CS781 Practice Problem Set 1 (Autumn 2023)

- *Be brief, complete and stick to what has been asked.*
- *Untidy presentation of answers, and random ramblings will be penalized by negative marks.*
- *Unless asked for explicitly, you may cite results/proofs covered in class without reproducing them.*
- ***If you need to make any assumptions, state them clearly.***

1. Consider a DNN N with $k \geq 3$ layers (layer 1 is the "input" layer, and layer k is the "output" layer). For layer i of the network, let $\mathcal{L}_{i-1}$ be the input domain and $\mathcal{L}_i$ be the output domain. Thus, the input domain of the overall network N is $\mathcal{L}_0$ and its output domain is $\mathcal{L}_k$.

   Suppose the transformer associated with the slice of the network from layers i through j, both inclusive, is $\nu_{i,j} : \mathcal{L}_{i-1} \to \mathcal{L}_j$. So, the overall transformer of the DNN N is $\nu_{1,k}$.

   Let $D_{i-1} : \mathcal{L}_{i-1} \times \mathcal{L}_{i-1} \to \mathbb{R}^{\geq 0}$ be a distance metric that maps a pair of inputs of the i^{th} layer to a non-negative real number, such that $D_{i-1}(I, I) = 0$ for every $I \in \mathcal{L}_{i-1}$. For $I, J \in \mathcal{L}_{i-1}$, we say $D_{i-1}(I, J)$ is the *distance* between the inputs I and J of layer i.

   For every layer $i < k$, we also have four parameters $\varepsilon_{1,i}, \varepsilon_{i,k}, \delta_{1,i}, \delta_{i,k} \geq 0$ such that the following two Hoare triples hold:

   - $\{D_0(I_0, J_0) \leq \varepsilon_{1,i}\}\ I_i \leftarrow \nu_{1,i}(I_0);\ J_i \leftarrow \nu_{1,i}(J_0);\ \{D_i(I_i, J_i) \leq \delta_{1,i}\}$
   - $\{D_i(I_i, J_i) \leq \varepsilon_{i,k}\}\ I_k \leftarrow \nu_{i+1,k}(I_i);\ J_k \leftarrow \nu_{i+1,k}(J_i);\ \{I_k = J_k\}$.

   (a) Let $A = \{i \mid \delta_{1,i} \leq \varepsilon_{i,k}\}$. What can you say about the input-output behaviour of the overall network N if:

   i. $A \neq \emptyset$?

   ii. $A = \emptyset$?

   Explain your answers clearly with reasons.

   (b) Suppose further that we know that there exist inputs $I^\star, J^\star$ such that $\nu_{1,k}(I^\star) \neq \nu_{1,k}(J^\star)$. Indicate with reasons which of the following are necessarily true.

   i. There is at least one $i < k$ such that $D_i(\nu_{1,i}(I^\star), \nu_{1,i}(J^\star)) \geq \delta_{1,i}$.

   ii. There is at least one $i < k - 1$ such that $D_i(\nu_{1,i}(I^\star), \nu_{1,i}(J^\star)) \geq \varepsilon_{i+1,k}$.

   iii. $D_{k-1}(\nu_{1,k-1}(I^\star), \nu_{1,k-1}(J^\star))$ cannot be arbitrarily large.

2. Consider an image classification DNN $\mathcal{N}$ with an associated input-output transformer ν. Suppose the input domain, denoted **Im**, consists of 32 pixel $\times$32 pixel gray-scale images of animals, where each pixel is a real number (hence, each pixel can take infinitely many values). Suppose the output domain is the set of labels $\{\text{Cat}, \text{Dog}, \text{Other}\}$.

   Let $\Delta : \mathbf{Im} \times \mathbf{Im} \to \mathbb{R}^{\geq 0}$ be a carefully designed image similarity metric that maps a pair of input images to a non-negative real number, also called their *similarity score*. Let $\varepsilon > 0$ be a small positive

real number, called *similarity threshold*. You are told that for every pair of input images $I_1, I_2 \in \mathbf{Im}$, the following hold: (i) $\Delta(I_1, I_2) = 0$ iff $I_1 = I_2$, and (ii) $\Delta(I_1, I_2) = \Delta(I_2, I_1)$. Additionally, for every finite set of images $\{I_1, I_2, \ldots I_k\} \subseteq \mathbf{Im}$, there exists at least one image $I_{k+1} \in \mathbf{Im}$ such that $\bigwedge_{j=1}^{k} \left(\Delta(I_{k+1}, I_j) > M\right)$ holds, where $M = \max_{i,j \in \{1,\ldots k\}} \Delta(I_i, I_j)$.

(a) We want a network like $\mathcal{N}$ to not classify similar looking images differently. A convenient way of expressing this is via the Hoare triple HT_1:

$$\{\Delta(I_1, I_2) < \varepsilon\} \quad \ell_1 = \nu(I_1); \ell_2 = \nu(I_2); \quad \{\ell_1 = \ell_2\}.$$

However, we have seen in the lectures that HT_1 may (rather unexpectedly) require $\mathcal{N}$ to classify everything with the same label, rendering HT_1 meaningless.

It turns out, however, that under certain conditions on the input image space, the triple HT_1 can indeed be meaningful and can express our intuitive ask, i.e. similar looking images should be classified the same.

Give a first order logic sentence (i.e. formula with no free variables), say α, over elements of **Im** such that if **Im** satisfies α, then indeed HT_1 specifies the desired (or intended) property of $\mathcal{N}$. Give as weak a sentence α as you can, so that the restriction on **Im** is as mild as possible. Explain your answer clearly.

(b) A student has written the following Hoare triple HT_2 for $\mathcal{N}$:

$$\{\Delta(I_1, I_2) > \varepsilon\}\ \ell_1 = \nu(I_1); \ell_2 = \nu(I_2)\ \{\ell_1 \neq \ell_2\}$$

Intuitively, the student wishes to express the property that if two input images are hugely dissimilar, then they should not be classified the same.

Prove that if HT_2 holds, then for every pair of images $I_1, I_2 \in \Im$, we must have $\Delta(I_1, I_2) < \varepsilon$. In other words, the only way for HT_2 to hold is vacuously, i.e. the pre-condition itself is unsatisfiable.

3. Consider the DNN shown in Fig. 1 below. Assume that each node in the hidden and output layers uses a ReLU activation function.

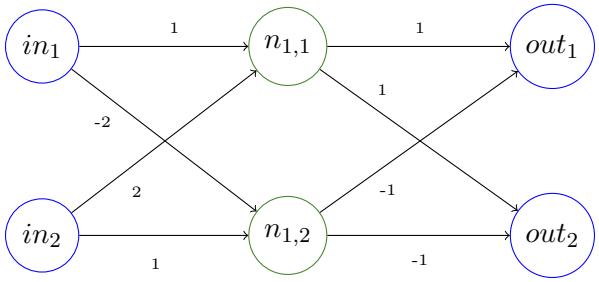

Figure 1: A simple DNN

(a) Write the constraints describing the relation between the inputs and outputs of all nodes in the hidden and output layers of this network.

(b) Let us call a conjunction of linear (in)equalities a *linear program*. For the DNN in Fig. 1, we wish to write an equivalent (not approximate) system of constraints that is a disjunction of linear programs, i.e. a disjunction of conjunction of linear (in)equalities. How many linear programs need to be reasoned about in general if we were to analyze the DNN in Figure 1 being studied in this question. reason about in order to analyze the DNN? Give clear explanation for your answer.

(c) Using ideas discussed in the lectures, write as good a linear program as you can that over-approximates the behaviour of the DNN in Fig. 1.

4. In the DeepPoly paper, we saw how a ReLU and a few other non-linear activation functions can be bounded by linear expressions. A saturating and leaky ReLU is defined as follows:

$$g(x) = \max\big(\beta, \alpha.x, \min(\gamma, x)\big), \tag{1}$$

where $\beta < 0$, $0 \leq \alpha < 1$ and $\gamma > 0$.

The input-output behaviour of such a ReLU is plotted in Fig. 2.

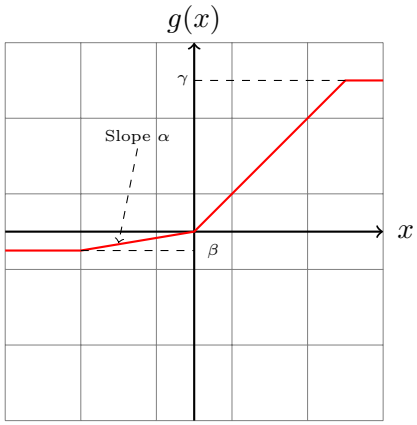

Figure 2: A saturating leaky ReLU

Indicate with justification how you would bound $g(x)$ by a linear expression in x (one each for lower bound and upper bound) to obtain the best 4-tuple abstraction (as used in the DeepPoly paper) of the relation between $g(x)$ and x. Since a saturating leaky ReLU becomes indistinguishable from a ReLU when $\beta = 0$ and $\gamma = \infty$, you must ensure your bounds reduce to those given in the DeepPoly paper when $\beta = 0$ and $\gamma = \infty$ in your bounds.

5. Consider the neural network shown in Fig. 3. This network consists of 1 input layer, 1 hidden layer and 1 output layer. Assume that all hidden and output layer nodes have bias 0, and ReLU functions are used only for hidden layer nodes.

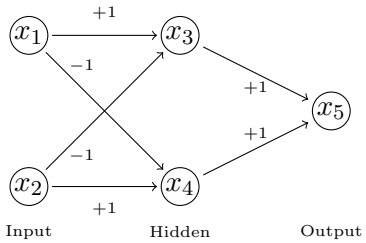

Figure 3: Simple neural network

In our study of DeepPoly, we have seen that approximations used in representing ReLUs can cause *conservative* intervals to be computed for outputs in general, i.e. the interval computed for an output contains at least one value that cannot actually appear at the output.

Under certain input interval conditions however, this conservatism can be avoided and we can compute *precise* intervals, i.e. all values in the interval computed for an output can indeed appear at the output. One such obvious condition is when the input intervals are such that the lower and upper bounds of the input of each ReLU are both non-negative or are both negative. Indeed, DeepPoly does not introduce approximations in representing ReLUs under this condition. We will call this condition SimpleCond for convenience.

Give an interval for each input of the network in Fig. 3 such that SimpleCond described above is not satisfied, and yet the interval computed for the output by DeepPoly is precise.

You must give the bounding inequalities and bounding constants for each hidden layer node and output layer node, as calculated by DeepPoly, starting from your input intervals. If you need to make any assumptions, state them clearly.

6. Suppose the ReLUs used in the nodes of the hidden layer in Fig. 3 are replaced by SpecialReLUs that have the following functionality: SpecialReLU$(x) = \max(0, \min(1, x))$.

   Assume that the inputs to the network are restricted to the set $\{0, 1\}$ and the output node uses a SpecialReLU after adding the sum of the hidden layer nodes. It turns out that in this case, the output of the neural network can be expressed as a propositional formula in terms of the inputs.

   Give a propositional formula φ expressing the output of the neural network as a function of its inputs.

7. The most time-consuming step in DeepPoly is propagation of affine constraints at every layer L back to the input layer in order to find constant bounds of neuron values. One way of limiting the time complexity of this step is to propagate the affine constraints back upto a fixed number, say K, of layers, instead of propagating back all the way to the input layer. Let us call this technique *K-bounded DeepPoly*.

   Show by means of a counter-example that K-bounded DeepPoly can yield unachievable constant bounds of the output values of neurons *even when there are no non-linear activation functions (like ReLUs) in the neural network.*

   To keep your answer simple, you may choose $K = 1$, i.e. affine constraints are propagated back only a single layer. Your answer should provide a specific neural network N without ReLUs (or other non-linear activation functions) along with intervals for each input, and show that there is at least one neuron in N for which the constant bounds obtained using 1-bounded DeepPoly are conservative and not achievable.

8. Consider the neural network shown in Fig. 4. Nodes 3, 4, 7 and 8 in this network linearly aggregate outputs of neurons from the previous layers and add a bias (b_3, b_4, b_7, b_8 respectively) to the resulting linear sum. There are two ReLUs, represented by the dotted lines, with outputs x_5 and x_6 respectively. We wish to apply the CROWN algorithm studies in class to identify if $x_7 > x_8$, when $-1 \leq x_1, x_2 \leq 1$.

   Assume that for each ReLU of the form $z = ReLU(w)$, where $l \leq w \leq u$ and $l < 0 < w$, we use the following inequality to bound the output from above: $z \leq \frac{u}{u-l}\big(w - l\big)$. Similarly, assume that the lower bound of the output is given by $z \geq \alpha w$, where $0 \leq \alpha \leq 1$ is a parameter that can be different for each ReLU.

   For the ReLUs with outputs x_5 and x_6 in Fig. 4, let the α parameters used in the lower bounds be denoted α_5 and α_6 respectively. Assume $b_3 = 1, b_4 = 0, b_7 = 1$ and $b_8 = -1$.

   (a) Use a simple interval propagation forward pass to find constant lower and upper bounds of x_3 and x_5.

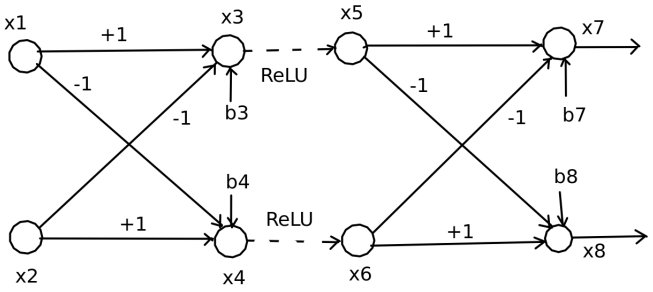

Figure 4: Neural network

(b) Let **In** denote the vector $(x_1, x_2)^T$ and let **Out** denote the vector $(x_7, x_8)^T$. Using the bounds obtained in part (a), and assuming $\alpha_5 = \alpha_6 = 0.5$, compute 2×2 matrices $\mathbf{\Lambda}^{(0)}$, $\mathbf{\Omega}^{(0)}$, and 2×1 vectors $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ as in the CROWN algorithm such that

$$\mathbf{\Omega}^{(0)}\mathbf{In} + \boldsymbol{\nu} \;\leq\; \mathbf{Out} \;\leq\; \mathbf{\Lambda}^{(0)}\mathbf{In} + \boldsymbol{\mu}, \tag{2}$$

where the inequalities are evaluated component-wise.

(c) Now suppose the values of α_5 and α_6 are not known, i.e. assume them to be symbols. Find symbolic expressions for the lower and upper bounds of x_7 and x_8 in terms of x_1, x_2, α_5 and α_6.

(d) Recalling that every value of α_5, α_6 in $[0, 1]$ yield sound bounds of x_7 and x_8 as long as $0 \leq \alpha_5, \alpha_6 \leq 1$, find the best numeric lower and upper bounds of $x_7 - x_8$. You must indicate values of α_5 and α_6 that allow you to compute these numeric bounds.