\epsilon)^{\mathcal{S}} \times \rho^{\mathcal{S}_h}/d_1!, \tag{3}$$

where ρ denotes the Lipschitz constant of σ_1 on the range of the hidden layer (i.e., $[-\sqrt{d_0}B(B_x) + 1), \sqrt{d_0}B(B_x + 1)]$), and $\mathcal{S}_h = d_0 d_1 + d_1$ is the total number of parameters in the linear transformation from input to the hidden layer, and $\mathcal{S} = d_0 \times d_1 + 2d_1 + 1$ is the total number of parameters.

- A reduced complexity (by $d_1!$) compared to existing studies [25, 3, 27, 23, 17]. For a shallow ReLU network with $d_1 = 128$, covering number reduced by $\approx 10^{215}$.
- Network is allowed to have bias vectors (crucial for approximation) [36, 19, 33].
- The covering number is in the uniform sense over existing studies defined on a finite sample [2, 3].

Improved Covering Number Bound

Theorem 3 (Covering number of shallow neural networks)

Consider the class of shallow neural networks $\mathcal{F} := \mathcal{F}(1, d_0, d_1, B)$ parameterized by $\theta \in \Theta = [-B, B]^{\mathcal{S}}$. Suppose the radius of the domain $\mathcal{X}$ of $f \in \mathcal{F}$ is bounded by some $B_x > 0$, and the activation σ_1 is continuous. Then for any $\epsilon > 0$, the covering number

$$\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty) \leq (16B^2(B_x + 1)\sqrt{d_0}d_1/\epsilon)^{\mathcal{S}} \times \rho^{\mathcal{S}_h}/d_1!, \tag{3}$$

where ρ denotes the Lipschitz constant of σ_1 on the range of the hidden layer (i.e., $[-\sqrt{d_0}B(B_x) + 1), \sqrt{d_0}B(B_x + 1)]$), and $\mathcal{S}_h = d_0 d_1 + d_1$ is the total number of parameters in the linear transformation from input to the hidden layer, and $\mathcal{S} = d_0 \times d_1 + 2d_1 + 1$ is the total number of parameters.

- A reduced complexity (by $d_1!$) compared to existing studies [25, 3, 27, 23, 17]. For a shallow ReLU network with $d_1 = 128$, covering number reduced by $\approx 10^{215}$.
- Network is allowed to have bias vectors (crucial for approximation) [36, 19, 33].
- The covering number is in the uniform sense over existing studies defined on a finite sample [2, 3].

Improved Covering Number Bound

Theorem 3 (Covering number of shallow neural networks)

Consider the class of shallow neural networks $\mathcal{F} := \mathcal{F}(1, d_0, d_1, B)$ parameterized by $\theta \in \Theta = [-B, B]^{\mathcal{S}}$. Suppose the radius of the domain $\mathcal{X}$ of $f \in \mathcal{F}$ is bounded by some $B_x > 0$, and the activation σ_1 is continuous. Then for any $\epsilon > 0$, the covering number

$$\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty) \leq (16B^2(B_x+1)\sqrt{d_0}d_1/\epsilon)^{\mathcal{S}} \times \rho^{\mathcal{S}_h}/d_1!, \tag{3}$$

where ρ denotes the Lipschitz constant of σ_1 on the range of the hidden layer (i.e., $[-\sqrt{d_0}B(B_x)+1), \sqrt{d_0}B(B_x+1)]$), and $\mathcal{S}_h = d_0 d_1 + d_1$ is the total number of parameters in the linear transformation from input to the hidden layer, and $\mathcal{S} = d_0 \times d_1 + 2d_1 + 1$ is the total number of parameters.

- A reduced complexity (by $d_1!$) compared to existing studies [25, 3, 27, 23, 17]. For a shallow ReLU network with $d_1 = 128$, covering number reduced by $\approx 10^{215}$.
- Network is allowed to have bias vectors (crucial for approximation) [36, 19, 33].
- The covering number is in the uniform sense over existing studies defined on a finite sample [2, 3].

Effective Parameter Space: Deep Networks

- A set of representatives $\Theta_0 = \Theta_0^{(1)} \times \Theta_0^{(2)} \times \cdots \times \Theta_0^{(L)} \times \Theta_0^{(L+1)}$ can be

$$\Theta_0^{(l)} = \Big\{(W^{(l)}, b^{(l)}) \in [-B, B]^{\mathcal{S}_l} : b_1^{(l)} \geq b_2^{(l)} \geq \cdots \geq b_{d_l}^{(l)}\}$$
$$\text{for } l = 1, \ldots, L, \Theta_0^{(L+1)} = \{(W^{(L+1)}, b^{(L+1)}) \in [-B, B]^{\mathcal{S}_{L+1}}\Big\}.$$

- The set of representatives Θ_0 has two important properties
  - Representation:
  $$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$
  - Efficiency:
  $$\text{Volume}(\Theta) = d_1! \times \cdots \times d_L! \times \text{Volume}(\Theta_0).$$

Effective Parameter Space: Deep Networks

- A set of representatives $\Theta_0 = \Theta_0^{(1)} \times \Theta_0^{(2)} \times \cdots \times \Theta_0^{(L)} \times \Theta_0^{(L+1)}$ can be

$$\Theta_0^{(l)} = \Big\{(W^{(l)}, b^{(l)}) \in [-B, B]^{\mathcal{S}_l} : b_1^{(l)} \geq b_2^{(l)} \geq \cdots \geq b_{d_l}^{(l)}\}$$
$$\text{for } l = 1, \ldots, L, \Theta_0^{(L+1)} = \{(W^{(L+1)}, b^{(L+1)}) \in [-B, B]^{\mathcal{S}_{L+1}}\Big\}.$$

- The set of representatives Θ_0 has two important properties
  - Representation:
$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$
  - Efficiency:
$$\text{Volume}(\Theta) = d_1! \times \cdots \times d_L! \times \text{Volume}(\Theta_0).$$

Effective Parameter Space: Deep Networks

- A set of representatives $\Theta_0 = \Theta_0^{(1)} \times \Theta_0^{(2)} \times \cdots \times \Theta_0^{(L)} \times \Theta_0^{(L+1)}$ can be

$$\Theta_0^{(l)} = \Big\{ (W^{(l)}, b^{(l)}) \in [-B, B]^{\mathcal{S}_l} : b_1^{(l)} \geq b_2^{(l)} \geq \cdots \geq b_{d_l}^{(l)} \}$$
$$\text{for } l = 1, \ldots, L, \Theta_0^{(L+1)} = \{ (W^{(L+1)}, b^{(L+1)}) \in [-B, B]^{\mathcal{S}_{L+1}} \Big\}.$$

- The set of representatives Θ_0 has two important properties
  - ▷ Representation:
$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$
  - ▷ Efficiency:
$$\text{Volume}(\Theta) = d_1! \times \cdots \times d_L! \times \text{Volume}(\Theta_0).$$

Effective Parameter Space: Deep Networks

- A set of representatives $\Theta_0 = \Theta_0^{(1)} \times \Theta_0^{(2)} \times \cdots \times \Theta_0^{(L)} \times \Theta_0^{(L+1)}$ can be

$$\Theta_0^{(l)} = \Big\{ (W^{(l)}, b^{(l)}) \in [-B, B]^{\mathcal{S}_l} : b_1^{(l)} \geq b_2^{(l)} \geq \cdots \geq b_{d_l}^{(l)} \}$$

$$\text{for } l = 1, \ldots, L, \Theta_0^{(L+1)} = \{ (W^{(L+1)}, b^{(L+1)}) \in [-B, B]^{\mathcal{S}_{L+1}} \Big\}.$$

- The set of representatives Θ_0 has two important properties
  - ▷ Representation:

$$\{f(\cdot;\theta) : \theta \in \Theta_0\} = \{f(\cdot;\theta) : \theta \in \Theta\}.$$

  - ▷ Efficiency:

$$\text{Volume}(\Theta) = d_1! \times \cdots \times d_L! \times \text{Volume}(\Theta_0).$$

Improved Covering Number Bound

Theorem 4 (Covering number of deep neural networks)

Consider the class of deep neural networks $\mathcal{F} := \mathcal{F}(1, d_0, d_1, \ldots, d_L, B)$ *parameterized by* $\theta \in \Theta = [-B, B]^{\mathcal{S}}$. *Suppose the radius of the domain* $\mathcal{X}$ *of* $f \in \mathcal{F}$ *is bounded by* B_x *for some* $B_x > 0$, *and the activations* $\sigma_1, \ldots, \sigma_L$ *are locally Lipschitz. Then for any* $\epsilon > 0$, *the covering number* $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ *is bounded by*

$$\frac{\left(4(L+1)(B_x+1)(2B)^{L+2}(\Pi_{j=1}^{L}\rho_j)(\Pi_{j=0}^{L}d_j)\cdot\epsilon^{-1}\right)^{\mathcal{S}}}{d_1! \times d_2! \times \cdots \times d_L!},$$

where $\mathcal{S} = \sum_{i=0}^{L} d_i d_{i+1} + d_{i+1}$ *and* ρ_i *denotes the Lipschitz constant of* σ_i *on the range of* $(i-1)$*-th hidden layer, especially the range of* $(i-1)$*-th hidden layer is bounded by* $[-B^{(i)}, B^{(i)}]$ *with* $B^{(i)} \leq (2B)^i \Pi_{j=1}^{i-1}\rho_j d_j$ *for* $i = 1, \ldots, L$.

- A reduced complexity (by $(d_1!d_2!\cdots d_L!)$) over existing studies [25, 3, 27, 23, 17].
- Increasing depth L does increase complexity. The increased hidden layer l will have a $(d_l!)$ discount on the complexity.
- Network is allowed to have bias vectors (crucial for approximation) [36, 19, 33].

Improved Covering Number Bound

Theorem 4 (Covering number of deep neural networks)

Consider the class of deep neural networks $\mathcal{F} := \mathcal{F}(1, d_0, d_1, \ldots, d_L, B)$ *parameterized by* $\theta \in \Theta = [-B, B]^{\mathcal{S}}$. *Suppose the radius of the domain* $\mathcal{X}$ *of* $f \in \mathcal{F}$ *is bounded by* B_x *for some* $B_x > 0$, *and the activations* $\sigma_1, \ldots, \sigma_L$ *are locally Lipschitz. Then for any* $\epsilon > 0$, *the covering number* $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ *is bounded by*

$$\frac{\left(4(L+1)(B_x+1)(2B)^{L+2}(\Pi_{j=1}^{L}\rho_j)(\Pi_{j=0}^{L}d_j)\cdot\epsilon^{-1}\right)^{\mathcal{S}}}{d_1! \times d_2! \times \cdots \times d_L!},$$

where $\mathcal{S} = \sum_{i=0}^{L} d_i d_{i+1} + d_{i+1}$ *and* ρ_i *denotes the Lipschitz constant of* σ_i *on the range of* $(i-1)$*-th hidden layer, especially the range of* $(i-1)$*-th hidden layer is bounded by* $[-B^{(i)}, B^{(i)}]$ *with* $B^{(i)} \leq (2B)^i \Pi_{j=1}^{i-1}\rho_j d_j$ *for* $i = 1, \ldots, L$.

- A reduced complexity (by $(d_1!d_2!\cdots d_L!)$) over existing studies [25, 3, 27, 23, 17].
- Increasing depth L does increase complexity. The increased hidden layer l will have a $(d_l!)$ discount on the complexity.
- Network is allowed to have bias vectors (crucial for approximation) [36, 19, 33].

Improved Covering Number Bound

Theorem 4 (Covering number of deep neural networks)

Consider the class of deep neural networks $\mathcal{F} := \mathcal{F}(1, d_0, d_1, \ldots, d_L, B)$ *parameterized by* $\theta \in \Theta = [-B, B]^{\mathcal{S}}$. *Suppose the radius of the domain* $\mathcal{X}$ *of* $f \in \mathcal{F}$ *is bounded by* B_x *for some* $B_x > 0$, *and the activations* $\sigma_1, \ldots, \sigma_L$ *are locally Lipschitz. Then for any* $\epsilon > 0$, *the covering number* $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ *is bounded by*

$$\frac{\left(4(L+1)(B_x+1)(2B)^{L+2}(\Pi_{j=1}^{L}\rho_j)(\Pi_{j=0}^{L}d_j)\cdot\epsilon^{-1}\right)^{\mathcal{S}}}{d_1! \times d_2! \times \cdots \times d_L!},$$

where $\mathcal{S} = \sum_{i=0}^{L} d_i d_{i+1} + d_{i+1}$ *and* ρ_i *denotes the Lipschitz constant of* σ_i *on the range of* $(i-1)$*-th hidden layer, especially the range of* $(i-1)$*-th hidden layer is bounded by* $[-B^{(i)}, B^{(i)}]$ *with* $B^{(i)} \leq (2B)^i \Pi_{j=1}^{i-1}\rho_j d_j$ *for* $i = 1, \ldots, L$.

- A reduced complexity (by $(d_1!d_2!\cdots d_L!)$) over existing studies [25, 3, 27, 23, 17].
- Increasing depth L does increase complexity. The increased hidden layer l will have a $(d_l!)$ discount on the complexity.
- Network is allowed to have bias vectors (crucial for approximation) [36, 19, 33].

Comparison to existing results

Table: A comparison of recent results on the complexity of feedforward neural networks.

Paper	Complexity	Bias Vec.	General Activations	Perm. Inv.
[3]	$B_x^2(\bar{\rho}\bar{s})^2\mathcal{U}\log(W)/\epsilon^2$	✗	✗	✗
[27]	$B_x^2(\bar{\rho}\bar{s})^2\mathcal{S}L^2\log(WL)/\epsilon^2$	✗	✗	✗
[17]	$B_x(\bar{\rho}\bar{s})\mathcal{S}^2L/\epsilon$	✗	✗	✗
[4]	$L\mathcal{S}\log(\mathcal{S})\log(\bar{\rho}\bar{s}B_x/\epsilon)$	✓	✓	✗
This paper	$L\mathcal{S}\log(\bar{\rho}\bar{s}B_x^{1/L}/((d_1!\cdots d_L!)^{1/\mathcal{S}}\epsilon)^{1/L})$	✓	✓	✓

Notations: $\mathcal{S}$ number of parameters; $\mathcal{U}$ number of hidden neurons; L number of hidden layers; W maximum hidden layers width; B_x, L2 norm of input; $\bar{\rho} = \Pi_{j=1}^{L}\rho_j$, products of Lipschitz constants of activations; $\bar{s} = \Pi_{j=1}^{L}s_j$, products of spectral norms of hidden layer weight matrices; ϵ, radius for covering number.

Outline

- Stochastic error:
  - For regressions and classifications: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq [\sqrt{\frac{\log[\mathcal{N}(\mathcal{F}, \frac{1}{n}, \|\cdot\|_\infty)]}{n}}]^k$ for $k = 1, 2$ [4, 16].
- Optimization Error:
  - Quantitative analysis is limited [35] even for convergence, due to high non-convexity.
  - [34] studied the geometry (in terms of manifold and connected affine subspace) of sets of minima and critical points when increasing the width of a network. Overparameterized networks bear more minima solutions.
  - The symmetry structure of parameter space implies larger probability of achieving zero (or some level of) optimization error.

Benefits to Generalization

- Stochastic error:
  - For regressions and classifications:
    $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq [\sqrt{\frac{\log[\mathcal{N}(\mathcal{F}, \frac{1}{n}, \|\cdot\|_\infty)]}{n}}]^k$ for $k = 1, 2$ [4, 16].
- Optimization Error:
  - Quantitative analysis is limited [35] even for convergence, due to high non-convexity.
  - [34] studied the geometry (in terms of manifold and connected affine subspace) of sets of minima and critical points when increasing the width of a network. Overparameterized networks bear more minima solutions.
  - The symmetry structure of parameter space implies larger probability of achieving zero (or some level of) optimization error.

Benefits to Generalization

- Stochastic error:
  - ▷ For regressions and classifications:
    $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq [\sqrt{\frac{\log[\mathcal{N}(\mathcal{F}, \frac{1}{n}, \|\cdot\|_\infty)]}{n}}]^k$ for $k = 1, 2$ [4, 16].
- Optimization Error:
  - ▷ Quantitative analysis is limited [35] even for convergence, due to high non-convexity.
  - ▷ [34] studied the geometry (in terms of manifold and connected affine subspace) of sets of minima and critical points when increasing the width of a network. Overparameterized networks bear more minima solutions.
  - ▷ The symmetry structure of parameter space implies larger probability of achieving zero (or some level of) optimization error.

Benefits to Generalization

- Stochastic error:
  - For regressions and classifications: $\mathcal{R}(\hat{f}_n) - \mathcal{R}(f_n^*) \leq [\sqrt{\frac{\log[\mathcal{N}(\mathcal{F}, \frac{1}{n}, \|\cdot\|_\infty)]}{n}}]^k$ for $k = 1, 2$ [4, 16].
- Optimization Error:
  - Quantitative analysis is limited [35] even for convergence, due to high non-convexity.
  - [34] studied the geometry (in terms of manifold and connected affine subspace) of sets of minima and critical points when increasing the width of a network. Overparameterized networks bear more minima solutions.
  - The symmetry structure of parameter space implies larger probability of achieving zero (or some level of) optimization error.

Benefits to Optimization

Theorem 5

Suppose we have an ERM $f_{\theta_n}(\cdot) = f(\cdot;\theta_n)$ with parameter θ_n having $(d_1^*, \dots, d_L^*)$ distinct permutations and $\Delta_{\min}(\theta_n) = \delta$. For any optimization algorithm $\mathcal{A}$, if it guarantees producing a convergent solution of θ_n when its initialization $\theta_n^{(0)}$ satisfies $\Delta_{\max}(\theta_n^{(0)} - \theta_n) \le \delta/2$, then any initialization scheme that uses identical random dist. for the entries of weights and biases within a layer will produce a convergent solution with probability at least $d_1^* \times \cdots \times d_L^* \times \mathbb{P}(\Delta_{\max}(\theta^{(0)} - \theta_n) \le \delta/2)$. Here, $\theta^{(0)}$ denotes the random initialization, and $\mathbb{P}(\cdot)$ is with respect to the randomness from initialization.

- A solution θ_n implies other solution $\tilde{\theta}_n$ by permutation.
- Volume $(2B)^{\mathcal{S}}/(d_1! \cdots d_L!)$ approaches zero when $d_l \to \infty$ for $l = 1, \dots, L$.
- The initialization schemes Xavier and He's methods, reduce the optimization difficulty due to the permutation invariance [10, 13, 32, 31].

Extension to CNN

Example (Permutation within Pooling Regions)

Consider two shallow CNNs defined by $f(x;\theta_1) = Pool(W_1x + b_1)$ *and* $f(x;\theta_2) = Pool(W_2x + b_2)$ *respectively where "Pool" is a pooling operator. Let* $\mathcal{I}_1, \ldots, \mathcal{I}_K$ *be the non-overlapping index sets (correspond to the pooling operator) of rows of* $W_1x + b_1$ *and* $W_2x + b_2$*. Then* $f(\cdot;\theta_1)$ *and* $f(\cdot;\theta_2)$ *are functional equivalent if there exists a permutation matrix* P *such that* $\forall k \in \{1, \ldots, K\}$

$$(PW_2)_{\mathcal{I}_k} \cong (W_1)_{\mathcal{I}_k} \text{ and } (Pb_2)_{\mathcal{I}_k} \cong (b_1)_{\mathcal{I}_k},$$

where $A_{\mathcal{I}_k}$ *denotes the* $\mathcal{I}_k$ *rows of* A *and* $A \cong B$ *denotes that* A *equals to* B *up to row permutations.*

- Permutation within non-overlapping regions in CNNs preserves max/min/avg values.
- FNN and CNN can be converted with the same order of parameter. [39, 38, 8, 20].

Extension to ResNet

Example (Equivalence of Residual Layer)

Consider two residual layers $f(x;\theta_1) = x + F(x;\theta_1)$ and $f(x;\theta_2) = x + F(x;\theta_2)$. Then $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ are functionally equivalent if and only if $F(\cdot;\theta_1)$ and $F(\cdot;\theta_2)$ are functionally equivalent.

- ResNet uses skip connections $f(x;\theta) = x + \mathrm{F}(x;\theta)$.
- Then the equivalence of F implies that of the residual layer.

Extension to Attention Model

Example (Permutation within Attention map)

Let $X_{n\times d}$ denote the input of a n-sequence of d-dimensional embeddings, and let $W^Q_{d\times d_q}$, $W^K_{d\times d_k}$ and $W^V_{d\times d_v}$ be the weight matrices where $d_q = d_k$. Then the self-attention map outputs

$$Softmax\left(\frac{XW^Q(W^K)'X'}{\sqrt{d_k}}\right)XW^V$$

where the $Softmax(\cdot)$ is applied to each row of its input and A' denotes the transpose of a matrix A. Consider two attention maps $f(x;\theta_1)$ and $f(x;\theta_2)$ with $f(x;\theta_i) = Softmax(XW_i^Q(W_i^K)'X'/\sqrt{d_k})XW_i^V$ for $i = 1, 2$. Then $f(\cdot;\theta_1)$ and $f(\cdot;\theta_2)$ are functionally equivalent if there exists $d_k \times d_k$ permutation matrix P such that

$$W_2^Q P = W_1^Q \quad \text{and} \quad W_2^K P = W_1^K.$$

- No activation function between the key and query matrices.
- Symmetry considered for any equivalent linear maps $W^Q(W^K)'$.
- Output of *Softmax* is invariant to the row shift of its input.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
    1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
    2. Implications for understanding generalization and optimization
    3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
    1. Sign flip and scaling invariance (relevant for specific activations)
    2. Functional equivalence defined on a finite sample
    3. Deriving the lower bound of the covering numbers.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
  1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
  2. Implications for understanding generalization and optimization
  3. Functional equivalence in convolutional, residual and attention-based networks.

- Limitation and future work
  1. Sign flip and scaling invariance (relevant for specific activations)
  2. Functional equivalence defined on a finite sample
  3. Deriving the lower bound of the covering numbers.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
  1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
  2. Implications for understanding generalization and optimization
  3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
  1. Sign flip and scaling invariance (relevant for specific activations)
  2. Functional equivalence defined on a finite sample
  3. Deriving the lower bound of the covering numbers.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
    1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
    2. Implications for understanding generalization and optimization
    3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
    1. Sign flip and scaling invariance (relevant for specific activations)
    2. Functional equivalence defined on a finite sample
    3. Deriving the lower bound of the covering numbers.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
  1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
  2. Implications for understanding generalization and optimization
  3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
  1. Sign flip and scaling invariance (relevant for specific activations)
  2. Functional equivalence defined on a finite sample
  3. Deriving the lower bound of the covering numbers.

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
  1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
  2. Implications for understanding generalization and optimization
  3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
  1. Sign flip and scaling invariance (relevant for specific activations)
  2. Functional equivalence defined on a finite sample
  3. Deriving the lower bound of the covering numbers.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
  1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
  2. Implications for understanding generalization and optimization
  3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
  1. Sign flip and scaling invariance (relevant for specific activations)
  2. Functional equivalence defined on a finite sample
  3. Deriving the lower bound of the covering numbers.

Conclusion

- Characterized the redundancy in the parameterization of deep neural networks based on functional equivalence
  1. A tighter, explicit bound of the covering number, allowing bias vectors and general activations
  2. Implications for understanding generalization and optimization
  3. Functional equivalence in convolutional, residual and attention-based networks.
- Limitation and future work
  1. Sign flip and scaling invariance (relevant for specific activations)
  2. Functional equivalence defined on a finite sample
  3. Deriving the lower bound of the covering numbers.

References

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

[2] M. Anthony, P. L. Bartlett, P. L. Bartlett, et al. *Neural network learning: Theoretical foundations*, volume 9. cambridge university press Cambridge, 1999.

[3] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

[4] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*, 20(1):2285–2301, 2019.

[5] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116 (32):15849–15854, 2019.

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

[7] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

[8] Z. Fang, H. Feng, S. Huang, and D.-X. Zhou. Theory of deep convolutional neural networks ii: Spherical analysis. *Neural Networks*, 131:154–162, 2020.

References

[9] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJl-b3RcF7`.

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[11] 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. JMLR Workshop and Conference Proceedings, 2011.

[12] B. Hanin and D. Rolnick. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems*, 32, 2019.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[14] C. Hertrich, A. Basu, M. Di Summa, and M. Skutella. Towards lower bounds on the depth of relu neural networks. *Advances in Neural Information Processing Systems*, 34:3336–3348, 2021.

[15] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

[16] M. Kohler and S. Langer. On the rate of convergence of fully connected deep neural network regression estimates. *The Annals of Statistics*, 49(4):2231–2249, 2021.

References

[17] S. Lin and J. Zhang. Generalization bounds for convolutional neural networks. *arXiv preprint arXiv:1910.01487*, 2019.

[18] J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

[19] J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

[20] T. Mao, Z. Shi, and D.-X. Zhou. Theory of deep convolutional neural networks iii: Approximating radial functions. *Neural Networks*, 144:778–790, 2021.

[21] F. Martinelli, B. Simsek, J. Brea, and W. Gerstner. Expand-and-cluster: exact parameter recovery of neural networks. *arXiv preprint arXiv:2304.12794*, 2023.

[22] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[23] B. Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[24] B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[25] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR, 2015.

[26] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

References

[27] B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

[28] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BygfghAcYX`.

[29] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJC2SzZCW`.

[30] N. Razin and N. Cohen. Implicit regularization in deep learning may not be explainable by norms. *Advances in neural information processing systems*, 33:21174–21187, 2020.

[31] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[32] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pages 2217–2225. PMLR, 2016.

[33] G. Shen, Y. Jiao, Y. Lin, and J. Huang. Approximation with cnns in sobolev space: with applications to classification. *Advances in Neural Information Processing Systems*, 35: 2876–2888, 2022.

[34] B. Simsek, F. Ged, A. Jacot, F. Spadaro, C. Hongler, W. Gerstner, and J. Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pages 9722–9732. PMLR, 2021.

[35] R.-Y. Sun. Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 8(2):249–294, 2020.

[36] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94: 103–114, 2017.

[37] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy8gdB9xx`.

[38] D.-X. Zhou. Theory of deep convolutional neural networks: Downsampling. *Neural Networks*, 124:319–327, 2020.

[39] D.-X. Zhou. Universality of deep convolutional neural networks. *Appl. Comput. Harmon. Anal.*, 48(2):787–794, 2020.

Thank you!