A Solver-Free Learning Framework for ILP

Rishabh Ranjan, Yatin Nandwani, Mausam, Parag Singla

BTP Final Presentation – 8 January 2022

Introduction

Motivation

Task

Perception → Reasoning

Model

Neural → Symbolic

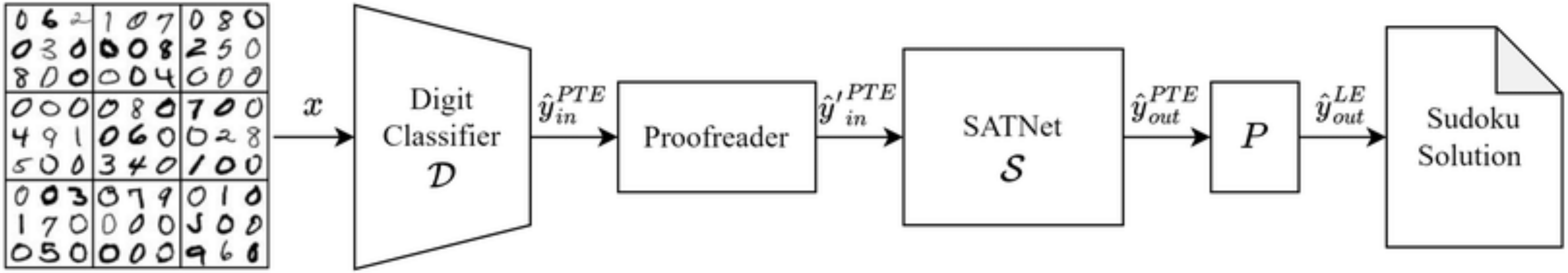


Figure from *Topan, Sever & Rolnick, David & Si, Xujie. (2021). Techniques for Symbol Grounding with SATNet.*

Integer Linear Programming

$x \rightarrow$ input

$A(x) \rightarrow$ constraint matrix

$b(x) \rightarrow$ bias vector

$c(x) \rightarrow$ cost vector

$y(A, b, c) \rightarrow$ solution vector / output

$n \rightarrow$ number of variables

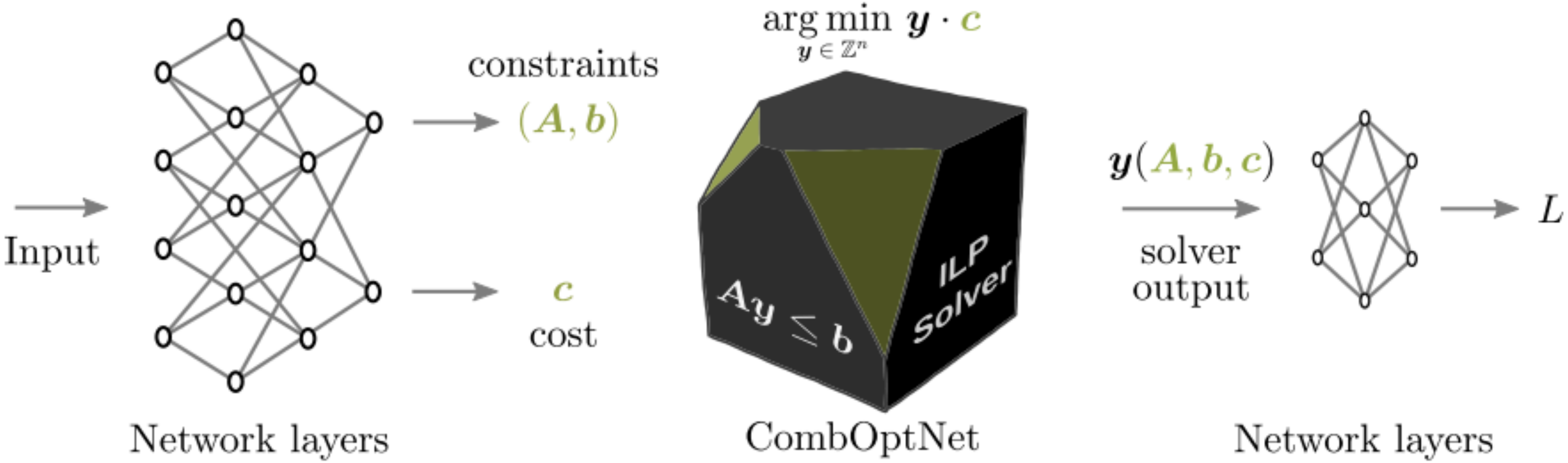
$\min c \cdot y$ such that $Ay + b \geq 0, y \in \mathbb{Z}^n$

Related Work 1: CombOptNet

CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints

Anselm Paulus, Michal Rolínek, Vít Musil, Brandon Amos, and Georg Martius

ICML 2021

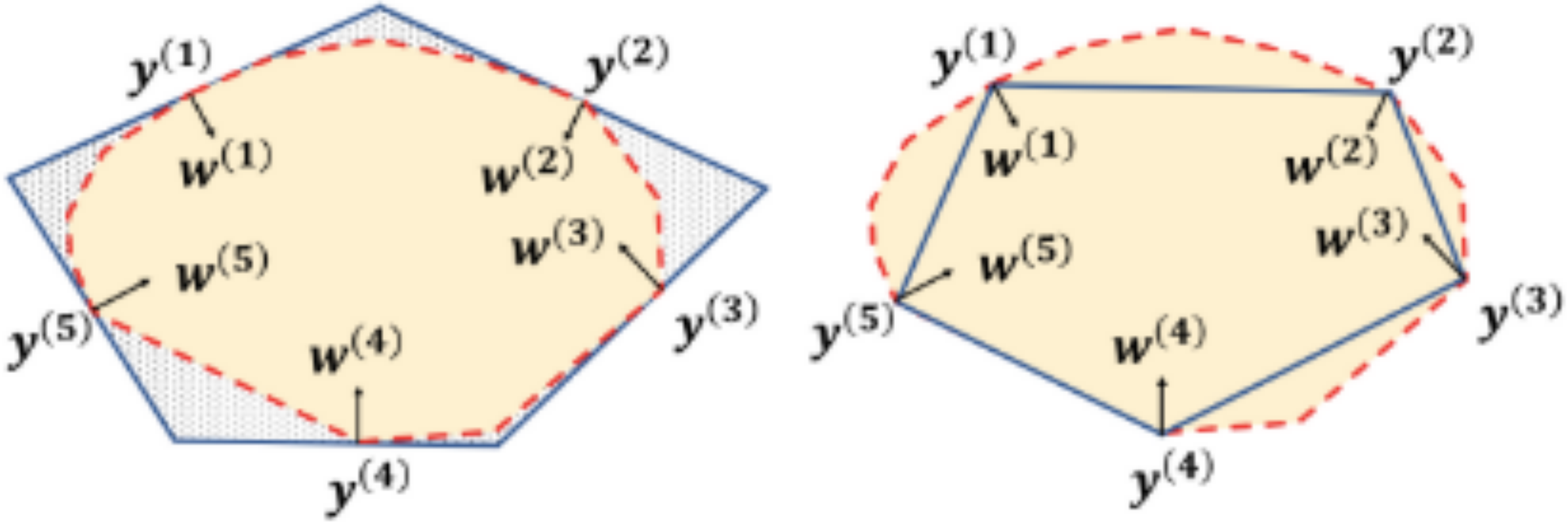


Related Work 2: Constraint Mining

An Integer Linear Programming Framework for Mining Constraints from Data

Tao Meng, Kai-Wei Chang

ICML 2021



(a) Outer polytope (S_O) (b) Inner polytope (S_I)

Related Work 3: Rectifier Networks

Learning Constraints for Structured Prediction Using Rectifier Networks
Xingyuan Pan, Maitrey Mehta, Vivek Srikumar
ACL 2020

- **Training:** 2-layer ReLU network
- **Inference:** network weights → ILP constraints
- Arbitrary constraint parameterization not supported
- Learnable cost not supported

Methodology

Constraint Satisfaction Framework

Cost is ZERO

- Geometry & ML perspective:

  contraint $\leftrightarrow$ hyperplane $\leftrightarrow$ linear classifier

  all constraints $\leftrightarrow$ polytope $\leftrightarrow$ *veto-ensemble* of linear classifiers

- Supervision dichotomy:

  Inference $\Longrightarrow$ combinatorial search via ILP solver

  Training $\Longrightarrow$ binary classification on +ve & -ve examples

- Example source:

  +ve example $\leftarrow$ unique, target solution

  -ve example(s) $\leftarrow$ exponential, sampling strategy?

Proposed Loss Function

- $d_j(A, b, y) = \dfrac{A_j \cdot y + b_j}{||A_j||}$ → signed distance from hyperplane
- $\ell(d_j, \pm)$ → some binary classification loss
- $\mathcal{L}(A, b, y_+) = \max_j \ell(d_j, +)$
- $\mathcal{L}(A, b, y_-) = \min_j \ell(d_j, -)$
- Smooth max/min with temperature τ
- Hinge Loss with margin μ:

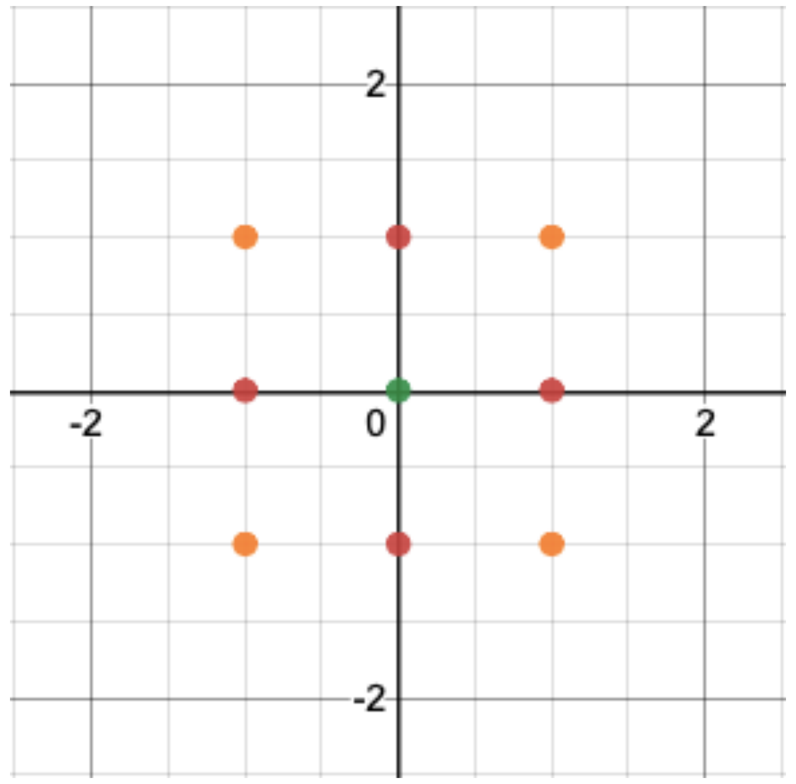
$$\ell(d_j, +) = \mathrm{ReLU}(\mu - d_j), \qquad \ell(d_j, -) = \mathrm{ReLU}(\mu + d_j)$$

- Cross-Entropy Loss:

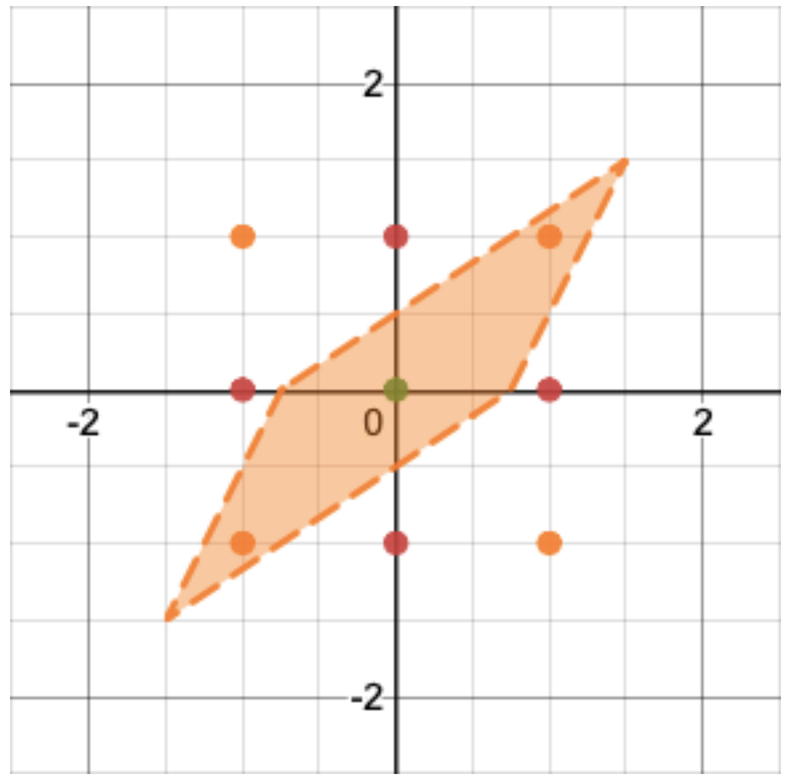
$$\ell(d_j, +) = -\log(\sigma(d_j)), \qquad \ell(d_j, -) = -\log(1 - \sigma(d_j))$$

Solver-Free Negative Sampling

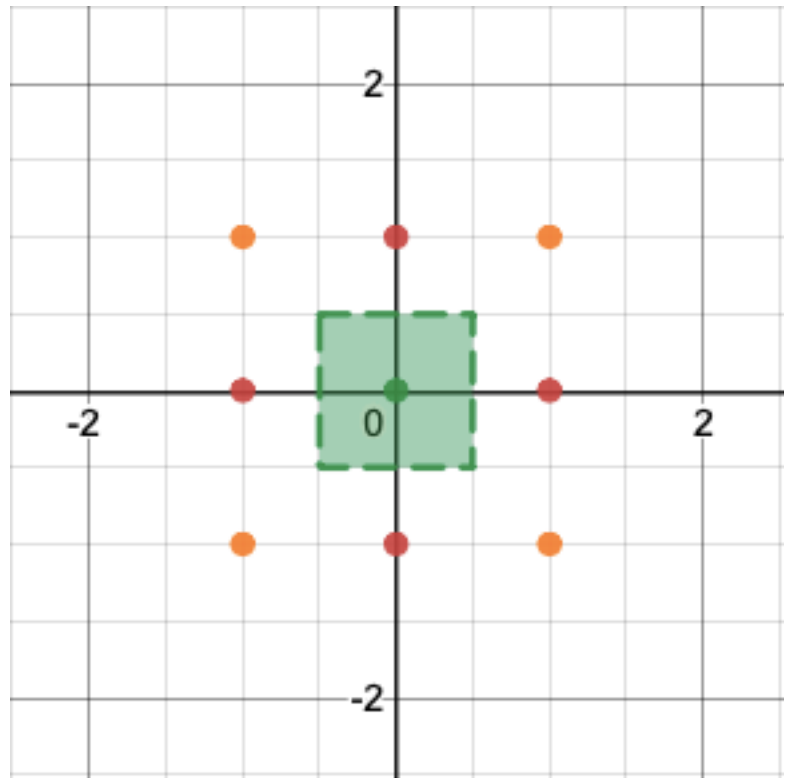
Nearest Neighbours



+ve & -ve examples

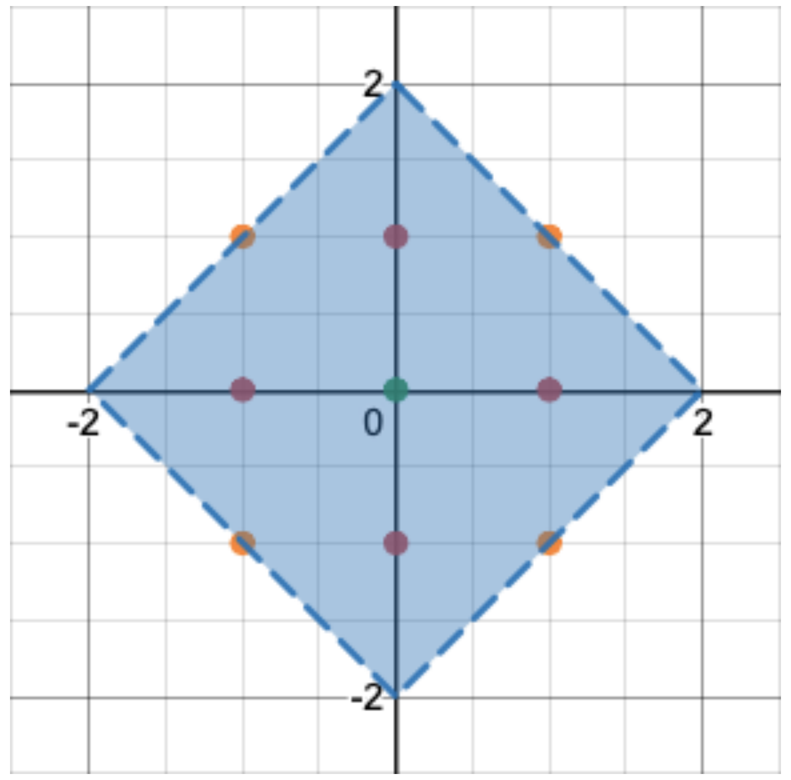


a polytope excluding nearest neighbours

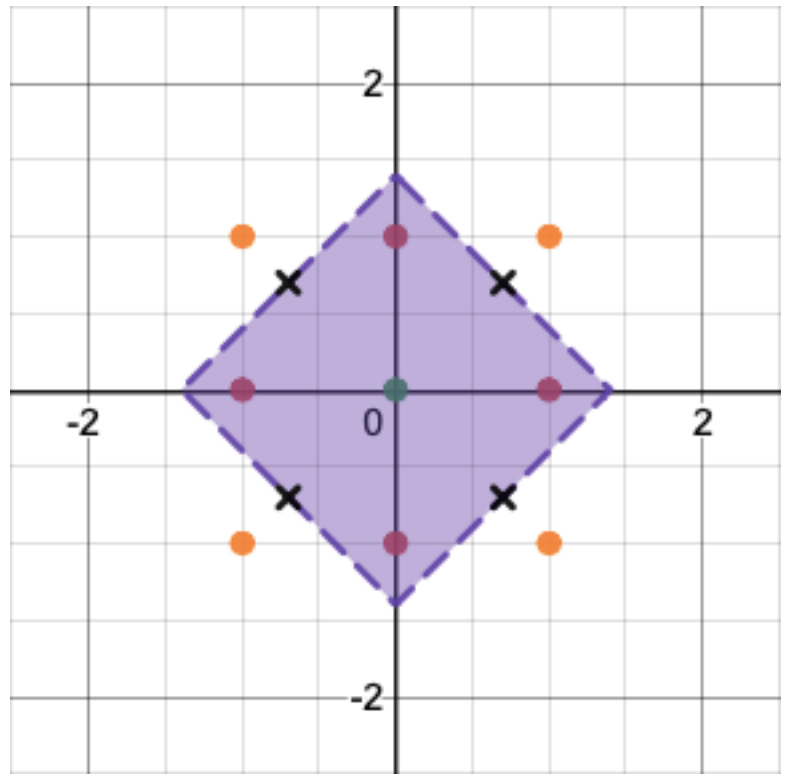


a possible learnt polytope

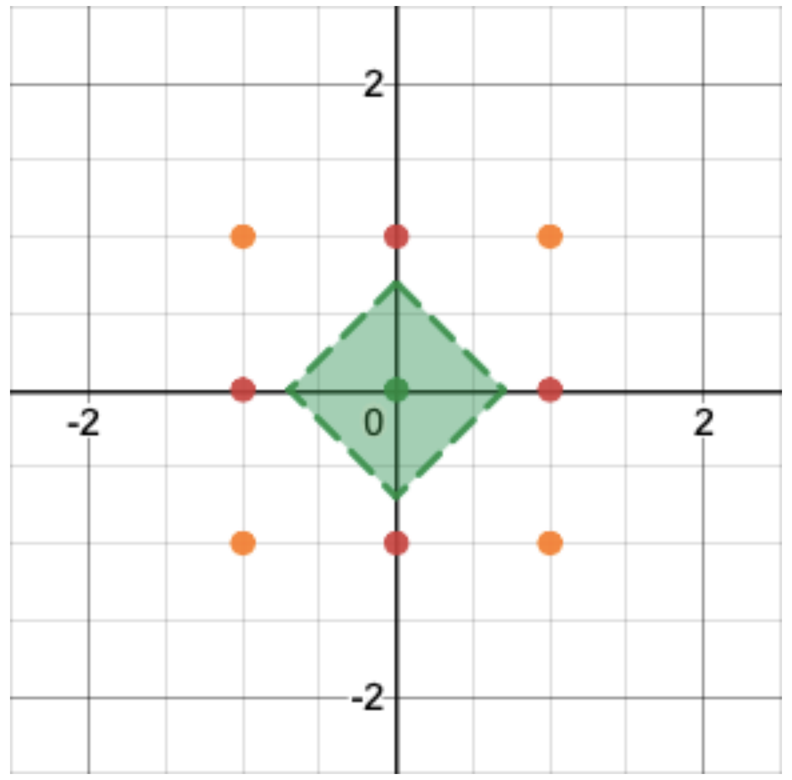
Project and Round



an interim learnt polytope



non-integral projected negatives



a possible learnt polytope

Solver-Based Negative Sampling

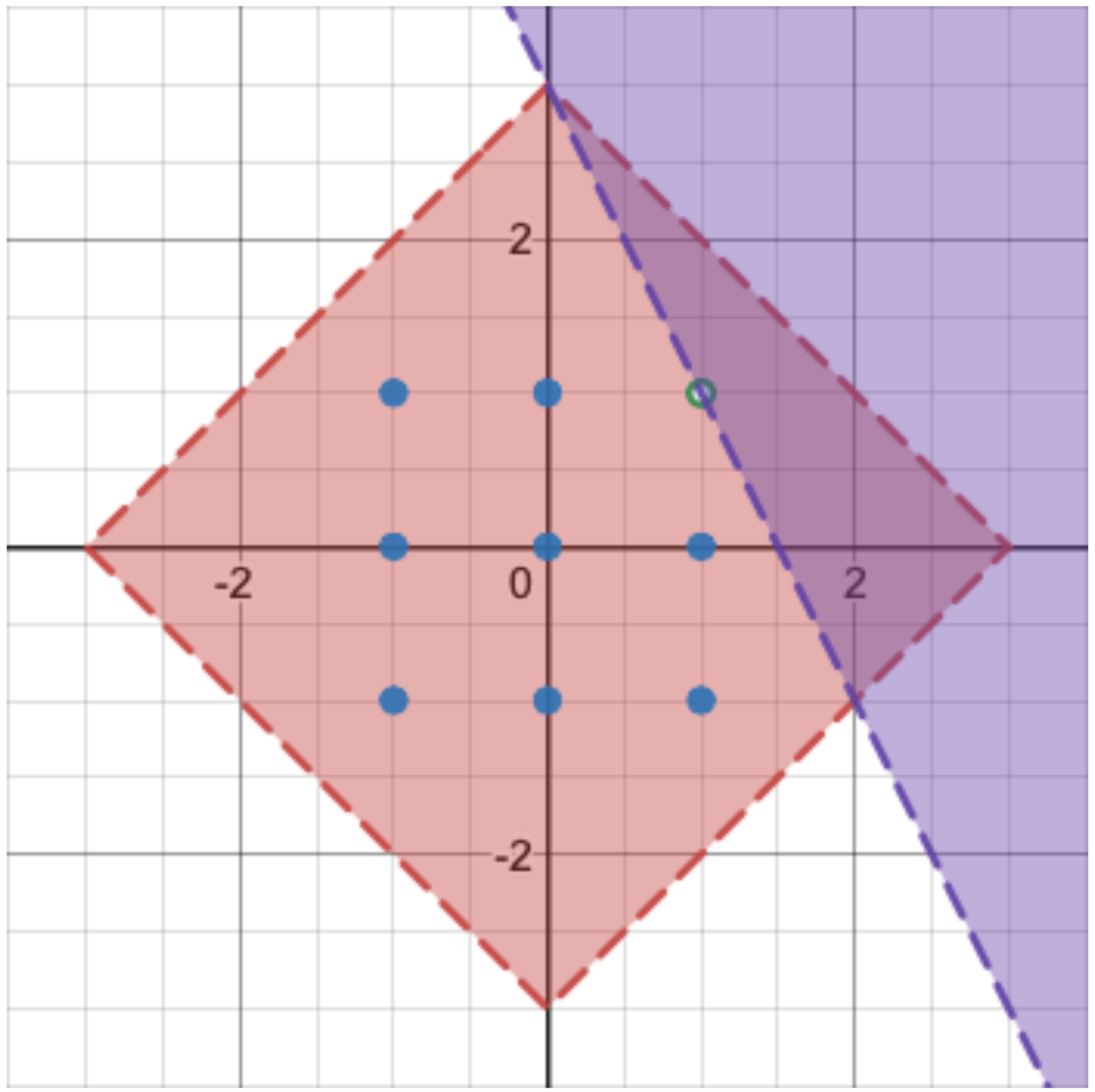
- Solution to current **ILP** (if $\neq$ target solution)
- Solution to **LP relaxation** (+ rounding, if $\neq$ target solution)

Regularization via Variance Loss

- Constraint directions may collapse
- Minimize $\sum_i \sum_j \frac{A_i \cdot A_j}{\|A_i\| \, \|A_j\|}$
- Equivalently, minimize $\mathscr{L}_v = \left\| \sum_i \frac{A_i}{\|A_i\|} \right\|^2$
- Crucial for projected negative samples

Extending to Constraint Optimization

Capture cost via new constraint: $c \cdot y \leq c \cdot y_+$



Results

Experimental Setup

Task: Symbolic Sudoku

- **Sudoku sizes:** 4x4, 6x6
- **Sudoku modelling:** clues as constraints, clues as cost weights
- **Binary classification loss:** Cross-Entropy, Hinge
- **Negative sampling:** Nearest Neighbours, Projection, ILP solution, LP solution
- **Baselines:** CombOptNet, Rectifier Networks
- Test accuracy is either 100% or 0%

Takeaways 1: Learnability

- **CombOptNet:**
  - Ultimately works for all cases
  - Training with cost is visibly more unstable
- **Rectifier Networks:**
  - Clues as constraints: not supported
  - Clues as cost: loss decreases but fails to learn – inadequate negatives
- **Our approach:**
  - Cross-Entropy loss doesn't work except for some cases
  - All negative samplers work with Hinge Loss (with suitable hyperparams)

Takeaways 2: Training Time

- **Solver-based:**
  - training takes hours (~11 hrs for CombOptNet at 4x4 with cost), even with upto 64 cores
  - cost optimization is a lot slower than mere satisfaction (~5x slower for CombOptNet)
  - GPU acceleration is pointless
- **Solver-free:**
  - converge in minutes (~3 mins for our approach at 4x4), on a single core
  - effective GPU acceleration
  - cost optimization is as fast as satisfaction
  - required learning iterations are only negligibly worse

Next Steps

For BTP2

- Scale to 9x9 sudokus
- Perceptual component
- Other tasks

Thank You!

Questions?