SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver
Po-Wei Wang 1, Priya L. Donti 1, Bryan Wilder 2, J. Zico Kolter 1,3
1School of Computer Science, Carnegie Mellon University 2School of Engineering and Applied Sciences, Harvard University 3Bosch Center for AI

Summary

Logical reasoning within DL architectures has been a major goal of modern AI. We provide a layer that enables end-to-end learning of the structure and solutions of logic problems within deep networks.

This work is about

A layer that enables end-to-end learning of both the constraints and solutions of logic problems within deep networks. A smoothed differentiable (maximum) satisfiability solver that can be integrated into the deep learning system.

This work is not about

Not about learning to find SAT solutions [Selman et al., 1992]
Not about learning to solve logic problems from examples - we have another paper at AAAI comparing to the state-of-the-art MAXSAT solvers when the rules are given.
Not about using DL and SAT in a multi-staged manner - doing so requires prior knowledge on the structure and constraints
-further, current SAT solvers cannot accept probability inputs.

MAXSAT Problem

MAXSAT is the optimization variant of SAT solving
SAT: Find false v_i, v_j, v_k ∨ v_k' ∈ (v_i ∨ v_k') A (v_i ∨ v_k') MAXSAT: maximize # of unsatisfiable clauses
Relax the binary variables to smooth & continuous surfaces
v_i ∈ {+-1}, v_i ∈ Rŋ, i ∈ [1, . . . , n], v_i ∈ R^n

Satisfiability learning (Goemans-Williamson, 1995): X i∈V ⊥ Y i∈V minimize (S(S,X,Y) i∈V, s.t. X i∈V ⊥ Y i∈V = 1, s ∈ Z)

Semidefinite relaxation (Goemans-Williamson, 1995): X i∈V minimize (S(S,X,Y) i∈V, s.t. X i∈V ⊥ Y i∈V = 1, s ∈ Z)

The core part is still a block coordinate method, solving the linear system from the implicit function differentiation:
\[
\nabla \Pi_{i,j} = \frac{1}{2} \left( \sum_{k \neq i} \left( \frac{\partial \Pi}{\partial v_k} \right) \right) + \left( \frac{\partial \Pi}{\partial v_i} \right)
\]

Provable bounds on the variance of the gradient:
\[
\frac{\text{Var}(\nabla \Pi_{i,j})}{\text{Var}(\nabla \Pi_{i,j}) + \text{Var}(\nabla \Pi_{i,j})} \leq \frac{1}{2}
\]

This result is obtained by studying the joint distribution of random variables.

Other ingredients in SATNet

Low-rank regularization on \( \mathbf{A} \)
- Double exponentially many possible feature functions
- Low-rank \( \mathbf{W} \) regularizes the number of clauses
- Auxiliary variable (hidden nodes)
- Only SDP with dual-optimal constraints, limiting representation
- Adding auxiliary variable (augment) increases representation power

Inside SATNet: the forward pass

The forward pass is responsible for relaxing/rounding the probability inputs/outputs and solving the low-rank MAXSAT SDP

procedure FORWARD(\( \mathbf{Z}_j \), \( \mathbf{Z}_k \))
1. compute \( \mathbf{x}_j \) from \( \mathbf{Z}_j \) via \( \mathbf{x}_j = \text{sign}(\mathbf{w}_j^T \mathbf{z}_j + \mathbf{b}_j) \)
2. compute \( \mathbf{y}_j \) from \( \mathbf{y}_j \) via block coordinate descent
3. compute \( \mathbf{Z}_j \) from \( \mathbf{Z}_j \) via SDP \( \mathbf{Z}_j = \mathbf{Z}_j^* \)

The core part is still a block coordinate descent method, which produces
\[
v_i = \text{normal}(\mathbf{Z}_i - \mathbf{x}_i \mathbf{x}_i^T, \sigma_i^2), \quad \text{for } i = 1, \ldots, m
\]

By maintaining \( \mathbf{D} = \mathbf{I} - \mathbf{Z} \), complexity of CD reduces from \( O(n^2m^2) \) to \( O(mn^2) \).

Inside SATNet: the backward pass

The backward pass is responsible for computing the gradient and solving the implicit function derivatives of the fixed-point equation.

procedure BACKWARD(\( \mathbf{Z}_j \), \( \mathbf{Z}_k \))
1. compute \( \mathbf{x}_j \) from \( \mathbf{Z}_j \) via \( \mathbf{x}_j = \text{sign}(\mathbf{w}_j^T \mathbf{z}_j + \mathbf{b}_j) \)
2. compute \( \mathbf{y}_j \) via block coordinate descent
3. compute \( \mathbf{Z}_j \) from \( \mathbf{Z}_j \) via SDP \( \mathbf{Z}_j = \mathbf{Z}_j^* \)
4. compute \( \mathbf{y}_j = -\left( \left( \frac{\partial \Pi}{\partial \mathbf{w}_j} \right) \right)^T \mathbf{Z}_j \)

The core part is still a block coordinate method, solving the linear system from the implicit function differentiation:
\[
\mathbf{u}_j = \left( 1 - \alpha \right) (\mathbf{U} \mathbf{y}_j) - \alpha \left( \mathbf{U} \mathbf{x}_j \mathbf{x}_j^T \right) \mathbf{y}_j = -\left( \mathbf{u}_j \right)
\]

By maintaining \( \mathbf{U} = \mathbf{I} - \mathbf{Z} \), complexity of CD is again \( O(mn^2) \).

Similarity to Boltzmann machine

SATNet can be considered as an SO(2) approximation of Boltzmann machine
Both trying to minimize the energy
\[
E = \sum_{v_i \in \mathcal{V}} \omega_i v_i - \sum_{\mathcal{E} \in \mathcal{E}} \mathcal{E}(v_i) v_i
\]
Boltzmann minimize by sampling (simulated annealing) in the discrete space \( v_i \in \{-1, 1\} \)
but SATNet optimize by relaxing binary variables to a smoother higher-dimension space in \( \mathbb{R}^n \)
That is, the internal optimization of SATNet is easier, thus much faster than Boltzmann machine.

Illustration: Learning Parity from data

Parity function is hard for deep networks to learn [Shalev-Shwartz et al., 2017]. For a sequence of length 40 with single-bit supervision only from the end, the recurrently trained SATNet-based network can learn the parity function easily from 16k examples while LSTM will stack.

Illustration: Learning Sudoku from data

Instead of simply solving the endgame (which is easy), the SATNet can learn both the constraints and solution of Sudokus solely from unsolved and solved puzzles without any structure information from only 1k examples. Here, the parameter of the SATNet is the clause matrix \( \mathbf{S} \), which is randomly initialized. The SATNet achieves almost perfect accuracy in both training and testing set. Note that we are reported that after code release, the performance of ConvNet might increase given 1M data and more epochs.

Illustration: MNIST Sudoku

We also form a MNIST version of Sudoku to test the end-to-end learning ability. To do so, we first apply the ConvNet to each digit, feeding the output to a SATNet layer, connecting it to a loss, and train them in an end-to-end fashion.

Illustration: SATNet vs. LSTM

SATNet can beat the LSTM in terms of accuracy, training time, and test time. The LSTM still stuck, while the SATNet finish quickly.

Illustration: SATNet vs. ConvNet

SATNet can beat ConvNet in terms of accuracy, training time, and test time. The ConvNet still stuck, while the SATNet finish quickly.

Illustration: SDP-Based SATNet

SATNet can be considered as an SO(2) approximation of Boltzmann machine
Both trying to minimize the energy
\[
E = \sum_{v_i \in \mathcal{V}} \omega_i v_i - \sum_{\mathcal{E} \in \mathcal{E}} \mathcal{E}(v_i) v_i
\]
Boltzmann minimize by sampling (simulated annealing) in the discrete space \( v_i \in \{-1, 1\} \)
but SATNet optimize by relaxing binary variables to a smoother higher-dimension space in \( \mathbb{R}^n \)
That is, the internal optimization of SATNet is easier, thus much faster than Boltzmann machine.

Illustration: SATNet vs. LSTM

SATNet can beat the LSTM in terms of accuracy, training time, and test time. The LSTM still stuck, while the SATNet finish quickly.