Advanced Machine Learning
Neural-Symbolic Thinking

Yu Wang
Assistant Professor
Department of Computer Science
University of Oregon

Background

Who can go first ?

A. The red car

B. The blue van

C. The white car

Neural-Thinking

Symbolic-Thinking

Background

Who can go first ?

A. The red car

B. The blue van

C. The white car

Neural-Thinking – System 1

- This is intersection
- There are three cars
- One car is turning left, the other car is going straight, the third car is going straight

Symbolic-Thinking – System 2

- Left should wait for straight

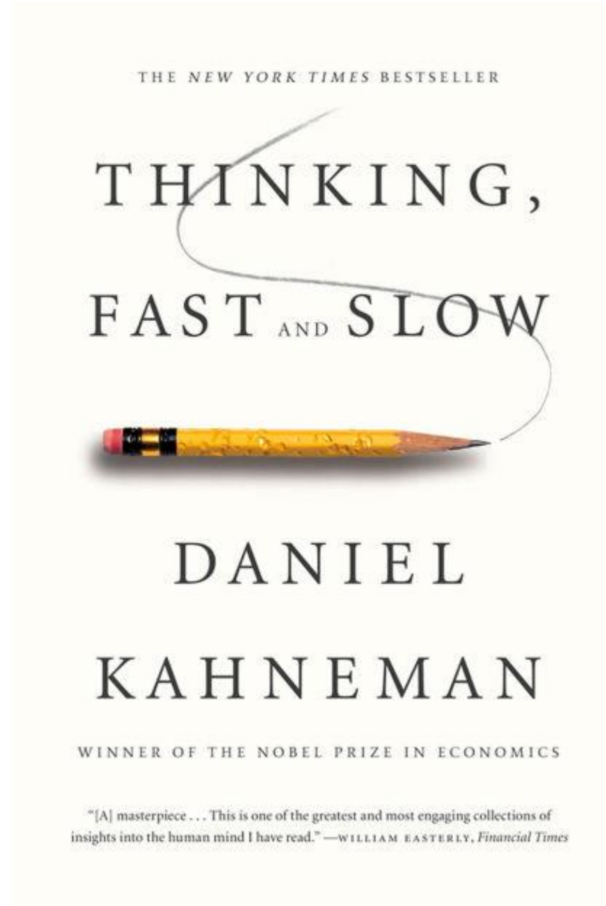
Background

Neural-Thinking – System 1

- Thinking fast
- Intuitive, human recognition
- Data-driven

Neural-Thinking – System 1

- Thinking slow
- Rule-based, theoretical guarantee
- Knowledge-driven



Background

Neural-Thinking – System 1

- Thinking fast
- Intuitive, human recognition
- Data-driven

Neural-Thinking – System 1

- Thinking slow
- Rule-based, theoretical guarantee
- Knowledge-driven

Are there more trees than animals?

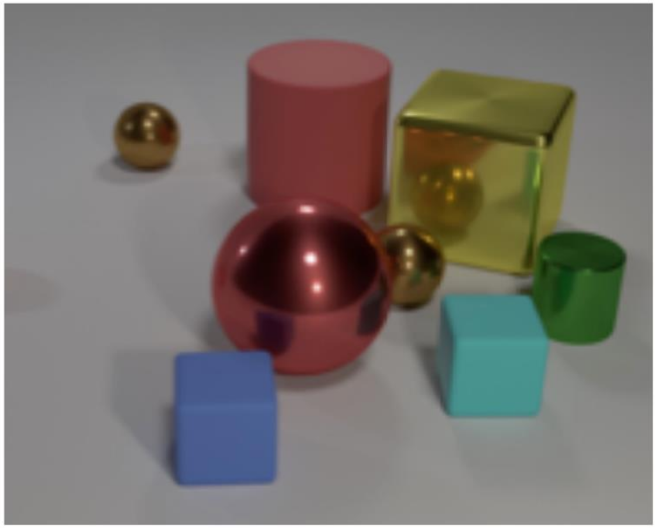
Background

Neural-Thinking – System 1

- Thinking fast
- Intuitive, human recognition
- Data-driven

Neural-Thinking – System 1

- Thinking slow
- Rule-based, theoretical guarantee
- Knowledge-driven



What is the shape of the object closest to the large cylinder?

Background

Neural-Thinking – System 1

- Thinking fast
- Intuitive, human recognition
- Data-driven

Neural-Thinking – System 1

- Thinking slow
- Rule-based, theoretical guarantee
- Knowledge-driven

Will the block tower fall if the top block is removed?

Background

Neural-Thinking – System 1

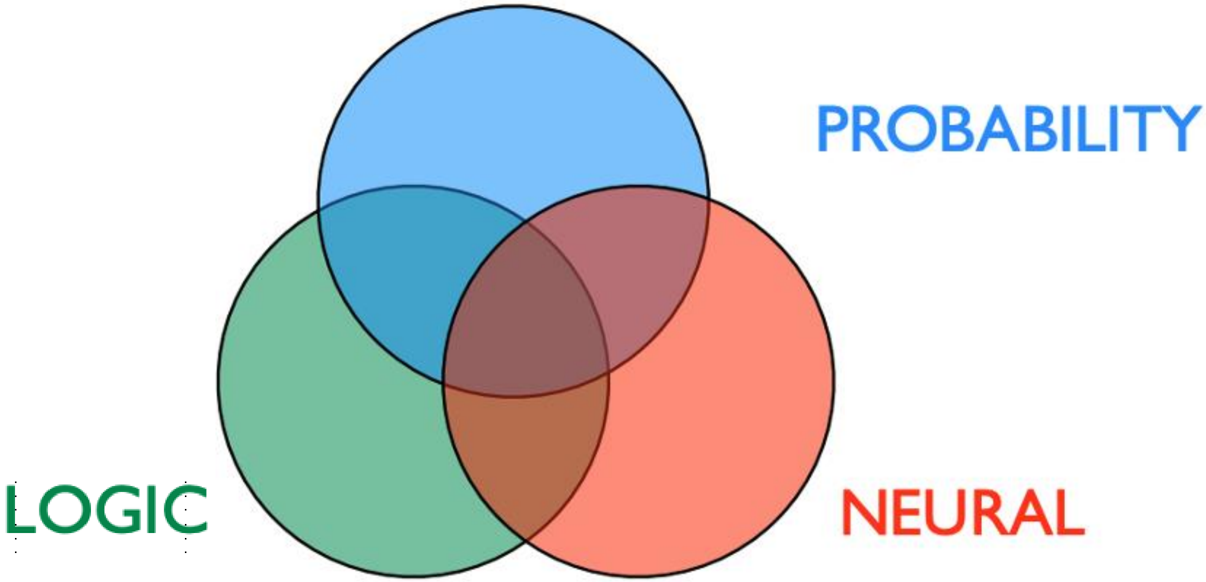
- Thinking fast
- Intuitive, human recognition
- Data-driven

Neural-Thinking – System 1

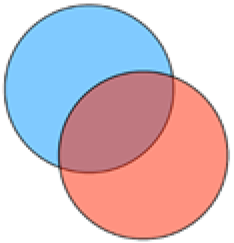
- Thinking slow
- Rule-based, theoretical guarantee
- Knowledge-driven

How many blocks are on the right of the three-level tower?

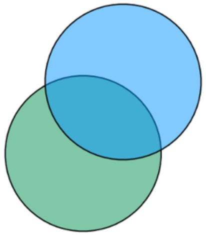
Probability, Logic, and Neural



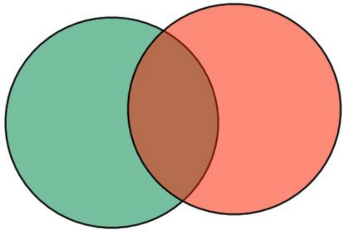
Probability - Neural



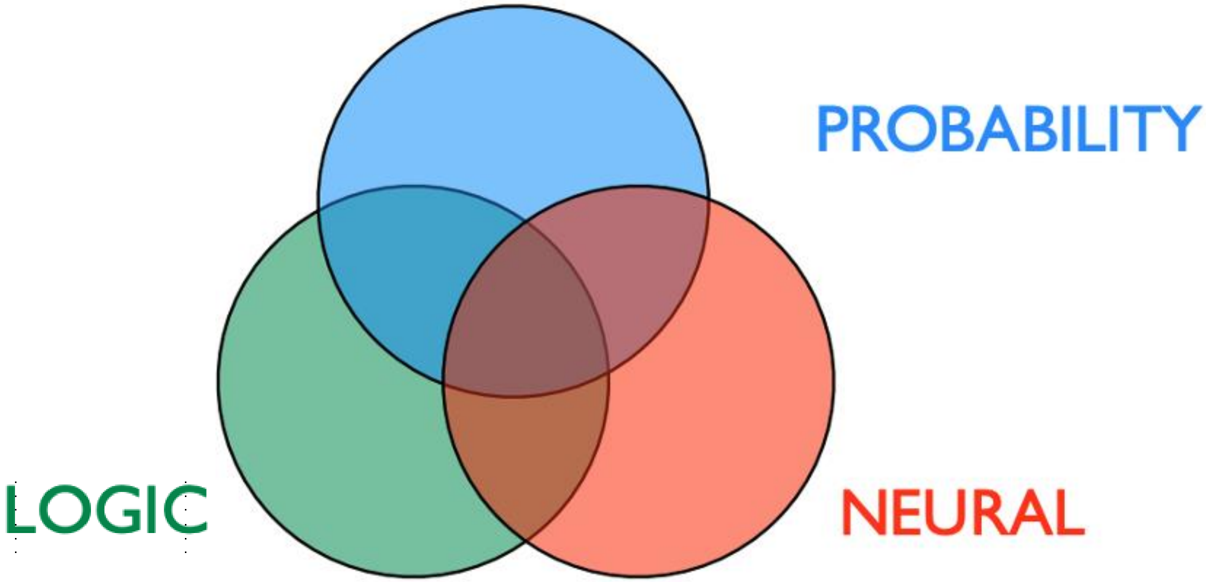
Probability - Logic



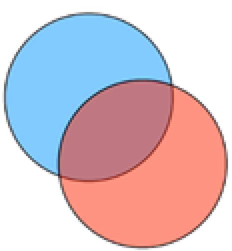
Logic - Neural



Probability, Logic, and Neural

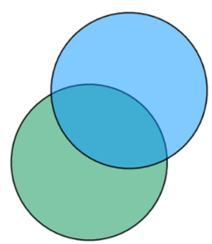


Probability - Neural



- VAEs
- Diffusion
- Policy Gradient

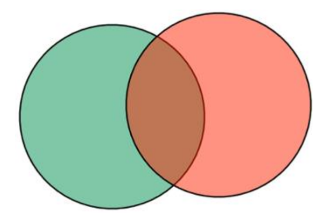
Probability - Logic



- C = A AND B

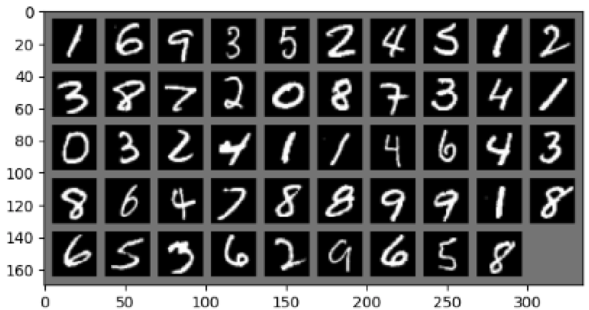
A: P(A>=0.5)
B: P(B>=0.5)

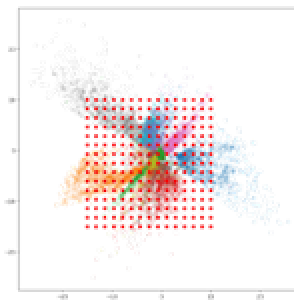
Logic - Neural



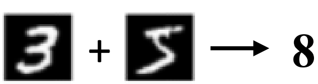
Why Neural-Symbolic Learning?

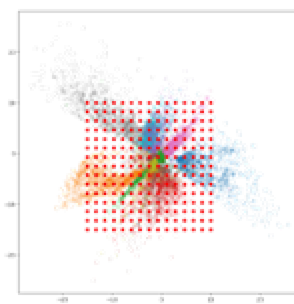
Benefit of Symbolic Learning on Neural Learning





Digit classification

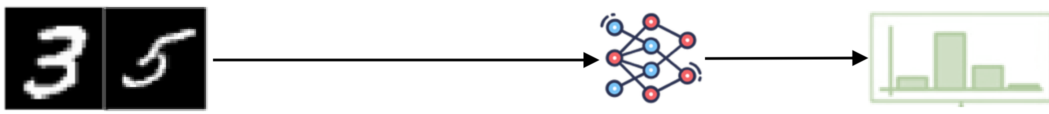




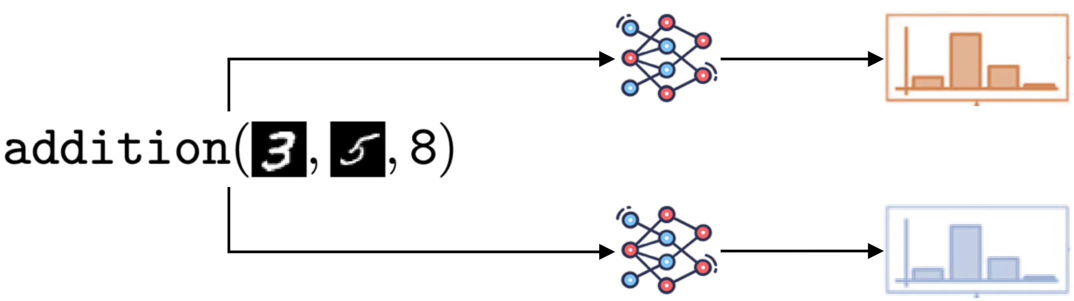
Digit classification with external calculation knowledge

Why Neural-Symbolic Learning?

Benefit of Symbolic Learning on Neural Learning



$$\mathcal{L} = -\sum_{k=0}^{18} (c_i = k) p_i^k$$

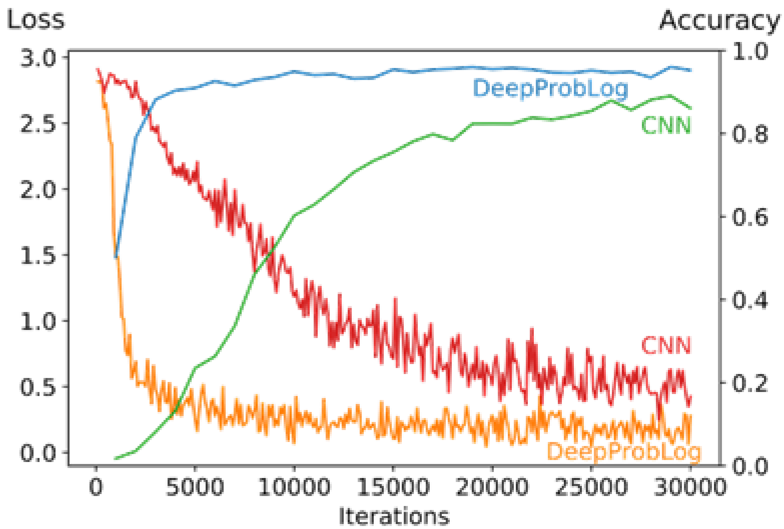


$$\mathcal{L} = -\sum_{k=1}^{18} (c_i = k) p_i^k$$

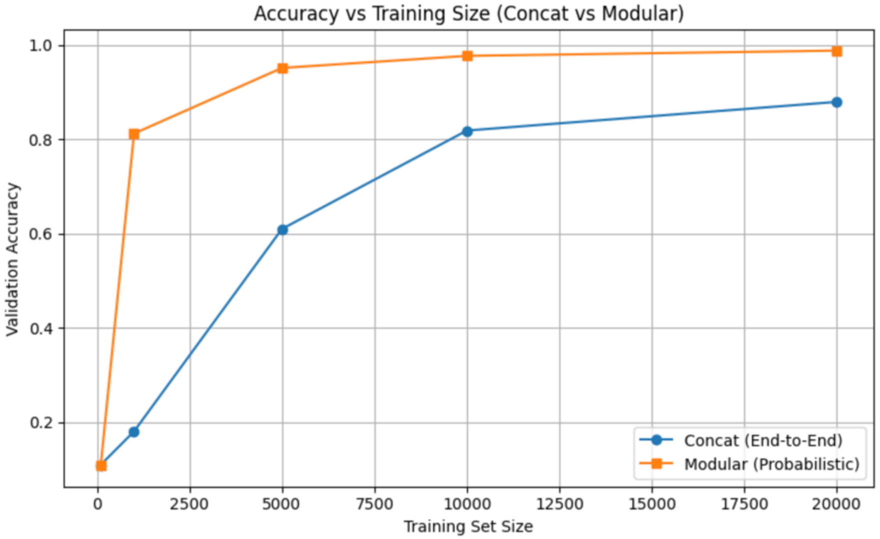
$$p_i^8 = \sum_{j=0}^{8} p_i^j p_i^{k-j}$$

Why Neural-Symbolic Learning?

Benefit of Symbolic Learning on Neural Learning



In the paper

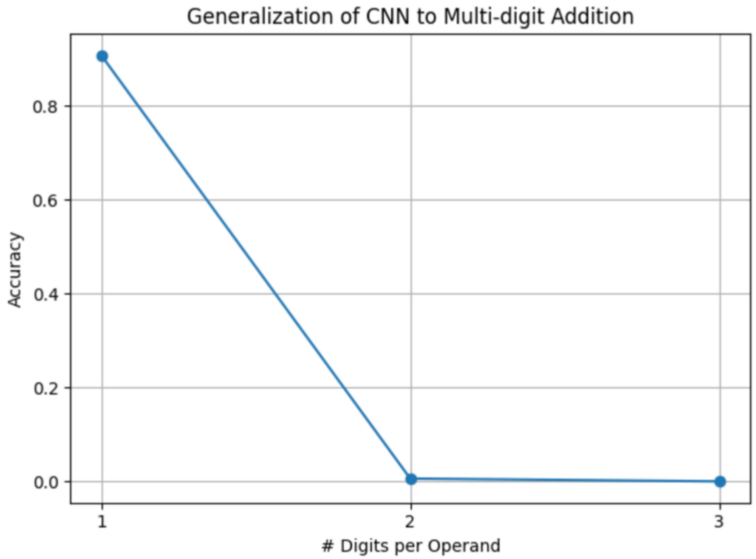


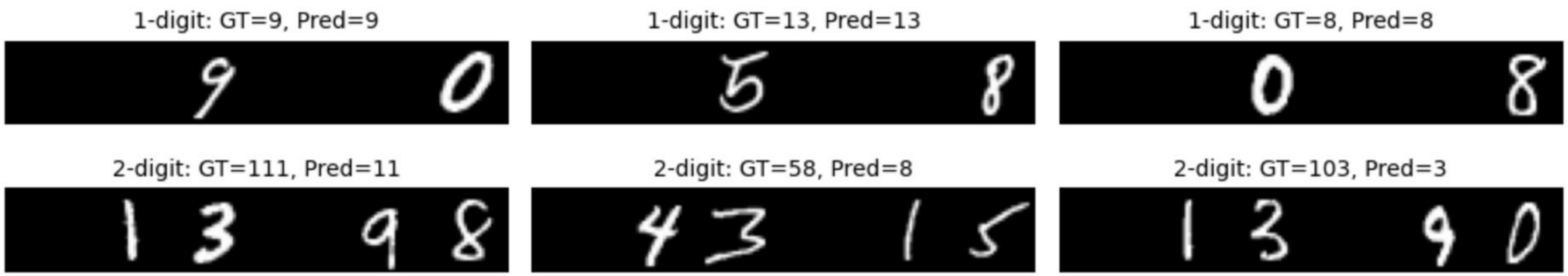
Our results

Benefit: more-easy to generalize

Why Neural-Symbolic Learning?

Benefit of Neural Learning on Symbolic Learning



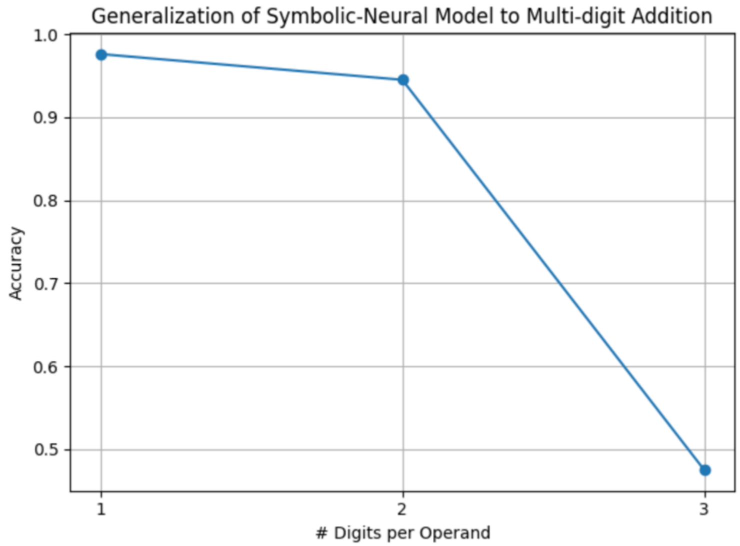


Why Neural-Symbolic Learning?

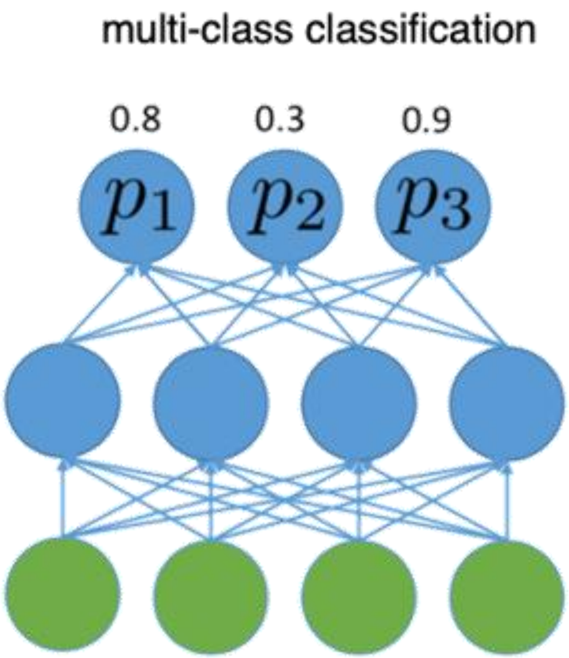
Benefit of Neural Learning on Symbolic Learning

```
# === Symbolic logic: compute sum from digit logits ===
def predict_sum_from_digits(logits, num_digits=1):
    B = logits.shape[0]
    probs = F.softmax(logits, dim=2)  # (B, 2D, 10)

    sums = []
    for b in range(B):
        s1, s2 = 0, 0
        for i in range(num_digits):
            d1 = torch.arange(10, device=logits.device)
            p1 = probs[b, 2*i]
            d2 = torch.arange(10, device=logits.device)
            p2 = probs[b, 2*i+1]
            exp1 = (d1 * p1).sum().round().item()
            exp2 = (d2 * p2).sum().round().item()
            s1 = s1 * 10 + exp1
            s2 = s2 * 10 + exp2
        sums.append(int(s1 + s2))
    return torch.tensor(sums, device=logits.device)
```



Symbolic as a kind of Constraints



Symbolic as a kind of Constraints

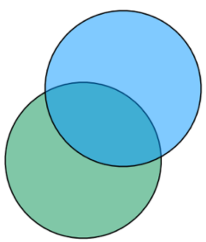
- Semantic loss $SLoss(T) \propto -\log \sum_{X \models T} \prod_{x \in X} p_i \prod_{\neg x \in X} (1 - p_i)$
- Used as regulariser $Loss = TraditionalLoss + w.SLoss$

How Neural-Symbolic Learning?

Symbolic as a kind of Constraints

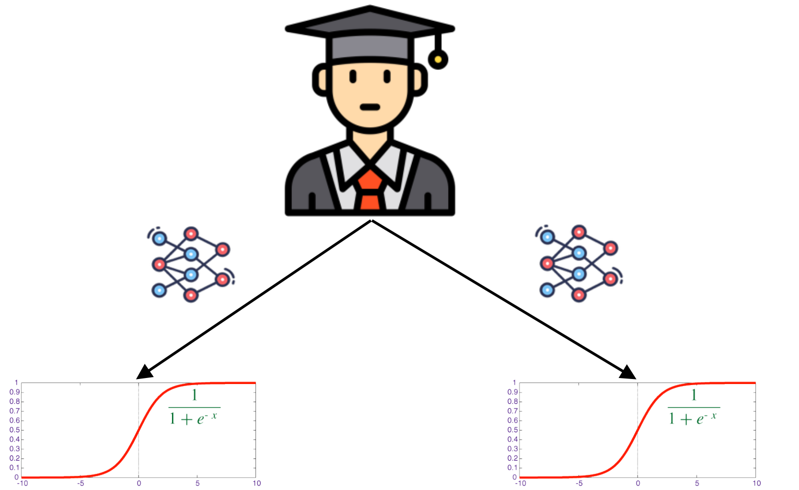
Fuzzy Logic

- $\text{AND}(a, b) = \min(a, b)$
- $\text{OR}(a, b) = \max(a, b)$
- $a \rightarrow b = \max(1 - a, b)$



Probability - Logic

Differentiable



Hard Working? **Appropriate Method?**

Hard Working **AND** Appropriate Method

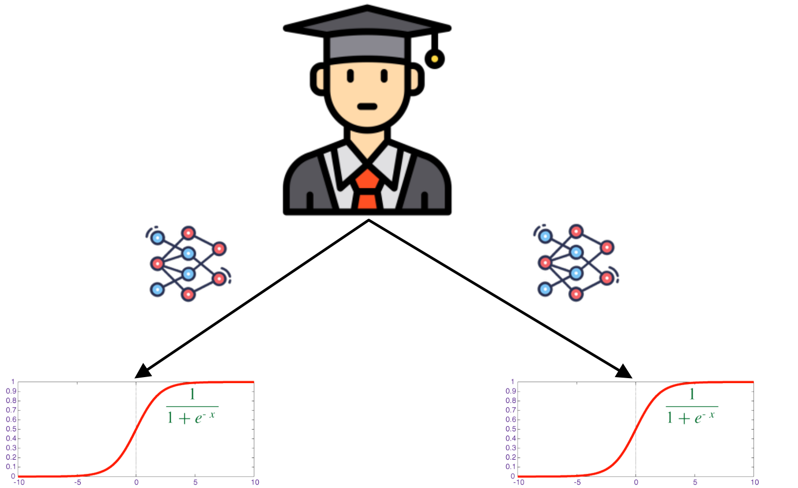
Hard Working **OR** Appropriate Method

Hard Working **--** Appropriate Method

How Neural-Symbolic Learning?

Symbolic as a kind of Constraints

$$\mathrm{AND}(a, b) = \min(a, b)$$

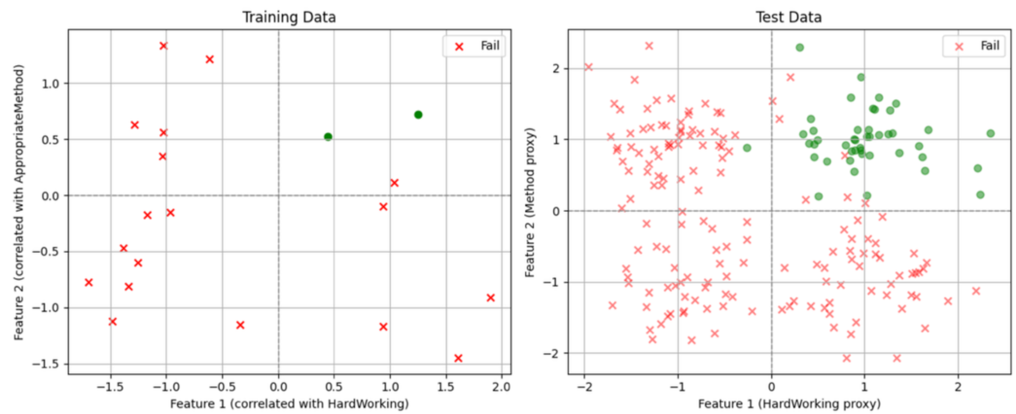


Hard Working?

Appropriate Method?

Hard Working **AND** Appropriate Method

Success(x) ⇐ HardWorking(x) ∧ AppropriateMethod(x)



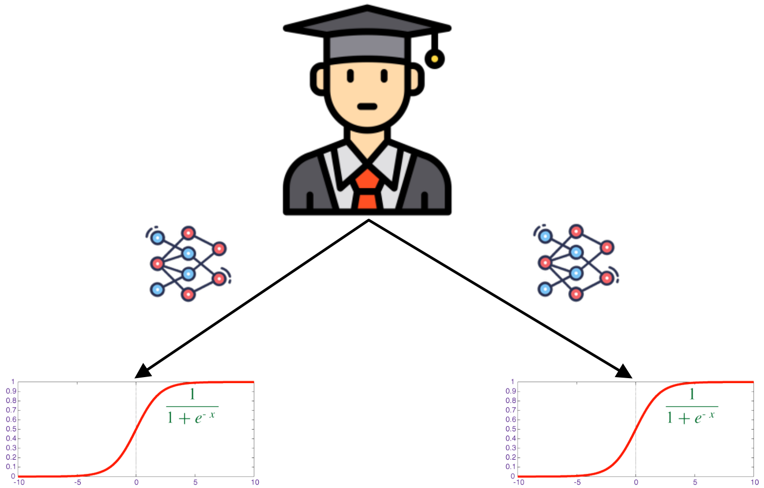

```python
# Helper to sample feature given predicate truth
def sample_feature(is_true):
    return random.gauss(1.0 if is_true else -1.0, 0.5)

# Generate training data
train_X, train_y = [], []
for i in range(n_train):
    # Random truth values for predicates
    H = 1 if random.random() < 0.5 else 0
    M = 1 if random.random() < 0.5 else 0
    S = 1 if (H == 1 and M == 1) else 0          # Success = H AND M
    # Features: correlated with H and M, plus noise
    f1 = sample_feature(H)                     # feature1 ~ N(+1 or -1)
    f2 = sample_feature(M)                     # feature2 ~ N(+1 or -1)
    f3 = random.gauss(0, 1)                    # feature3 ~ N(0, 1) noise
    train_X.append([f1, f2, f3]);  train_y.append([float(S)])
```

How Neural-Symbolic Learning?

Symbolic as a kind of Constraints

$$\mathrm{AND}(a, b) = \min(a, b)$$



Hard Working? **Appropriate Method?**

Hard Working **AND** Appropriate Method

Success(x) ⇐ HardWorking(x) ∧ AppropriateMethod(x)

```
p_H, p_M, p_S = logic_model(train_X)          # forward pass (3 outputs)
# Standard loss on success prediction:
loss_main = criterion(p_S, train_y)
# Logic constraint loss (MSE between p_S and min(p_H, p_M)):
loss_logic = torch.mean((p_S - torch.minimum(p_H, p_M))**2)
loss_total = loss_main + loss_logic           # combined loss
```

Baseline Model Accuracy: 85.00% Logic-Constrained Model Accuracy: 95.00% Logic-Constrained Model Rule Consistency (probabilities): 74.00% Logic-Constrained Model Rule Consistency (binary): 80.50%

How Neural-Symbolic Learning?

Symbolic as a kind of neural program