Supplement to Chapter 6: 
LNN: Logical Neural Networks

Supplement to Chapter 6 from Neuro Symbolic Reasoning and Learning - Current Advances and Future Directions

Additional Discussion

Interpreting Parameters. Note that, unlike PSL/NeurPSL, which assigns weights to rules [13] (which can be thought of as a single weight for an operator), LNN’s assign weights to the inputs of an operator. These inputs are associated with the lower-level logical constructs (i.e., atoms for subformulas). As a result, learned weights can be viewed as a level of importance for those components. The weights are not required to sum to 1, so they should be viewed in terms of “absolute” importance, as opposed to relative importance. The bias ($\beta$), which is associated with each logical construct, is somewhat less interpretable. While the designers of LNN recommend a common setting for the bias (e.g., $\beta = 1$ as mentioned earlier), they also view it as the “difficulty” involved in satisfying a given logical formula.

In the context of neuro symbolic reasoning, the concept of weighting arguments in a logical construct is singular to LNN’s. However, the novelty in the concept with respect to neuro symbolic reasoning is not the weighting itself, but the fact that the weights are learned through gradient descent. The idea had appeared earlier primarily in the context of developing aggregators for use in database queries (e.g., [3]).

Perhaps the greater question to raise is that is such a parameterization scheme viable as we seek to scale neuro symbolic methods? Questions such as should there be shared weights for various constructs, are parameterized operators leading to overfitting, and the relationship between weights for pure neural layers (e.g., lower level perceptual layers) and LNN layers will provide many topics for future research in the area of LNN’s.
Interpreting Fuzzy Outputs. With any fuzzy/real-valued logic framework (to include neurosymbolic approaches), interpreting the values associated with the atoms is an important issue. However, in the simplest case, a thresholding of the resulting confidence value can be determined with a typical machine learning validation approach. In other words, the target predicates (and associated ground atoms) can be treated as target classes and identifying a level of confidence (fuzzy value threshold) that provides the desired trade-off between precision and recall is a reasonable approach in many cases. Further, this allows for a direct comparison with standard black-box machine learning approaches. However, the introduction of the $\alpha$ hyper-parameter in LNN’s and the associated interpretation of the real values related to truth or falsehood (e.g., see Figure ?? add an additional layer of complexity. There are three reasons for this. First, the model is returning intervals rather than scalars. Second, there are pre-defined areas of truth and falsehood, and the resulting intervals may be subsumed by one of these levels and/or intersect with one or both levels. Understanding best practices around setting $\alpha$ prior to training as well as establishing proper thresholds for decision making in validation is important in the application of LNN’s to real-world systems. This is especially in terms of modularity - as it will be important to understand the semantics of the real-valued intervals provided for target atoms when integrating into other AI platforms.

References