Lecture 1: The Learning Problem

Module 1: Foundations of Neural Networks Deep and Reinforcement Learning

Srikanth Pai

Madras School of Economics

Why Not Just Write Rules?

The task: classify handwritten digits.

Rules-based attempt:

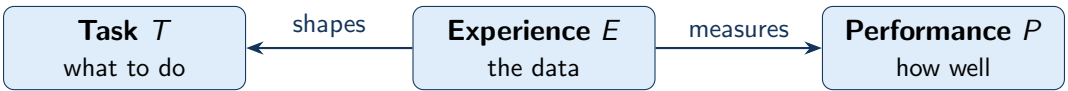
- A ‘2’ has a loop at top and a flat base.
- A ‘1’ is thin and mostly vertical.
- . . .

Problem: variation is too rich. Every rule has exceptions.

Same label, wildly different pixels.

Insight: instead of programming the computer, program a **learning algorithm** that finds the pattern from labelled examples.

Three Ingredients



Mitchell 1997

A program *learns* if its performance at T, measured by P, improves with experience E.

Classification
predict category $k \in \{1, \ldots, K\}$

Regression
predict real value $y \in \mathbb{R}$

Density estimation
model $P(x)$ directly

T, E, P in Action: Linear Regression

The simplest supervised learning problem

Task T:	Predict a real output y from features $x \in \mathbb{R}^d$ using a linear model $h(x;\, w, b) = w^\top x + b$.
Experience E:	A labelled dataset $\{(x_i, y_i)\}_{i=1}^n$. For example: house sizes and sale prices, or years of schooling and wages.
Performance P:	Mean squared error on a held-out test set.

Choosing the best model means choosing (w, b) to minimise training MSE.

This is **ERM over linear functions**: the familiar least-squares problem.

The insight: the same principle extends to any hypothesis class.
What changes in deep learning is $\mathcal{H}$, not the framework.

The Statistical Setup

Assumption

Data $(x_1, y_1), \ldots, (x_n, y_n)$ drawn **i.i.d.** from an unknown distribution P on $\mathcal{X} \times \mathcal{Y}$.

True risk (want to minimise, cannot compute):

$$R(h) = \mathbb{E}_{(x,y)\sim P}\big[L(h(x), y)\big]$$

Empirical risk (can compute):

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x_i), y_i)$$

Theorem: $\mathbb{E}_{\mathcal{D}}[\widehat{R}(h)] = R(h)$ for any *fixed* h.

Caveat: the $\hat{h}$ we select is *not* fixed; it is chosen to minimise $\widehat{R}$, so it can exploit fluctuations in the training set.

Empirical Risk Minimisation

ERM

Given a hypothesis class $\mathcal{H}$, choose

$$\hat{h} \;=\; \operatorname*{arg\,min}_{h \in \mathcal{H}} \; \widehat{R}(h).$$

- Linear classifier: $\mathcal{H} = \{x \mapsto w^\top x + b\}$.
- Neural network: $\mathcal{H} =$ compositions of linear maps and nonlinearities.
- The choice of $\mathcal{H}$ determines what patterns the model can express.

Key question: how do we choose $\mathcal{H}$?

Classwork: Empirical Risk for a Constant Predictor

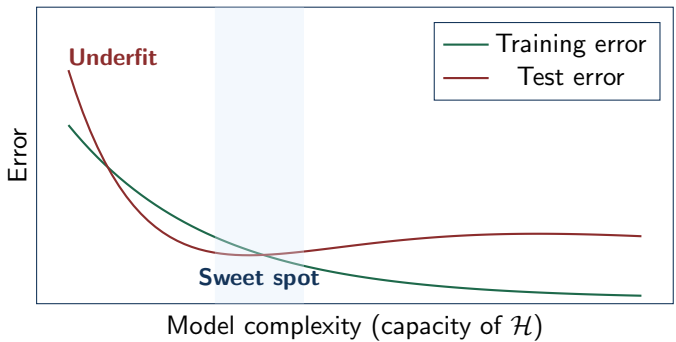
Try it: 5 minutes

Hypothesis class: $\mathcal{H} = \{h_b : b \in \mathbb{R}\}$, $h_b(x) = b$. Training data: $(1, 3)$, $(2, 5)$, $(3, 1)$. Loss: squared loss $L(\hat{y}, y) = (\hat{y} - y)^2$.

(a) Compute $\widehat{R}(h_b)$ as a function of b.
(b) Find the ERM solution $\hat{b}$.
(c) What is the minimum empirical risk?

[Hint: differentiate $\widehat{R}(h_b)$ with respect to b.]

Underfitting vs. Overfitting



Bias-Variance Decomposition

Suppose $y = f(x) + \varepsilon$, $\mathbb{E}[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$. Then:

$$\mathbb{E}\big[(\hat{h}(x) - y)^2\big] = \underbrace{(\bar{h}(x) - f(x))^2}_{\text{Bias}^2} + \underbrace{\mathrm{Var}(\hat{h}(x))}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Noise}}$$

where $\bar{h}(x) = \mathbb{E}[\hat{h}(x)]$ is the average prediction.

Low capacity
High bias, low variance.
Average prediction consistently wrong.

High capacity
Low bias, high variance.
Prediction varies wildly across training sets.

Noise σ^2
Irreducible.
No algorithm can beat this floor.

Key Takeaways: Statistical Framework

- Data is i.i.d. from an unknown distribution P.
- We minimise **empirical risk** $\widehat{R}(h)$ as a proxy for true risk $R(h)$.
- Empirical risk is unbiased, but the **selected** hypothesis exploits data fluctuations.
- **Capacity** controls the bias-variance tradeoff: too low $\Rightarrow$ underfit; too high $\Rightarrow$ overfit.
- There is an optimal capacity for a given dataset size.

The Feature Engineering Bottleneck

Classical ML:

1. Human expert designs features $\phi(x)$.
2. Algorithm learns $h(\phi(x))$.

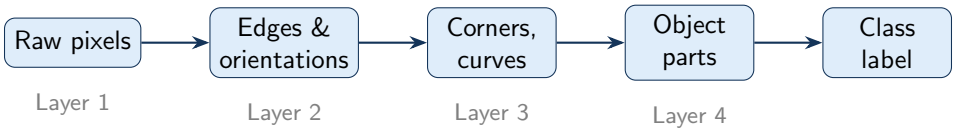
Brittle: features must be redesigned for every domain. Fails on unstructured data.

Deep Learning:

1. Each layer learns a representation.
2. Layers compose: raw input $\rightarrow$ useful features $\rightarrow$ decision.

Flexible: representations emerge from data. End-to-end training.

Hierarchy of Representations



"The key insight is that the features are not designed by humans but are learned from the data using a general-purpose learning procedure."

Goodfellow et al., Deep Learning, Ch. 1

Key Takeaways: Why Depth

- Feature engineering: powerful, brittle, expensive.
- Deep networks learn **hierarchical representations** end-to-end from raw data.
- Natural data lies near a low-dimensional **manifold** in a high-dimensional space. Depth lets a network navigate that manifold.
- **Parameter efficiency**: depth gives exponential representational gain over width alone.
- What changed around 2010: large data, GPUs, better algorithms.

Next lecture: feedforward networks, the basic unit. What can a single hidden layer represent?

Classwork: Bias or Variance?

Try it: 5 minutes

For each scenario, say whether the dominant problem is **high bias** (underfit) or **high variance** (overfit), and explain in one sentence.

(a) Training error: 2%. Test error: 18%.
(b) Training error: 35%. Test error: 37%.
(c) A degree-1 polynomial fitted to data from $f(x) = x^3 - x$.
(d) A degree-$(n-1)$ polynomial fitted to n training points, where the true f is linear.