COMS4995W32
Applied Machine Learning

Dr. Spencer W. Luo

Columbia University | Fall 2025

From Models to Systems:
The Evolution of Applied Machine Learning

Course Journey at a Glance 🚀

What we covered (Weeks 1 → 12):

- ML problems → Metrics-driven learning
- Data & features → Linear models
- Decision trees → Ensembles
- Deep learning → Representation learning
- Sequence models → Transformers
- LLMs → RLHF and Agentic systems

Path: From (un)supervised learning to LLM reasoning and agentic intelligence

What This Course Taught Us 🤔

How to formulate real-world problems as learning tasks

→ Define objectives, data requirements, and measurable success criteria

→ Translate business / scientific requirements into ML formulations

How to build end-to-end ML pipelines

→ Data cleaning, feature extraction, model training, evaluation, and deployment

→ Automate feedback loops and monitoring in production

What This Course Taught Us 🤔

How to design experiments and evaluate appropriately

→ Train/validation/test splits, cross-validation, ablation studies

→ Understand metrics: precision, recall, F1, AUCPR, perplexity

How to reason about trade-offs

→ Bias-variance, underfitting vs overfitting

→ Scaling vs cost (compute, labeling, iteration speed)

The Core ML Loop ♻️

The ML flywheel:

Data → Feature Engineering → Model → Loss → Optimization → Evaluation

- **The fundamental feedback loop underlying every ML system**
- Each stage influences and constrains the next - improving one improves all

The Core ML Loop ♻️

Practical Takeaways

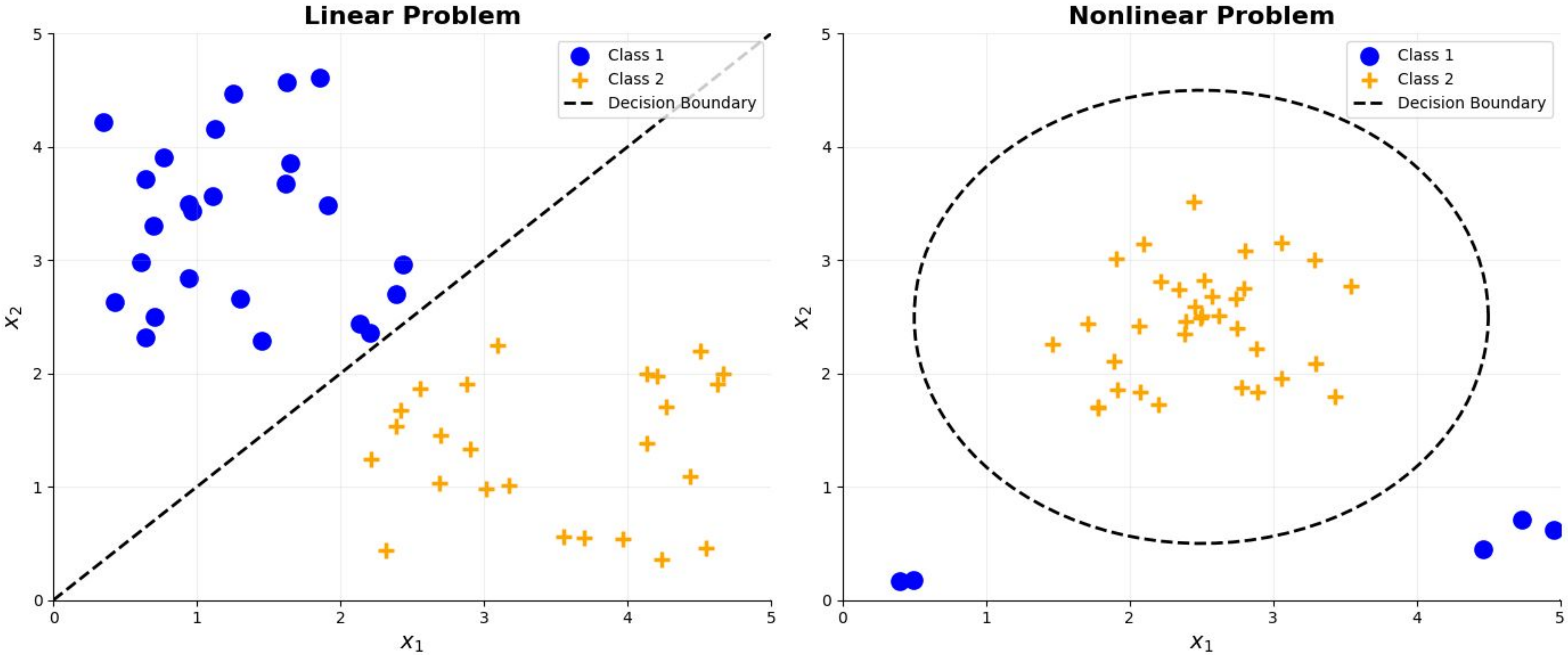
- Always visualize intermediate results (train vs validation loss etc.)
- Apply regularization, dropout, or early stopping to balance fit and generalization
- Periodically re-evaluate when data distributions shift (“data drift”)

From Linear to Non-Linear Models

Linear/Logistic regression: simple, interpretable

Decision trees: handle non-linearity, feature interactions

Non-linear models unlock representation power *beyond linear boundaries*



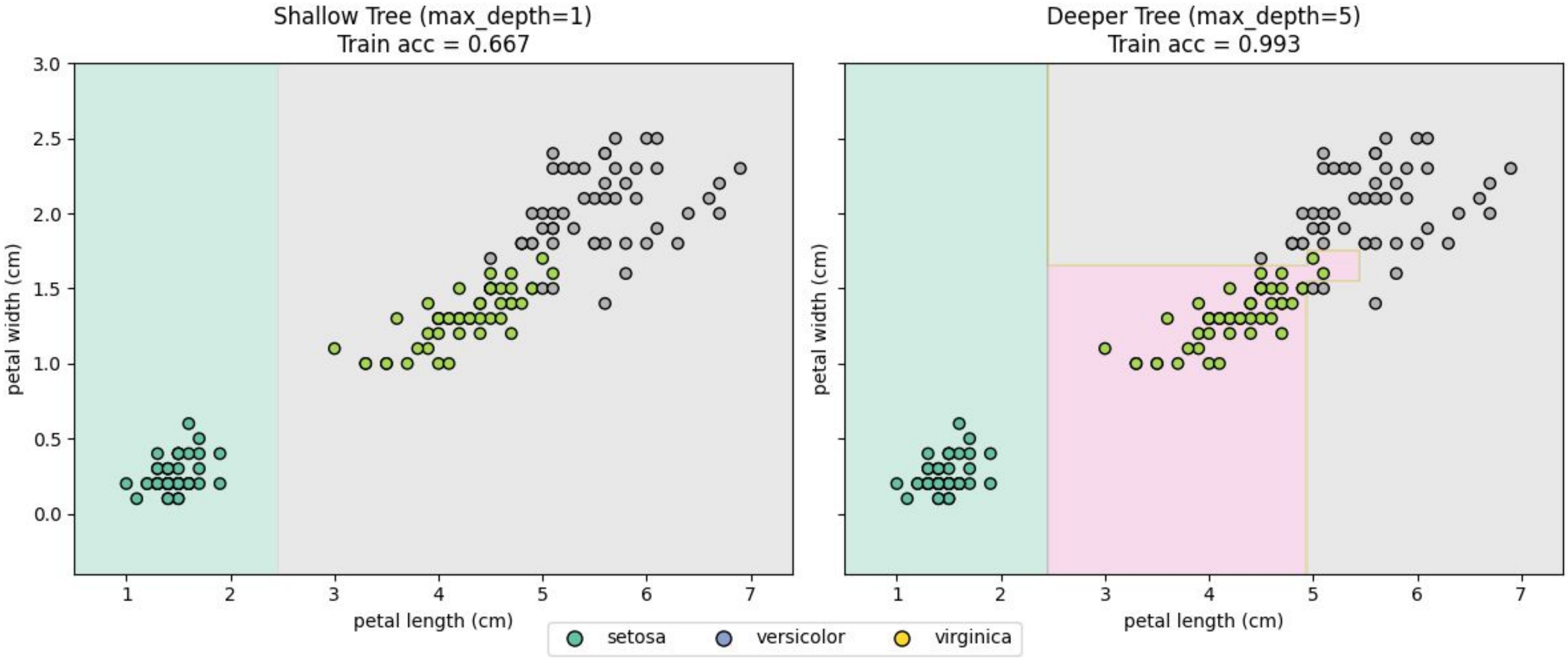
Linear Problem
Nonlinear Problem
Class 1
Class 2
Decision Boundary
x_1
x_2
0
1
2
3
4
5

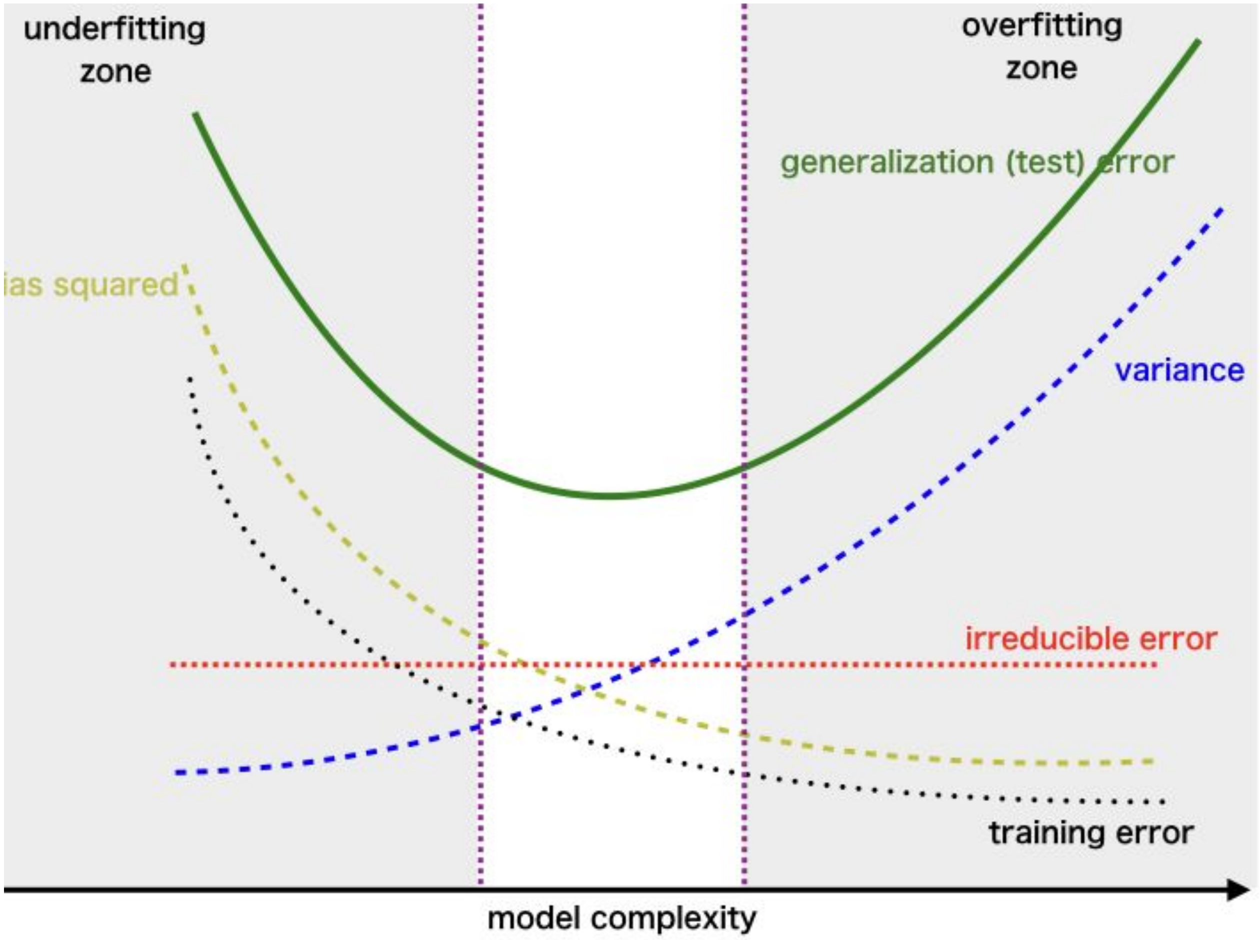
Ensembles: The Wisdom of Crowds

Bagging (Random Forest): reduce variance via many weak learners

Boosting (XGBoost): sequentially correct errors, improve performance

Mixture-of-experts: MoE to scale up Transformers


Shallow Tree (max_depth=1)
Train acc = 0.667
Deeper Tree (max_depth=5)
Train acc = 0.993
petal width (cm)
petal length (cm)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7
setosa
versicolor
virginica



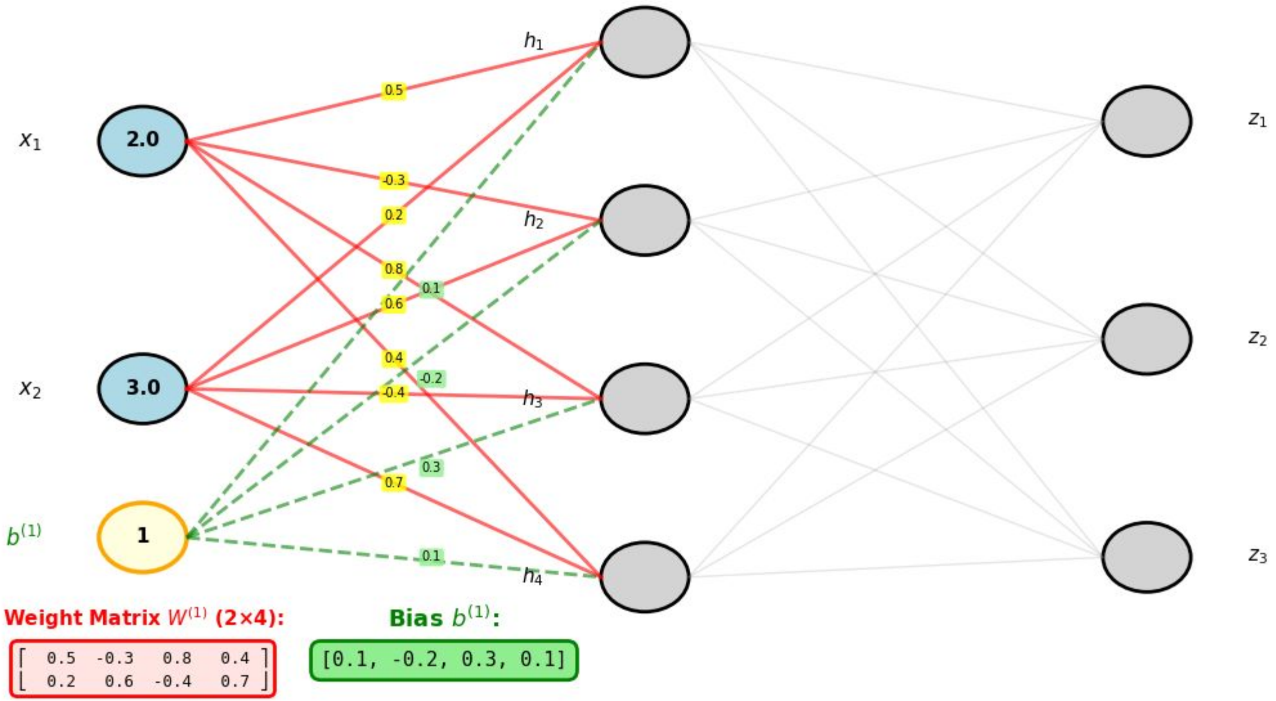
underfitting zone
overfitting zone
generalization (test) error
ias squared
variance
irreducible error
training error
model complexity

Deep Learning: Representation Learning

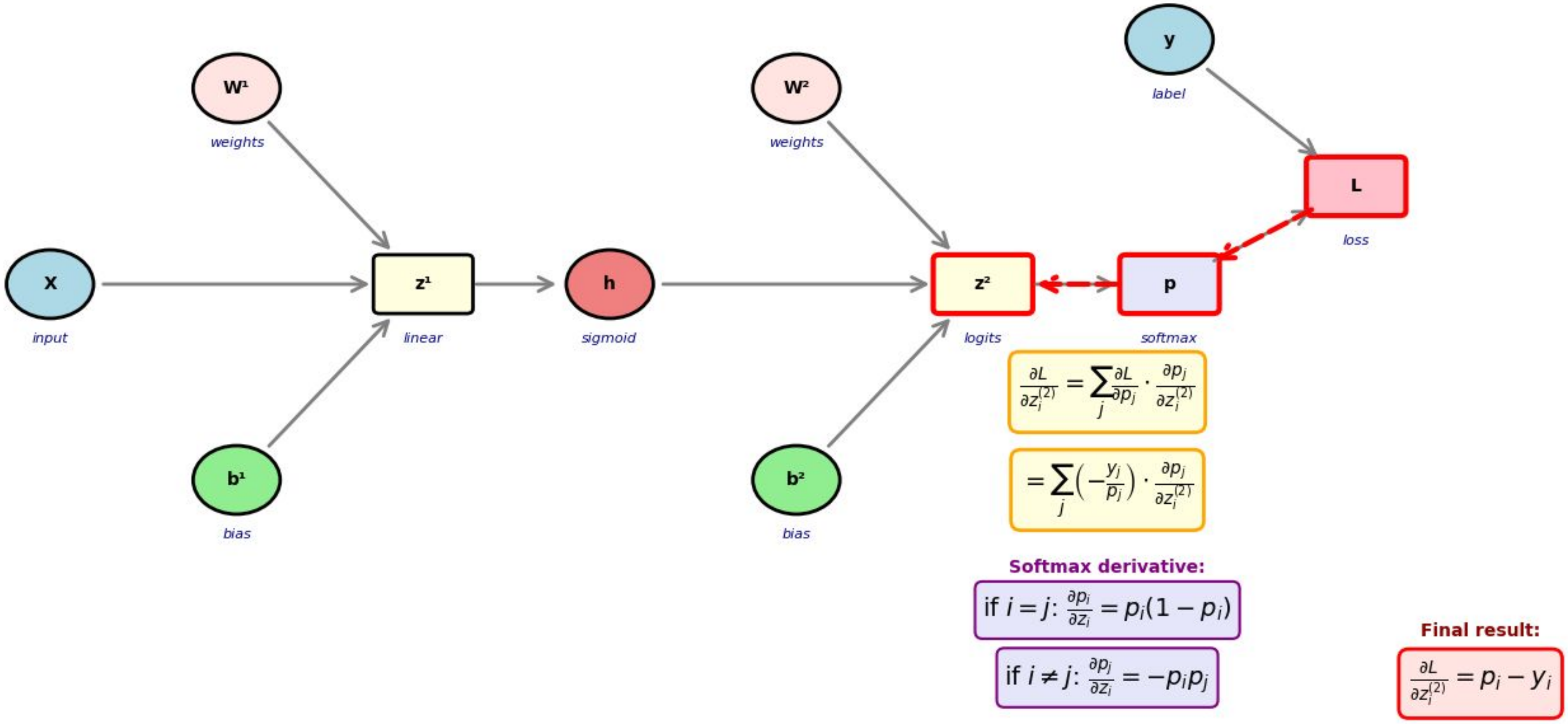
Forward pass

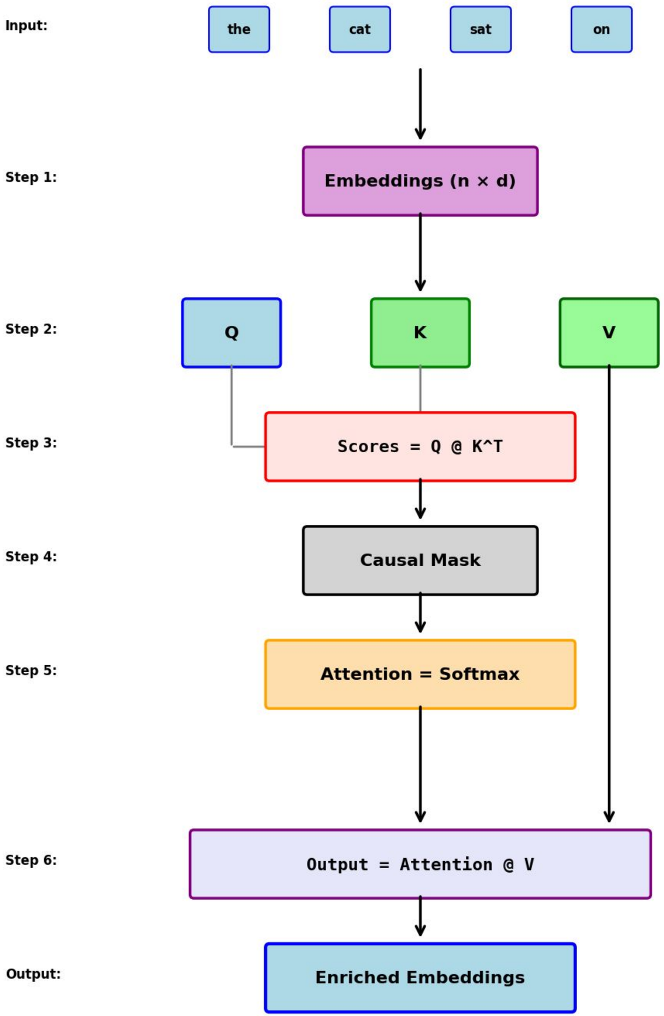
Loss computation

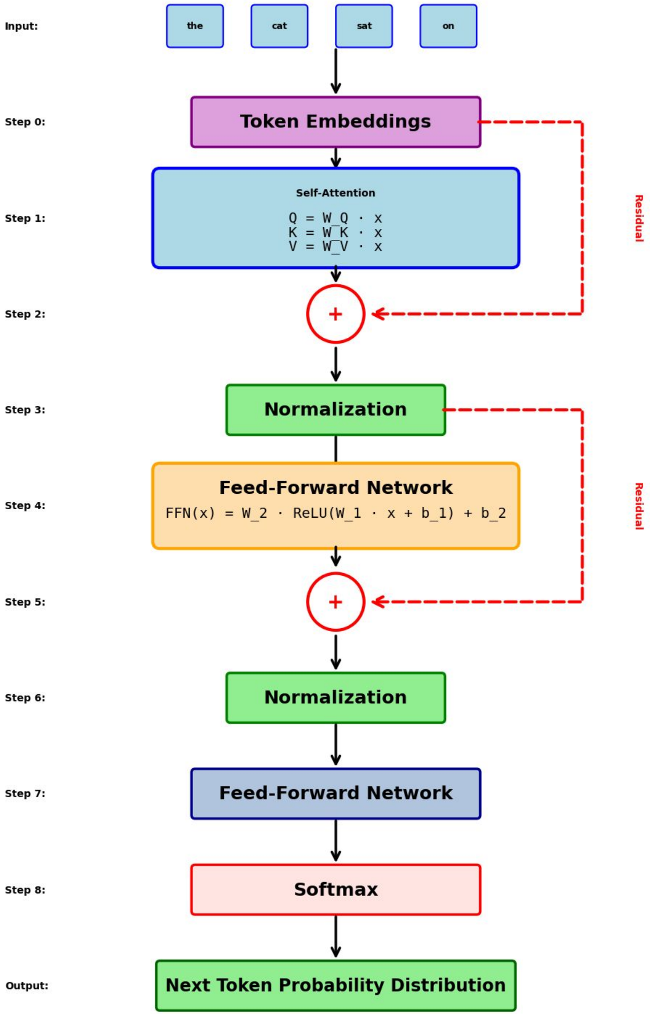
Backpropagation



x_1
2.0
x_2
3.0
$b^{(1)}$
1
h_1
h_2
h_3
h_4
z_1
z_2
z_3
0.5
-0.3
0.2
0.8
0.1
0.6
0.4
-0.2
-0.4
0.3
0.7
0.1
Weight Matrix $W^{(1)}$ (2×4):
$\begin{bmatrix} 0.5 & -0.3 & 0.8 & 0.4 \\ 0.2 & 0.6 & -0.4 & 0.7 \end{bmatrix}$
Bias $b^{(1)}$:
[0.1, -0.2, 0.3, 0.1]


W^1
weights
W^2
weights
y
label
L
loss
X
input
z^1
linear
h
sigmoid
z^2
logits
p
softmax
b^1
bias
b^2
bias
$\frac{\partial L}{\partial z_i^{(2)}} = \sum_j \frac{\partial L}{\partial p_j} \cdot \frac{\partial p_j}{\partial z_i^{(2)}}$
$= \sum_j \left(-\frac{y_j}{p_j}\right) \cdot \frac{\partial p_j}{\partial z_i^{(2)}}$
Softmax derivative:
if $i = j$: $\frac{\partial p_i}{\partial z_i} = p_i(1 - p_i)$
if $i \neq j$: $\frac{\partial p_j}{\partial z_i} = -p_i p_j$
Final result:
$\frac{\partial L}{\partial z_i^{(2)}} = p_i - y_i$

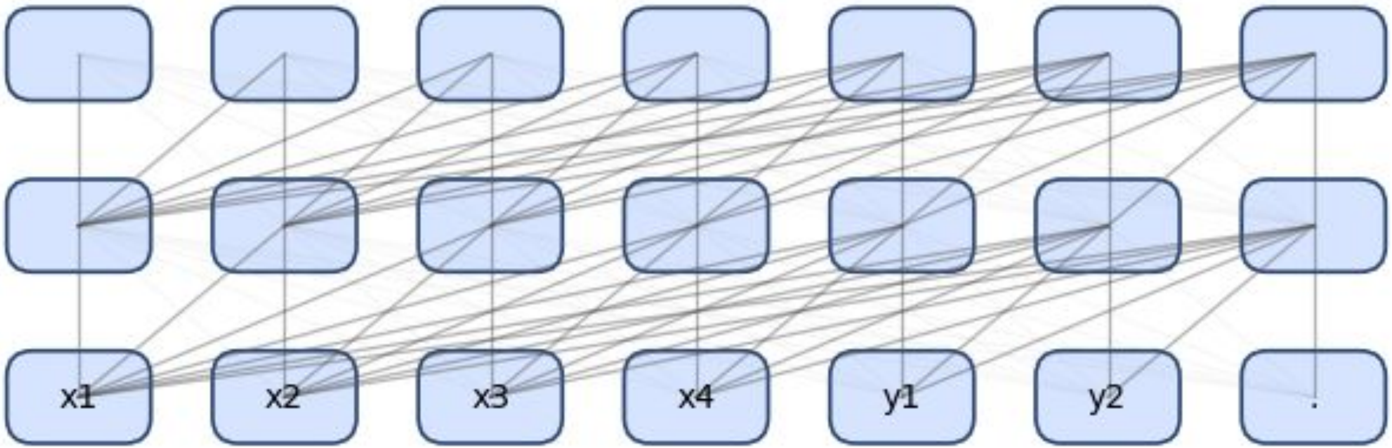

Input:
the
cat
sat
on
Step 1:
Embeddings (n × d)
Step 2:
Q
K
V
Step 3:
Scores = Q @ K^T
Step 4:
Causal Mask
Step 5:
Attention = Softmax
Step 6:
Output = Attention @ V
Output:
Enriched Embeddings

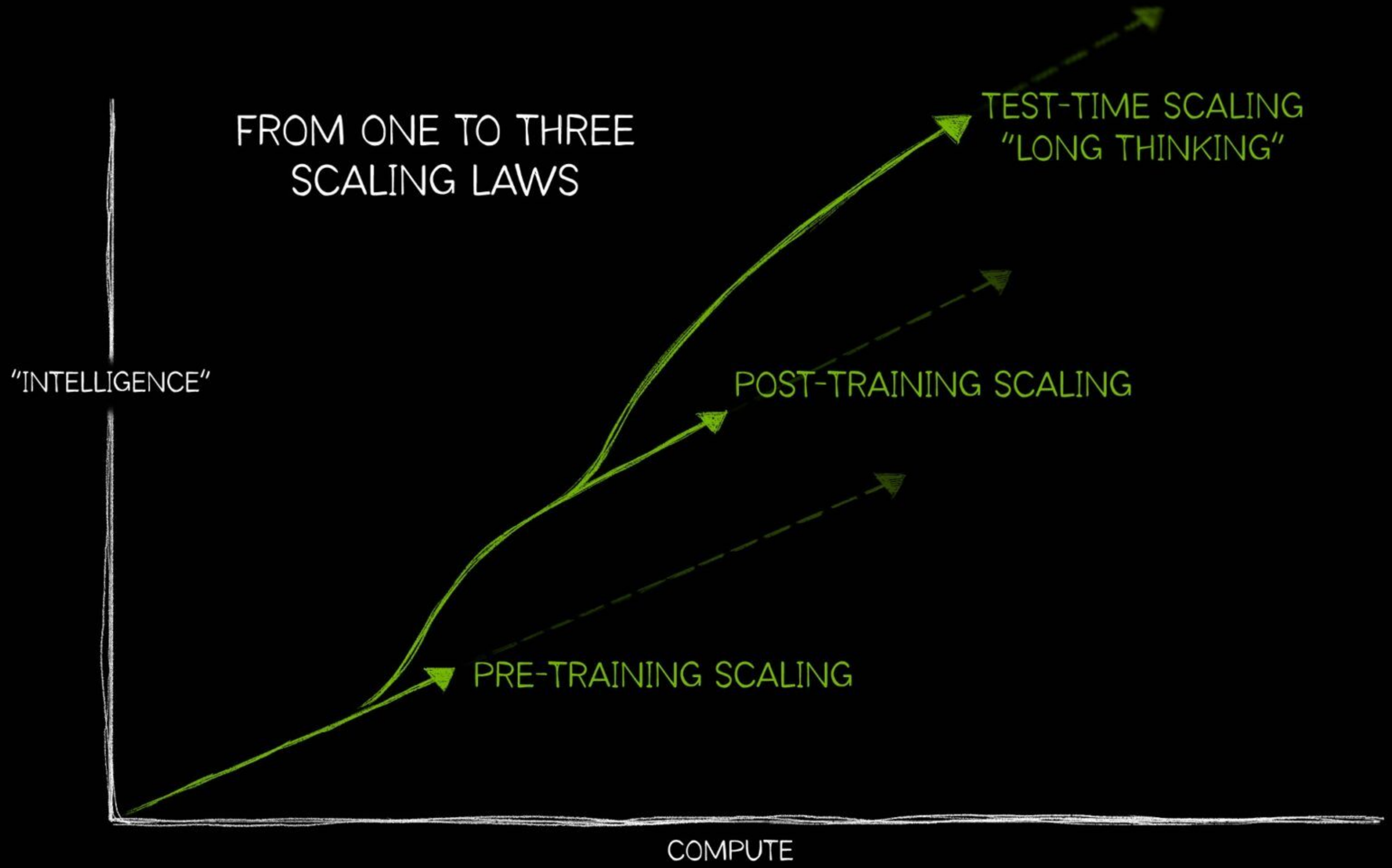


🤖 Attention is All You Need

Decoder-only

causal self-attention




FROM ONE TO THREE
SCALING LAWS
TEST-TIME SCALING
"LONG THINKING"
POST-TRAINING SCALING
PRE-TRAINING SCALING
"INTELLIGENCE"
COMPUTE

Pre-training on Everything

Self-supervised pre-training:

- Train on massive text corpora by predicting masked or next tokens

What it learns:

- Syntax, semantics, reasoning patterns, and broad world knowledge

Outcome:

- General-purpose representations adaptable to many tasks

Impact:

- Pre-training builds the foundation; fine-tuning shapes the purpose

Supervised Fine-Tuning

Converts a pretrained foundation model into a task-specific LLM

Key process:

- Train on curated instruction-reasoning-answer triplets

Data quality > quantity:

- Clean, diverse, well-structured prompts answers yield stronger generalization

Purpose:

- Adapt broad “general intelligence” into practical skills - reasoning for the most

Alignment through RLxF

RLxF = Reinforcement Learning from X Feedback

- Uses human preference + reward models + verifiable rewards to guide behavior

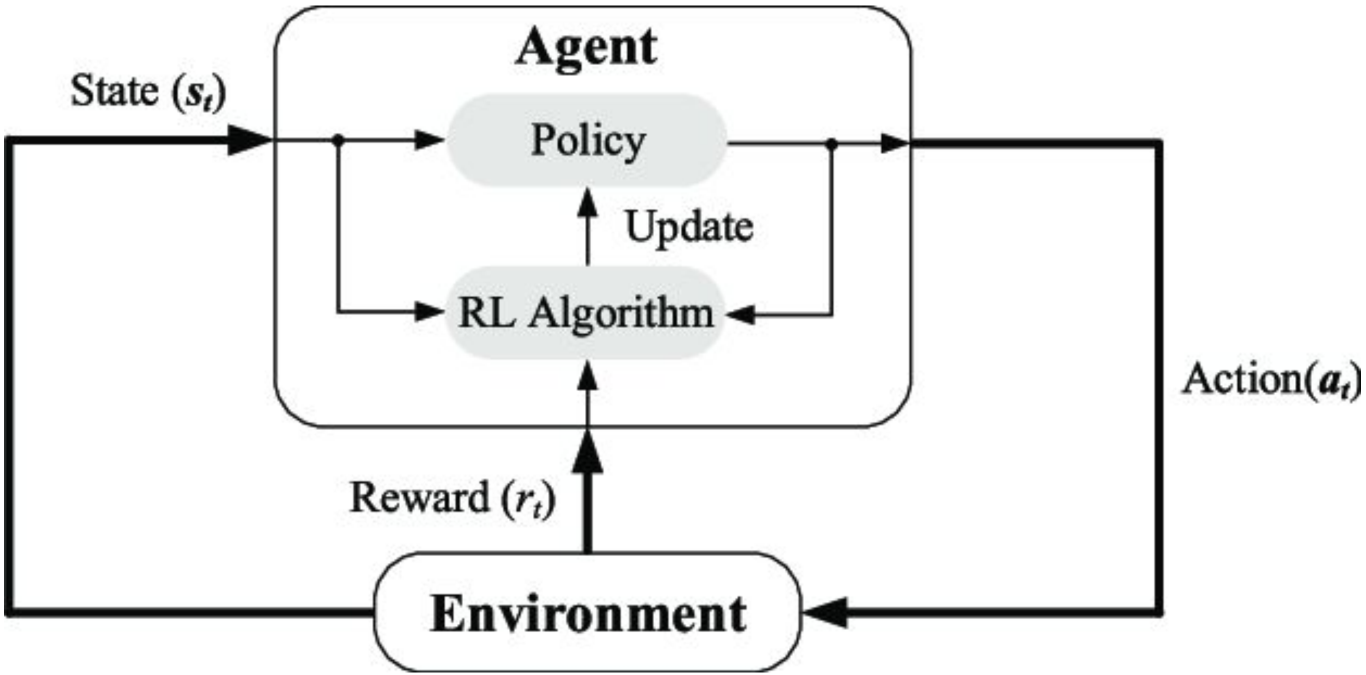
Process:

1️⃣Humans/LLMs rank outputs

2️⃣Reward model learns verifiable rewards/preferences

3️⃣LLM optimized via PPO to maximize those signals

Goal: Align models with verifiable/human preference signals 🏅


Agent
State (s_t)
Policy
Update
RL Algorithm
Action(a_t)
Reward (r_t)
Environment

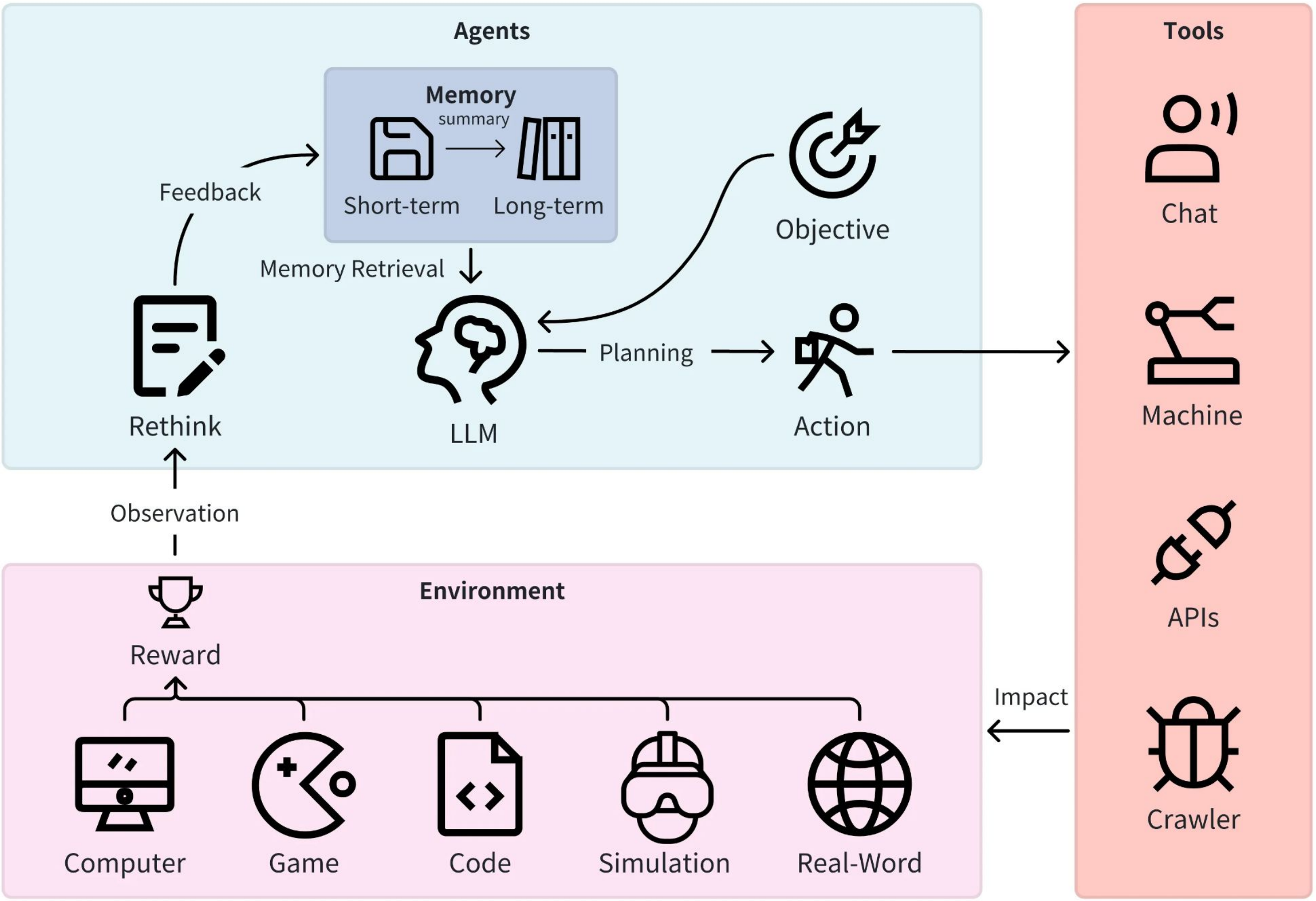
PPO: Proximal Policy Optimization 🧩

Core idea: small, controlled policy updates

MAXIMIZE objective (simplified version):

$$L(\theta) = \mathbb{E}_t\Big[\mathrm{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\, A_t\Big] \;-\; \beta\, \mathrm{KL}\big(\pi_\theta(a_t \mid s_t) \,\|\, \pi_{\mathrm{ref}}(a_t \mid s_t)\big)$$

$$\text{where} \quad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\mathrm{old}}(a_t \mid s_t)}$$



Agents
Memory
summary
Short-term
Long-term
Feedback
Memory Retrieval
Objective
Rethink
LLM
Planning
Action
Observation
Environment
Reward
Computer
Game
Code
Simulation
Real-Word
Tools
Chat
Machine
APIs
Crawler
Impact

What is Next?

Smaller, faster, cheaper LLMs with competitive performance

Agentic application everywhere

Multimodality: integrating text, vision, audio, action…

Advanced alignment: RLHF → RLAIF → Verifiable Rewards → 🤔

New architectures beyond Transformers

Skills to build in the Age of AI

Short term (to landing a job): Strong Python/C++ (vibe debugging/coding), solid ML fundamentals, hands-on project experience, and the ability to clearly COMMUNICATE ideas and work with others

Long term (to stay competitive): Develop rapid learning skills, the ability to build and debug quickly with AI, and cross-domain understanding to leverage AI for fast prototyping and real-world impact - then share your lessons to establish your brand and reputation

Mindset: Aim to become a full-stack problem solver - someone who can use AI tools to build, analyze, and deploy ideas **FASTER** 🚀 than before, and learn from success and failures