Machine learning for optimization (2/5)

Clément W. Royer

M2 MODO - 2024/2025

January 15, 2025

Today's session

- Paradigm: Integrate optimization solvers within an ML (deep learning) pipeline.

Today's session

- Paradigm: Integrate optimization solvers within an ML (deep learning) pipeline.
- Challenges:
  - Efficient implementation.
  - Compatibility between solver/deep learning environment.

Today's session

- Paradigm: Integrate optimization solvers within an ML (deep learning) pipeline.
- Challenges:
  - Efficient implementation.
  - Compatibility between solver/deep learning environment.
- Key concepts:
  - Implicit layers in neural networks.
  - Automatic differentiation/Backpropagation.

Roadmap

1. Crash course on neural networks
2. Implicit layer paradigm
3. Application: OptNet
4. Application: SATNet

Roadmap

Intro to neural networks and automatic differentiation

See online notes.

Roadmap

From explicit to implicit layers

Explicit layers

- A layer≡A differentiable parametric function.
- Most layers are explicitly defined by a mapping $\boldsymbol{z} = f(\boldsymbol{x})$
  Ex) Fully connected, convolutional, recurrent.

From explicit to implicit layers

Explicit layers

- A layer≡A differentiable parametric function.
- Most layers are explicitly defined by a mapping $\boldsymbol{z} = f(\boldsymbol{x})$
  Ex) Fully connected, convolutional, recurrent.

Implicit layers

- Defined in terms of a joint condition on the input and output:
  Given $\boldsymbol{x}$, find $\boldsymbol{z}$ such that $c(\boldsymbol{x}, \boldsymbol{z}) = \mathbf{0}$.
- Decouples the purpose of the network from its computation.

Roadmap

Setup

Broader: Differentiable convex optimization layers

$$
\begin{array}{rcll}
z^*(\theta) & = & \text{minimize}_z & f(z,\theta) \\
 & & \text{s.t.} & g(z,\theta) \leq \mathbf{0} \\
 & & & h(z,\theta) = \mathbf{0}.
\end{array}
$$

- Maps θ to $z^*(\theta)$.
- Convex objective function+Convex constraints.

Approach

- Use solvers that allow differentiation of z^* w.r.t. θ.
- Learn some of the problem/solver parameters using data.

OptNet (Amos and Kolter '17-'19)

Quadratic programming setup

$$\begin{array}{rl} \text{minimize}_{\boldsymbol{z}} & \frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x})\boldsymbol{z} + \boldsymbol{q}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{z} \\ \text{s.t.} & \boldsymbol{G}(\boldsymbol{x})\boldsymbol{z} \le \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{A}(\boldsymbol{x})\boldsymbol{z} = \boldsymbol{b}(\boldsymbol{x}). \end{array}$$

Layer perspective: Input $\boldsymbol{x}$, get $\boldsymbol{z}^*(\boldsymbol{x}) \in \text{argmin}_{\boldsymbol{z}} \{\cdots\}$ as output.

OptNet (Amos and Kolter '17-'19)

Quadratic programming setup

$$\begin{array}{rl} \text{minimize}_{\boldsymbol{z}} & \frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x})\boldsymbol{z} + \boldsymbol{q}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{z} \\ \text{s.t.} & \boldsymbol{G}(\boldsymbol{x})\boldsymbol{z} \le \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{A}(\boldsymbol{x})\boldsymbol{z} = \boldsymbol{b}(\boldsymbol{x}). \end{array}$$

Layer perspective: Input $\boldsymbol{x}$, get $\boldsymbol{z}^*(\boldsymbol{x}) \in \text{argmin}_{\boldsymbol{z}} \{\cdots\}$ as output.

Challenges

- Gurobi/CPLEX/etc solve a single problem efficiently but are harder to deploy in batches on GPU
  ⇒Ad hoc QP solver (scaling limitations).
- Solver must be differentiable
  ⇒Careful implementation of forward pass.
- Difference between input and trainable parameters.

Differentiation through a QP layer

QP: minimize$_{\boldsymbol{z}}$ $\frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{z} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{z}$ s.t. $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b},\ \boldsymbol{G}\boldsymbol{z} \le \boldsymbol{h}$.

KKT conditions

If $\boldsymbol{z}^*$ is a solution, there exist $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$ such that

$$
\begin{array}{rcll}
\boldsymbol{Q}\boldsymbol{z}^* + \boldsymbol{q} + \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\lambda}^* + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\mu}^* & = & \mathbf{0} & \\
\boldsymbol{\lambda}_i^* \left[\boldsymbol{G}\boldsymbol{z}^* - \boldsymbol{h}\right]_i & = & 0 & \forall i \\
\boldsymbol{A}\boldsymbol{z}^* - \boldsymbol{b} & = & \mathbf{0} & \\
\boldsymbol{G}\boldsymbol{z}^* - \boldsymbol{h} & \le & \mathbf{0} & \\
\boldsymbol{\lambda}^* & \ge & \mathbf{0}. &
\end{array}
$$

Solver (IPM type): Apply Newton's method to the first 3 equations!

Differentiation through a QP layer

QP: minimize$_{\boldsymbol{z}}$ $\frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{z} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{z}$ s.t. $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b},\ \boldsymbol{G}\boldsymbol{z} \le \boldsymbol{h}$.

Differentiation

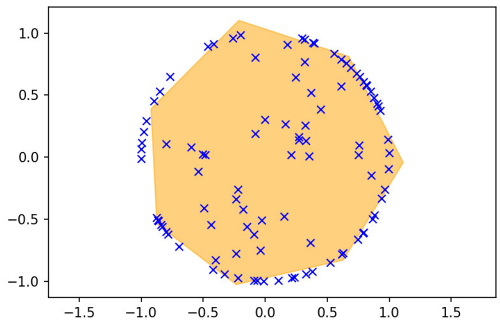
- Implicit function theorem applied to the first 3 KKT conditions.
- Example: For any function ℓ of $\boldsymbol{z}^*$,

$$\mathrm{D}_Q \ell = \frac{1}{2}\left(\boldsymbol{d}_z [\boldsymbol{z}^*]^{\mathrm{T}} + \boldsymbol{z}^*\, \boldsymbol{d}_z^{\mathrm{T}}\right)$$

where

$$\begin{bmatrix} \boldsymbol{d}_z \\ \boldsymbol{d}_\lambda \\ \boldsymbol{d}_\mu \end{bmatrix} = -\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{G}^{\mathrm{T}} \operatorname{diag}(\boldsymbol{\lambda}^*) & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{G} & \operatorname{diag}(\boldsymbol{G}\boldsymbol{z}^* - \boldsymbol{h}) & \boldsymbol{0} \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{D}_{\boldsymbol{z}^*} \ell^{\mathrm{T}} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}$$

Nice OptNet example



- Data $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1,\dots,100}$, $\boldsymbol{y}_i$ projection of $\boldsymbol{x}_i$ of some set.
- OptNet layer approximates $\boldsymbol{y}_i$ with projection onto polytope

$$\operatorname*{minimize}_{\boldsymbol{z}\in\mathbb{R}^n} \|\boldsymbol{z}-\boldsymbol{x}\|^2 \quad \text{s.t.} \quad \boldsymbol{G}\boldsymbol{z} \le \boldsymbol{h}.$$

- $\boldsymbol{G}, \boldsymbol{h}$ learned through training (40 epochs).

Roadmap

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

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

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

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

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

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

- Equivalent to the continuous program

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

- Remove rank constraint: Get the SDP relaxation!

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

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 ⇒ 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

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

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

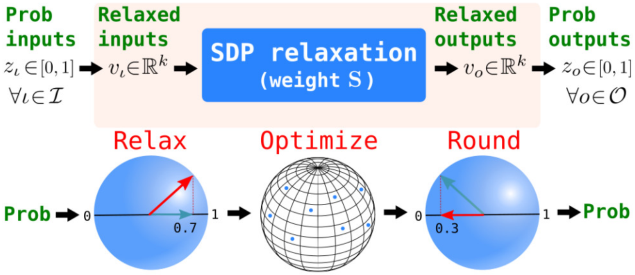
$$\operatorname*{maximize}_{\tilde{\boldsymbol{v}} \in \{-1,1\}^n} \sum_{j=1}^{m} \bigvee_{i=1}^{n} \mathbf{1}\left\{\tilde{s}_{ij}\tilde{v}_i > 0\right\}$$

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.

SATNet (Wang et al. '19)



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.

Cool SATNet example

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.

Summary: Implicit layers/Optimization solvers

Optimization solvers as a layer

- Implicit layer paradigm.
- Key: Allow for **automatic differentiation**.
- Benefit: Parallelism+integration within a neural architecture.

Key examples: Essentially convex optimization solves

- Quadratic programming layers (e.g. OptNet).
- Relaxation of combinatorial problems (e.g. SATNet).

References

- A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, J. Z. Kolter, *Differentiable convex optimization layers*, NeurIPS, 2019.
- B. Amos and J. Z. Kolter, *OptNet: Differentiable Optimization as a Layer in Neural Networks*, ICML, 2017.
- J. Djolonga and A. Krause, *Differentiable learning of submodular models*, NeurIPS, 2017.
- M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 1995.
- M. Hardt and B. Recht, *Patterns, Predictions and Actions: Foundations of Machine Learning*, Princeton University Press, 2022.
- J. Z. Kolter, D. Duvenaud and M. Johnson, *Deep Implicit Layers: Neural ODEs, Deep Equilibrium Models and beyond*, NeurIPS tutorial, 2020.
- 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.