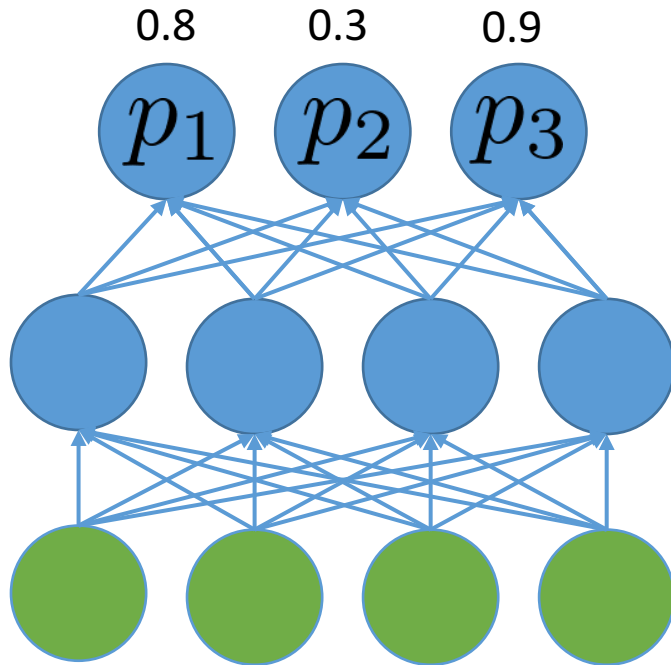


# A Semantic Loss Function for Deep Learning with Symbolic Knowledge

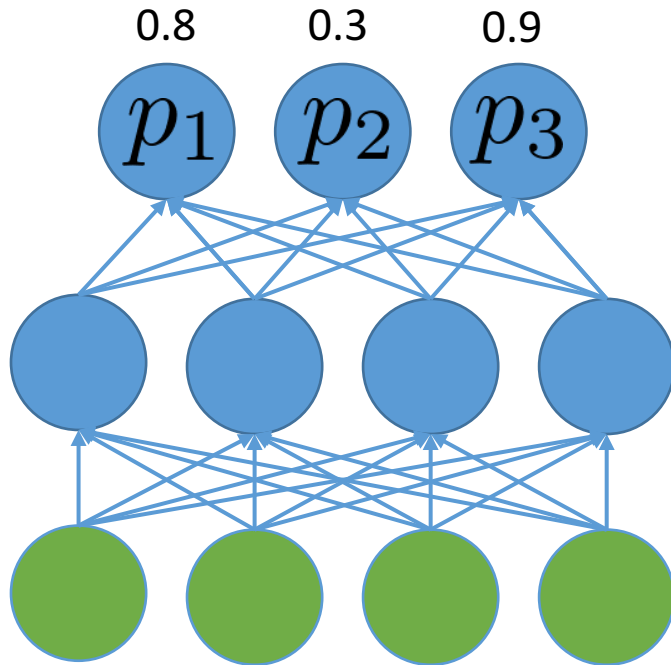
Jingyi Xu, Zilu Zhang, **Tal Friedman**, Yitao Liang,  
Guy Van den Broeck

***Goal: Constrain neural network outputs using logic***

# Multiclass Classification

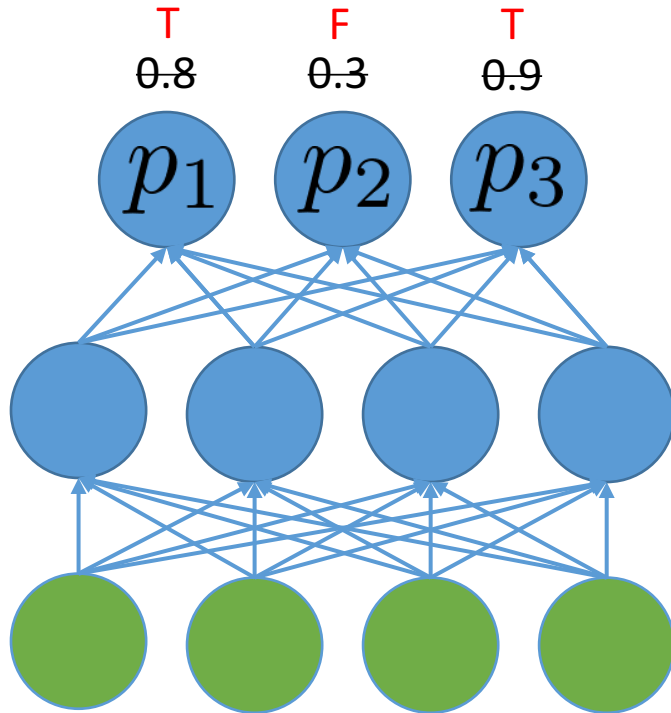


# Multiclass Classification



Want exactly one class: 
$$\begin{cases} x_1 \neg x_2 \neg x_3 \\ \vee \\ \neg x_1 x_2 \neg x_3 \\ \vee \\ \neg x_1 \neg x_2 x_3 \end{cases}$$

# Multiclass Classification

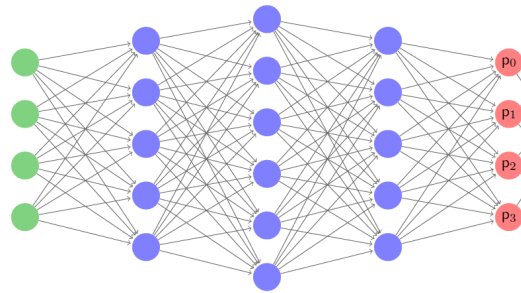


Want exactly one class: 
$$\begin{cases} x_1 \neg x_2 \neg x_3 \\ \vee \\ \neg x_1 x_2 \neg x_3 \\ \vee \\ \neg x_1 \neg x_2 x_3 \end{cases}$$

No information gained!

# Why is mixing so difficult?

## Deep Learning



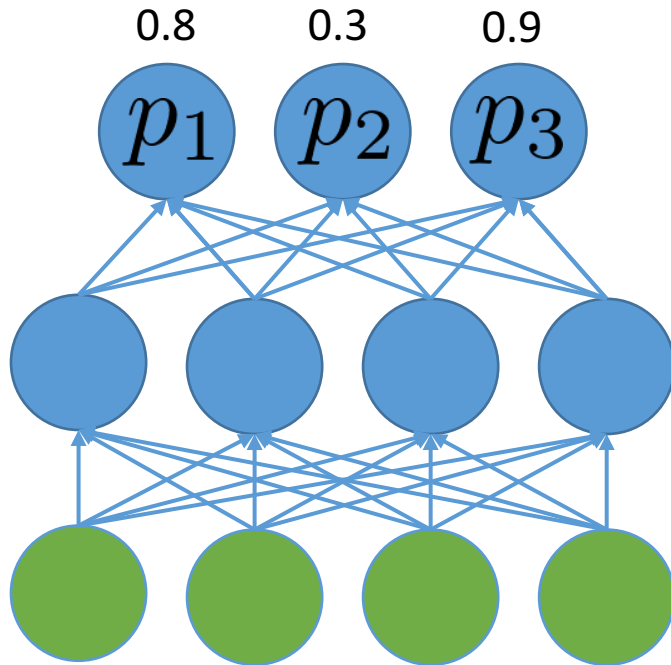
- Continuous
- Smooth
- Differentiable

## Logic

$$\begin{aligned} P \vee L \\ A \Rightarrow P \\ K \Rightarrow (P \vee L) \end{aligned}$$

- Discrete
- Symbolic
- Strong semantics

# Multiclass Classification



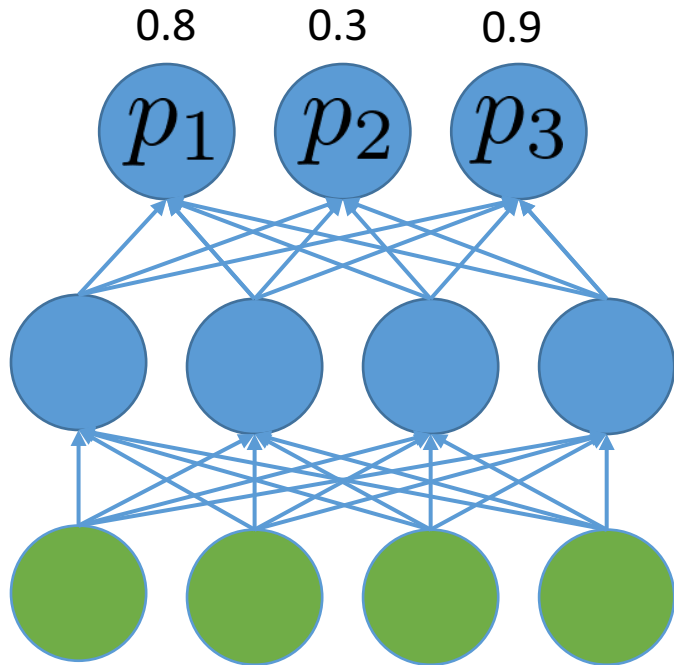
Want exactly one class: 
$$\begin{cases} x_1 \neg x_2 \neg x_3 \\ \vee \\ \neg x_1 x_2 \neg x_3 \\ \vee \\ \neg x_1 \neg x_2 x_3 \end{cases}$$

**Probability** constraint is satisfied

*Use a **probabilistic** interpretation!*



# Multiclass Classification



Want exactly one class: 
$$\begin{cases} x_1 \neg x_2 \neg x_3 \\ \vee \\ \neg x_1 x_2 \neg x_3 \\ \vee \\ \neg x_1 \neg x_2 x_3 \end{cases}$$

**Probability** constraint is satisfied

$$\begin{aligned} & x_1(1 - x_2)(1 - x_3) \\ & + (1 - x_1)x_2(1 - x_3) \\ & + (1 - x_1)(1 - x_2)x_3 \\ & = \mathbf{0.188} \end{aligned}$$

# Semantic Loss



- Continuous, smooth, easily differentiable function
- Represents how close outputs are to satisfying the constraint
- Axiomatically respects semantics of logic, maintains precise meaning
  - independent of syntax

*How do we compute semantic loss?*

# Logical Circuits

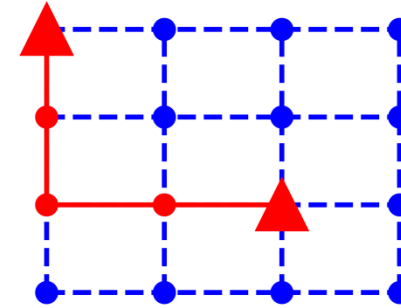
- In general: #P-hard
- Linear in size of circuit

$$L(\alpha, \mathbf{p}) = L(\text{Circuit}, \mathbf{p}) = -\log(\text{Circuit})$$

The diagram illustrates the relationship between a logical circuit and its probabilistic interpretation. On the left, a circuit with three AND gates is shown. The inputs to the circuit are  $x_1$ ,  $\neg x_2$ ,  $\neg x_3$ ,  $\neg x_1$ ,  $x_2$ , and  $x_3$ . The circuit's output is represented by the expression  $L(\alpha, \mathbf{p}) = L(\text{Circuit}, \mathbf{p})$ . On the right, the same circuit is shown as a tree diagram, where the output is the negative logarithm of the sum of the products of the probabilities of the inputs that satisfy the circuit's output. The inputs to the tree are  $\Pr(x_1)$ ,  $\Pr(\neg x_2)$ ,  $\Pr(\neg x_3)$ ,  $\Pr(\neg x_1)$ ,  $\Pr(x_2)$ , and  $\Pr(x_3)$ .

# Supervised Learning

- Predict shortest paths
- Add semantic loss representing paths



Test accuracy %	Coherent	Incoherent	Constraint
5-layer MLP	5.62	<b>85.91</b>	6.99
Semantic loss	<b>28.51</b>	83.14	<b>69.89</b>

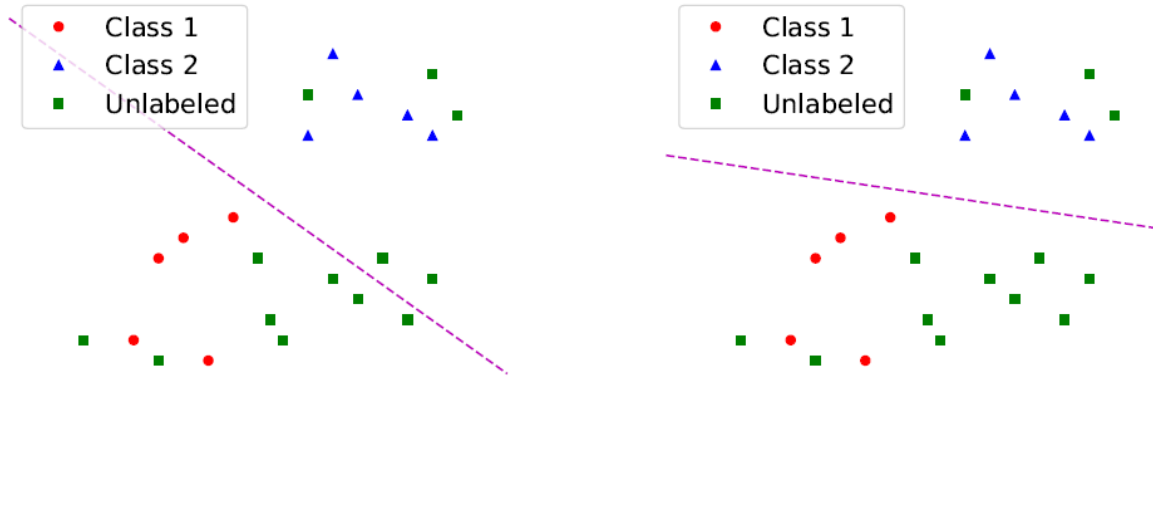
*Is output  
the true shortest path?*

*Does output  
have true edges?*

*Is output  
a path?*

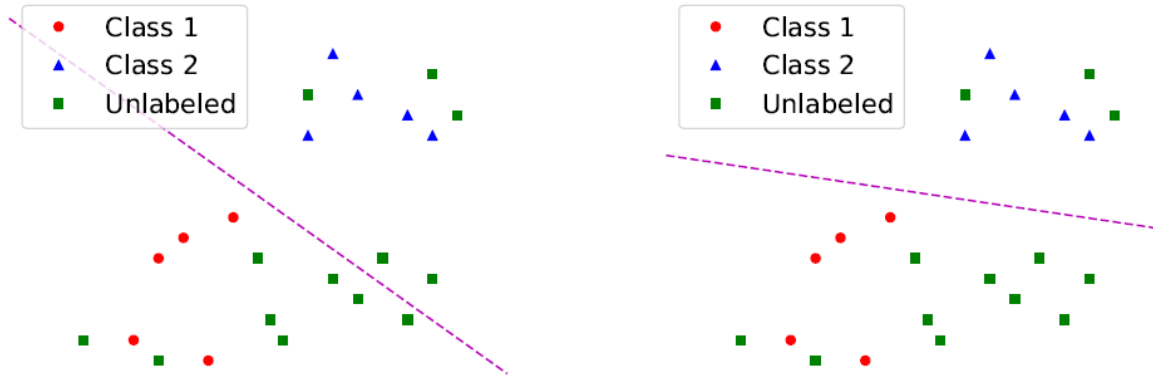
# Semi-Supervised Learning

- Unlabeled data must have some label



# Semi-Supervised Learning

- Unlabeled data must have some label



- Exactly-one constraint increases confidence



Table 2: FASHION. Test accuracy comparison between MLP with semantic loss and ladder nets.

Accuracy % with # of used labels	100	500	1000	ALL
Ladder Net (Rasmus et al., 2015)	81.46 ( $\pm 0.64$ )	85.18 ( $\pm 0.27$ )	86.48 ( $\pm 0.15$ )	<b>90.46</b>
Baseline: MLP, Gaussian Noise	69.45 ( $\pm 2.03$ )	78.12 ( $\pm 1.41$ )	80.94 ( $\pm 0.84$ )	89.87
MLP with Semantic Loss	<b>86.74</b> ( $\pm 0.71$ )	<b>89.49</b> ( $\pm 0.24$ )	<b>89.67</b> ( $\pm 0.09$ )	89.81



# Main Takeaway



- Deep learning and logic **can** be combined by using a probabilistic approach
- Maintain precise meaning while fitting into the deep learning framework

Thanks!