# 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 $\lambda$'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 $\lambda$.

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

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

where $Z'_i$ are the zonotopes of the "violations" $x_i := o_i - o_t$. 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 $Z_{out}$ (and so $Z'_i$ and $Z''$) depends on the ReLU-transformation parameters. So the loss function

$$L(\lambda) = \max_i \max_{x_i \in Z'_i} x_i \tag{1}$$

can be computed by propagating the input zonotope $Z_{in}$ through the network. The network is verifiably robust if there exists $\lambda$ 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.