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 $a_0 \in R^d$ and a non-negative tensor $A \in R^{k \times d}$ 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 lambdas

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

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

$$\begin{aligned}\forall (o_0, ..., o_9) \in Z, \forall i \in \{0, ..., 9\}, o_i &\leq o_t \\ \forall (x_0, ..., x_9) \in Z', \forall i \in \{0, ..., 9\}, x_i &\leq 0 \\ \max_{x \in Z'} \max_i x_i &\leq 0 \\ \max_i \max_{x \in Z'_i} x_i &\leq 0\end{aligned}$$

where Z' is the zonotope of the "violations" $[o_0 - o_t, ..., o_0 - o_t]$, and Z'_i is the zonotope of $o_i - o_t$.

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.

Recall that Z'_i depends on the ReLU-transformation parameters λ. The loss function:

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

can be computed by propagating the input zonotope 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 grid 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.