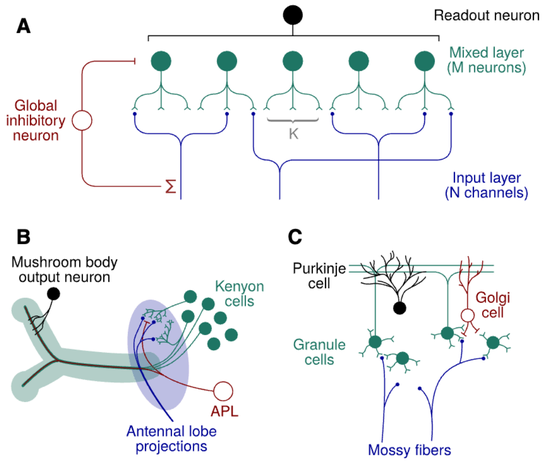
Bayesian theory of neural networks

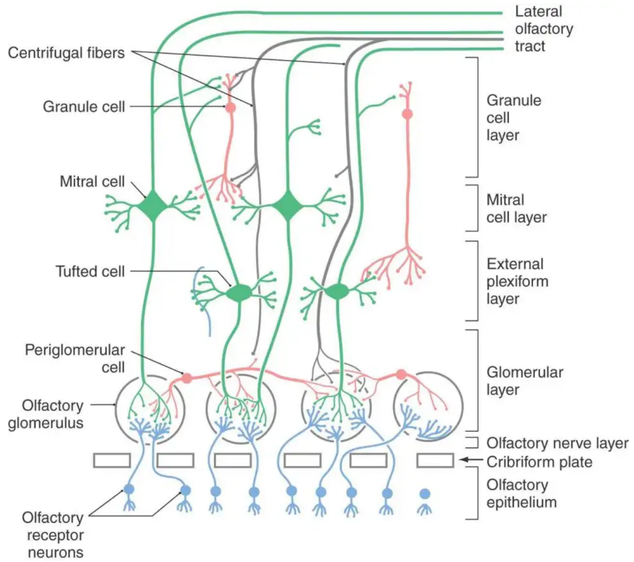
(Statistical mechanics) (deep, feedforward)

Haozhe Shan 2/12/2025

@ Advanced Topics in Theoretical Neuroscience

Why do we care about deep feedforward neural networks?





We know this is how the brain is wired in some places

- Cerebellum-like structures
- Early olfaction

They can be good models of more complex circuits

- Deep convolutional NNs can model mammalian visual areas at a neuron level
- Potentially good system-level models for capabilities such as language and abstraction

They are the backbone of advances in AI

- Even in LLMs, MLP layers contain the most* parameters and demand the most* compute
- They point to phenomena (e.g., neural collapse, in-context learning, lazy vs. rich learning) that can be universal in neural computation

What do we want from a theory of neural networks?

Feedforward network, supervised learning

NN: $f(\Theta, x)$ loss function $L(f(\Theta, x), D)$

Iterative optimization: initialize $\Theta(0)$, use D_{train} to update $\Theta(t) \rightarrow \Theta(t+1)$

To answer a basic question: why does deep learning work?

- How come that we can get training loss to zero? (What about non-convexity and local minima?)
- How come that we can get good generalization performance? (What about the bias-variance tradeoff?)

What has the NN learned?

- How should we think about the “knowledge” or “strategy” gained through training in $\Theta(T)$?
- Important for interpretability, better architecture/training protocol

Outline

This lecture: “Bayesian” theories of deep neural networks

General conceptual idea (some equations)

Concrete example to discuss insights (many equations)

Summary of this line of work (a few equations)

Connections to other theoretical approaches, in light of rich vs lazy learning (no equations)

Difficulties in studying DNNs

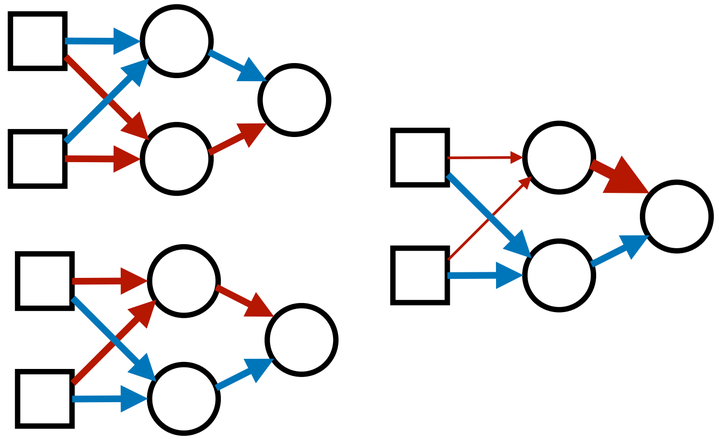
(Stochastic) difference equation: $\Theta(t+1) - \Theta(t) = \mathcal{F}(\Theta(t), D_{train}, \text{optimizer, loss fn})$

Problem 1: dynamics of $\Theta(t)$ are difficult to study

- Solvable only in very simple cases (e.g., linear regression where $f(\Theta, x) = \Theta^T x$ and loss is mean squared error)
- In very restricted cases: reducible to lower-D dynamical systems (Saad & Solla, 1995; Saxe et al., 2013) or "distributional dynamics" (Mei et al., 2018)

Problem 2: degeneracy in $\Theta(T)$

- For non-convex cases (anything with >1 layer), $\Theta(T)$ depends on Θ_0
- In overparameterized NNs, many $\Theta(T)$ "do the same thing" computationally (symmetry)

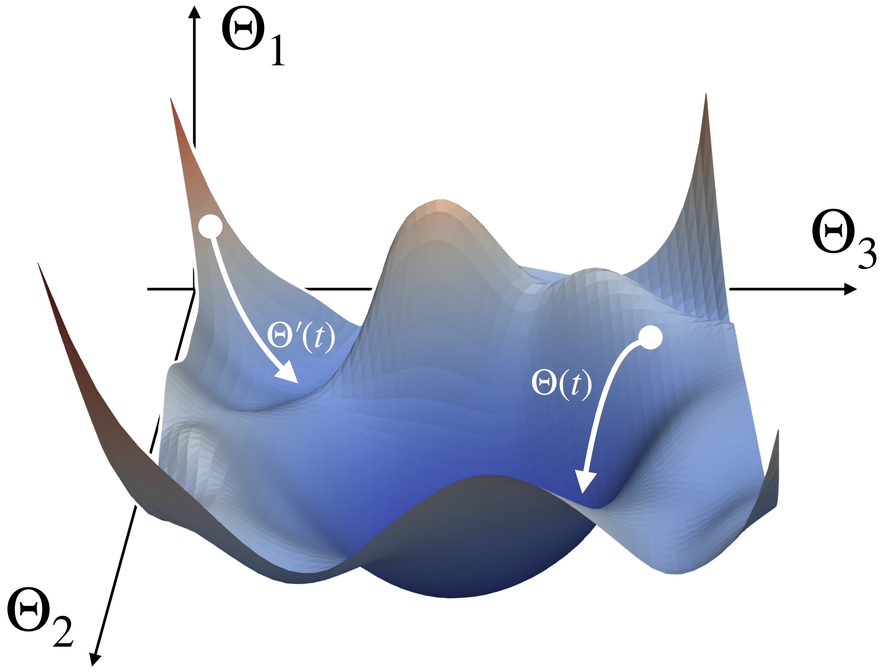


permutation and scale symmetries in ReLU networks

Treat it as a “thermodynamical” system

High-level idea of this approach:

- Sidestep dynamics and directly look at trained NNs
- Average over an ensemble of trained NNs to find something general



$$p(\Theta \mid D_{train}) = Z^{-1} \exp\left[-\beta E(f(\Theta, x), D_{train})\right]$$

FYI: Equilibrium distribution of Langevin dynamics

$$\Theta(t+1) - \Theta(t) = -\eta \nabla_\Theta E(f(\Theta, x), D_{train}) + \beta^{-1}\epsilon(t) \quad \epsilon(t) \sim \mathcal{N}(0, \mathbb{I})$$

$$\Theta(\infty) \sim p(\Theta \mid D_{train})$$

What do we assume about $E(f(\Theta, x), D_{train})$?

- Contains the loss function (low E = good performance)
- Is sufficiently constrained
- Is easy for subsequent calculations

$$E = \sum_\mu \left\| f(\Theta, x_\mu) - y_\mu \right\|^2 + \beta^{-1}\sigma^{-2}\|\Theta\|^2 \qquad D_{train} = \left\{ (x_\mu, y_\mu) \right\}_{\mu=1,\ldots,P}$$

likelihood prior

What do we do with this?

$$p(\Theta \mid D_{train}) = Z^{-1} \exp\left[-\beta E(f(\Theta, x), D_{train})\right]$$

This lecture: $f(\Theta, x)$ is a random function — compute $\langle f(\Theta, x)\rangle_\Theta$, $\langle f(\Theta, x)^2\rangle_\Theta$

- This will tell us the performance of the trained network on any data
- The performance will be a function of D_{train} — So we can compare the effectiveness of different training sets
- $f(\Theta, x)$ is what the network has learned — in the process we can understand inner workings of the NN as well

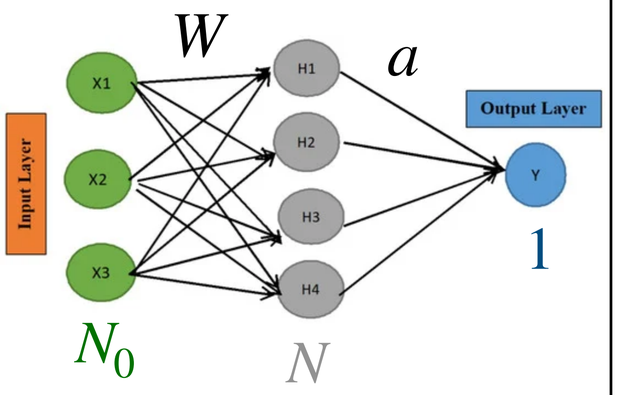
Approach: moment generating function

$$M(l) = \int d\Theta p(\Theta \mid D_{train}) \exp(l f\left(\Theta, x\right))$$

$$\langle f(\Theta, x)\rangle_\Theta = \left.\frac{dM(l)}{dl}\right|_{l=0} \qquad \langle f(\Theta, x)^2\rangle_\Theta = \left.\frac{d^2M(l)}{dl^2}\right|_{l=0}$$

Example: Two-layer MLP

network



$$\Theta = \left\{ a \in \mathbb{R}^N, W \in \mathbb{R}^{N \times N_0} \right\}$$

$$f(\Theta, x) = N^{-1/2} a^T \Phi(N_0^{-1/2} W x)$$

training data

$$D_{train} = \left\{ (x_\mu, y_\mu)_{\mu=1,\ldots,P} \right\}$$

$$x_\mu \in \mathbb{R}^{N_0}, y_\mu \in \mathbb{R}$$

Hidden neuron activations

$$X \in \mathbb{R}^{P \times N} : X_{\mu j} = \Phi(N_0^{-1/2} W x_\mu)_j$$

$$X_* \in \mathbb{R}^{1 \times N} : X_{* j} = \Phi(N_0^{-1/2} W x)_j$$

Plug in various definitions

$$M(l) \propto \int da \int dW \exp\left[-\frac{1}{2}\beta \parallel N^{-1/2} X a - Y \parallel^2 - \frac{\sigma^{-2}}{2} \parallel a \parallel^2 - \frac{\sigma^{-2}}{2} \parallel W \parallel^2 + l N^{-1/2} X_* a \right]$$

The exponent is quadratic in a

Meaning: $p(a \mid W, D_{train})$ is a multivariate Gaussian distribution

Apply HS transform $\exp\left(-\frac{c}{2}\|U\|^2\right) = (2\pi c)^{-1/2} \int \exp\left[-(2c)^{-1}\|V\|^2 - iU^T V\right] dV$

$$\int da \int dW \int dV \exp\Big[-i \left(N^{-1/2} X a - Y\right)^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2 - \frac{\sigma^{-2}}{2} \parallel a \parallel^2 - \frac{\sigma^{-2}}{2} \parallel W \parallel^2 + l N^{-1/2} X_* a \Big]$$

Gaussian integral over a

$$\int dW \int dV \exp\left[\frac{1}{2} N^{-1} \sigma^2 \left(l^2 X_* X_*^T - V^T X X^T V - 2 i l V^T X X_*^T \right) + i Y^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2 - \frac{\sigma^{-2}}{2} \parallel W \parallel^2 \right]$$

A mean-field description

Hidden neuron activations

$$X \in \mathbb{R}^{P\times N} : X_{\mu j} = \Phi(N_0^{-1/2} W x_\mu)_j$$

$$X_* \in \mathbb{R}^{1\times N} : X_{*j} = \Phi(N_0^{-1/2} W x)_j$$

All the W-dependent terms are summed over the hidden neuron index $j = 1,...,N$

$$\int dW \int dV \exp\left[\frac{1}{2}N^{-1}\sigma^2\left(l^2 \underbrace{X_* X_*^T}_{\sum_j X_{*j}^2} - V^T \underbrace{XX^T}_{\sum_j X_{:,j}X_{:,j}^T} V - 2il V^T \underbrace{XX_*^T}_{\sum_j X_{:,j}X_{*j}}\right) + iY^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2 - \frac{\sigma^{-2}}{2} \underbrace{\parallel W \parallel^2}_{\sum_j \|W_j\|^2}\right]$$

$$\int dW \int dV \exp\left[\sum_j \left[\frac{1}{2}N^{-1}\sigma^2\left(l^2 X_{*j}^2 - V^T X_{:,j} X_{:,j}^T V - 2il V^T X_{:,j} X_{*j}\right) - \frac{\sigma^{-2}}{2} \parallel W_j \parallel^2\right] + iY^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2\right]$$

X_{*j} and $X_{:,j}$ are responses of the jth hidden neuron — they only depend on W_j not $W_{j'\neq j}$

$$\int dV \exp\left[iY^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2\right] \prod_j \left\{ \int dW_j \exp\left[\frac{1}{2}N^{-1}\sigma^2\left(l^2 X_{*j}^2 - V^T X_{:,j} X_{:,j}^T V - 2il V^T X_{:,j} X_{*j}\right) - \frac{\sigma^{-2}}{2} \parallel W_j \parallel^2\right]\right\}$$

Integrals over $W_{j=1,...,N}$ are all the same — reducing equations about N different neurons to a single-neuron equation (mean field theory)

Integrating over W_j is intractable in general — X_* and $X_{:,j}$ are nonlinear functions of W_j. What do we do?

Gaussian Process Limit: $N \to \infty, P, N_0 \sim O(1)$

Hidden neuron activations

$$X \in \mathbb{R}^{P \times N} : X_{\mu j} = \Phi(N_0^{-1/2} W x_\mu)_j$$

$$X_* \in \mathbb{R}^{1 \times N} : X_{* j} = \Phi(N_0^{-1/2} W x)_j$$

$$\int dV \exp\left[iY^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2\right] \prod_j \left\{ \int dW_j \exp\left[\underbrace{\frac{1}{2} N^{-1}\sigma^2 \left(l^2 X_{*j}^2 - V^T X_{:,j} X_{:,j}^T V - 2ilV^T X_{:,j} X_{*j}\right)}_{O(P/N),\ \text{call it } z(W_j)} - \underbrace{\frac{\sigma^{-2}}{2} \parallel W_j \parallel^2}_{O(1)} \right] \right\}$$

---

when $z(W_j)$ is small, we can Taylor expand around $\exp(z(W_j)) \approx z(W_j) + 1$

$$\approx \int dW_j z(W_j) \exp\left[-\frac{\sigma^{-2}}{2} \parallel W_j \parallel^2\right] = \langle z(W_j) \rangle_{W_j \sim \mathcal{N}(0, \sigma^2 \mathbb{I})}$$

Meaning: in this limit $p(W \mid D_{train}) \approx p(W)$

$$\int dV \exp\left[iY^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2\right] \frac{1}{2} N^{-1}\sigma^2 \left(l^2 \langle \|X_*\|^2 \rangle_W - V^T \langle XX^T \rangle_W V - 2ilV^T \langle XX_*^T \rangle_W\right)$$

Integrate over V, which is a Gaussian variable, and some more algebra later…

$$\langle f(\Theta, x) \rangle_{\Theta \sim p(\Theta \mid D_{train})} = \langle N^{-1} X_* X^T \rangle_{W \sim p(W)} \left(\langle N^{-1} XX^T \rangle_{W \sim p(W)} + \beta^{-1} \mathbb{I} \right)^{-1} Y$$

What has the network learned?

Hidden neuron activations

$$X \in \mathbb{R}^{P\times N} : X_{\mu j} = \Phi(N_0^{-1/2} W x_\mu)_j$$

$$X_* \in \mathbb{R}^{1\times N} : X_{* j} = \Phi(N_0^{-1/2} W x)_j$$

$$\langle f(\Theta, x)\rangle_{\Theta\sim p(\Theta|D_{train})} = \underbrace{\langle N^{-1} X_* X^T\rangle_{W\sim p(W)}}_{1\times P} \Big(\underbrace{\langle N^{-1} X X^T\rangle_{W\sim p(W)}}_{P\times P} + \beta^{-1}\mathbb{I} \Big)^{-1} Y$$

representation similarity matrices, averaged over random weights

---

Define a kernel function $K(x,x') = \left\langle \sum_{j=1}^{N} N^{-1}\Phi(N_0^{-1/2} W_j\cdot x)\cdot\Phi(N_0^{-1/2} W_j\cdot x') \right\rangle_{W_j\sim\mathcal{N}(0,\sigma^2\mathbb{I})}$

empirical mean of N i.i.d. random variables — at large N it will just be its true mean ("self-averaging")

$$K(x,x') = \left\langle \Phi(N_0^{-1/2} W_j\cdot x)\cdot\Phi(N_0^{-1/2} W_j\cdot x') \right\rangle_{W_j\sim\mathcal{N}(0,\sigma^2\mathbb{I})}$$

---

denote the training data as $X_0 \in \mathbb{R}^{P\times X_0}$

$$\langle f(\Theta, x)\rangle_{\Theta\sim p(\Theta|D_{train})} = N^{-1} K(x, X_0)\big(N^{-1} K(X_0, X_0) + \beta^{-1}\mathbb{I}\big)^{-1} Y$$

The networks learn to do kernel regression with $K(x,x')$ on D_{train}

Equivalently: linear regression on $\{\Phi(N_0^{-1/2} W x_\mu)\}_{\mu=1,\ldots,P}$ in the N-dim hidden-neuron space with random W

Zooming out: summary of this line of work

$$\langle f(\Theta, x)\rangle_{\Theta\sim p(\Theta|D_{train})} = N^{-1}K(x, X_0)\left(N^{-1}K(X_0, X_0) + \beta^{-1}\mathbb{I}\right)^{-1} Y$$

- Equivalent to kernel regression, which has known convergence/generalization properties
- Performance fully determined by the data X_0, Y and the kernel function K — which in turn depends on architectural choices (depth, nonlinearity, $p(W)$, convolution…)
- Captures the performance of some "real" MLPs (SGD-trained, finite width) reasonably well* (see table)

Main problem with the GP limit: Has no "feature learning" — $p(W|D_{train}) \approx p(W)$

- **Feature learning is important**
  - sometimes "Real" NNs, especially Convents etc., significantly outperform their kernel counterparts. So it's definitely important for performance
  - Some phenomena (neural collapse, disentanglement of manifolds, etc.) clearly require feature learning to explain
- **Sometimes tractable at the limit of** $P, N \to \infty, P/N \sim O(1)$ — calculations become **much** more involved (see Li & Sompolinsky, 2021; Cui et al., 2023)

Num training	Model (ReLU)	Test accuracy
MNIST:100	NN-2-5000-0.10-0.11	**0.7786**
	GP-100-1.79-0.83	0.7735
MNIST:200	NN-2-2000-0.52-0.00	**0.8298**
	GP-100-1.79-0.83	0.8282
MNIST:500	NN-2-5000-1.82-0.77	**0.9028**
	GP-100-1.79-0.83	0.8995
MNIST:1k	NN-2-5000-3.19-0.00	0.9252
	GP-20-1.45-0.28	**0.9279**
MNIST:2k	NN-2-5000-2.88-0.01	0.9468
	GP-10-1.11-0.55	**0.9485**
MNIST:5k	NN-3-500-2.92-0.22	0.9675
	GP-7-0.61-0.07	0.9692
MNIST:10k	NN-2-2000-0.42-0.16	0.9771
	GP-7-0.61-0.07	0.9765
MNIST:20k	NN-3-1000-2.45-0.98	0.9825
	GP-5-1.62-0.83	0.9830
MNIST:50k	NN-2-2000-0.60-0.44	0.9864
	GP-1-0.10-0.48	0.9875
CIFAR:100	NN-5-500-1.88-1.00	0.2586
	GP-3-4.49-0.97	0.2673
CIFAR:200	NN-3-200-0.17-0.00	0.2719
	GP-3-3.99-1.72	**0.3022**
CIFAR:500	NN-1-100-1.26-0.63	0.3132
	GP-20-1.79-0.21	**0.3395**
CIFAR:1k	NN-5-500-1.29-0.28	0.3225
	GP-7-1.28-0.00	0.3608
CIFAR:2k	NN-3-5000-5.59-0.57	0.3894
	GP-3-4.16-1.17	0.3953
CIFAR:5k	NN-5-2000-5.26-1.74	0.4241
	GP-3-4.66-1.03	**0.4454**
CIFAR:10k	NN-5-2000-1.60-1.07	0.4545
	GP-5-2.97-0.28	**0.4780**
CIFAR:20k	NN-3-5000-4.18-0.18	0.5041
	GP-3-5.00-0.83	0.5118
CIFAR:45k	NN-3-5000-0.53-0.01	0.5313
	GP-3-3.31-1.86	**0.5566**

from Lee et al., 2017

Connecting to neural tangent kernels

Quick summary of NTK

- At a certain limit, throughout GD learning, parameter changes are sufficiently small such that $f(\Theta_t, x) \approx f(\Theta_0, x) + (\Theta_t - \Theta_0) \cdot \nabla_\Theta f(\Theta, x)\,|_{\Theta=\Theta_0}$
- At convergence, $f(\Theta, x)$ does kernel regression with the NTK kernel:

$$K_{NTK}(x, x') = \langle \nabla_\Theta f(\Theta, x) \cdot \nabla_\Theta f(\Theta, x') \rangle_{\Theta \sim p(\Theta_0)}$$

$$K_{NTK}(x, x') = \langle \nabla_a f(\Theta, x) \cdot \nabla_a f(\Theta, x') \rangle_\Theta + \langle \nabla_W f(\Theta, x) \cdot \nabla_W f(\Theta, x') \rangle_\Theta$$

$$\frac{d}{da_j} f(\Theta, x) = N^{-1/2} \Phi(N_0^{-1/2} W_j \cdot x)$$

This part of the NTK is exactly the NNGP kernel

Comparison of the physical meanings

NNGP (change=difference between prior/posterior)

- Changes to W are negligible (their effects on $f(\Theta, x)$ or $\nabla_a f(\Theta, x)$ are $\ll O(1)$)
- a changes significantly in a data-dependent way
- Equivalent to linear regression in N dim

NTK (change=difference between Θ_0 and Θ_t)

- Changes to W have a negligible effect on $\nabla_a f(\Theta, x)$ but $O(1)$ effect on $f(\Theta, x)$
- Changes to a have a negligible effect on $\nabla_W f(\Theta, x)$ but $O(1)$ effect on $f(\Theta, x)$
- Equivalent to linear regression in $N + N_0 N$ dim

“rich” and “lazy” learning in the Bayesian setting

“Infinitely wide networks are ‘lazy’ learners/do not go through feature learning.”

What is a lazy learner?

In the Bayesian MLP context, laziness can be defined as

$p(W \mid D_{train}) \approx p(W)$ or their difference is not big enough to change the kernel values

---

What makes the network lazy?

Retracing our steps…

$$\int dV \exp\left[iY^T V - \frac{1}{2}\beta^{-1} \parallel V \parallel^2\right] \prod_j \left\{ \int dW_j \exp\left[\underbrace{\frac{1}{2} N^{-1}\sigma^2 \left(l^2 X_{*j}^2 - V^T X_{:,j} X_{:,j}^T V - 2ilV^T X_{:,j} X_{*j}\right)}_{O(P/N)} - \frac{\sigma^{-2}}{2} \underbrace{\parallel W_j \parallel^2}_{O(1)} \right] \right\}$$

Taking the hidden-layer width (N) to infinity was necessary… but actually not sufficient!

E.g., (1) if the amount of data is extensive ($P/N \sim O(1)$), the first term is not small

(2) the first term is not necessarily $O(P/N)$ if we mess with the prior strength and normalization factors

$$f(\Theta, x) = N^{-1/2} a^T \Phi(N_0^{-1/2} W x)$$

If we set up the prior/normalization factors such that $f(\theta, x)$ cannot be $O(1)$ with $W \sim P(W)$, then fitting the data requires substantially changing W

Rich and lazy learning in general

For nonlinear NNs: common to define it by the magnitude of changes to $\Phi(N_0^{-1/2}Wx)$

What changes?

In the Bayesian setting

"change" = difference between $p(W)$ and $p(W|D)$

In the GD setting

"change" = difference between $W(0) \sim p_{init}(W)$ and $W(t)$

How much change is "rich"?

Is $\langle \Phi(N_0^{-1/2}w \cdot x) \cdot \Phi(N_0^{-1/2}w \cdot x') \rangle$ different for $w \sim p(W)$ and $w \sim p(W|D)$?

If they are different, it suggests that the trained NNs are using kernels ($w \sim p(W|D)$) different from the NNGP kernels ($w \sim p(W)$)

Are weights changing enough to break the Taylor expansion $f(\Theta_t, x) \approx f(\Theta_0, x) + (\Theta_t - \Theta_0) \cdot \nabla_\Theta f(\Theta, x)|_{\Theta=\Theta_0}$?

If so, the NTK is said to "evolve" over time; the trained NNs are using "final NTK" kernels $\nabla_\Theta f(\Theta, x)\Big|_{\Theta=\Theta(t)} \cdot \nabla_\Theta f(\Theta, x')\Big|_{\Theta=\Theta(t)}$

Some common themes and caveats

- Rich learning — not equivalent to using a kernel that is data-independent
- Lazy learning is not inevitable at infinite width (N) — rather it depends on how initialization/prior and normalization scale with N (e.g., Chizat et al., 2019; Yang & Hu, 2021)
- IMO: rich learning does not necessarily lead to a "better" solution — it just tells you how far you travel from initialization/prior

Bibliography

Wide Neural Networks as Gaussian Processes

- Neal, R. M. (1996). Priors for infinite networks. *Bayesian learning for neural networks*, 29-53.
- Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., & Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*.
- Matthews, A. G. D. G., Rowland, M., Hron, J., Turner, R. E., & Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.
- Hron, J., Bahri, Y., Sohl-Dickstein, J., & Novak, R. (2020, November). Infinite attention: NNGP and NTK for deep attention networks. In *International Conference on Machine Learning* (pp. 4376-4386). PMLR.

Theory at the $P/N \sim O(1)$ limit

- Li, Q., & Sompolinsky, H. (2021). Statistical mechanics of deep linear neural networks: The backpropagating kernel renormalization. *Physical Review X*, *11*(3), 031059.
- Cui, H., Krzakala, F., & Zdeborová, L. (2023, July). Bayes-optimal learning of deep random networks of extensive-width. In *International Conference on Machine Learning* (pp. 6468-6521). PMLR.
- van Meegen, A., & Sompolinsky, H. (2024). Coding schemes in neural networks learning classification tasks. *arXiv preprint arXiv:2406.16689*.
- Shan, H., Li, Q., & Sompolinsky, H. (2024). Order parameters and phase transitions of continual learning in deep neural networks. *arXiv preprint arXiv:2407.10315*.

Some general results on rich vs lazy learning for gradient descent

- Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. *Advances in neural information processing systems*, *32*.
- Yang, G., & Hu, E. J. (2021, July). Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning* (pp. 11727-11737). PMLR.

Rich and lazy learning in the brain?

- Flesch, T., Juechems, K., Dumbalska, T., Saxe, A., & Summerfield, C. (2022). Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, *110*(7), 1258-1270.
- Farrell, M., Recanatesi, S., & Shea-Brown, E. (2023). From lazy to rich to exclusive task representations in neural networks and neural codes. *Current opinion in neurobiology*, *83*, 102780.