SATNet : Bridging deep learning and logical reasoning using a differentiable satisfiability solver

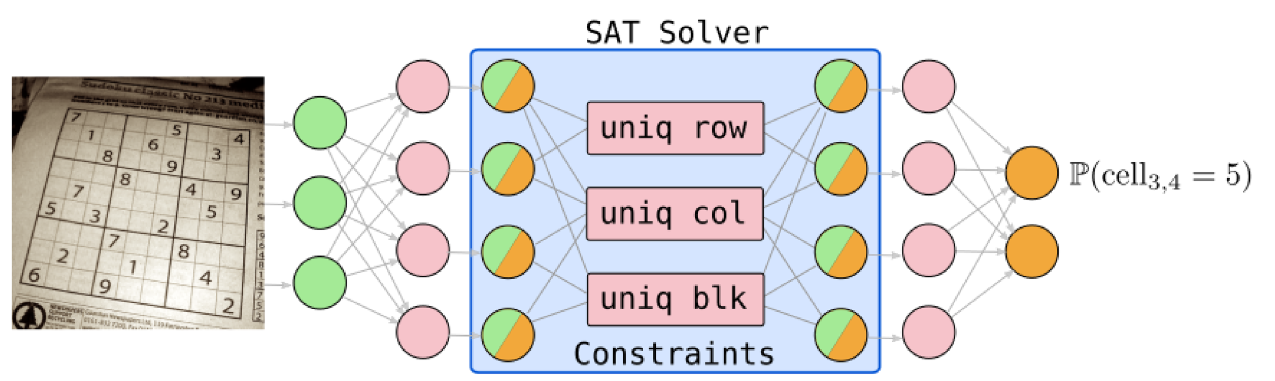
Po-Wei Wang [1] Priya L. Donti [1] Bryan Wilder [2] J. Zico Kolter [1,3]

[1]School of Computer Science, Carnegie Mellon University [2]School of Engineering and Applied Sciences, Harvard University [3]Bosch Center for AI

Code available at: github.com/locuslab/SATNet

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 loop of deep learning systems.

This work is not about …

Not about learning to find SAT solutions [Selsam et al. 2019]

- but about learning both **constraints** and **solution** 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 feasible v_i s.t. $v_2 \wedge (v_1 \vee \neg v_2) \wedge (v_2 \vee \neg v_3)$

MAXSAT: maximize # of satisfiable clauses

Relax the binary variables to smooth & continuous spheres

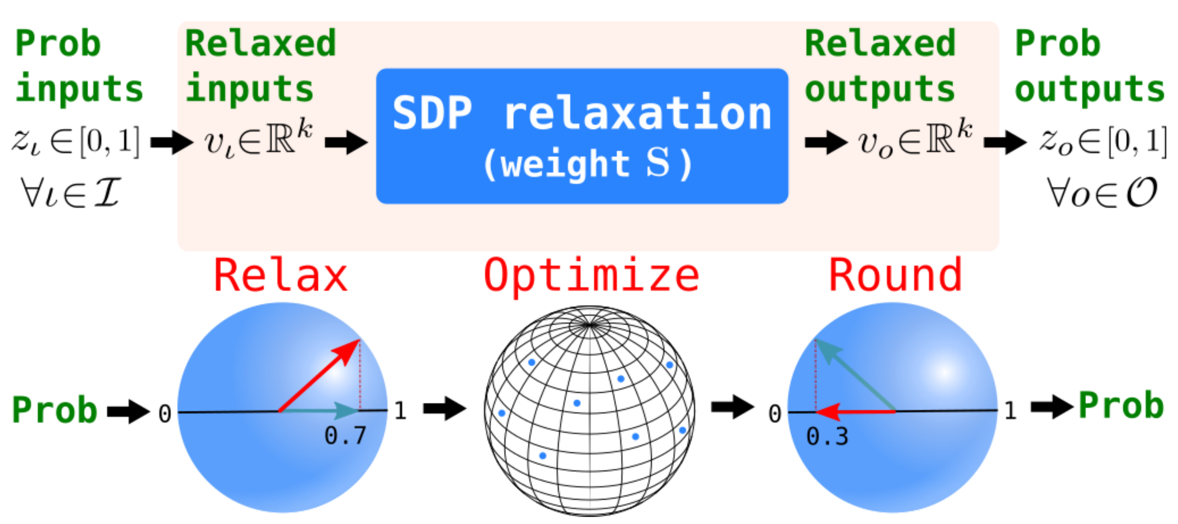
$$v_i \in \{+1, -1\} \xrightarrow{equiv} |v_i| = 1,\ v_i \in \mathbb{R}^1 \xrightarrow{relax} \|v_i\| = 1,\ v_i \in \mathbb{R}^k$$

Semidefinite relaxation (Goemans-Williamson, 1995), $X = V^T V$

$$\text{minimize } \langle S^T S, X \rangle, \quad \text{s.t. } X \succeq 0,\ \mathrm{diag}(X) = 1$$

where $s_{ij} \in \{-1, 0, +1\}$ denotes sign of ith variable in jth clause.

SATNet: MAXSAT SDP as a layer

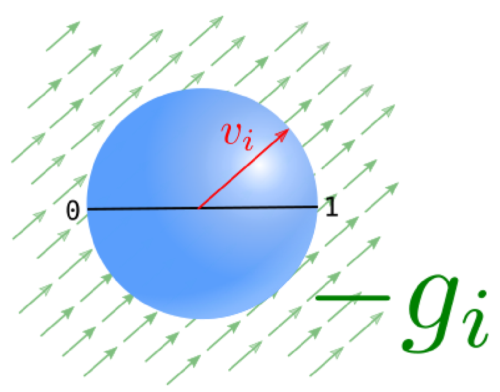


Fast solution to MAXSAT SDP approximation

Efficiently solve via low-rank factorization (Burer-Monteiro method): $X = V^T V$, $V \in \mathbb{R}^{k \times n}$, $\|v_i\| = 1$. Block coordinate descent updates:

$$v_i = -\mathrm{normalize}(V S^T s_i - \|s_i\|^2 v_i).$$

Derivation of the update



The low-rank optimization problem is

$$\text{minimize}_V \ \langle S^T S, V^T V \rangle, \text{ s.t. } \|v_i\| = 1,\ \forall i$$

Focusing on the subproblem w.r.t. a specific v_i:

$$\text{minimize}_{v_i} \ g_i^T v_i, \text{ s.t. } \|v_i\| = 1,$$

where $g_i = V S^T s_i - \|s_i\|^2 v_i$ is a constant.

Closed-form solution $v_i = -\mathrm{normalize}(g_i)$.

Theoretical analysis

- For $k > \sqrt{2n}$, the non-convex iterates are guaranteed to converge to global optima of SDP [Wang et al., 2018; Erdogdu et al., 2018]
- Complexity reduced from $O(n^6 \log\log\frac{1}{\epsilon})$ of interior point methods to $O(n^{1.5} m \log\frac{1}{\epsilon})$ of our coordinate descent method, where m is #clauses.

Global Convergence proof Illustration

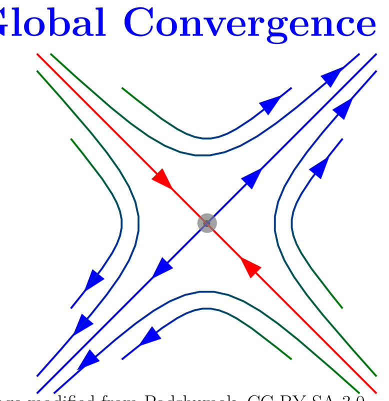
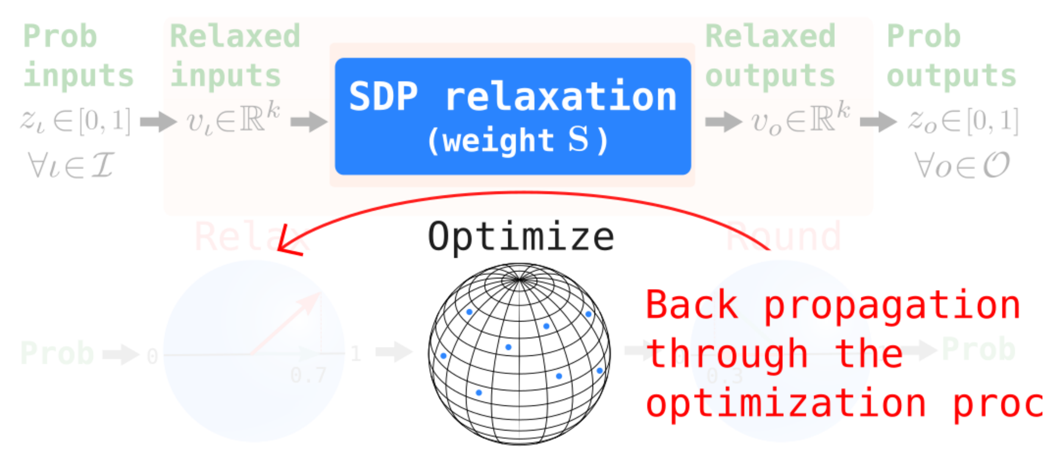


Image modified from Podshumok, CC BY-SA 3.0

We proved all the non-optimal critical points are unstable because their Jacobian admits a spectral radius greater than 1. Then, the stable manifold theorem in control theory implies after random initialization, the method has zero possibility converging to non-opts, thus almost surely converges to the global optimum.

Differentiate through the optimization problem



When converged, the block coordinate descent satisfies the fixed-point equation

$$v_i = -\mathrm{normalize}(V S^T s_i - \|s_i\|^2 v_i),\ \forall i$$

The **fixed point equation** of the block coordinate descent provides an implicit function definition of the solution [Amos et al. 2017]

$$F_i(S, V(S)) = v_i + \mathrm{normalize}(V S^T s_i - \|s_i\|^2 v_i) = 0,\ \forall i$$

Thus, can apply implicit function theorem on the total derivatives

$$\frac{\partial \vec{F}(\vec{S}, \vec{V}(S))}{\partial \vec{S}} = 0 \implies \frac{\partial \vec{F}(\vec{S}, \vec{V})}{\partial \vec{S}} + \frac{\partial \vec{F}(\vec{S}, \vec{V})}{\partial \vec{V}} \cdot \frac{\partial \vec{V}}{\partial \vec{S}} = 0$$

Solve the above **linear system** of $\partial \vec{V} / \partial \vec{S}$ to backprop

Other ingredients in SATNet

Low-rank regularization on S

- Doubly-exponentially many possible Boolean functions !
- Low-rank $\Rightarrow$ Regularize the complexity through number of clauses

Auxiliary variable (hidden nodes)

- Only SDP with diagonal constraints, limiting representation
- Adding auxiliary variable (gadget) 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 ($Z_\mathcal{I}$)

1: **compute** $V_\mathcal{I}$ **from** $Z_\mathcal{I}$ **via** $v_\iota = -\cos(\pi z_\iota) v_\top + \sin(\pi z_\iota)(I_k - v_\top v_\top^T) v_\iota^{\text{rand}}$

2: **compute** $V_\mathcal{O}$ **from** $V_\mathcal{I}$ **via** block coordinate descent

3: **compute** $Z_\mathcal{O}$ **from** $V_\mathcal{O}$ **via** $P(\tilde{v}_o) = \cos^{-1}(-v_o^T v_\top)/\pi$

The core part is a blocking coordinate descent method, which performs

$$v_i = -\mathrm{normalize}(V S^T s_i - \|s_i\|^2 v_i), \quad \text{for } i = 1 \ldots n.$$

By maintaining $\Omega = V S^T$, complexity of CD reduces from $O(n^3)$ to $O(n^{1.5} m)$.

Inside SATNet: the backward pass

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

procedure BACKWARD ($\partial \ell / \partial Z_\mathcal{O}$)

1: **compute** $\partial \ell / \partial V_\mathcal{O}$ from $\frac{\partial \ell}{\partial v_o} = \left(\frac{\partial \ell}{\partial z_o}\right) \frac{1}{\pi \sin(\pi z_o)} v_\top$,

2: **compute** U **from** $\partial \ell / \partial V_\mathcal{O}$ **via** block coordinate descent

3: **compute** $\partial \ell / \partial Z_\mathcal{I} = \frac{\partial \ell}{\partial z_\iota^\star} - \left(\frac{\partial v_\iota}{\partial z_\iota}\right)^T \left(\sum_{o \in \mathcal{O}} u_o s_o^T\right) s_\iota$

4: **compute** $\partial \ell / \partial S = -\left(\sum_{o \in \mathcal{O}} u_o s_o^T\right)^T V - (S V^T) U$

The core part is still a block coordinate method, solving the linear system from the implicit function differentiation:

$$u_i = -\left(\frac{I - v_i v_i^T}{\|g_i\|}\right)(U S^T s_i - \|s_i\|^2 u_i - \partial l / \partial v_i), \quad \text{for } i = 1 \ldots n.$$

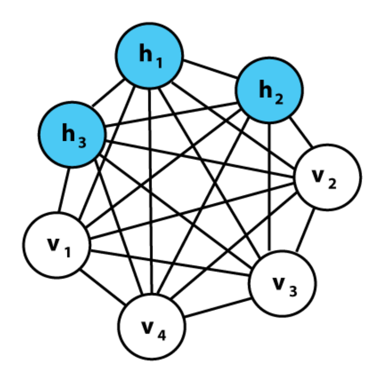
By maintaining $\Psi = U S^T$, complexity of CD is again $O(n^{1.5} m)$.

Similarity to Boltzmann machine

SATNet can be consider as an SOS(2) approximation of Boltzmann machine

Both tries to minimizing the energy

$$E = -\sum_{ij} w_{ij} v_i v_j, \text{ where } w_{ij} = -s_i^T s_j$$



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 sphere in $\mathbb{R}^k$.

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-Swartz et al., 2017]. For a sequence of length 40 with single-bit supervision only from the end, the recurrently chained SATNet-based network can learn the parity function easily form 10k examples while LSTM will stuck.

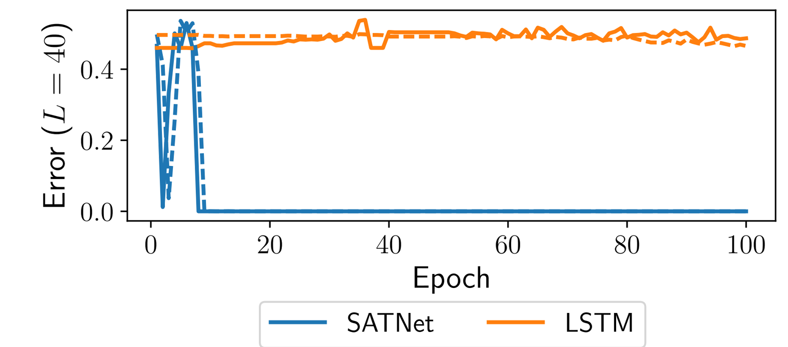
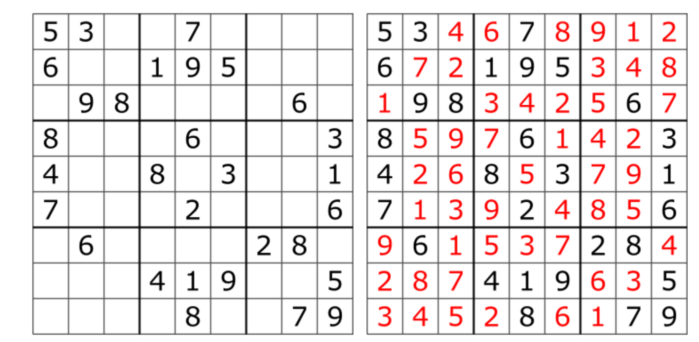


Illustration: Learning Sudoku from data

Instead of simply solving the sudoku (which is easy), the SATNet can learn both the constraints and solution of Sudoku solely from unsolved and solved puzzles **without any structure information** from only 9K examples. Here, the parameter of the SATNet is the clause matrix 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.



Model	Train	Test
ConvNet	72.6%	0.04%
ConvNetMask	91.4%	15.1%
SATNet (ours)	99.8%	**98.3%**

Original Sudoku

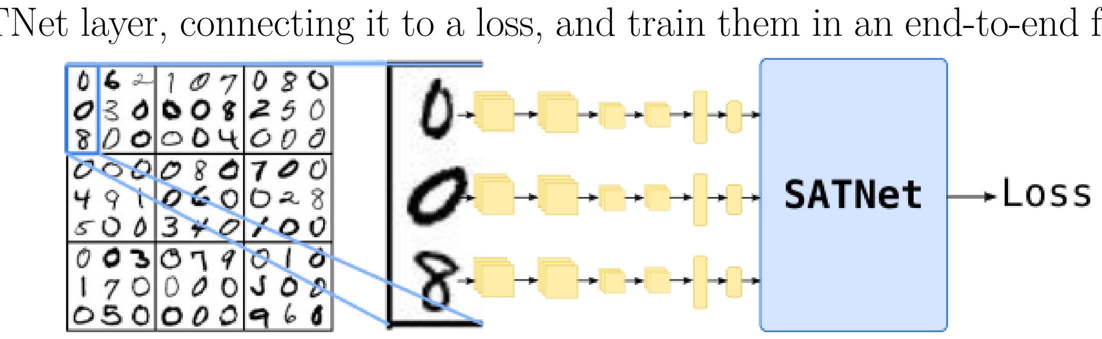
Shall we permute the Sudoku problem to remove all the structure information and locality, SATNet still learns the logic, but ConvNet fails completely.

Model	Train	Test
ConvNet	0%	0%
ConvNetMask	0.01%	0%
SATNet (ours)	99.7%	**98.3%**

Permuted Sudoku

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 digits, feeding the output to a SATNet layer, connecting it to a loss, and train them in an end-to-end fashion.



Model	Train	Test
ConvNet	0.31%	0%
ConvNetMask	89%	0.1%
SATNet (ours)	93.6%	**63.2%**

- Getting example "correct" requires correct Sudoku solution *and* predicting all MNIST test digits correctly
- 85% accuracy on correct ConvNet input

[1] B. Amos and J. Z. Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR, 2017.

[2] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[3] M. Shub. *Global stability of dynamical systems*. Springer Science & Business Media, 2013.