Preferred Networks

BNN verification dataset for Max-SAT Evaluation 2020

Masahiro Sakai (sakai@preferred.jp)

NII Shonan Meeting (No.180) on "The Art of SAT"
Oct. 2023

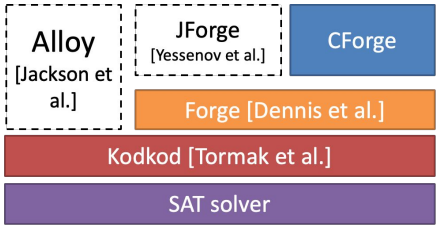
Masahiro Sakai (酒井 政裕) @ Preferred Networks, Inc.

Twitter: @masahiro_sakai

Some of my SAT/SMT-related activities

1. Japanese Translation of "Software Abstractions"
A book about Alloy (a SAT-based finite model finder) and Formal Methods

2. CForge: Model-Checking C Programs against JML-like Specification Language



(Work in Toshiba)

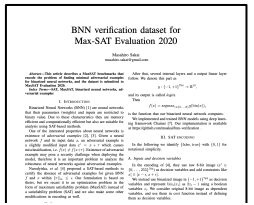
Also we combined BMC and UNSAT-core enumeration for precondition inference

3. toysat
SAT/Max-SAT/PB/SMT/QBF/MILP solver written in Haskell

A hobby project for learning solver algorithms, but submitted to some past PB / Max-SAT / SMT competitions.

github:msakai/toysolver

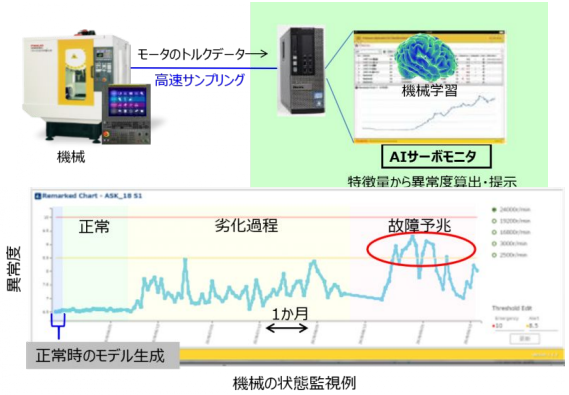
4. Binarized Neural Network verification instances for Max-SAT Evaluation 2020

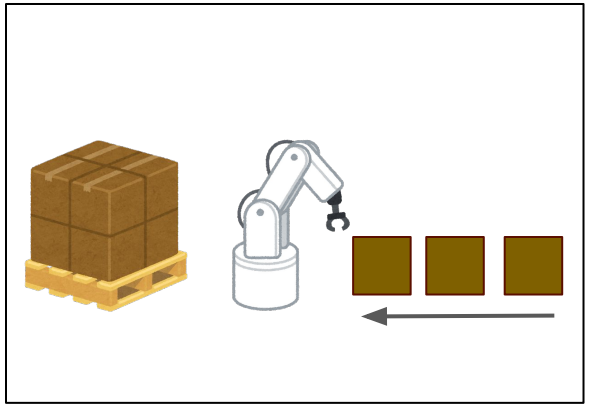


github:msakai/bnn-verification

Personal Motivation

I have been using SAT/SMT and also implementing my own solver.
Then, I learned machine learning, joined Preferred Networks in 2017, and worked on projects that uses (mainly) machine learning.





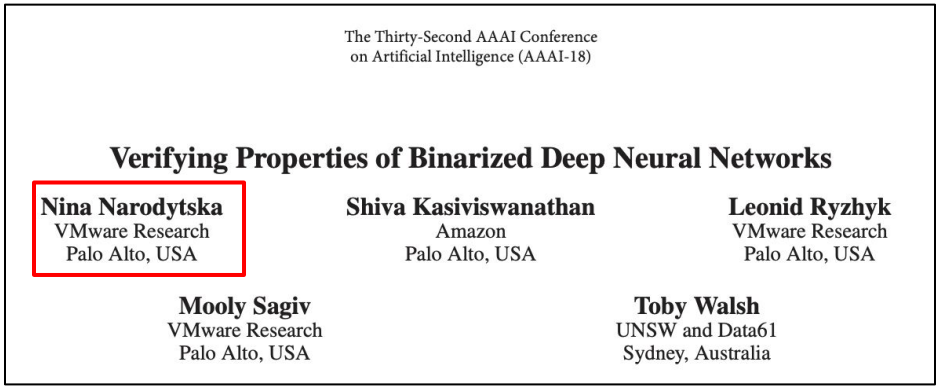
So, I'm interested in the overlapping area of machine learning and SAT/SMT.

Preferred Networks

Acknowledgement

Thank you to the authors of [Narodytska+, AAAI-18] and organizers of the MaxSAT Evaluation 2022.

This work is based on the paper:



The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Verifying Properties of Binarized Deep Neural Networks

Nina Narodytska
VMware Research
Palo Alto, USA

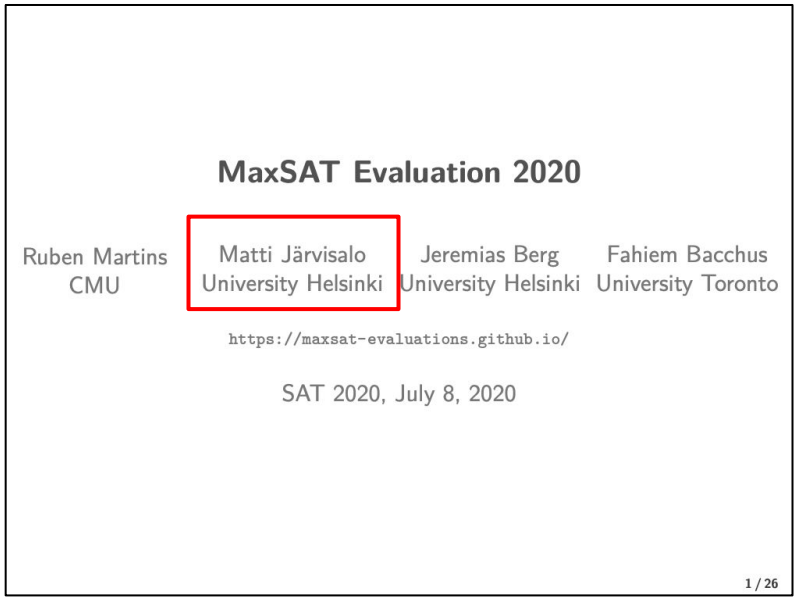
Shiva Kasiviswanathan
Amazon
Palo Alto, USA

Leonid Ryzhyk
VMware Research
Palo Alto, USA

Mooly Sagiv
VMware Research
Palo Alto, USA

Toby Walsh
UNSW and Data61
Sydney, Australia

https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16898



MaxSAT Evaluation 2020

Ruben Martins
CMU

Matti Järvisalo
University Helsinki

Jeremias Berg
University Helsinki

Fahiem Bacchus
University Toronto

https://maxsat-evaluations.github.io/

SAT 2020, July 8, 2020

1 / 26

https://maxsat-evaluations.github.io/2020/mse20-talk.pdf

Preferred Networks

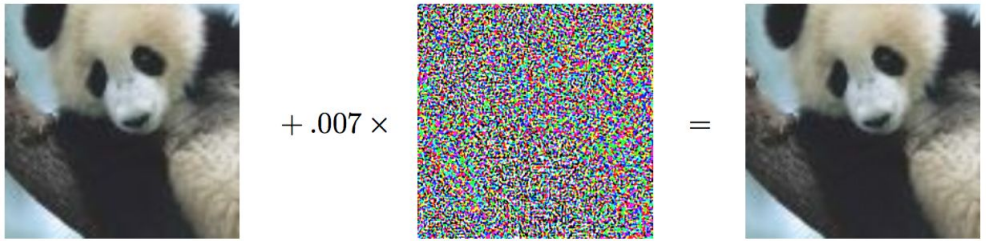
Outline

- 1. Background: DNN verification
- 2. Background: Binarized Neural Networks (BNN)
- 3. (Max-)SAT-based verification
  - Models
  - SAT-encoding
  - Cost Function and Soft Clauses
- 4. Result in the Max-SAT Evaluation 2020
- 5. Conclusion

1. Background on DNN verification

Adversarial perturbations

- Deep neural networks (DNN) achieved impressive performance in various tasks, But …



- "Explaining and Harnessing Adversarial Examples" [Goodfellow+, ICLR 2015]
- "Intriguing properties of neural networks" [Szegedy+, ICLR 2014]
- …

Preferred Networks

Formal guarantees on DNN behavior

- Such non-robustness may result in **unsafe systems**, or **restrict the use of DNNs in safety-critical applications**.
  - e.g. autonomous vehicles, airborne collision avoidance systems, deep-brain stimulation against epileptic seizures, robots, etc.
- Hence, there is a need for **formal guarantees about their behavior**.
- There are lots of research using e.g. SAT, SMT, CEGAR, MILP

Basic approach

For example, a property called **local adversarial robustness** can be checked by SAT/SMT by checking if

$$\|\tau\| \leq \varepsilon \land f(x+\tau) \neq f(x)$$

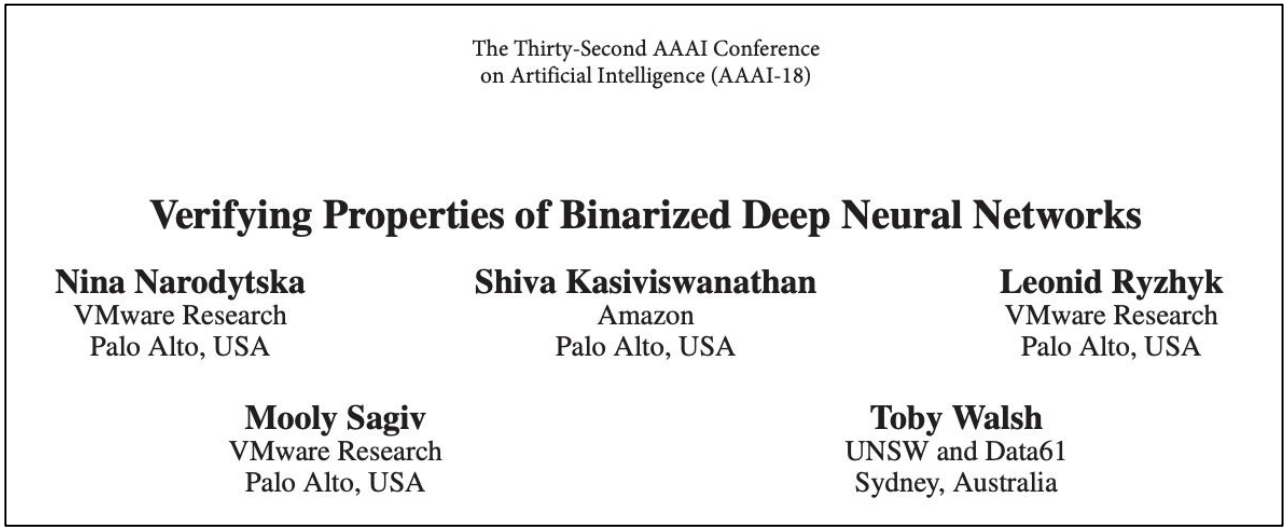
is satisfiable.

However, neural network **f** is often **very large**, **complex**, and **contains massive amounts of floating point operations.**
⇒ Difficulty for SAT/SMT solvers

There are many researches for overcoming the problem.

Focusing on specific type of NN

- [Narodytska+, AAAI-18] is one such research
- They focus on a specific type of neural networks called Binalized Deep Neural Networks (BNN)



The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Verifying Properties of Binarized Deep Neural Networks

Nina Narodytska
VMware Research
Palo Alto, USA

Shiva Kasiviswanathan
Amazon
Palo Alto, USA

Leonid Ryzhyk
VMware Research
Palo Alto, USA

Mooly Sagiv
VMware Research
Palo Alto, USA

Toby Walsh
UNSW and Data61
Sydney, Australia

https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16898

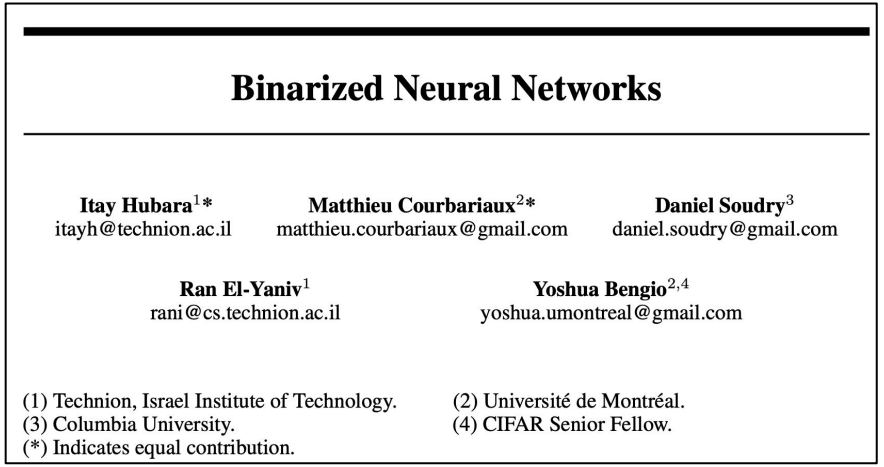
Preferred Networks

2. Background: Binarized Neural Networks (BNN)

Binarized Neural Networks (BNN)

- **Large number of floating point operations** are computationally expensive and require big memory
- Usually **float32** is used
- Quantization is a technique to use **reduces bit-width floating point**, or **small bit-width integer** instead
- In the extreme case, binary values {-1, +1} is used

- One such attempt is Binarized Neural Networks [Hubara+, NIPS 2016]



Binarized Neural Networks

Itay Hubara[1]* itayh@technion.ac.il

Matthieu Courbariaux[2]* matthieu.courbariaux@gmail.com

Daniel Soudry[3] daniel.soudry@gmail.com

Ran El-Yaniv[1] rani@cs.technion.ac.il

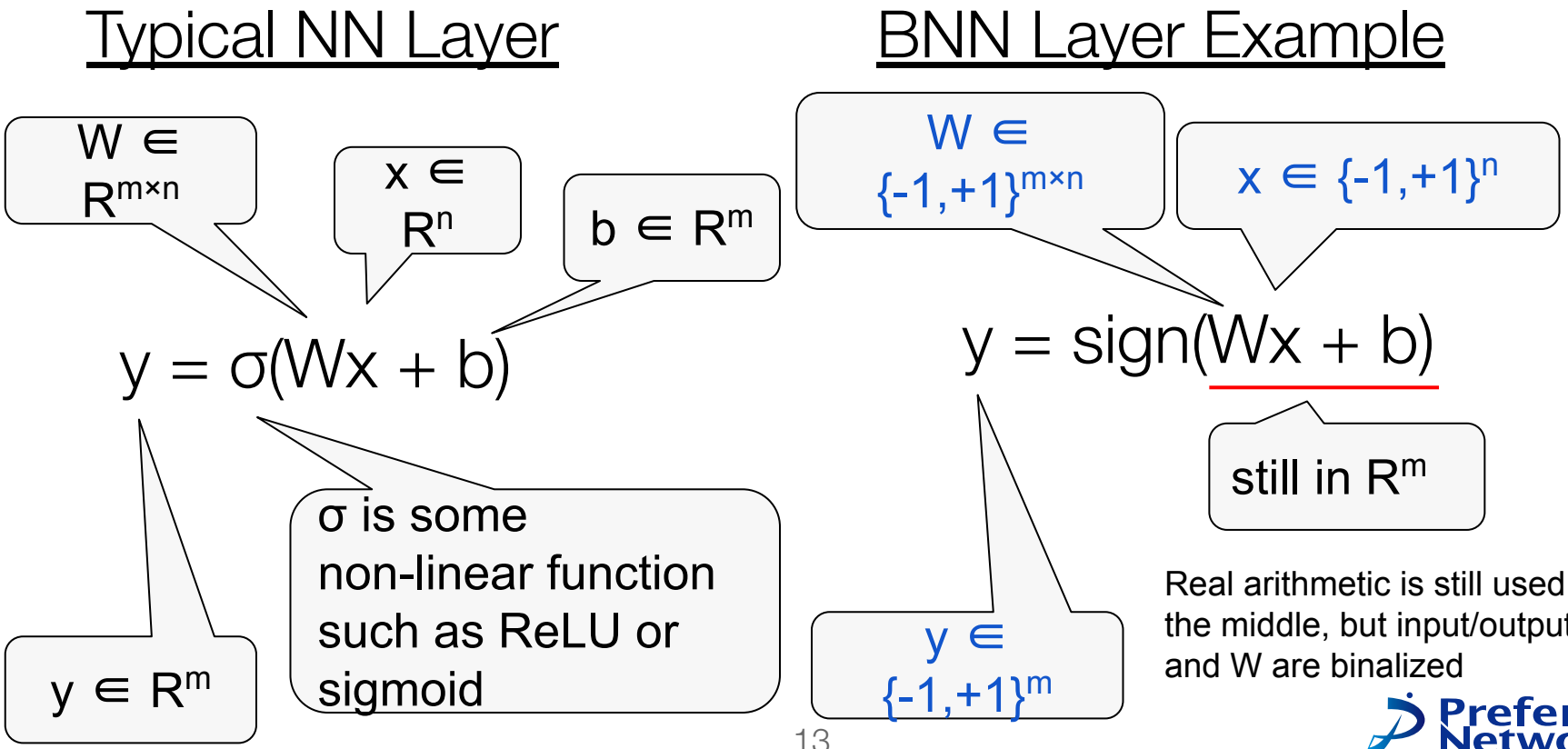
Yoshua Bengio[2,4] yoshua.umontreal@gmail.com

(1) Technion, Israel Institute of Technology.
(2) Université de Montréal.
(3) Columbia University.
(4) CIFAR Senior Fellow.
(*) Indicates equal contribution.

https://papers.nips.cc/paper/6573-binarized-neural-networks

Preferred Networks

Binarized Neural Networks (BNN)



Preferred Networks

Binarized Neural Networks (BNN)

- Binarization is for performance (e.g. computation time, memory usage, energy consumption, circuit area, …)

- But, at the same time, it is very convenient for SAT solvers

3. (Max-)SAT-based verification

Our approach is based on [Narodytska+, AAAI-18]
Since many parts are the same, I will talk only the overall idea and the details of the difference.

Decision problem ⇒ Optimization probem

[Narodytska+, AAAI-18] verifies absence of adversarial examples (among other properties) by showing

$$\|\tau\|_\infty \leq \varepsilon \ \wedge \ f(x_{orig}) \neq f(x_{orig} + \tau)$$

is UNSAT for given f, x, and ε.

However, this can be generalized to an optimization problem

$$\text{min. } \|\tau\|_\infty \quad \text{s.t. } \ f(x_{orig}) \neq f(x_{orig} + \tau)$$

i.e. finding minimum adversarial perturbation.

(Actually, we encode the objective function as a set of weighted soft clauses to use Max-SAT solvers)

Models

High-level structure of our model and input data

- High-level structure of our model:

$$f(x) = \mathrm{argmax}_{c \in \{0,\ldots,9\}} g(\mathrm{bin}(x))_c$$

- Input image $x \in \{0, \ldots, 255\}^{784}$
  - 784 (= 28×28) dimension vector of 8-bit integers
- Binarized image $\mathrm{bin}(x) \in \{-1, +1\}^{784}$
  - x is converted to binarized image $\{-1, +1\}^{784}$
  - Note that the binarization threshold is different depending on components (i.e. pixel locations)

Internal and output layers

- High-level structure of our model:

$$f(x) = \mathrm{argmax}_{c \in \{0,\ldots,9\}} g(\mathrm{bin}(x))_c$$

- We denote composition of internal layers and a output linear layer as $g : \{-1, +1\}^{784} \to \mathbb{R}^{10}$
- The output $g(\mathrm{bin}(x)) \in \mathbb{R}^{10}$ is called **logits**

Internal and output layers

- High-level structure of our model:

$$f(x) = \underline{\mathrm{argmax}_{c \in \{0,\ldots,9\}}} g(\mathrm{bin}(x))_c$$

- $g(\mathrm{bin}(x)) \in \mathbb{R}^{10}$ is called **logits**
- **argmax** compare the logits and choose a class c with maximal logit

Implementation and Training

- High-level structure of our model:

$$f(x) = \mathrm{argmax}_{c \in \{0,\ldots,9\}} g(\mathrm{bin}(x))_c$$

- We implemented the model and trained using deep learning framework Chainer

Preferred Networks

Implementation and Training

- **Trained three models for three datasets**
  - MNIST and its two variants: MNIST-rot and MNIST-back-image

100 sample images from these dataset

MNIST

MNIST-rot

MNIST-back-image

Preferred Networks

SAT encoding

Inputs and decision variables

$$f(x) = \mathrm{argmax}_{c \in \{0,\ldots,9\}} g(\mathrm{bin}(x))_c$$

- [Narodytska+, AAAI-18] uses perturbation τ as decision variables and add constraints like $\tau \in [-\varepsilon, +\varepsilon]^{784}$ and $x_{orig}+\tau \in [0, 255]^{784}$ which involve integer addition and comparison

- We instead use binarized image $z = \mathrm{bin}(x_{orig}+\tau) \in \{-1, +1\}^{784}$ as decision variables
  - Once you find a desired binarized image, it is easy to construct smallest perturbation in original image space
  - Reduced number of decision variables and constraints

Relationship between inputs and logits

$$f(x) = \mathrm{argmax}_{c \in \{0,\ldots,9\}} g(\mathrm{bin}(x))_c$$

- The g part is encoded to CNF in a way similar to [Narodytska+, AAAI-18]
  - Detail omitted here, but can be encoded to clauses and cardinality constraints
- We use **totalizer** [Bailleux, CP 2003] instead of **sequential counter** [Sinz+, CP 2005] for encoding cardinality constraints to reduce formula size

Relationship between logits and outputs

$$f(x) = \underline{\mathrm{argmax}_{c \in \{0,\ldots,9\}}} g(\mathrm{bin}(x))_c$$

- Let logits := g(bin(x))
- Let y := $\{y_c\}_c$ be ten boolean variables such that $y_c \Leftrightarrow f(x)=c$

- Since y is one-hot vector
  - $\Sigma_c\, y_c = 1$
- If y_c is true, c-th logit must be largest
  - $y_c \rightarrow \underbrace{\mathrm{logits}_c \geq \mathrm{logits}_{c'}}$ for all c′.

A cardinality constraints if we expand the definition of logits

(Conditional) cardinality constraints are encoded using totalizer

Preferred Networks

Relationship between logits and outputs

$$f(x) = \underline{\mathrm{argmax}_{c \in \{0,\ldots,9\}}} g(\mathrm{bin}(x))_c$$

- Let logits := g(bin(x))
- Let y := $\{y_c\}_c$ be ten boolean variables such that $y_c \Leftrightarrow f(x)=c$

- Finally, we want the input to be misclassified
- If the (true) label of the image is c, we add $(\neg\, y_c)$

Cost Function and Soft Clauses

Turning cost functions into soft constraints

- To find the smallest adversarial perturbation, we have to penalize modification from x_is
- Our decision variables are binarized pixels z_is instead of τ_is
- Therefore, the basic idea is to add the following as soft constraints with appropriate weights

$$z_i = \mathrm{bin}_i(x_{\mathrm{orig},i})$$

This is a constant

Preferred Networks

Some preparation

- Let τ_i be the change of x_i
  - $x_i = x_{orig,i} + \tau_i$
- Let δ_i be the smallest perturbation to change z_i
  - $bin_i(x_{orig,i} + \delta_i) \neq bin_i(x_{orig,i})$
- Since we are minimizing τ, we can let
  - τ_i = (if $z_i = bin_i(x_{orig,i})$ then 0 else δ_i)

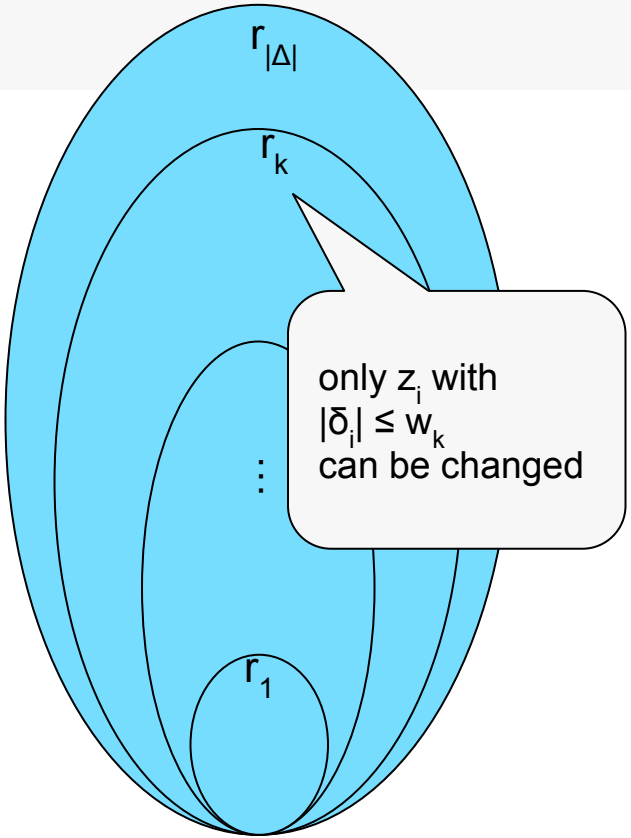
L_p-norm case for $p \neq \infty$

- $\|\tau\|_p = (|\tau_1|^p + \ldots + |\tau_n|^p)^{(1/p)}$
- Since $(-)^{(1/p)}$ is monotone, minimizing $\|\tau\|_p$ is equivalent to minimizing $|\tau_1|^p + \ldots + |\tau_n|^p$
- Since τ_i = (if $z_i = \mathrm{bin}_i(x_{\mathrm{orig},i})$ then 0 else δ_i), it is equivalent to have $(z_i = \mathrm{bin}_i(x_{\mathrm{orig},i}))$ as a soft constraint with weight $|\delta_i|^p$

L_∞-norm case

- $\|\tau\|_\infty = \max(|\tau_1|, \ldots, |\tau_n|)$

- Let $\Delta = \{ |\delta_i| \} = \{w_1 < \ldots < w_{|\Delta|}\}$
- Introduce relaxation variables $\{r_k\}_{k \in 1..|\Delta|}$ with
  - $\neg r_k \longrightarrow (z_i = \mathrm{bin}_i(x_{\mathrm{orig},i}))$
    for all i with $|\delta_i| = w_k$
  - $r_k \longrightarrow r_{k-1}$ *(they are lower closed)*
  - $(\neg r_k)$ with cost $(w_k - w_{k-1})$



Preferred Networks

Result in the Max-SAT Evaluation 2020

Preferred Networks

Instance generation and submission

- Average size of generated instance
  - #variables = 1.8 M
  - #clauses = 132 M
  - (uncompressed) WCNF filesize = 3.6 GB
- Submitted 60 instances
  - 3 dataset and trained model
  - 2 images for each of 10 digit class
  - only used L_∞-norm setting

5 among 60 submitted instances were used

Instance	Image	Label
bnn_mnist_7_label9_adversarial_norm_inf_totalizer.wcnf.gz		9
bnn_mnist_back_image_32_label3_adversarial_norm_inf_totalizer.wcnf.gz		3
bnn_mnist_rot_16_label5_adversarial_norm_inf_totalizer.wcnf.gz		5
bnn_mnist_rot_8_label1_adversarial_norm_inf_totalizer.wcnf.gz		1
bnn_mnist_back_image_73_label5_adversarial_norm_inf_totalizer.wcnf.gz		5

Result: Complete track (Weighted)

https://maxsat-evaluations.github.io/2020/rankings.html

Instance	maxino-pref	maxino	Pacose	UWrMaxSat	MaxHS	QMaxSAT	RC2-B / RC2-A / smax-minisat / smax-mergesat
…	270.62	269.06	402.17	648.45	991.52	141.42	3600.0
…	279.84	277.76	1101.24	795.81	1733.77	1729.06	
…	367.28	367.06	221.87	657.69	1006.6	704.83	
…	84.87	84.06	347.71	588.25	1083.57	3600.0	
…	2215.51	2232.61	3600.0	3600.0	3600.0	3600.0	

Result: optimal objective values

Very small $\|\tau\|_\infty$ is enough ⇒ Those models are not robust at all… 😢

Instance	Image	Label	Minimal $\|\tau\|_\infty$
bnn_mnist_7_label9_adversarial_norm_inf_totalizer.wcnf.gz		9	1
bnn_mnist_back_image_32_label3_adversarial_norm_inf_totalizer.wcnf.gz		3	2
bnn_mnist_rot_16_label5_adversarial_norm_inf_totalizer.wcnf.gz		5	1
bnn_mnist_rot_8_label1_adversarial_norm_inf_totalizer.wcnf.gz		1	1
bnn_mnist_back_image_73_label5_adversarial_norm_inf_totalizer.wcnf.gz		5	4

Some slides from organizer's talk

MaxSAT Applications

- Many real-world applications can be encoded to MaxSAT:
  - Software package upgradeability
  - Error localization in C code
  - Haplotyping with pedigrees
  - Verification of binarized networks
  - ...
- MaxSAT algorithms are **very effective** for solving real-word problems

3 / 26

New benchmarks

- Verification of binarized neural networks
- Maximum probability minimal cut sets
- Data flow security weaknesses of reconfigurable scan networks
- Coalition Structure Generation
- Role-Based Access Control maintenance
- Synthesis of minimal hardware exploits
- Most compatible phylogenetic trees over multi-state characters
- Railway timetabling
- Single-machine scheduling
- Analysis of large networks
- Set covering problems from Italian railways
- Set covering problems from Steiner Triple Systems
- Program disambiguation
- Inference of tumor evolutionary history
- User authorization query problems

8 / 26

https://maxsat-evaluations.github.io/2020/mse20-talk.pdf

Preferred Networks

Some slides from organizer's talk

New benchmarks

- MaxSAT is being used in many applications!
  - 15 different domains
- We received a lot of new benchmarks:
  - **5,091 new benchmarks!**
- Benchmark size is increasing significantly:
  - **81 GB** (after gzip compression)
- MaxSAT solvers can solve very large problems:
  - More than 20 **million variables** and 130 **million clauses**

9 / 26

Complete track: Weighted

- Improvements with respect to best solver at MSE 2019:
  - RC2: 417 solved instances, 205.88 seconds (average)
- VBS solves 479 instances:
  - Smaller gap than in 2019
- Large benchmarks can be solved:
  - pa-3.wcnf.gz:
    - 19,264,629 variables, 22,951,677 clauses
    - Solved by MaxHS in 731.64 seconds!
  - bnn_mnist_back_image_73_label5_adversarial_norm_inf_totalizer.wcnf.gz
    - 1,825,399 variables, 132,682,679 clauses
    - Solved by Maxino in 2215.51 seconds!

12 / 26

https://maxsat-evaluations.github.io/2020/mse20-talk.pdf

Conclusion

Some concluding remarks (1)

- **Max-SAT solvers can solve big instances (as mentioned by organizer)**
  - Even w/ 1.8M variables and 132M constraints
  - I thought that the problem size MaxSAT solver can handle was limited (compared to SAT), but it can solve larger than I thought!
- **Disclaimer:**
  - This is based on the 2018 paper and the work was done on 2020, so some part of the talk may already be out of date

Some concluding remarks (2)

- **Submitting problems was an interesting and enjoyable experience for me**

- **Future interaction of Machine Learning and SAT/SMT**
  - This is still a toy problem and model (MNIST is like “Hello World” in machine learning)
  - But I hope these two areas to have more fruitful interactions in the future

Any Questions or Comments?

Source code and dataset are available at:
github.com/msakai/bnn-verification

Preferred Networks

Preferred Networks

Making the real world computable