Neural Inductive Logic Programming

Architecture allows us to induce rules of predefined structure

We can, for instance, incorporate the inductive bias of a transitivity relationship in the knowledge base

\[ \theta_1(X, Y) : \theta_2(X, Z), \theta_3(Z, Y). \]

\( \theta \) are vector representations for unknown predicates

They can be learned like all other vector representations

They can be decoded at test time by finding the closest known relation using the RBF kernel

Rule confidence is minimum RBF similarity over all decodings

Confidence is an upper bound on the proof success that can be achieved when applying the induced rule

Recursion

Iterate through all rules in the knowledge base and unify goal with rule heads

1. and(\( G, d, S \) = \( S' \mid S' \in and(\( G, d, S' \) for \( H - B \in |G| \))

Recursively prove subgoals in rule body

1. and(\( d, B \) - \( 0 \), \( F \) = \( F \))

3. and(\( d, S \) = \( S \))

4. and(\( G, d, S \) = \( S' \mid S' \in and(\( G, d, S' \) for \( S' \in \text{substitute}(G, S_0), d = 1, S_0 \))

Training Objective

Loss: \( L(r(e_i, e_j), y) = -y \log(NTP(r(e_i, e_j), 1-y) \log(1-NTP(r(e_i, e_j))) \)

\( \text{NTP} \) variant: SotA neural link prediction model (ComplEx) as auxiliary task

Limitations and Future Work

Scale to larger knowledge bases (beyond 10k facts)

Hierarchical attention for unification with facts

Reinforcement learning for pruning proof tree

Train jointly with RNNs that encode natural language statements which can then be used in proofs

Learn to prove mathematical theorems

Incorporate commonsense knowledge for Visual Q&A