ECE 57000

Guest Lecture:

Over-parameterization & Transition to Linearity of Neural Networks

Chaoyue Liu

Fall 2025

October 15, 2025

Observations & Questions

- Neural network architectures are often designed through heuristics and trial-and-error
  - MLP, CNN, RNN, LSTM, Transformers etc.
  - Much like Alchemy
  - Neural networks are considered very complicated: "black-box"
  - Loss function looks super complicated, too: highly non-convex

Question: How complex are neural networks, in reality?

- However, optimization algorithms are quite simple
  - Pure SGD often works well enough

Question:
Why does the simple SGD optimize NN so well in practice?

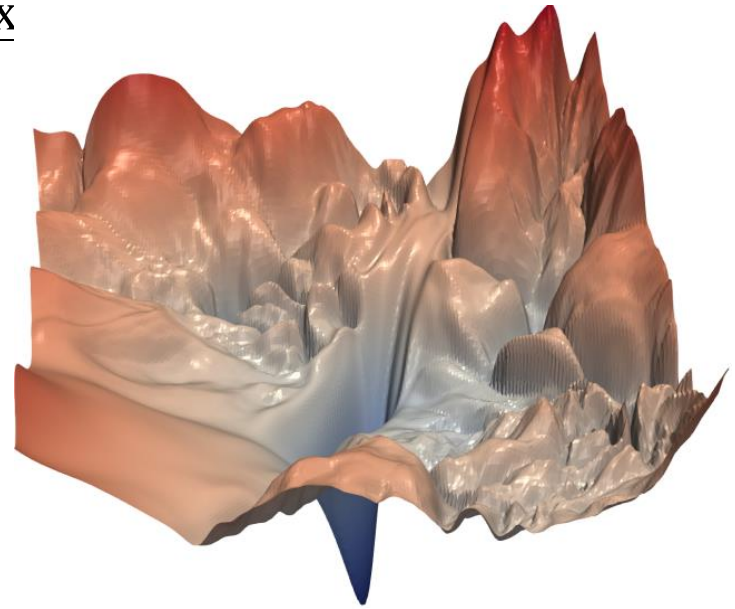


Figure source: https://arxiv.org/pdf/1712.09913

Over-parameterization

Over-parameterization helps to simplify!

- NNs are not as complicated as we thought, at least when NN is over-parameterized.

The number of model parameters p is greater than the number of training samples n

Over-parameterization is real!

- Training a ResNet50 on CIFAR-10 dataset
  - $p \approx 25$ million, $n = 50k$
- Training a 5-layer MLP (500 hidden neuron each layer) on MNIST dataset
  - $p \approx 1$ million, $n = 60k$

Over-parameterization

The number of model parameters p is greater than the number of training samples n

The goal of ML training is to fit the data (or at least *approximately* fit):

$$f_i(\mathbf{w}) = f(\mathbf{w}; \mathbf{x}_i) = y_i$$

Another perspective: solve the set of equations

$$\begin{aligned} f_1(\mathbf{w}) &= y_1 \\ f_2(\mathbf{w}) &= y_2 \\ &\cdots \\ f_n(\mathbf{w}) &= y_n \end{aligned}$$

n equations (constraints)

p parameters (degrees of freedom)

In over-parameterization regime, the data can be exactly fit (namely, the set of equations can be exactly solved) -- **Interpolation**

Interpolation

the data can be exactly fit :

$$f_i(\mathbf{w}) = f(\mathbf{w}; \mathbf{x}_i) = y_i, \qquad \forall i \in [n]$$

Training loss can be exactly zero $\mathcal{L}(\mathbf{w}^*) = 0$.

Global minima always have zero training loss

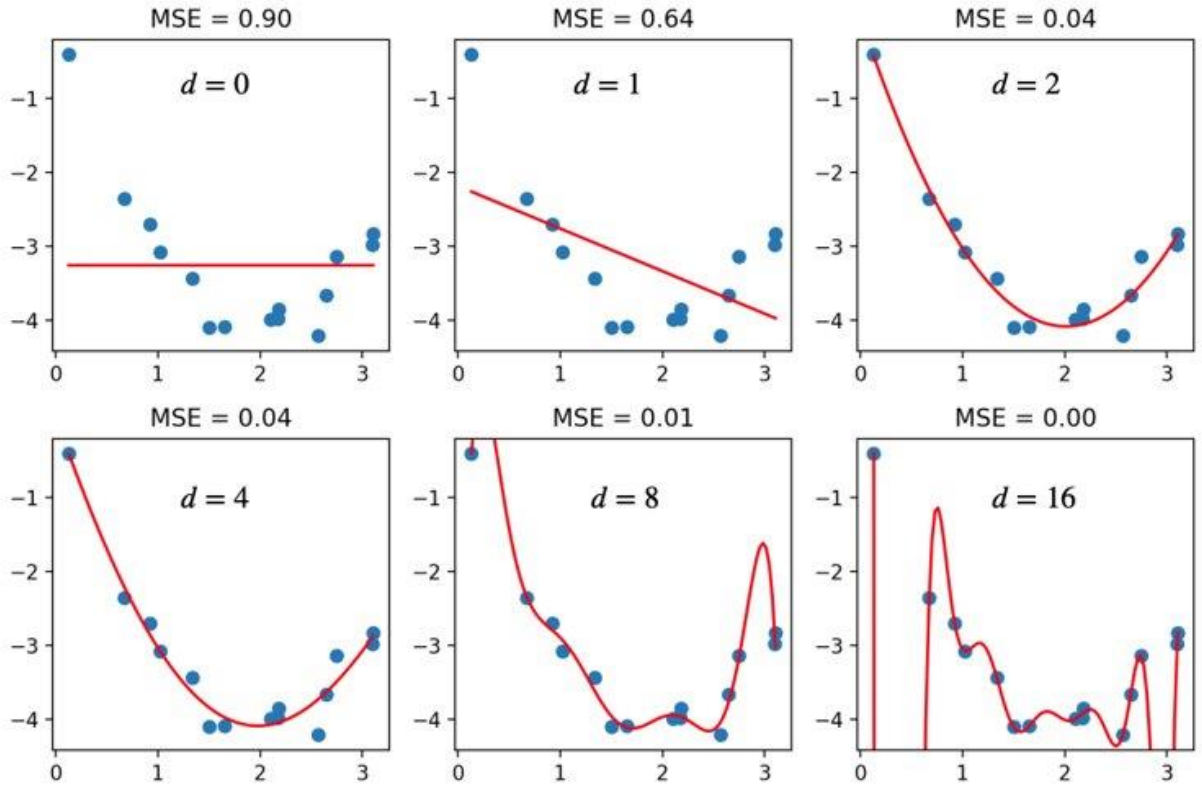
Moreover, each individual loss can be exactly zero:

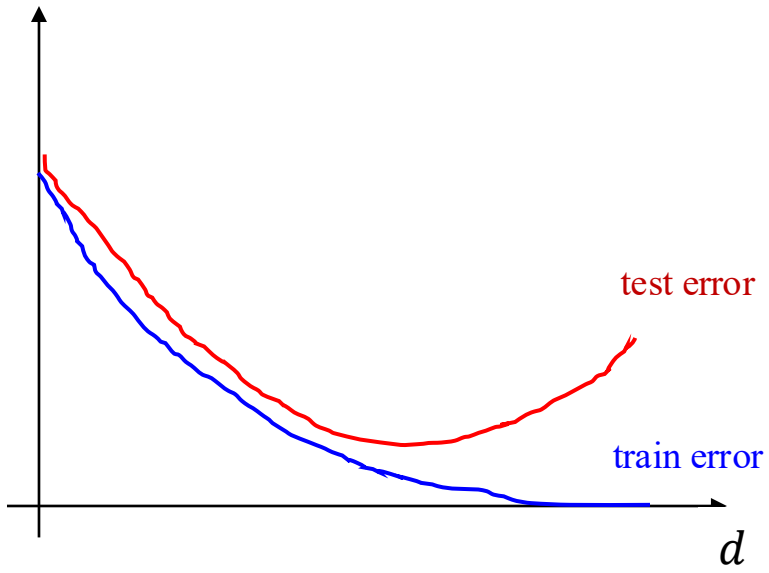
$$\ell_i(\mathbf{w}^*) = 0, \qquad \forall i \in [n]$$

Overfitting?

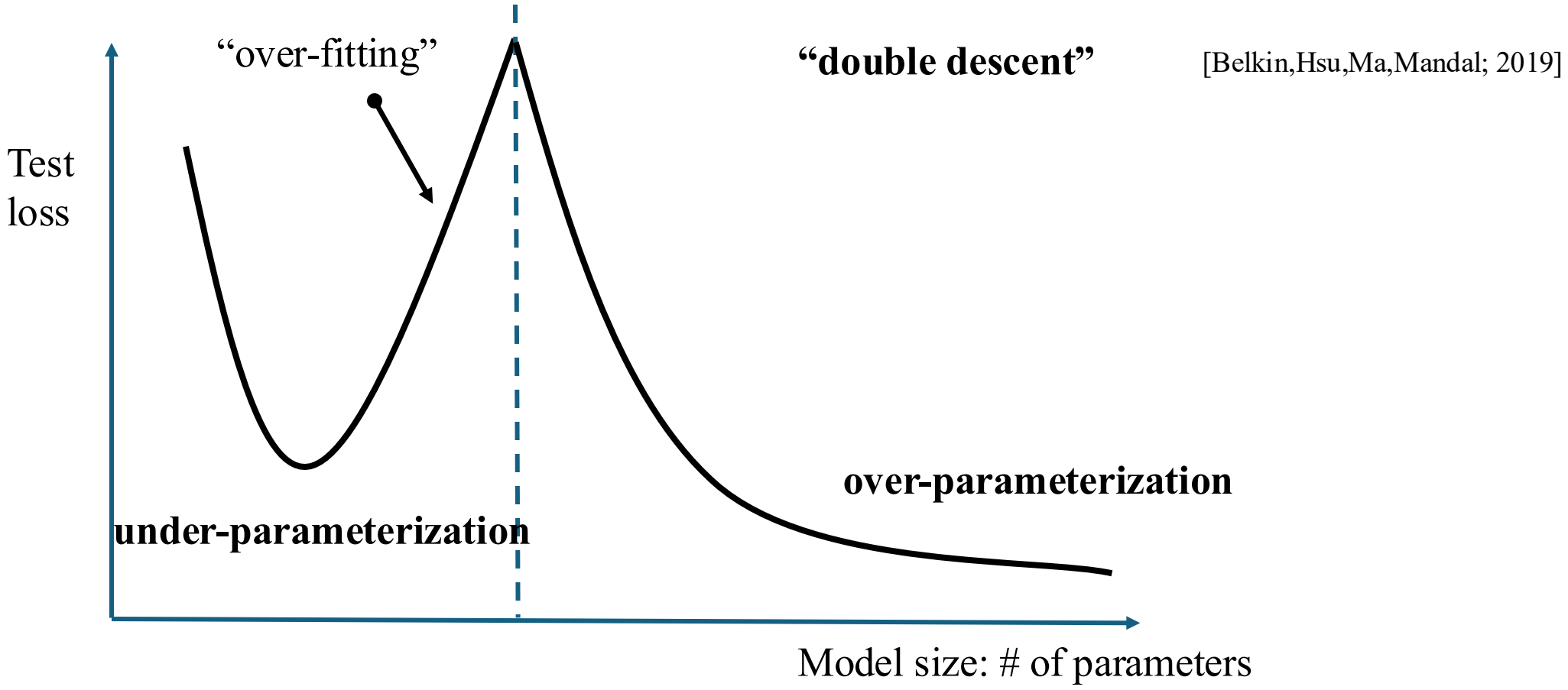
Fitting data with polynomial features:

- Number of training samples n = 16
- Number of model parameters p = d





Double descent



[Belkin,Hsu,Ma,Mandal; 2019]: Reconciling modern machine learning practice and the classical bias–variance trade-off. PNAS 2019.

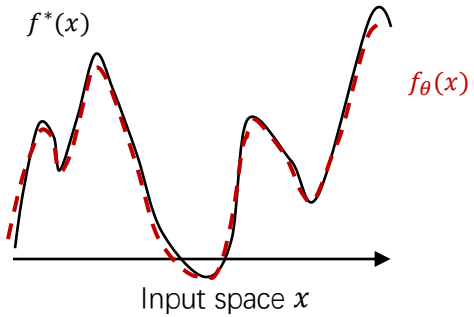
Transition to linearity

A neural network f is a function $f(\theta, x)$

network f as a **function of input $f_\theta(x)$**

Universal approximation [Hornik et al. 1989]:

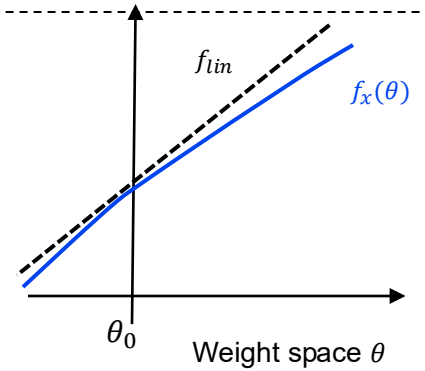
- Neural networks can approximate any *continuous* functions on a finite domain
- Larger network width $\Longrightarrow$ better approximation
- Infinite network width $\Longrightarrow$ exactly match the target function.



Fixing input x, network f as a **function of weights $f_x(\theta)$**

Transition to linearity [**Liu**, Zhu, Belkin NeurIPS 20]:

- Neural networks $f_x(\theta)$ is close to a linear function: non-linear terms are small, on a finite domain
- Larger network width $\Longrightarrow$ smaller non-linear term
- Infinite network width $\Longrightarrow$ non-linear term vanishes

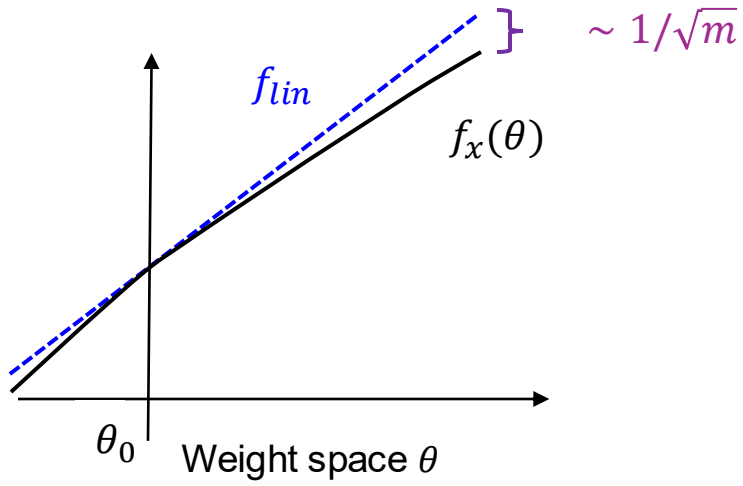


[**Liu**, Zhu, Belkin NeurIPS 20]: On the linearity of large non-linear models: when and why the tangent kernel is constant. NeurIPS 2020.

Transition to linearity

$$f(\theta) = f(\theta_0) + \nabla f(\theta_0)(\theta - \theta_0) + \underbrace{\frac{1}{2}(\theta - \theta_0)^T H(\theta_0)(\theta - \theta_0) + \cdots}$$

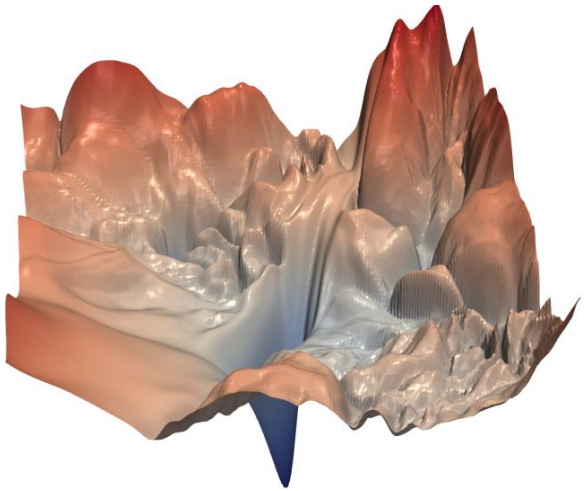
Vanishes as network width $m \to \infty$



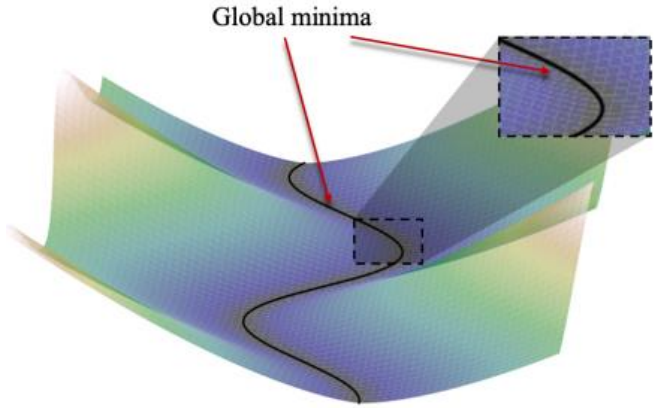
Transition to linearity

Why is *transition to linearity* useful?

- Simplifies the neural network function (w.r.t. parameters θ)
- Simplifies loss landscape*
- Theoretical guarantees for convergence of gradient descent algorithms* (including SGD)**



Under-parameterized



Over-parameterized

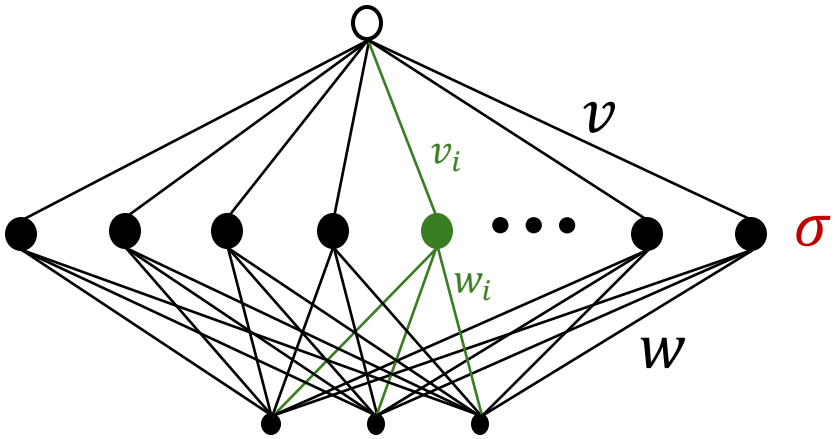
*: **Liu**, Zhu, Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. ACHA 2022.
: **Liu, Drusvyatskiy, Belkin, Davis, Ma. Aiming towards the minimizers: fast convergence of SGD for overparametrized problems. NeurIPS 2023.

Illustration: two-layer network

Two-layer neural network (f):

$$f(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma\left(w_i x\right)$$

Initialization: $v_{0,i} \sim \mathcal{N}(0,1)$; $w_{0,i} \sim \mathcal{N}(0,1)$.



Notations

- weights $\theta = \{\theta_i\}_{i=1}^m$, $\theta_i = (v_i, w_i)$,
- Non-linear activation σ: e.g., *sigmoid*, *tanh*, ReLU

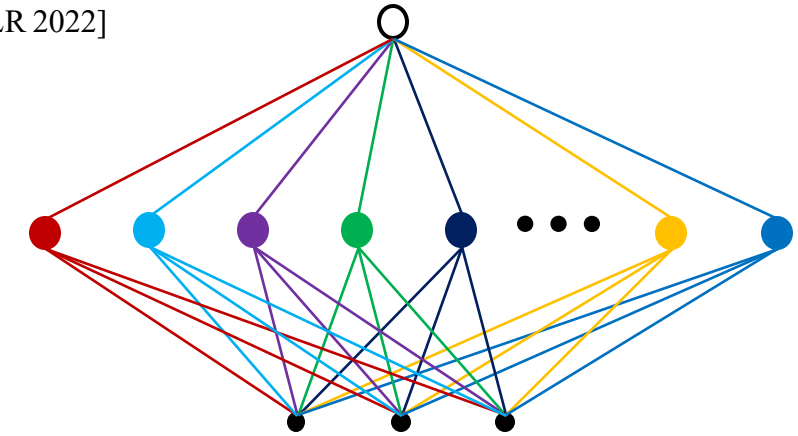
Kaiming Initialization (default in Pytorch):

- $v_{0,i} \sim \mathcal{N}(0, 1/m)$;

Assembly model view [**Liu**, Zhu, Belkin ICLR 2022]

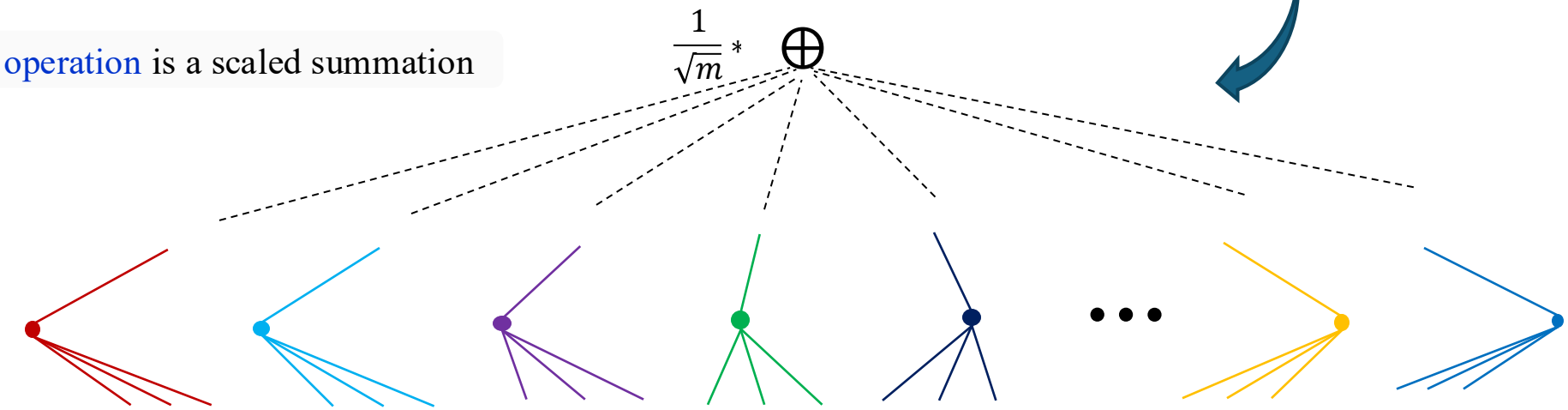
Network

$$f(\theta) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma\,(w_i x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} g_i$$



Assembly model

Observation 1: Assembly operation is a scaled summation



Sub-models $g_i = v_i \sigma(w_i x)$

Observation 2: sub-model independence -- sub-models share no weights

[**Liu**, Zhu, Belkin ICLR 2022]: Transition to Linearity of Wide Neural Networks is an Emerging Property of Assembling Weak Models, ICLR 2022.

Transferring structure to mathematical properties

Network $f(\theta) = \frac{1}{\sqrt{m}}\sum_{i=1}^{m} v_i \sigma(w_i x) \triangleq \frac{1}{\sqrt{m}}\sum_{i=1}^{m} g_i(\theta)$

Taylor expansion: (at initialization θ_0)

Lagrange remainder

$$f(\theta) = \underbrace{f(\theta_0) + \nabla f(\theta_0)(\theta - \theta_0)}_{\text{linear part: } f_{lin}(\theta)} + \underbrace{\frac{1}{2}(\theta - \theta_0)^T H(\xi)(\theta - \theta_0)}_{\text{non-linear part:}\mathcal{R}(\theta)}$$

$*$: ξ is between θ and θ_0

Hessian $H = \frac{\partial^2 f}{\partial \theta^2}$

Transferring structure to mathematical properties

Lagrange remainder (the non-linear part):

$$\mathcal{R}(\theta) = \frac{1}{2}(\theta - \theta_0)^T H_f(\xi)(\theta - \theta_0) = \frac{1}{2\sqrt{m}} \sum_{i=1}^{m} (\theta - \theta_0)^T H_{g_i}(\xi)(\theta - \theta_0)$$

Obs 1: assembly

$$H_f(\xi) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} H_{g_i}(\xi)$$

Transferring structure to mathematical properties

Lagrange remainder (the non-linear part):

$$\mathcal{R}(\theta) = \frac{1}{2}(\theta - \theta_0)^T H_f(\xi)(\theta - \theta_0) = \frac{1}{2\sqrt{m}} \sum_{i=1}^{m} (\theta - \theta_0)^T H_{g_i}(\xi)(\theta - \theta_0) = \frac{1}{2\sqrt{m}} \sum_{i=1}^{m} (\theta_i - \theta_{0,i})^T H_{g_i}(\xi)(\theta_i - \theta_{0,i})$$

Obs 1: assembly

Obs 2: Independence

$$(\theta_j - \theta_{0,j})^T H_{g_i}(\xi)(\theta_j - \theta_{0,j}) = 0$$

$$\forall j \neq i$$

Transferring structure to mathematical properties

Lagrange remainder (the non-linear part):

$$\mathcal{R}(\theta) = \frac{1}{2\sqrt{m}} \sum_{i=1}^{m} (\theta_i - \theta_{0,i})^T H_{g_i}(\xi)(\theta_i - \theta_{0,i})$$

each sub-model g_i is smooth:
$||H_{g_i}(\xi)||_{sp} \leq \beta$
β is a constant

$$|\mathcal{R}(\theta)| \leq \frac{1}{2\sqrt{m}} \sum_{i=1}^{m} ||H_{g_i}(\xi)||_{sp} \cdot ||\theta_i - \theta_{0,i}||^2$$
$$\leq \frac{\beta}{2\sqrt{m}} ||\theta - \theta_0||^2$$
$$\sim O\left(\frac{1}{\sqrt{m}}\right), \quad \text{for finite } ||\theta - \theta_0||^2$$

in a finite domain. e.g., a ball
$B(\theta_0, R)$ of finite radius R

Transition to Linearity

Lagrange remainder (the non-linear part):

$$|\mathcal{R}(\theta)| \sim O(\frac{1}{\sqrt{m}}), \quad \text{for finite } ||\theta - \theta_0||^2$$

When m is large, $|\mathcal{R}(\theta)|$ is small;
When $m \to \infty$, $|\mathcal{R}(\theta)| \to 0$.

Transition to linearity:

$$f(\theta) = \underbrace{f(\theta_0) + \nabla f(\theta_0)(\theta - \theta_0)}_{\text{Linear part: } f_{lin}(\theta)} + \underbrace{\frac{1}{2}(\theta - \theta_0)^T H(\xi)(\theta - \theta_0)}_{\text{Non-linear part: } \mathcal{R}(\theta)}$$

equivalently $\|H(\theta)\|_{sp} \sim O\left(\frac{1}{\sqrt{m}}\right), \forall \theta \in B(\theta_0, R)$, for $m \to \infty$.

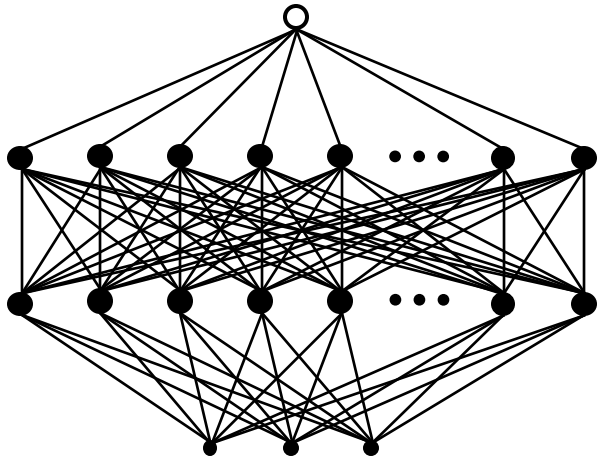
How about deep networks?

$$\alpha^{(0)} = x$$

$$\tilde{\alpha}^{(l)} = \frac{1}{\sqrt{m_{l-1}}} \sum_{i=1}^{m_{l-1}} w_i^{(l)} \alpha_i^{(l-1)}, \forall l = 1,2,\cdots,L$$

$$\alpha^{(l)} = \sigma\left(\tilde{\alpha}^{(l)}\right), \forall l = 1,2,\cdots,L-1$$

$$f = \tilde{\alpha}^{(L)}$$



Transition to Linearity holds, because:

- Random initialization: each weight $w_{ij}^{(l)} \sim N(0,1)$, i,i,d.
- Each layer has the scaled summation assembly form: $\frac{1}{\sqrt{m}}\sum$ (i.e., **Obs 1**)
- Independence of sub-models hold, after an appropriate rotation. (i.e., **Obs 2**)

Take home messages

In theory, neural networks are much simpler than we once believed.

Traditional view		**Modern view**
• Arbitrarily complex functions	→	• Much simpler functions of **weights $\boldsymbol{\theta}$** • Transition to linearity
• Overfitting when over-parameterized	→	• Double descent
• Loss function highly non-convex	→	• Satisfy good math properties