Machine learning for optimization (3/5)

Clément W. Royer

M2 MODO - 2025/2026

December 8, 2026

Today's session

An example (not for a lab session)

- Continuous relaxations of SAT problems.
- Differentiable solvers (Wang et al '19).

A more toy example (easier for a lab session)

- Learning solver hyperparameters.
- Unfolded ISTA (Ablin et al '19).

Roadmap

1. SATNet
2. Learning with unfolded solvers

Outline

1. SATNet
2. Learning with unfolded solvers

Context: Continuous relaxations of combinatorial problems

Classical example: MaxCut (Goemans, Williamson '95).

- Problem: Given graph (V, E) with weighted edges, find a cut with maximum edge weight.

Context: Continuous relaxations of combinatorial problems

Classical example: MaxCut (Goemans, Williamson '95).

- Problem: Given graph (V, E) with weighted edges, find a cut with maximum edge weight.
- Using graph Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$, can be formulated as

$$\underset{\boldsymbol{x} \in \{-1,1\}^n}{\text{maximize}} \, \boldsymbol{x}^{\mathrm{T}} \boldsymbol{L} \boldsymbol{x}.$$

Classical example: MaxCut (Goemans, Williamson '95).

- Problem: Given graph (V, E) with weighted edges, find a cut with maximum edge weight.
- Using graph Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$, can be formulated as

$$\operatorname*{maximize}_{\boldsymbol{x} \in \{-1,1\}^n} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{L} \boldsymbol{x}.$$

- Equivalent to the continuous program

$$\operatorname*{maximize}_{\boldsymbol{X} \in \mathcal{S}^{n \times n}} \operatorname{trace}(\boldsymbol{L}^{\mathrm{T}} \boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{X}_{ii} = 1, \boldsymbol{X} \succeq \boldsymbol{0}, \operatorname{rank}(\boldsymbol{X}) = 1.$$

Context: Continuous relaxations of combinatorial problems

Classical example: MaxCut (Goemans, Williamson '95).

- Problem: Given graph (V, E) with weighted edges, find a cut with maximum edge weight.
- Using graph Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$, can be formulated as

$$\operatorname*{maximize}_{\boldsymbol{x} \in \{-1,1\}^n} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{L} \boldsymbol{x}.$$

- Equivalent to the continuous program

$$\operatorname*{maximize}_{\boldsymbol{X} \in \mathcal{S}^{n \times n}} \mathrm{trace}(\boldsymbol{L}^{\mathrm{T}} \boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{X}_{ii} = 1, \boldsymbol{X} \succeq \mathbf{0}, \mathrm{rank}(\boldsymbol{X}) = 1.$$

- Remove rank constraint: Get the SDP relaxation!

$$\operatorname*{maximize}_{\boldsymbol{X} \in \mathcal{S}^{n \times n}} \mathrm{trace}(\boldsymbol{L}^{\mathrm{T}} \boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{X}_{ii} = 1, \boldsymbol{X} \succeq \mathbf{0}.$$

From the relaxation to a solution

1. Solve Max-Cut SDP $\Rightarrow$ Solution $\boldsymbol{X}^* \succeq \mathbf{0}$, $\boldsymbol{X}^*_{ii} = 1$.

From the relaxation to a solution

1. Solve Max-Cut SDP $\Rightarrow$ Solution $\boldsymbol{X}^* \succeq \mathbf{0}$, $\boldsymbol{X}^*_{ii} = 1$.
2. Write $\boldsymbol{X}^* = [\boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{v}_j]$ with $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ unit vectors in $\mathbb{R}^n$.

From the relaxation to a solution

1. Solve Max-Cut SDP $\Rightarrow$ Solution $\boldsymbol{X}^* \succeq \mathbf{0}$, $\boldsymbol{X}^*_{ii} = 1$.
2. Write $\boldsymbol{X}^* = [\boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{v}_j]$ with $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ unit vectors in $\mathbb{R}^n$.
3. Draw $\boldsymbol{u}$ uniformly at random in the unit sphere, and set

$$\forall i = 1, \dots, n, \qquad x_i^* = \begin{cases} -1 & \text{if } \boldsymbol{u}^{\mathrm{T}} \boldsymbol{v}_i \le 0 \\ 1 & \text{if } \boldsymbol{u}^{\mathrm{T}} \boldsymbol{v}_i > 0. \end{cases}$$

From the relaxation to a solution

1. Solve Max-Cut SDP $\Rightarrow$ Solution $\boldsymbol{X}^* \succeq \mathbf{0}$, $\boldsymbol{X}^*_{ii} = 1$.
2. Write $\boldsymbol{X}^* = [\boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{v}_j]$ with $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ unit vectors in $\mathbb{R}^n$.
3. Draw $\boldsymbol{u}$ uniformly at random in the unit sphere, and set

$$\forall i = 1, \dots, n, \qquad x_i^* = \begin{cases} -1 & \text{if } \boldsymbol{u}^{\mathrm{T}} \boldsymbol{v}_i \le 0 \\ 1 & \text{if } \boldsymbol{u}^{\mathrm{T}} \boldsymbol{v}_i > 0. \end{cases}$$

Guarantees

- Randomized rounding above finds an 0.87856-approximation!
- Similar guarantees can be obtained for other problems, such as MAXSAT.
- Challenge: SDPs are difficult to solve at scale.

MaxSAT case

MAXSAT problem

Given m vectors $\{\tilde{\boldsymbol{s}}_i\} \subset \{-1,0,1\}^m$, solve

$$\underset{\tilde{\boldsymbol{v}} \in \{-1,1\}^n}{\text{maximize}} \sum_{j=1}^{m} \bigvee_{i=1}^{n} \mathbf{1}\left\{\tilde{s}_{ij}\tilde{v}_i > 0\right\}$$

MaxSAT case

MAXSAT problem

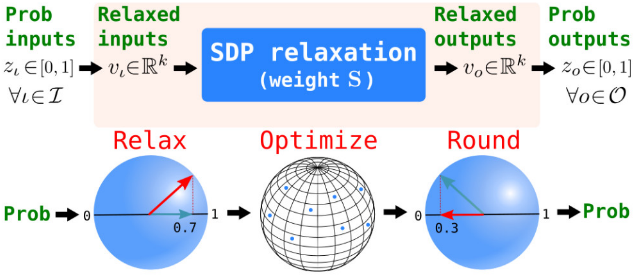
Given m vectors $\{\tilde{\boldsymbol{s}}_i\} \subset \{-1,0,1\}^m$, solve

$$\operatorname*{maximize}_{\tilde{\boldsymbol{v}} \in \{-1,1\}^n} \sum_{j=1}^{m} \bigvee_{i=1}^{n} \mathbf{1}\left\{\tilde{s}_{ij}\tilde{v}_i > 0\right\}$$

Continuous SDP relaxation

$$\operatorname*{minimize}_{\boldsymbol{V} \in \mathbb{R}^{k \times (n+1)}} \left\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}, \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} \right\rangle \quad \text{s.t.} \quad \|\boldsymbol{v}_i\| = 1 \forall i = 1, \dots, n+1.$$

- Relax $\tilde{v}_i$ into $\boldsymbol{v}_i \in \mathbb{R}^k$, $\|\boldsymbol{v}_i\| = 1$.
- Add a variable $\boldsymbol{v}_0$ to apply randomized rounding.
- $\boldsymbol{S}$ built from the $\tilde{\boldsymbol{s}}_i$ with scaling.
- If $k > \sqrt{2n}$, recovers the original solution.



Optimization solver

- Use vector representation of SDP matrix.
- Cheap update, one vector at a time.
- Amenable to batch parallelism.
- Can differentiate through the solver!

SDP layer

- Careful encoding of backpropagation.
- Continuous relaxation and randomized rounding encoded through probability distributions.

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

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

(a) Original Sudoku.

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

(b) Permuted Sudoku.

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

(c) Visual Sudoku. (Note: the theoretical "best" test accuracy for our architecture is 74.7%.)

Table 1. Results for 9×9 Sudoku experiments with 9K train/1K test examples. We compare our SATNet model against a vanilla convolutional neural network (ConvNet) as well as one that receives a binary mask indicating which bits need to be learned (ConvNetMask).

- Setup: Learn rules and fill out Sudoku grids, represented as vectors.
- Convolutional networks treat grids as images, must learn the masked bits.
- Permuting the inputs does not change the rules to learn$\Rightarrow$Clear advantage of SATNet.

Outline

1. SATNet

2. Learning with unfolded solvers

Problem

Goal: Sparse representation/coding

$$\operatorname*{minimize}_{\boldsymbol{z}\in\mathbb{R}^m} f(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{z}\|^2 + \lambda\|\boldsymbol{z}\|_1.$$

- $\boldsymbol{x} \in \mathbb{R}^n$: Target vector.
- $\boldsymbol{D} \in \mathbb{R}^{n\times m}$: Dictionary.
- $\|\boldsymbol{z}\|_1 = \sum_{j=1}^d |z_i|$: Promotes sparsity of $\boldsymbol{z}$.

An algorithm: ISTA

Problem minimize$_{\boldsymbol{z}\in\mathbb{R}^m}$ $f(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{z}\|^2 + \lambda\|\boldsymbol{z}\|_1$..

ISTA iteration

$$\boldsymbol{z}_{k+1} = \mathrm{ST}(\boldsymbol{z}_k - \alpha_k \boldsymbol{D}^{\mathrm{T}}(\boldsymbol{D}\boldsymbol{z}_k - \boldsymbol{x}); \lambda\alpha_k)$$

- Gradient descent step+ Soft-thresholding (ST) ($\equiv$ Proximal gradient)

$$\mathrm{ST}(t;\mu) = \begin{cases} t-\mu & \text{if } t > \mu \\ t+\mu & \text{if } t < \mu \\ 0 & \text{otherwise.} \end{cases}$$

An algorithm: ISTA

Problem minimize$_{\boldsymbol{z}\in\mathbb{R}^m}$ $f(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{z}\|^2 + \lambda\|\boldsymbol{z}\|_1$..

ISTA iteration

$$\boldsymbol{z}_{k+1} = \mathrm{ST}(\boldsymbol{z}_k - \alpha_k \boldsymbol{D}^{\mathrm{T}}(\boldsymbol{D}\boldsymbol{z}_k - \boldsymbol{x}); \lambda\alpha_k)$$

- Gradient descent step+ Soft-thresholding (ST) ($\equiv$ Proximal gradient)

$$\mathrm{ST}(t;\mu) = \begin{cases} t-\mu & \text{if } t > \mu \\ t+\mu & \text{if } t < \mu \\ 0 & \text{otherwise.} \end{cases}$$

- Key parameter: Stepsize α_k.
- Popular choice: $\alpha_k = \frac{1}{L}$ for every k, where $L = \|\boldsymbol{D}\|^2$ (Lipschitz constant for the gradient of $\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{z}\|^2$).

Unrolling ISTA

Neural net representation of ISTA

- One "layer" $= K$ iterations of ISTA starting from $\boldsymbol{z}_0$.
- Network parameters: $\alpha_0, \dots, \alpha_{K-1}$.
- Fixed parameters: $\boldsymbol{D}$, λ.

Neural net representation of ISTA

- One "layer" $= K$ iterations of ISTA starting from $\boldsymbol{z}_0$.
- Network parameters: $\alpha_0, \dots, \alpha_{K-1}$.
- Fixed parameters: $\boldsymbol{D}$, λ.

Training

- Forward: Feed $\boldsymbol{x}$, get $f(\boldsymbol{z}_K(\boldsymbol{x}))$ from explicit ISTA iterations.
- Backward: Automatic differentation (nothing to do!).
- Training data: Unsupervised, try to minimize

$$\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{z}_K(\boldsymbol{x}^i)).$$

References

- P. Ablin, T. Moreau, M. Massias, A. Gramfort, *Learning step sizes for unfolded sparse coding*, NeurIPS, 2019.
- M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 1995.
- P.-W. Wang, P. L. Donti, B. Wilder and J. Z. Kolter, *SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver*, ICML, 2019.