Reliable and Interpretable Artificial Intelligence
Project Report
Autumn Semester 2019

Francesco Saverio Varini & Jakob Beckmann

December 17, 2018

1 Abstract

The intention of this report is to discuss about how it has been tackled the trade off between a fast but imprecise Interval Propagation analysis versus the slow but precise Linear Programming analysis on the output verification a network.
Clearly, given the constraints for the project, namely the possibility to use only Gurobi linear programming and ELINA to get better approximations of the nework output verification, some strategy and heuristics had to be taken into account.

1.1 Full Layerwise Linear Programming

For the smallest network, strictly speaking, the mnist_relu_3_10, mnist_relu_3_20, mnist_relu_3_50, mnist_relu_6_20, mnist_relu_6_50 and mnist_relu_6_100, no particular strategy has been taken into account, since all the networks above run with layerwise linear programming with all the possible epsilon between 0 and 0.1 in less than the timeout of 7 minutes. Therefore, it was possible to obtain the best approximations in terms of verification using only Gurobi Linear programming.

1.2 Heuristics

For the bigger networks, namely, mnist_relu_4_1024, mnist_relu_6_200, mnist_relu_9_100, mnist_relu_9_200 different heuristics were deployed in order to speed up the verification using less LP variables and constraints, still improving as much as possible the precision.

1.2.1 Neuronwise Heuristic

Produce a scoring mechanism that scores a neuron based on some importance criteria and thus chooses the best neurons on which to perform linear programming per layer.

1.2.2 Weight Scores Heuristic

This heuristic looks at a neuron's outgoing weights and the neurons's bounds to determine its score. The score is obtained by multiplying the neuron's outgoing weights times either its upper bound or the lower bound or the absolute difference between these. The scoring policy is quite slow in the current implementation.

1.2.3 Moving Window Linear Programming Heuristic

This heuristic, as the name suggests, consists on performing linear programming on an arbitrary window of layers of the network. This partial model is then moved across the network as if it were a window. This heuristic verifies with very high precision but takes fractions of the time of the full layerwise linear programming.

1.2.4 Recursive Back-Prop of High impact neurons

This heuristic takes inspiration from the standard backpropagation procedure while training the neural networks. Firstly, it is performed the fast interval propagation to have a general idea of the intervals each neuron inside the network can take. Then, starting at the output layer:

- **1** For each neuron n in the layer l, check which neurons in the previous layer $l-1$ can affect its value the most. This is performed by checking the possible interval size of each incoming neuron m and multiplying it by the weight between m and n. This gives a general extimation how much m affect the output interval of n. The scoring policy can be one of the one explained in the Neuronwise heuristic.
- **2** Based on the scores computed in the previous step, take the highest capacity neurons that affect neuron n and store them.
- **3** Compute the union of all high impact sets returned.
- **4** Repeat from step 1 using the previous layer (layer that so far contained the m neurons), but only check for high impact neurons in the list returned from step 2.

Back-propagate as explained above until the input layer is reached.
The algorithm end up having a list of high impact neuron sets for each layer. Therefore, starting at the first hidden layer, linear programming is computed to better approximate the bounds of the high impact neurons. For the non-high impact neurons within the current layer, interval propagation is performed in order to approximate their bounds.
By default, all neurons in the output layer are considered "high impact", hence linear programming will always be performed on all output neurons.

1.2.5 Final Remarks