Reliable and Interpretable Artificial Intelligence project report

Simone Barbaro, Guillaume Wang

December 2019 (HS2019)

1 Zonotope representation and transformation

We represent a zonotope Z by its center $a_0 \in \mathbb{R}^d$ and a tensor $A \in \mathbb{R}^{k\times d}$ representing the coefficient of the k error terms.

Zonotope propagation through the neural network is straightforward using the transformations presented during the course. For convolutional layers, it suffices to apply the convolution to A itself (excluding bias). The proof is not reproduced here due to space restrictions.

2 Loss function and learning λ's

Let $[o_0, o_2, ...o_9]$ be the output layer of the neural network (the logits for the MNIST digit classification). Let $Z_{out} =: Z$ be the zonotope region at the output layer, for a given input region Z_{in}, and for given ReLU-transformation parameters λ.

Then the network is verifiably robust on the input region if:

$$\begin{aligned} &\forall (o_0, ..., o_9) \in Z, \forall i \in \{0, ..., 9\}, o_i \le o_t \\ \iff &\forall (x_0, ..., x_9) \in Z', \forall i \in \{0, ..., 9\}, x_i \le 0 \quad \iff \quad Z' \subset \mathbb{R}_-^{10} \end{aligned} \tag{*}$$

where Z' is the zonotope of the "violations" $[x_0, ..., x_9] := [o_0 - o_t, ..., o_9 - o_t]$. There are multiple ways of translating this property into a loss function.

Maximum violation (`zMaxViolation`)

$$\begin{aligned} (*) \iff &\forall x \in Z', \forall i, x_i \le 0 \\ \iff &\forall i, \forall x \in Z'_i, x_i \le 0 \\ \iff &\max_i \max_{x \in Z'_i} x_i \le 0 \end{aligned}$$

where Z'_i is the zonotope of x_i ($Z'_i = \{x_i; \exists y \in Z', y_i = x_i\}$). Since Z'_i are one-dimensional zonotopes, the innermost max can be computed in $O(1)$ by assigning all the error terms to the sign of the corresponding coefficients.

Sum of maximum individual violations (`zSumOfMaxIndividualViolations`)

$$\begin{aligned} (*) \iff &\forall i, \max_{x \in Z'_i} x_i \le 0 \\ \iff &\sum_i \left(\max_{x \in Z'_i} x_i \right)^+ \le 0 \end{aligned}$$

Note that $\max_{x \in Z'_i} x_i$ is just a scalar, so taking its positive part is easy.

Maximum sum of violations (`zMaxSumOfViolations`)

$$\begin{aligned}(*) &\iff \forall x \in Z', \forall i, x_i \leq 0 \\ &\iff \forall x \in Z', \sum_i x_i^+ = \sum_i \mathrm{ReLU}(x_i) \leq 0 \\ &\iff \max_{x \in Z'} \sum_i \mathrm{ReLU}(x_i) = \max Z'' \leq 0\end{aligned}$$

where Z'' is the zonotope of $\sum_i \mathrm{ReLU}(x_i)$ Note that computing this requires performing an additional ReLU zonotope transformation.

Recall that Z_{out} (and so Z'_i and Z'') depends on the ReLU-transformation parameters λ. Each of the possible loss functions: [1]

$$L(\lambda) = \max_i \max_{x_i \in Z'_i} x_i \qquad \text{(maximum violation)}$$

$$L(\lambda) = \sum_i \left(\max_{x \in Z'_i} x_i \right)^+ \qquad \text{(sum of maximum individual violations)}$$

$$L(\lambda) = \max Z'' \qquad \text{(maximum sum of violations)}$$

can be computed by propagating the input zonotope Z_{in} through the network. The network is verifiably robust if there exists λ such that $L(\lambda) \leq 0$.

Finally, $L(\lambda)$ is differentiable, so we use gradient-based methods to minimize it.

3 Optimizer selection

We used the optimizers from `pytorch.optim` to optimize the loss function $L(\lambda)$. To select which method and which hyperparameters (e.g. learning rate) to use, we performed a hyperparameter search using Ax. The criterion used was the verifier execution time (capped by a timeout). We were able to do this by using additional test cases, which we generated ourselves.

4 Test case generation and self-evaluation

We generated additional test cases by doing the following:

- Generate random datapoints of MNIST images and epsilons (randomly drawn in $[0.005, 0.2]$).
- Use adversarial attacks implemented in the ART library to classify them into "maybe robust" and "not robust".
- Make sure our verifier is sound, by running it on the "not robust" datapoints with a long timeout.
- Use our own verifier to further refine this classification, by running it on the "maybe robust" datapoints with a long timeout, thus yielding some "verifiable" datapoints.

[1] In practice, all three loss functions seemed to yield similar performances, with "maximum sum of violations" being slightly slower due to the additional ReLU transformation.