STAT 946 – Deep Learning Theory

Majid Ghasemi

Session 1

1 Instructor's Contact

Dr. Mufan Li **(Office M3 4006)**

2 Grading Scheme

- Attendance: 25% (Scribing notes as part of it)
- Presentation: 25%
- Report: 50%

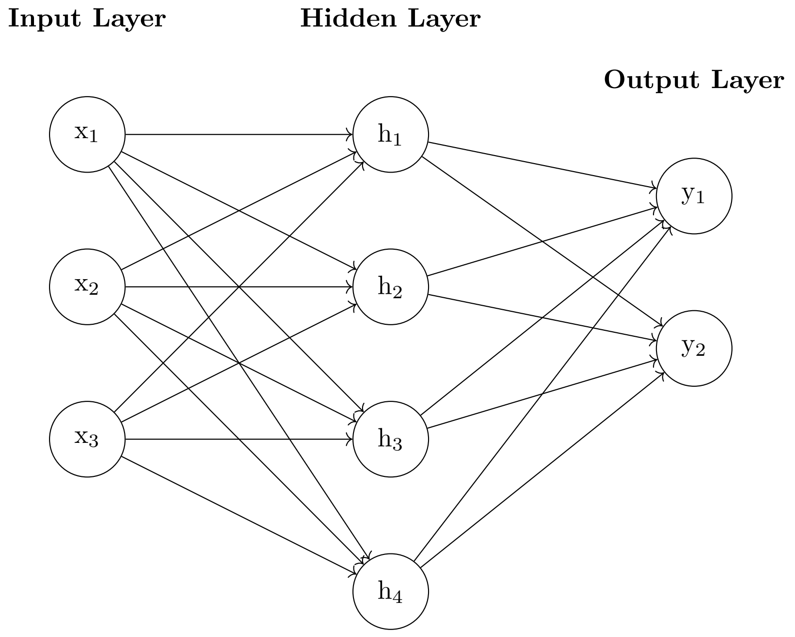
We need to talk with the prof about the report's idea around the week of Sept 15.

3 Background Topics

- Markov Chains
- Measure Theory
- Stochastic Calculus
- Random Matrix Theory
- NTK (Neural Tangent Kernel)
- Deep Learning Training (which will not be covered in this course)

4 Brief History of Neural Networks

Neural Networks (NNs) root back to Allan Turing (1948), Rosenblatt (1958, 1962). NNs resemble a brain based on the mentioned peoples beliefs but they are simply two weights in a way that $f(x;\theta) = W_1\phi(W_0x)$, where $\theta = \{W_0, W1\}$.



In the early development of NNs, there was no efficient way to update the weights across multiple layers. Gradient descent as an optimization method dates back to Cauchy in the 19th century, but it was not until the 1980s that the combination of stochastic gradient descent (SGD) and backpropagation became the standard approach for training deep networks (Rumelhart, Hinton, and Williams, 1986). Backpropagation is essentially an efficient application of the chain rule, allowing the gradients of all weights to be computed by reusing intermediate results instead of recalculating derivatives repeatedly. This efficiency is what made training multi-layer neural networks practical.

$$\theta_{k+1} = \theta_k - \eta\nabla\mathcal{L}(\theta_k), \qquad \eta > 0 \text{ (step size)} \tag{1}$$

$$f(x;\theta) = W^{(n)}h^{(d)} \tag{2}$$

$$h_{\ell+1} = W_\ell\,\phi(h_\ell) \qquad \text{(MLP)} \tag{3}$$

$$\frac{\partial f}{\partial W_{\ell,ij}} = \frac{\partial f}{\partial h_{\ell+1}}\frac{\partial h_{\ell+1}}{\partial h_\ell}\cdots\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_{\ell,ij}} \tag{4}$$

If we were to apply gradient descent to optimize the parameters θ of a neural network, the iterative update rule is given by:

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k), \tag{5}$$

where η is the learning rate and $\nabla_\theta \mathcal{L}(\theta_k)$ is the gradient of the loss function with respect to the parameters. The calculation of these gradients across layers is enabled by *automatic differentiation*, which makes training deep networks computationally feasible.

Neural networks initially fell out of favour in the 1990s, largely due to their tendency to overfit and theoretical challenges related to statistical learning theory (e.g., VC theory). However, two key advances around the early 2000s reignited interest:

- The practical use of **Auto differentiation**, enabling efficient computation of gradients in large networks.
- The adoption of **GPUs for neural network training** (around 2004), providing significant computational acceleration. A notable milestone was the work of Raina, Madhavan, and Ng (2009), who trained models with over 100 million parameters using GPUs, reporting speedups of up to $100\times$ compared to CPU-based training.

This set the stage for the modern deep learning era. The breakthrough of **AlexNet** (2012), which achieved state-of-the-art results on the ImageNet image classification benchmark, marked the second wave of neural network popularity.

Several major contributions further shaped the field:

- **Adam optimizer (2014)**: widely adopted as an alternative to SGD due to its adaptive learning rates and robustness.
- **Transformers (2017)**: introduced as a new architecture, they became the backbone of modern natural language processing and large-scale AI systems.
- **Scaling laws (2020)**: demonstrated that transformer performance improves predictably with increased model size, dataset size, and compute resources (see GPT-4 Technical Report).

The central technical takeaway of this era was the principle that *"bigger is consistently and predictably better."* However, more recent research has raised questions about the universality of this rule, suggesting possible limits or exceptions to the scaling paradigm.
Deep Learning (DL) research is in a perpetual cycle:

$$\begin{tikzcd}
\text{Propose Algo.} \arrow[d] \\
\text{Scale up.} \arrow[u, bend left=90]
\end{tikzcd}$$

There are limits to DL progress; a practical way to phrase the question is: *"Can we evaluate the limit given our compute budget?"* A simple training-compute proxy is proportional to model size, data processed per step, and training steps:

$$C \;\propto\; N_{\text{params}} \times B \times S, \tag{6}$$

where C is the compute budget, N_{params} is the number of parameters, B is the batch size (tokens/examples per step), and S is the number of optimization steps.

5 Overview of "Other DL Theory"

5.1 Universal Approximation

Let

$$f^* : [0,1]^d \to R, \qquad f^* \in \text{Smoothness class } C^h,\, C^\alpha,\, W^{k,p}.$$

Then, for every $\varepsilon > 0$, there exists a neural network with width n and depth d such that

$$\|f - f^*\| \;\leq\; \varepsilon.$$

(The approximation error depends on the smoothness properties of f^*.)

5.2 Statistical Learning Theory

- $\mathcal{X}, \mathcal{Y}$: Data domain
- $\mathcal{H}$: Hypothesis class, $h : \mathcal{X} \to \mathcal{Y}$
- $\mathcal{D} \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$: Data distribution
- ℓ : Loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \to R_{\geq 0}$

- $\mathcal{A}$: Learning algorithm, $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^m \to \mathcal{H}$

The standard learning guarantee can be expressed as follows:

$$\forall \varepsilon > 0,\ \delta \in (0,1), \quad \exists \mathcal{A} \text{ and } m^* > 0 \text{ s.t.}$$

$$\forall \mathcal{D}, \quad (x_i, y_i)_{i=1}^m \overset{\text{i.i.d.}}{\sim} \mathcal{D}, \quad m \geq m^*,$$

$$h_m = \mathcal{A}\big((x_i, y_i)_{i=1}^m\big),$$

$$\text{then with probability at least } 1 - \delta, \quad \ell(h_m(x), y) \ \leq \ \inf_{h \in \mathcal{H}} \ell(h(x), y) + \varepsilon.$$

5.3 Model Agnostic

Optimization

Assume the loss function $\mathcal{L}$ is convex and has a Lipschitz-continuous gradient. Then, using stochastic gradient descent (SGD), we can guarantee that

$$\mathcal{L} \ \leq \ \varepsilon \quad \text{after} \ldots \text{ steps of SGD.}$$

Edge of Stability

The training dynamics of SGD often operate near the *edge of stability*, where the step size is close to the largest value that still ensures convergence. This regime is empirically observed to play a role in the efficiency and generalization of modern deep networks.

6 Open Problems in Deep Learning Theory

Why do we study DL? Several important open theoretical problems remain unresolved:

6.1 Compare and Improve Methods

How can we say which method is better? This raises several sub-questions:

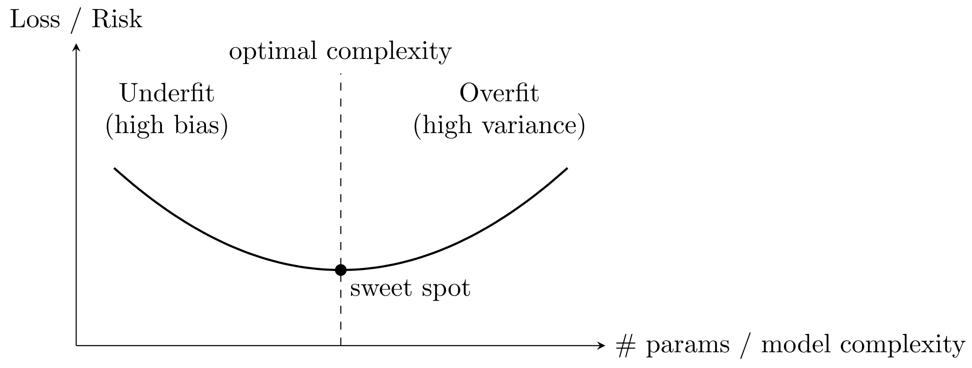
- **Why do scaling laws work?**
- **Compare benchmarks:** are results consistent across different datasets and tasks?
- **Hyperparameter tuning without full training:** can we predict optimal values (e.g., step size η in SGD, depth-to-width ratio) without running the entire algorithm?

6.2 Why Does Deep Learning Work?

Two central issues are:

- **Optimization for non-convex loss $\mathcal{L}$:** training deep nets is NP-hard in general, yet gradient-based methods perform well in practice.
- **Overfitting and generalization:** despite highly overparameterized models, deep nets often generalize well.

This is often illustrated by the **bias-variance trade-off**, where increasing parameters first decreases risk (bias reduction), but past a point leads to overfitting. Deep learning, however, often appears to break this classical curve.



6.3 Scaling Limits (Important)

Deep learning research often operates in a *scaling limit* regime:

- Number of random variables (or sources) $\to \infty$.
- Contribution of each variable $\to 0$.

This mirrors the Central Limit Theorem:

$$\frac{\sum_{i=1}^{n}(X_i - \mu)}{\sigma\sqrt{n}} \xrightarrow{d} \mathcal{N}(0,1), \qquad n \to \infty,$$

where a complex system converges to a *nice limit object.*

For neural networks:

$$\text{Finite NN} \longrightarrow \text{“nice limit”}$$

such as Gaussian processes or neural tangent kernels.

6.4 Summary

The central open problems can be grouped as:

- Understanding scaling laws and limits.
- Explaining why non-convex optimization works.
- Explaining why deep nets generalize despite over-parameterization.
- Developing principled methods for model and hyperparameter selection.