Reduced Implication-bias Logic Loss for Neuro Symbolic Learning

Presenter: Haoyuan He / Paper: Link

Introduction

Begin of our story.

Currently, machine learning methods are achieving significant success in perception. However, real-world learning tasks usually require not only the perception ability but also the logical reasoning ability. It is reasonable to believe that the next generation of advanced artificial intelligence will be seamlessly constructed by combining logical reasoning with machine learning.

Neuro-Symbolic Learning

Neuro-Symbolic (NeSy) AI aims to combine neural networks and symbolic reasoning.

Some people try to implement a hybrid system by implementing an interface between neural networks and symbolic reasoning systems. Zhou and Dai 2019 introduce the Abductive Learning (ABL) framework to integrate first-order logic with machine learning models and abductive reasoning. Manhaeve 2018 propose DeepProbLog, which integrates probabilistic logic programming with deep learning by interfacing them with neural predicates.

However, these models are difficult to train because of the high complexity caused by the joint optimisation of the neural and symbolic modules. Therefore, many researchers propose to approximate logical reasoning with differentiable operators, thereby making it possible to turn symbolic knowledge into loss functions and train an end-to-end model by gradient descent.

Implication Bias [breif intro]

In the process of approximating implication rules such as $p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q$ with differentiable operations, a phenomenon that we called **Implication Bias** could degrade the model performance. Informally, this rule could be satisfied via vacuous truth, i.e., by negating the premises $p_1 \wedge p_2 \wedge \cdots \wedge p_n$. Under ideal circumstances, samples that do not satisfy the rule premises, their loss values and gradients computed from this rule should be zero. However, we found this is not the case for semantic loss and many other widely used logic loss functions derived from fuzzy operators. Furthermore, the shortcuts problem of neural network training would amplify this bias and make the model tends to negate the premise literals when the training data is insufficient.

We point out that the major problem behind this phenomenon is that the model keeps inferring unwanted negations during the training process. In logic programming, people employ Closed World Assumption and Negation As Failure (NAF) to prevent this issue and guarantee the soundness of inference. Nevertheless, in NeSy systems, there lacks such a method to reduce this bias.

Preliminaries

First Order Logic

A First Order Logic (FOL) $\mathcal{L}$ consists of a set $\mathcal{C}$ of constant symbols, a set $\mathcal{F}$ of functional symbols, a set $\mathcal{P}$ of predicates (relations) symbols, and a set $\mathcal{V}$ of variable symbols, a set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ of connectives, a set $\{\forall, \exists\}$ of quantifiers.

The connectives set can be reduced to $\{\neg, \rightarrow\}$ because other operations can be replaced with a combination of these two operations. For example, $p \wedge q \equiv \neg(p \rightarrow \neg q), p \leftrightarrow q \equiv \neg((p \rightarrow q) \rightarrow \neg(q \rightarrow p)), p \vee q \equiv \neg p \rightarrow q$. Atom is the basic component of logic formulas, such as $\mathrm{BiggerThan}(x, \mathrm{Add10}(y)), \mathrm{Equal}(\mathrm{Succ}(x), y)$. Denote the set of atoms as $\mathcal{A}$, the set of formulas as $\mathbb{F}$.

Closed World Assumption

The closed-world assumption (CWA) is a presumption that a true statement is also known to be true. Conversely, what is not currently known to be true, is false. The opposite of the closed-world assumption is the open-world assumption (OWA), holding that lack of knowledge does not imply falsity. The CWA is a very natural rule to use in a database context. Information not explicitly presented in the database is taken to be false. For example:

Statement	Bob and Cam are students.
Question	Is Alice a student?
Closed world answer	No.
Open world answer	Unknown.

Negation as Failure

Suppose we wish to infer $\neg A$ from a program P. To use the CWA, we have to show that A is not a logical consequence of P. Unfortunately, because of the undecidability of the validity problem of first-order logic, there is no algorithm which will take an arbitrary A as input and respond in a finite amount of time with the answer whether A is or is not a logical consequence of P. If A is not a logical consequence, it may loop forever.

To alleviate this problem, Clark 1978 proposed the *negation as failure* rule. It states that if A is in the finite failure set of P, then infer $\neg A$. We can see that the negation as a failure rule is less powerful than the CWA. However, implementing anything beyond negation as a failure rule is difficult in practice.

Semantics of Continuous-valued Logic

To transform logic rules into loss functions, we have to assign every grounding term in logic rules a continuous value between 0 and 1. In most cases, this assignment will be given by the machine learning model's output.

Definition. *Valuation of Atom*

A valuation of atom is a function $G(\cdot)\colon \mathcal{A} \to [0,1]$.

It is natural to define the valuation of negation of an atom $G(\neg A)$ as $1 - G(A)$.

Definition. *Implication Likelihood*

A function $I(\cdot,\cdot)\colon [0,1] \times [0,1] \to [0,1]$ is implication likelihood if it is differentiable.

Implication likelihood is a function that is used to estimate the degree of satisfiability of implication rules, which is closely related to fuzzy t-norms. For example, $I(G(\mathrm{Raven}(x)), G(\mathrm{Black}(x))) = 0.95$ says the satisfiability of this rule : $\mathrm{Raven}(x) \to \mathrm{Black}(x)$ is 0.95.

Semantics of Continuous-valued Logic (Con't)

Since computing the satisfiability of negation and implication rules being available, as mentioned before, in FOL, $\{\neg, \rightarrow\}$ as connectives are enough. It is easy to calculate any logic rule's satisfiability by $\rightarrow$ and $\neg$ recursively.

> **Definition.** *Logic Likelihood*
>
> Logic Likelihood $s(\cdot,\cdot):\mathbb{F}\times[0,1]^n\rightarrow[0,1]$ is a function that used to estimate the degree of satisfiablity of logic formula. $s(\cdot,\cdot)$ takes two inputs, the first one is a logic formula, and the second one is the valuation vector of this logic formula.

For example:$s(P\rightarrow Q,(x,y))=I(x,y)$

Semantics of Continuous-valued Logic (Con't)

Definition. *Confidence Monotonic*

A Logic Likelihood is called δ-Confidence Monotonic if its Implication Likelihood satisfies:

- For any $y \in [0, \delta)$, $I(x, y)$ is monotonically decreasing on x.
- For any $x \in (1 - \delta, 1]$, $I(x, y)$ is monotonically increasing on y.

For example, the Implication Likelihood used most commonly is defined as $I(x, y) = 1 - x + x \cdot y$. It is a 1 -Confidence Monotonic Implication Likelihood.

An example for understanding this definition may be helpful. Suppose we have an implication rule, $\mathrm{Raven}(x) \to \mathrm{Black}(x)$. When $G(\mathrm{Black}(x))$ is nearly 0, that is to say, x is not black w.h.p., the higher confidence we trust $\mathrm{Raven}(x)$, the less satisfiability of this rule, vice versa.

Definition. *Logic Loss*

A Logic Loss ℓ_{logic} is a function that estimates the degree of unsatisfiability of logic formula r. It is derived from a Logic Likelihood: $\ell_{\text{logic}}(r, z) = g(s(r, z))$ where g is a monotonically decreasing function, w.l.o.g., assume $g(1) = 0$.

Impact of additional logic loss

During the training process of NeSy system which using logic loss as inference of knowledge bases, the optimization object should be set as:

$$\ell = \ell_{\text{task}} + \lambda \cdot \ell_{\text{logic}}$$

Combining logic loss with a task-specific loss for machine learning should be mutually beneficial. In one way, the task-specific loss can direct the model to achieve a pretty good performance so it can give out valid logic primitive facts. In the other way, the logic loss can help shrink the empirical optimal space that can make the model easier to achieve the optimum.

However, a logic loss with δ-confidence monotonic implication likelihood may introduce bias into NeSy systems during the training process.

Proposed Method

Implication Bias

Definition. *Implication Biased Loss*

A Logic Loss is Implication Biased if its Implication Likelihood $I(x, y)$ satisfies: $\forall x, y \in (0, \delta)$

$$\frac{\partial I}{\partial x} < 0$$

For simplicity, a logic loss is called implication biased if there exists some $\delta \in (0, 1]$ that make this logic loss δ-Implication Biased.

Theorem.

A Logic Loss ℓ_{logic} with δ-Confidence Monotonic Logic Likelihood is also δ-Implication Biased.

Implication biased means that when both the premise and the consequent are close to zero, the gradient given by the logic loss will still be in the direction of negating the premise. This suggests that **implication-biased logic loss tends to negate the premise of implication rules.**

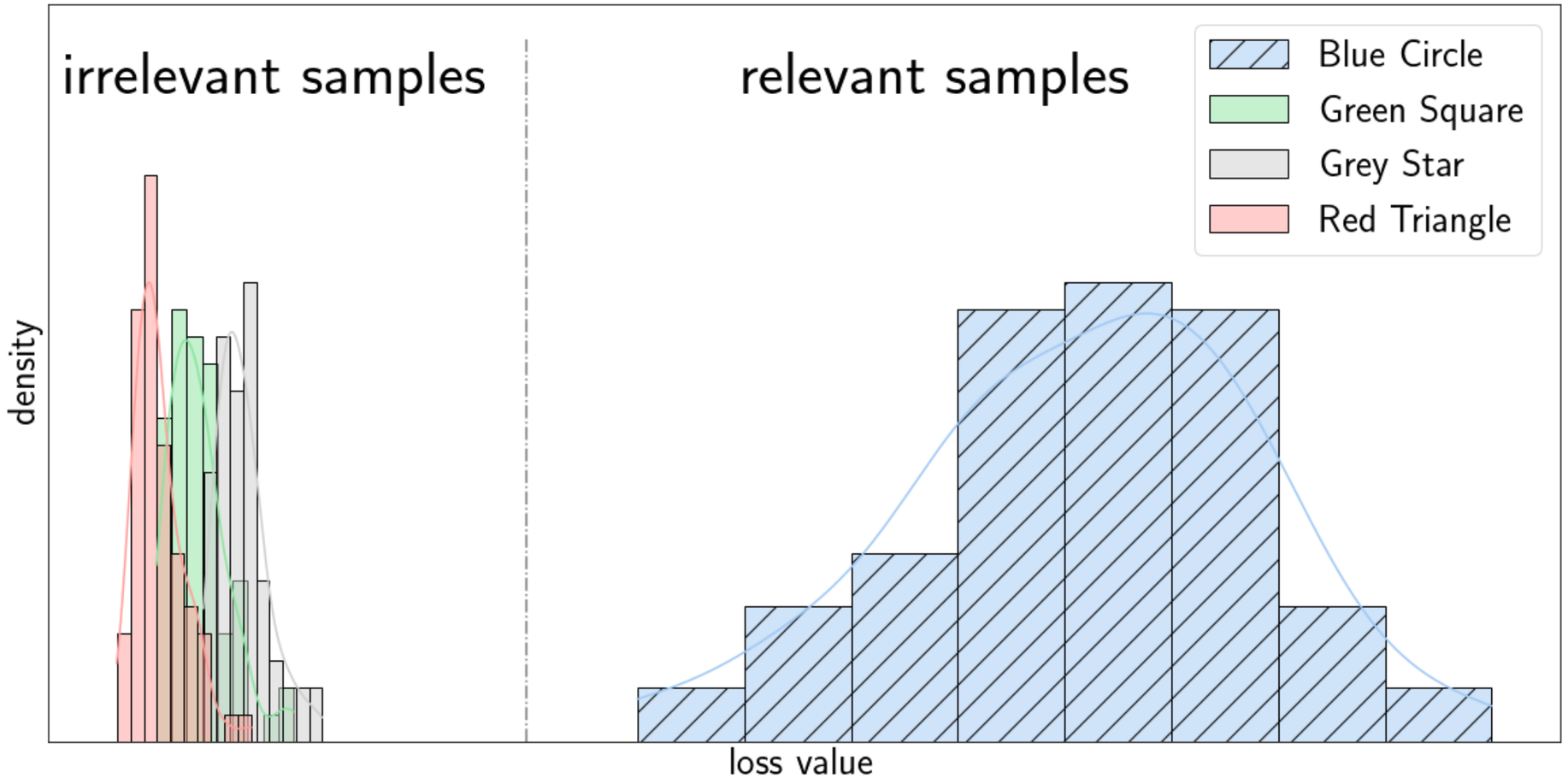
Implication Bias:Case Study

The training set is split from the whole dataset. It holds 25% data and only contains the shape label. Two different knowledge bases shown in figure are $\{\mathrm{Blue}(x) \rightarrow \mathrm{Circle}(x)\}$ and $\{\mathrm{Circle}(x) \rightarrow \mathrm{Blue}(x)\}$.

Task-specific loss ℓ_{task} is CrossEntropy and logic loss ℓ_{logic} derived from Implication Likelihood $I(x, y) = 1 - x + x \cdot y$.

As the left bottom picture, when ℓ_{logic} is implication biased, the optimised model will refuse to predict any sample's colour as blue, although the initial model can predict colour-label right. This phenomenon also happened in the right bottom picture. Both of them negate their rule's premise.

Implication Bias:Case Study (Con’t)



Reduced Implication-bias Logic Loss

Let us analyse implication bias from two perspectives.

Optimization of Machine Learning

Implication bias will induce neural networks to negate the premise of implication rules, which is a shortcut in the learning process. Considering neural networks tend to learn shortcuts, this bias will bring a nuisance to the training process.

Logic Programming

Implication bias will induce the model to negate the premise of implication rules. However, in an open world, negating a predicate is undecidable. For example, if we have a statement $\mathrm{Raven}(x) \to \mathrm{Black}(x)$. If we try to negate its premise, that is to say, $\neg\mathrm{Raven}(x)$. Since so many predicates can be "not raven", we do not have an explicit target to improve the model.

Reduced Implication-bias Logic Loss (Con’t)

> Insight of `Reduced Implication-bias Logic Loss`
>
> **Samples in the sensitive region should be assigned less importance to the training process.**

- Samples in the sensitive region usually get a relatively small loss value, thus having less importance to the optimisation process than others. Furthermore, since these samples’ gradients can introduce nuisance into the training process, it is natural to reduce their importance.

- Samples in the sensitive region have lower confidence in the premise, which will introduce undecidability into the training process. As Clark’s completion try to handle the undecidability of negative predicates by utilising information explicitly present in the knowledge base. It is natural to reduce the information that is not explicitly present.

Reduce Implication Bias by Transformation

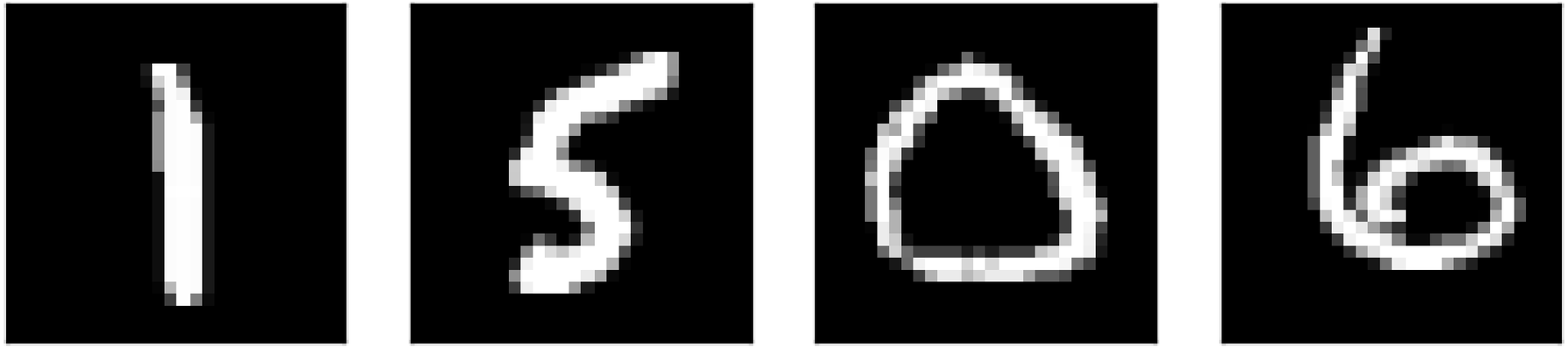
Let us use the notation $\mathcal{R}(\cdot)$ to represent a mapping that transforms a loss function into another loss function. Three different ways were designed to reduce the implication bias.

> Remark
> All these transformations are managing to reduce the sensitive region's impact on the training process. This approach will be very efficient as we only transform a biased logic loss through one-dimension mapping.

Empirical Study

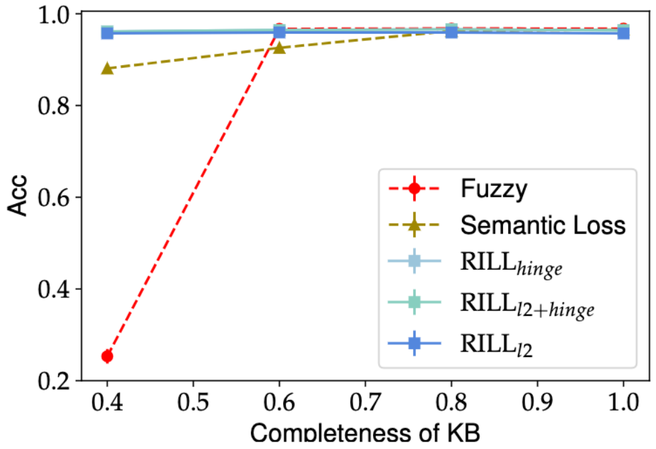
Just as in the real world, the lack of sufficient labelling data is a common occurrence, and so is the lack of a complete knowledge base. These situations leave training our model becoming more challenging. For this reason, two challenging situations **incomplete knowledge base** and **insufficient labelled data** will be discussed.

Task1: Addition Equations

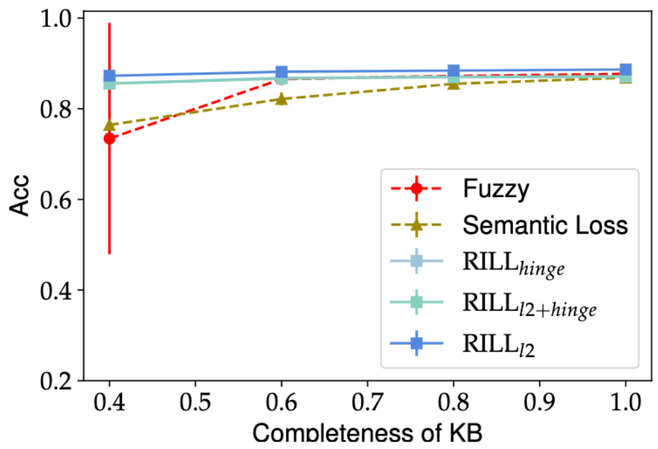


Task2: Hierarchical Classification

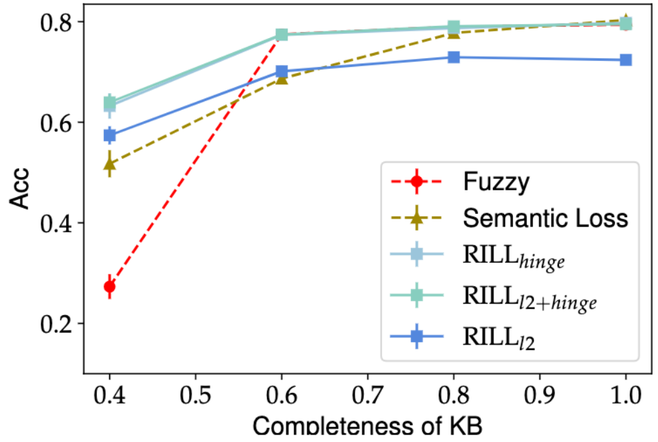
Incomplete Knowledge Base



(a) Add-MNIST



(b) Add-FashionMNIST



(c) Add-CIFAR10

Insufficient labelled data

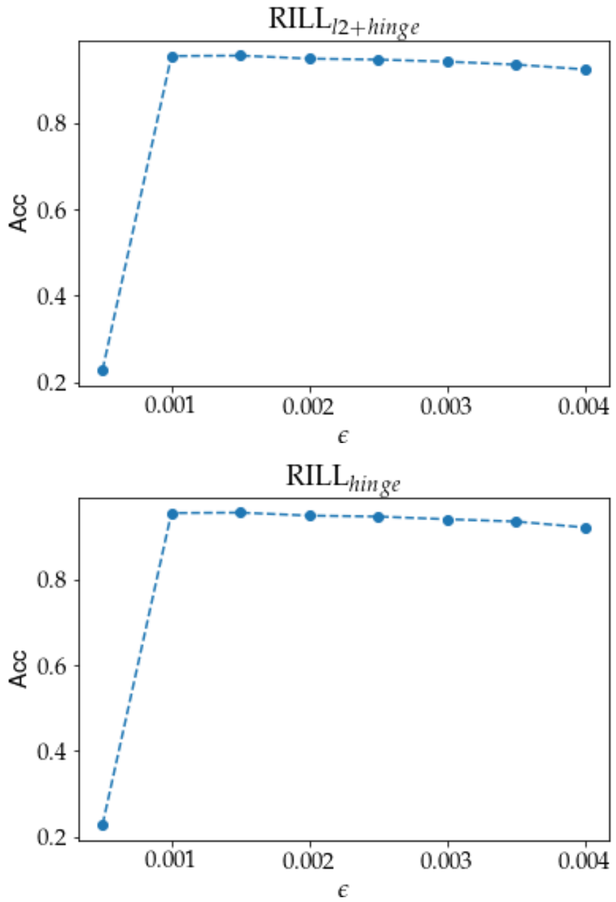
labelled data	8	4	2	1
Fuzzy	.19±.001	.19±.004	.19±.003	.19±.002
SL	.72±.006	.72±.007	.71±.029	.57±.227
Vanilia	.29±.016	.24±.025	.18±.015	.17±.012
RILL$_{l2}$	.81±.008	.82±.003	.82±.004	.83±.004
RILL$_{l2+hinge}$	**.85**±.003	**.86**±.004	.84±.010	**.86**±.005
RILL$_{hinge}$	**.85**±.006	.85±.007	**.85**±.005	**.86**±.004

Table 1: Task1 Add-CIFAR-10, accuracies that change the size of the labelled dataset from eight per class to one per class. The process of getting this table is similar to Tab.2.

labelled data	1		3		5		10	
	Acc	SC-Acc	Acc	SC-Acc	Acc	SC-Acc	Acc	SC-Acc
Fuzzy	.2291±.004	.7484±.014	.3246±.011	.7570±.020	**.3858**±.005	**.7661**±.012	.4449±.010	.7585±.014
SL	.2281±.007	.7384±.021	**.3301**±.011	**.7590**±.022	.3787±.011	.7502±.015	.4310±.034	.7462±.037
Vanilia	.2297±.007	.7415±.014	.3153±.015	.7333±.026	.3819±.007	.7630±.009	.4520±.004	.7670±.010
RILL$_{l2+hinge}$	**.2363**±.009	**.7575**±.004	**.3290**±.005	.7581±.007	.3843±.013	.7649±.016	.4509±.002	.7649±.008
RILL$_{l2}$	**.2357**±.010	.7449±.014	.3262±.010	**.7585**±.011	.3803±.012	.7593±.017	**.4538**±.003	.7699±.005
RILL$_{hinge}$	.2334±.006	.7463±.014	.3228±.012	.7520±.008	**.3851**±.004	**.7654**±.006	**.4562**±.004	**.7750**±.006

Table 2: Task2-Hierarchical-Classification accuracy that changes the labelled dataset size from ten per class to one per class. Experiments results (mean±std of Acc/SC-Acc) repeat for five times with seed $\{2020, 2021, 2022, 2023, 2024\}$. The boldfaces denote the best result in terms of Acc/SC-Acc, according to the Wilcoxon test at the significance level of 5%.

Sensitivity Analysis



RILL$_{hinge}$, RILL$_{l2+hinge}$ both contain a hyper-parameter ϵ in their transformation. In this section, the sensitivity of ϵ is revealed. The relation between ϵ and the model's accuracy is shown in the picture, with the experiment of Add-MNIST when the incompleteness of the knowledge base equals 40%.

Conclusion

Limitations

The limitations of this work are listed in three folds.

- First, this paper only computes logic loss from each single logic rule of knowledge bases. It remains future work to consider the more complex interaction ways between different logic rules. It is promising to measure complex reasoning processes by loss functions.
- Second, this paper's analysis and formal definition may not be suitable for probabilistic logic-based loss functions. However, it seems complicated to use the probabilistic logic-based loss function for the high complexity of compiling logic circuits, which remains a hard problem.
- Third, discussion of the sensitive region is not enough. We do not give a formal definition of this region which remains future work. This topic is closely related to the field of Learning with Noisy Labels. Samples in this sensitive region introduce implication bias into the training process, which can be seen as the hard samples of learning with the noisy label. Nevertheless, how to find these samples still remains a key problem in this field.

Contributions

- We analyse the phenomenon of implication bias caused by differentiable implication operators and point out what kinds of loss functions will suffer from this bias.
- We propose a simple yet effective method that can reduce implication bias named Reduced Implication-bias Logic Loss (RILL). Empirical study shows that RILL can achieve significant improvements compared with other forms of logic loss, especially when the knowledge base is incomplete or labelled data is insufficient.

Q & A