Binarized Neural Network

Inês Jesus, 2073570

Auriane Mahfouz, 2072042

Università degli Studi di Padova

Dipartimento Matematica

What are BNN?

UNIVERSITÀ DEGLI STUDI DI PADOVA

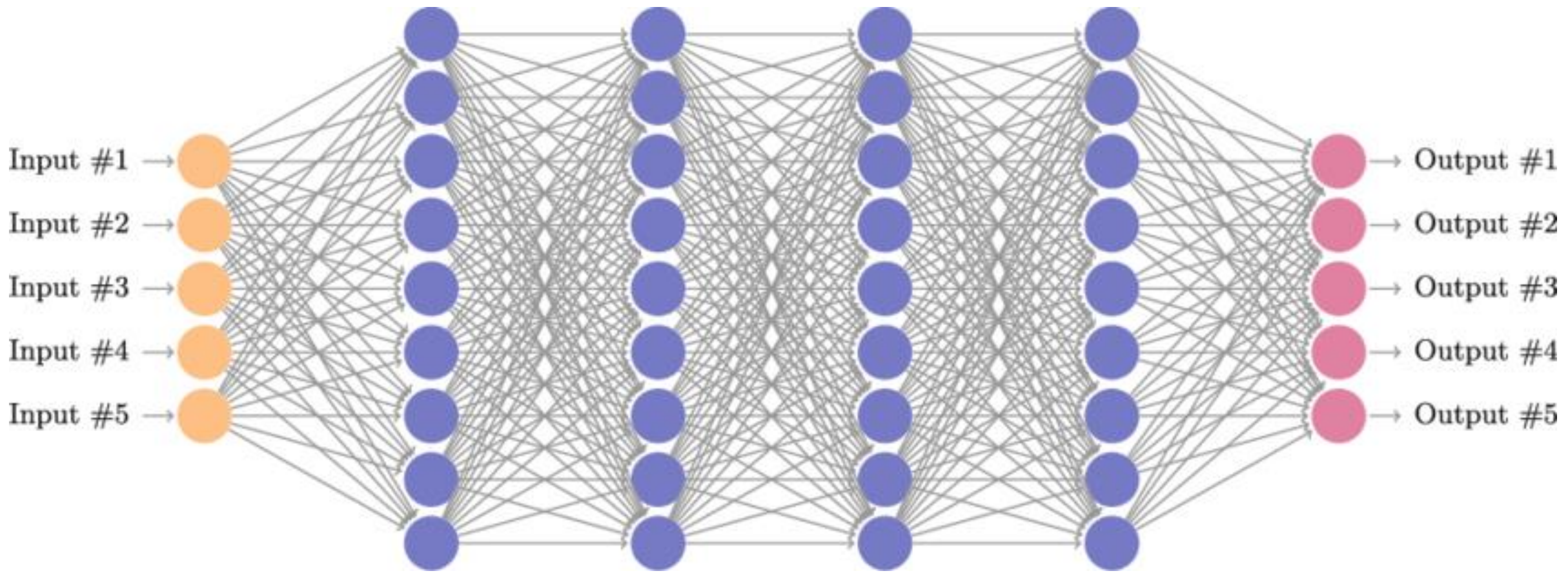
- BNNs consist of binary inputs, outputs, and weights, with values limited to -1 and 1

- The objective of a BNN is to learn the optimal combination of weights to maximize classification accuracy

DIPARTIMENTO MATEMATICA

Aim

The goal is to generate Max-SAT encoding of a binarized neural network layer with 5 input nodes and 5 output nodes.



$l = 1$ $l = 2$ $l = 3$ $l = 4$ $l = 5$ $l = 6$

Data Generation

Università degli Studi di Padova

- The dataset consists of binary input-output pairs $(x^{(t)}, y^{(t)})$ where $x^{(t)} \in \{-1,1\}^5$ and $y^{(t)} \in \{-1,1\}^5$

- We generated all possible combinations of 5 binary inputs, resulting in 32 unique input vectors

- These vectors are used to train and test the binarized neural network model.

Functions

UNIVERSITÀ DEGLI STUDI DI PADOVA

We implement 3 binary functions with 5 inputs and 5 outputs:

- $f_1(x) = [x_1 \wedge x_2, x_3 \wedge x_4, x_2 \wedge x_5, x_1 \wedge x_4, x_3 \wedge x_5]$
- $f_2(x) = [x_1 \vee (x_2 \wedge x_3), \neg x_2, \ \big((x_1 \vee x_2) \wedge (x_3 \vee x_4)\big) \vee \big(x_5 \wedge (\neg x_1 \wedge \neg x_3)\big),$
  $(x_1 \wedge x_2 \ \wedge x_3) \vee (x_4 \wedge x_5), \neg x_2 \wedge \neg x_4 \wedge \neg x_5]$
- $f_3(x) = [majority(x), parity(x), FirstOrLast2(x), \neg x_3, XNOR(x_2, x_3)]$

DIPARTIMENTO MATEMATICA

Functions: Example

For example: $f_1(x) = [x_1 \wedge x_2, x_3 \wedge x_4, x_2 \wedge x_5, x_1 \wedge x_4, x_3 \wedge x_5]$

$x = (x_1, x_2, x_3, x_4, x_5)$	$f_1(x)$
(1, 1, 1, 1, 1)	(1, 1, 1, 1, 1)
(1, -1, -1, 1, 1)	(-1, -1, -1, 1, -1)

DIPARTIMENTO
MATEMATICA

Train and Test Sets

Università degli Studi di Padova

- The dataset is split into training (75%) and testing (25%) sets
- We ensured that the dataset is mostly balanced for each function to prevent biased predictions

DIPARTIMENTO MATEMATICA

Encoding BNN layers into WCNF

- **Weighted Conjunctive Normal Form (WCNF)**: The BNN is encoded in WCNF, where each layer's neurons and connections' constraints are formulated as clauses

- **Layer Connections**: Neurons are connected to the next layer through binary combinations of input signals, with hard clauses enforcing their behavior and encoding all possible input-weight interactions to ensure proper signal propagation

- **Output Layer Encoding**: The binary output values (either -1 or 1) are represented as soft clauses, aligning each prediction with the expected output

- The objective is to minimize the number of unsatisfied soft clauses

SAT Indexes

Università degli Studi di Padova

- A BNN is structured into multiple layers, including an input layer, hidden layers, and an output layer
- Each neuron is associated with a unique index based on its position in the layer and the specific training sample
- Weights connect neurons between layers and are also indexed to track their specific locations
- They are formulated such that the first index is 1

DIPARTIMENTO MATEMATICA

SAT Indexes

Università degli Studi di Padova

- **Input feature** i from sample t :

$$x_i^{(t)} = t \times N_1 + i$$

- **Hidden node** i of layer l from sample t :

$$h_{l,i}^{(t)} = \sum_{k=1}^{l-1} N_k \times N + t \times N_l + i$$

N_k is the number of neurons in layer k.

Layer 1 is the input layer with a fixed number of 5 neurons.

N is the number of training samples.

DIPARTIMENTO MATEMATICA

SAT Indexes

- **Output node** j from sample t:

$$o_j^{(t)} = \underbrace{\sum_{k=1}^{L-1} N_k \times N}_{\text{neurons of previous layers}} + t \times N_L + j$$

- **Weight** from node i of layer l to node j of the next:

$$w_{i,j}^{l} = \underbrace{\sum_{k=1}^{L} N_k \times N}_{\text{all neurons}} + \underbrace{\sum_{k=1}^{l-1} N_k \times N_{k+1}}_{\text{weights of previous layers}} + i \times N_{l+1} + j$$

N_k is the number of neurons in layer k.

L is the total number of layers.

Example: Input Index

Input feature i from sample t :

$$x_i^{(t)} = t \times N_1 + i$$

- Where N_1 = 5 (input neurons), N = 10 (samples), i = input neuron index, t = sample index

- For i = 1 and t = 0:

$$x_1^0 = 0 \times 5 + 1 = 1$$

- For t = 1:

$$x_1^1 = 1 \times 5 + 1 = 6$$

DIPARTIMENTO MATEMATICA

Example: Input Index

Università degli Studi di Padova

If we have 10 examples:

1st sample 2nd sample 10th sample

- $x_1^0 = 1, \; x_1^1 = 6, \ldots, \; x_1^9 = 46$
- $x_2^0 = 2, \; x_2^1 = 7, \ldots, \; x_2^9 = 47$

…

- $x_5^0 = 5, \; x_5^1 = 10, \ldots, \; x_5^9 = 50$

DIPARTIMENTO MATEMATICA

Example: Hidden Index

- **Hidden node** i of layer l from sample t :

$$h_{l,i}^{(t)} = \sum_{k=1}^{l-1} N_k \times N \; + \; t \; \times N_l + \; i$$

- For hidden node $h_{2,1}^{0}$ with l = 2, N_1 = 5, N_2= 3, N = 10:

$$h_{2,1}^{0} = \; 5 \; \times \; 10 \; + \; 1 \; = \; 51$$

- For t = 1:

$$h_{2,1}^{1} = \; 50 \; + \; 3 \; + \; 1 \; = \; 54$$

DIPARTIMENTO MATEMATICA

Example: output Index

- **Output node** j from sample t:

$$o_j^{(t)} = \sum_{k=1}^{L-1} N_k \times N \ + \ t \ \times N_L + \ j$$

- For output node o_1^t with L = 3, N_1= 5, N_2 = 3, N_3 = 5, N = 10:

$$o_1^0 = (5 \times 10) + (3 \times 10) + \ 1 \ = 50 + 30 + 1 = \ 81$$

- For t = 1:

$$o_1^1 = 80 + 1 \times 5 \ + \ 1 \ = \ 86$$

Example: Weight Index

- **Weight** from node i of layer l to node j of the next:

$$w_{i,j}^{l} = \sum_{k=1}^{L} N_k \times N + \sum_{k=1}^{l-1} N_k \times N_{k+1} + i \times N_{l+1} + j$$

- For weight $w_{1,1}^{2}$ with N = 10, N_1= 5, N_2 = 3, N_3= 5:

$w_{1,1}^{2}$= (50+30+50) + 15 + 5 + 1 = 151

Università degli Studi di Padova

Activation function:

$$o_j^{(t)} = sign\left(\sum_{i=1}^{5} x_i^{(t)}\, w_{i,j}^{1}\right)$$

When $x_i^{(t)}$ and $w_{i,j}^{1}$ have the same sign, we add 1; otherwise, we subtract 1, so the result is positive if:

$$\#\left(x_i^{(t)} \Leftrightarrow w_{i,j}^{1}\right) > \#\left(\neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^{1}\right)\right)$$

Problem Encoding

One fully connected layer

UNIVERSITÀ DEGLI STUDI DI PADOVA

As such, if we find a combination of i's such that:

$$\#\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right) \geq \left\lceil \frac{5}{2} \right\rceil$$

then $o_j^{(t)} = 1$.

And if we find a combination where:

$$\#\left(\neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right) \geq \left\lceil \frac{5}{2} \right\rceil$$

then $o_j^{(t)} = -1$.

The two are mutually exclusive.

- So we need to iterate over every combination of 3 input neurons.

- There are 10 sets of combinations of 5 neurons, 3 items at a time. We define the set:

$$C_5 = \left\{ \binom{5}{3}_1, \binom{5}{3}_2, \dots, \binom{5}{3}_{10} \right\}$$

or more generally

$$C_k = \left\{ \binom{k}{\left\lceil \frac{k}{2} \right\rceil}_1, \binom{k}{\left\lceil \frac{k}{2} \right\rceil}_2, \dots, \binom{k}{\left\lceil \frac{k}{2} \right\rceil}_{\binom{k}{\left\lceil \frac{k}{2} \right\rceil}} \right\}$$

Problem Encoding

One fully connected layer

So, we can respectively rewrite

$$\text{“}\mathit{If}\ \#\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right) \geq \left\lceil \frac{5}{2} \right\rceil \mathit{for\ some\ combination\ of}\ \text{i's},\ \mathit{then}\ o_j^{(t)} = 1\text{”}$$

$$\text{“}\mathit{If}\ \#\left(\neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right) \geq \left\lceil \frac{5}{2} \right\rceil \mathit{for\ some\ combination\ of}\ \text{i's},\ \mathit{then}\ o_j^{(t)} = -1\text{”}$$

as

$$\left[\bigvee_{c \in C_5} \bigwedge_{i \in c} \left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right] \Rightarrow o_j^{(t)}$$

$$\left[\bigvee_{c \in C_5} \bigwedge_{i \in c} \neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right] \Rightarrow \neg o_j^{(t)}$$

Università degli Studi di Padova
Dipartimento Matematica

CNF Formulation

One fully connected layer

Università degli Studi di Padova

$$\left[\bigvee_{c\in C_5}\bigwedge_{i\in c}\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right] \Rightarrow o_j^{(t)}$$

$$\neg\left[\bigvee_{c\in C_5}\bigwedge_{i\in c}\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right] \vee o_j^{(t)}$$

$$\bigwedge_{c\in C_5}\bigvee_{i\in c}\neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right) \vee o_j^{(t)}$$

$$\bigwedge_{c\in C_5}\bigvee_{i\in c}\left(\neg x_i^{(t)} \wedge w_{i,j}^1\right) \vee \left(x_i^{(t)} \wedge \neg w_{i,j}^1\right) \vee o_j^{(t)}$$

$$\bigwedge_{c\in C_5}\boxed{\bigvee_{i\in c}}\left(x_i^{(t)} \vee w_{i,j}^1 \vee o_j^{(t)}\right) \wedge \left(\neg x_i^{(t)} \vee \neg w_{i,j}^1 \vee o_j^{(t)}\right)$$

$$\left[\bigvee_{c\in C_5}\bigwedge_{i\in c}\neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right] \Rightarrow \neg o_j^{(t)}$$

$$\neg\left[\bigvee_{c\in C_5}\bigwedge_{i\in c}\neg\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right)\right] \vee \neg o_j^{(t)}$$

$$\bigwedge_{c\in C_5}\bigvee_{i\in c}\left(x_i^{(t)} \Leftrightarrow w_{i,j}^1\right) \vee \neg o_j^{(t)}$$

$$\bigwedge_{c\in C_5}\bigvee_{i\in c}\left(\left(\neg x_i^{(t)} \vee w_{i,j}^1\right) \wedge \left(x_i^{(t)} \vee \neg w_{i,j}^1\right)\right) \vee \neg o_j^{(t)}$$

$$\bigwedge_{c\in C_5}\boxed{\bigvee_{i\in c}}\left(\neg x_i^{(t)} \vee w_{i,j}^1 \vee \neg o_j^{(t)}\right) \wedge \left(x_i^{(t)} \vee \neg w_{i,j}^1 \vee \neg o_j^{(t)}\right)$$

… still not in CNF.

DIPARTIMENTO MATEMATICA

CNF Formulation

One fully connected layer

So, for each combination, we expand the "big OR", taking into account that for the first formulation we have input nodes and weights with the same sign and for the second with different signs.

As an example, the first combination with neurons $x_1^{(t)}$, $x_2^{(t)}$ and $x_3^{(t)}$ will look like this:

$$\bigwedge_{c \in C_5} \bigvee_{i \in c} \left(x_i^{(t)} \vee w_{i,j}^1 \vee o_j^{(t)} \right) \wedge \left(\neg x_i^{(t)} \vee \neg w_{i,j}^1 \vee o_j^{(t)} \right)$$

$$\left(x_1^{(t)} \vee w_{1,j}^1 \vee x_2^{(t)} \vee w_{2,j}^1 \vee x_3^{(t)} \vee w_{3,j}^1 \vee o_j^{(t)} \right) \wedge$$
$$\left(x_1^{(t)} \vee w_{1,j}^1 \vee x_2^{(t)} \vee w_{2,j}^1 \vee \neg x_3^{(t)} \vee \neg w_{3,j}^1 \vee o_j^{(t)} \right) \wedge$$

$\vdots$

$$\left(\neg x_1^{(t)} \vee \neg w_{1,j}^1 \vee \neg x_2^{(t)} \vee \neg w_{2,j}^1 \vee \neg x_3^{(t)} \vee \neg w_{3,j}^1 \vee o_j^{(t)} \right)$$

$$\bigwedge_{c \in C_5} \bigvee_{i \in c} \left(\neg x_i^{(t)} \vee w_{i,j}^1 \vee \neg o_j^{(t)} \right) \wedge \left(x_i^{(t)} \vee \neg w_{i,j}^1 \vee \neg o_j^{(t)} \right)$$

$$\left(x_1^{(t)} \vee \neg w_{1,j}^1 \vee x_2^{(t)} \vee \neg w_{2,j}^1 \vee x_3^{(t)} \vee \neg w_{3,j}^1 \vee \neg o_j^{(t)} \right) \wedge$$
$$\left(x_1^{(t)} \vee \neg w_{1,j}^1 \vee x_2^{(t)} \vee \neg w_{2,j}^1 \vee \neg x_3^{(t)} \vee w_{3,j}^1 \vee \neg o_j^{(t)} \right) \wedge$$

$\vdots$

$$\left(\neg x_1^{(t)} \vee w_{1,j}^1 \vee \neg x_2^{(t)} \vee w_{2,j}^1 \vee \neg x_3^{(t)} \vee w_{3,j}^1 \vee \neg o_j^{(t)} \right)$$

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO MATEMATICA

Hard clauses
One fully connected layer

$$\left(x_1^{(t)} \vee w_{1,j}^1 \vee x_2^{(t)} \vee w_{2,j}^1 \vee x_3^{(t)} \vee w_{3,j}^1 \vee o_j^{(t)}\right) \wedge$$
$$\left(x_1^{(t)} \vee w_{1,j}^1 \vee x_2^{(t)} \vee w_{2,j}^1 \vee \neg x_3^{(t)} \vee \neg w_{3,j}^1 \vee o_j^{(t)}\right) \wedge$$
$$\vdots$$
$$\left(\neg x_1^{(t)} \vee \neg w_{1,j}^1 \vee \neg x_2^{(t)} \vee \neg w_{2,j}^1 \vee \neg x_3^{(t)} \vee \neg w_{3,j}^1 \vee o_j^{(t)}\right)$$

$$\left(x_1^{(t)} \vee \neg w_{1,j}^1 \vee x_2^{(t)} \vee \neg w_{2,j}^1 \vee x_3^{(t)} \vee \neg w_{3,j}^1 \vee \neg o_j^{(t)}\right) \wedge$$
$$\left(x_1^{(t)} \vee \neg w_{1,j}^1 \vee x_2^{(t)} \vee \neg w_{2,j}^1 \vee \neg x_3^{(t)} \vee w_{3,j}^1 \vee \neg o_j^{(t)}\right) \wedge$$
$$\vdots$$
$$\left(\neg x_1^{(t)} \vee w_{1,j}^1 \vee \neg x_2^{(t)} \vee w_{2,j}^1 \vee \neg x_3^{(t)} \vee w_{3,j}^1 \vee \neg o_j^{(t)}\right)$$

- Iterating over all combinations as described before, each of these formulas will be divided into clauses, and these will be the hard clauses of the MaxSAT problem.
- They define the mathematical operations to get the activation of the output neuron.

UNIVERSITÀ DEGLI STUDI DI PADOVA

$$\left(x_1^{(t)} \vee w_{1,j}^1 \vee x_2^{(t)} \vee w_{2,j}^1 \vee x_3^{(t)} \vee w_{3,j}^1 \vee o_j^{(t)}\right) \wedge$$
$$\left(x_1^{(t)} \vee w_{1,j}^1 \vee x_2^{(t)} \vee w_{2,j}^1 \vee \neg x_3^{(t)} \vee \neg w_{3,j}^1 \vee o_j^{(t)}\right) \wedge$$
$$\vdots$$
$$\left(\neg x_1^{(t)} \vee \neg w_{1,j}^1 \vee \neg x_2^{(t)} \vee \neg w_{2,j}^1 \vee \neg x_3^{(t)} \vee \neg w_{3,j}^1 \vee o_j^{(t)}\right)$$

$$\left(x_1^{(t)} \vee \neg w_{1,j}^1 \vee x_2^{(t)} \vee \neg w_{2,j}^1 \vee x_3^{(t)} \vee \neg w_{3,j}^1 \vee \neg o_j^{(t)}\right) \wedge$$
$$\left(x_1^{(t)} \vee \neg w_{1,j}^1 \vee x_2^{(t)} \vee \neg w_{2,j}^1 \vee \neg x_3^{(t)} \vee w_{3,j}^1 \vee \neg o_j^{(t)}\right) \wedge$$
$$\vdots$$
$$\left(\neg x_1^{(t)} \vee w_{1,j}^1 \vee \neg x_2^{(t)} \vee w_{2,j}^1 \vee \neg x_3^{(t)} \vee w_{3,j}^1 \vee \neg o_j^{(t)}\right)$$

- Since we already know the input values, we can simplify the clauses, removing all $x_i^{(t)}$'s with the rule:

  If $x_i^{(t)} = 1$ (resp. -1), then we keep the clauses where $\neg x_i^{(t)}$ (resp. $x_i^{(t)}$).

DIPARTIMENTO MATEMATICA

Soft clauses

One fully connected layer

UNIVERSITÀ DEGLI STUDI DI PADOVA

- For training the network, we wish to enforce the training labels on the respective samples, and we do so by adding the training labels as a soft clause.

- We follow this rule to append to the WCNF formula the following clauses with weight 1:

"If $y_j^{(t)} = 1$, then append $o_j^{(t)}$; else append $\neg o_j^{(t)}$."

which is simplified as:

$$o_j^{(t)} * sign\left(y_j^{(t)}\right)$$

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA

Activation functions:

$$h_{2,i}^{(t)} = sign\left(\sum_{i=1}^{5} x_i^{(t)}\, w_{i,j}^{1}\right)$$

$$o_j^{(t)} = sign\left(\sum_{i=1}^{N_2} h_{2,i}^{(t)}\, w_{i,j}^{2}\right)$$

N_2 is the number of neurons in layer 2 (the hidden layer).

Hard clauses

Two fully stacked layers

- The formulation for the first layer is the same but with a hidden neuron instead of an output one:

$$\bigwedge_{c \in C_5} \bigvee_{i \in c} \left(x_i^{(t)} \vee w_{i,j}^1 \vee h_{2,j}^{(t)}\right) \wedge \left(\neg x_i^{(t)} \vee \neg w_{i,j}^1 \vee h_{2,j}^{(t)}\right)$$

$$\bigwedge_{c \in C_5} \bigvee_{i \in c} \left(\neg x_i^{(t)} \vee w_{i,j}^1 \vee \neg h_{2,j}^{(t)}\right) \wedge \left(x_i^{(t)} \vee \neg w_{i,j}^1 \vee \neg h_{2,j}^{(t)}\right)$$

- The second layer follows the same structure but from a hidden neuron to the output. We iterate on combinations of $\mathrm{k} = \left\lceil \frac{N_2}{2} \right\rceil$ hidden neurons.

$$\bigwedge_{c \in C_k} \bigvee_{i \in c} \left(h_{2,i}^{(t)} \vee w_{i,j}^2 \vee o_j^{(t)}\right) \wedge \left(\neg h_{2,i}^{(t)} \vee \neg w_{i,j}^2 \vee o_j^{(t)}\right)$$

$$\bigwedge_{c \in C_k} \bigvee_{i \in c} \left(\neg h_{2,i}^{(t)} \vee w_{i,j}^2 \vee \neg o_j^{(t)}\right) \wedge \left(h_{2,i}^{(t)} \vee \neg w_{i,j}^2 \vee \neg o_j^{(t)}\right)$$

- We expand all these formulations and simplify the first layer as before, knowing the training samples.

Soft clauses

Two fully stacked layers

- The output layer is encoded exactly the same way.

$$o_j^{(t)} * \ sign\left(y_j^{(t)}\right)$$

- Following this reasoning, it is easy to further encode more layers. However, it can get computationally expensive to solve the problem.

Model Solver

- Once all constraints are encoded, a Max-SAT solver should be used to find the best set of weights of the BNN that satisfy all the hard clauses and the maximum number of soft clauses.

- For that, PySAT offers 3 algorithmic options:
  - LSU (doesn't support weighted problems yet)
  - FM
  - RC2[1] (outperforms FM)

[1] Alexey Ignatiev, António Morgado, Joao Marques-Silva. *RC2: An Efficient MaxSAT Solver*. MaxSAT Evaluation 2018. JSAT 11. 2019. pp. 53-64

Model Solver: RC2

Università degli Studi di Padova

- We used PySAT where the MaxSAT problem is encoded with **hard** and **soft constraints** in **WCNF** (Weighted Constraint Normal Form)
- We use the RC2 (*Relaxable cardinality constraints*) algorithm which has implemented optimization techniques which result in a good average time complexity compared to other solvers
- We use it with the default underlying SAT oracle Glucose 3

Dipartimento Matematica

Results: First Function

Università degli Studi di Padova

- Without any hidden layer:

```
Layers: [5, 5]
Train size: 24

Number of hard clauses: 2400
Number of soft clauses: 120
Time to add clauses: 0min 00sec

Model computing time:  0min 00sec
Solver cost: 28
Size of model: 265

Train accuracy: 0.77
Train accuracy per function: [0.75 0.79 0.75 0.79 0.75]
Test accuracy: 0.45
Test accuracy per function: [0.5  0.38 0.5  0.38 0.5 ]
```

Results: First Function

UNIVERSITÀ DEGLI STUDI DI PADOVA

- With one hidden layer of 3 neurons:

```
Layers: [5, 3, 5]
Train size: 24

Number of hard clauses: 4320
Number of soft clauses: 120
Time to add clauses: 0min 00sec

Model computing time:  0min 07sec
Solver cost: 11
Size of model: 342

Train accuracy: 0.91
Train accuracy per function: [0.88 0.92 0.92 0.92 0.92]
Test accuracy: 0.78
Test accuracy per function: [0.88 0.75 0.75 0.75 0.75]
```

DIPARTIMENTO MATEMATICA

Results: First Function

Università degli Studi di Padova

7 hidden nodes are enough to find a perfect solution

```
Layers: [5, 7, 5]
Train size: 24

Number of hard clauses: 137760
Number of soft clauses: 120
Time to add clauses: 0min 13sec

Model computing time:  0min 00sec
Solver cost: 0
Size of model: 478

Train accuracy: 1.0
Train accuracy per function: [1. 1. 1. 1. 1.]
Test accuracy: 1.0
Test accuracy per function: [1. 1. 1. 1. 1.]
```

```
Layers: [5, 7, 7, 5]
Train size: 24

Number of hard clauses: 325920
Number of soft clauses: 120
Time to add clauses: 0min 27sec

Model computing time:  0min 56sec
Solver cost: 0
Size of model: 695

Train accuracy: 1.0
Train accuracy per function: [1. 1. 1. 1. 1.]
Test accuracy: 1.0
Test accuracy per function: [1. 1. 1. 1. 1.]
```

DIPARTIMENTO MATEMATICA

Results: Second Function

- Without any hidden layer:

```
Layers: [5, 5]
Train size: 24

Number of hard clauses: 2400
Number of soft clauses: 120
Time to add clauses: 0min 00sec

Model computing time:  0min 00sec
Solver cost: 30
Size of model: 265

Train accuracy: 0.75
Train accuracy per function: [0.75 0.75 0.75 0.75 0.75]
Test accuracy: 0.5
Test accuracy per function: [0.5 0.5 0.5 0.5 0.5]
```

Results: Second Function

- With one hidden layer of 3 neurons:

```
Layers: [5, 3, 5]
Train size: 24

Number of hard clauses: 4320
Number of soft clauses: 120
Time to add clauses: 0min 00sec

Model computing time:  2min 33sec
Solver cost: 16
Size of model: 342

Train accuracy: 0.87
Train accuracy per function: [0.88 0.71 0.92 0.92 0.92]
Test accuracy: 0.8
Test accuracy per function: [0.88 0.88 0.75 0.75 0.75]
```

Results: Second Function

UNIVERSITÀ DEGLI STUDI DI PADOVA

- With one hidden layer of 7 neurons:

```
Layers: [5, 7, 5]
Train size: 24

Number of hard clauses: 137760
Number of soft clauses: 120
Time to add clauses: 0min 09sec

Model computing time:  0min 00sec
Solver cost: 0
Size of model: 478

Train accuracy: 1.0
Train accuracy per function: [1. 1. 1. 1. 1.]
Test accuracy: 1.0
Test accuracy per function: [1. 1. 1. 1. 1.]
```

DIPARTIMENTO MATEMATICA

Results: Third Function

UNIVERSITÀ DEGLI STUDI DI PADOVA

- Without any hidden layer:

```
Layers: [5, 5]
Train size: 24

Number of hard clauses: 2400
Number of soft clauses: 120
Time to add clauses: 0min 00sec

Model computing time:  0min 00sec
Solver cost: 24
Size of model: 265

Train accuracy: 0.8
Train accuracy per function: [1.   0.75 0.83 0.75 0.67]
Test accuracy: 0.55
Test accuracy per function: [1.   0.5  0.75 0.5  0.   ]
```

DIPARTIMENTO MATEMATICA

Results: Third Function

- With one hidden layer of 3 neurons:

```
Layers: [5, 3, 5]
Train size: 24

Number of hard clauses: 4320
Number of soft clauses: 120
Time to add clauses: 0min 00sec

Model computing time:  42min 48sec
Solver cost: 22
Size of model: 342

Train accuracy: 0.82
Train accuracy per function: [0.88 0.88 0.88 0.88 0.58]
Test accuracy: 0.65
Test accuracy per function: [0.62 0.88 0.62 0.88 0.25]
```

Results: Third Function

- With one hidden layer of 7 neurons:

```
Layers: [5, 7, 5]
Train size: 24

Number of hard clauses: 137760
Number of soft clauses: 120
Time to add clauses: 0min 12sec

Model computing time:  85min 46sec
Solver cost: 11
Size of model: 478

Train accuracy: 0.91
Train accuracy per function: [1.   1.   0.88 1.   0.67]
Test accuracy: 0.72
Test accuracy per function: [1.   1.   0.62 1.   0.   ]
```

Summary

Università degli Studi di Padova

Test accuracy per binary function

	[5, 5]	[5, 3, 5]	[5, 7, 5]
$f_1(x)$	45%	78%	100%
$f_2(x)$	50%	80%	100%
$f_3(x)$	55%	65%	72%

Dipartimento Matematica

Discussion

- Adding more neurons results in more accuracy and less cost
- The first two functions, which are simpler than the third one, achieved a perfect accuracy with only one hidden layer of 7 neurons
- Complex functions are computationally expensive even with one hidden layer

Conclusion

- BNN can achieve good results when encoded as a MaxSAT problem
- For more complex problems, the solution can be too computationally expensive for state-of-the-art algorithms, when compared to common machine learning alternatives