DeepPoly - An abstract domain for certifying NN

Tuesday, 3 November 2020 10:35

- State-of-the-art methods for NN verification based on:
  1. SMT solving: Reluplex [Katz'17]
     precise, but can only handle very small networks.
  2. Abstract interpretation: AI2 [Gehr'18]
     relies on existing generic abstract domains
     which either ① like convex polyhedra do not scale to larger NNs.
     or ② like Zonotope are too imprecise.
  3. Linear approximations Towards Fast computation of certified robustness for ReLU networks [Weng'18]
     scales better but can only handle feedforward networks, can not handle convolutional networks

  both ① and ③ unsound for floating point arithmatics, and thus they suffer from false negatives.
- DeepPoly: is a novel abstract interpreter specifically tailored to the setting of neural networks; uses an abstract domain which is a combination of ① floating-point polyhedra ② with intervals, coupled with abstract transformers for common NN functions.
- Adversarial region; is the set of possible images induced by a modification of the original input to a perturbed version inside L_∞ ball.
  DeepPoly automatically prove that the output for all the inputs in the adversarial region is correct by soundly propagating the entire input adversarial region through the abstract transformers of the network.
- Abstract transformers for the output of a neuron are point-wise (can be computed in parallel)
  abstract interpretation on GPU Differentiable abstract interpretation [Mirman'18]
- The context:
  - verifying robustness for adversarial regions that can be represented using a set of interval constraints.
    $X = X_{i=1}^{m} [l_i, u_i]$ for $l_i, u_i \in \mathbb{R} \cup \{-\infty, +\infty\}$ and m is the number of the inputs.
  - NN is represented as a sequence of assignments ① ReLU $x_i \leftarrow \max(0, x_j)$
    ② sigmoid/tanh $x_i \leftarrow g(x_j)$ for sigmoid $g = \sigma(x) = \frac{e^x}{e^x+1}$ for tanh $g = \tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$
    ③ affine $x_i \leftarrow v + \sum_j w_j \cdot x_j$, can describe convolutional layers.
- Abstract domain and transformers:
  - Abstract domain A_n contains elements $a \in A_n$ consist of a set of polyhedral constraints over n variables
    each constraint relates one variable → a linear combination of variables of a smaller index.
    each variable has two concrete constraints lower bound l and upper bound u
    $a_i^{\leq}, a_i^{\geq}$ are the lower and upper relational polyhedra constraints for x_i.
  - An abstract element has additionally derived interval bounds $a^{\leq}, a^{\geq}$ → $a \in A_n = \langle a^{\leq}, a^{\geq}, l, u \rangle$
    where $a_i^{\leq}, a_i^{\geq} \in \{x \mapsto v + \sum_{j=1}^{i-1} w_j x_j \mid v \in \mathbb{R}, w \in \mathbb{R}^{i-1}\}$ for $i = \{1, \ldots, n\}$, $l, u \in \mathbb{R}^n$
  - The concretization function $\gamma_n: A_n \rightarrow P(\mathbb{R}^n)$ is given by:
    $$\gamma_n(a) = \{x \in \mathbb{R}^n \mid \forall i = \{1, \ldots, n\} : (a_i^{\leq}(x) \leq x_i \leq a_i^{\geq}(x))\}$$
  - Domain invariant: every abstract element in the abstract domains maintains concrete lower and upper bounds, i.e. $\gamma_n(a) \subseteq X_i [l_i, u_i]$ which over approximate the symbolic bounds.
  - Abstract transformers $T_f^{\#}$ for a deterministic function $f: A^m \rightarrow A^n$ satisfy the soundness property
    $T_f(\gamma_n(a)) \subseteq \gamma_n(T_f^{\#}(a))$ for all $a \in A^m$ where T_f is a concrete transformer $T_f(X) = \{f(x) \mid x \in X\}$
    - all variables are ① bounded, ② assigned exactly once (in increasing order of thier indices.
      $T_f^{\#}(\langle a^{\leq}, a^{\geq}, l, u \rangle) = \langle a'^{\leq}, a'^{\geq}, l', u' \rangle$
      for $k < i$ outputs $\langle a_k^{\leq}, a_k^{\geq}, l, u \rangle$

1. ReLU Abstract transformer: $f: \mathbb{R}^{i-1} \rightarrow \mathbb{R}^i$ executes $x_i \leftarrow \max(0, x_j)$ for some $j < i$

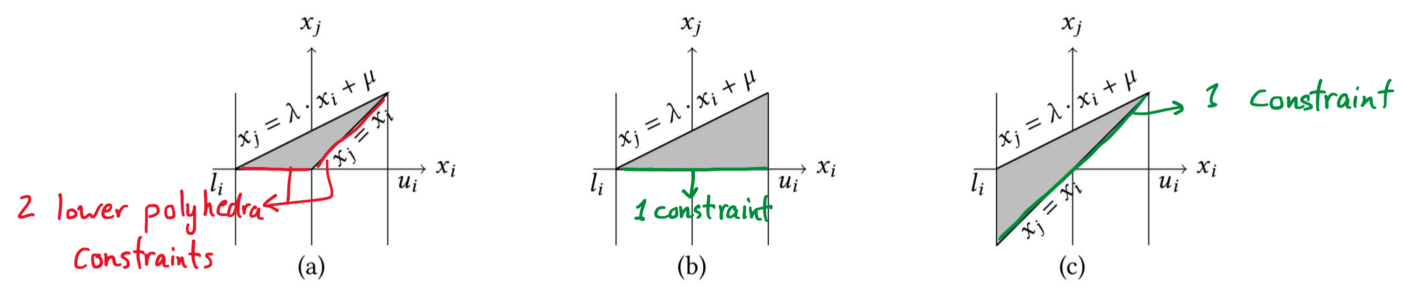

Fig. 4. Convex approximations for the ReLU function: (a) shows the convex approximation with the minimum area in the input-output plane, (b) and (c) show the two convex approximations proposed in this paper. In the figure, [redacted]

for new components i
- → if $u_j \leq 0$ then $\langle 0, 0, 0, 0 \rangle$ exact
- → if $0 \leq l_j$ then $\langle x_j, x_j, l_j, u_j \rangle$ $x_i \leq x_j \leq x_i$
- → otherwise $\langle u_j \frac{x_j - l_j}{u_j - l_j}, \lambda x_j, \lambda l_j, u_j \rangle$ $\lambda \in \{0, 1\}$ select that minimize the area. (λx_j → 0 or x_j)

2. Sigmoid and tanh abstract transformer: $\lambda = \frac{g(u_j) - g(l_j)}{u_j - l_j}$, $\lambda' = \min(g'(l_j) - g'(u_j))$; $g'(x) > 0$
   for new component i
   - → if $u_j \leq 0$ then $\langle g(\underline{u}_j) + \underline{\lambda}(x_j - \underline{u}_j), g(\underline{u}_j) + \underline{\lambda}'(x_j - \underline{u}_j), g(l_j), g(u_j) \rangle$
   - → if $0 < l_j$ then $\langle g(\underline{l}_j) + \underline{\lambda}(x_j - \underline{l}_j), g(\underline{l}_j) + \lambda'(x_j - \underline{l}_j), g(l_j), g(u_j) \rangle$
3. Max Pool abstract transformer: $x_i \leftarrow \max_{j \in J} x_j$ for some $J \subseteq \{1, \ldots, i-1\}$
   for new components
   - → $\exists k \in j : u_j < l_k$ for all $j \in J / \{k\}$ then $\langle x_k, x_k, l_k, u_k \rangle$
   - → otherwise we choose $k \in J$ that maximize l_k then $\langle x_k, \max_{j \in J} u_j, l_k, \max_{j \in J} u_j \rangle$
4. Affine abstract transformer: $x_i \leftarrow v + \sum_{j=1}^{i-1} w_j x_j$
   $a_i'^{\leq}(x) = a_i'^{\geq}(x) = v + \sum_{j=1}^{i-1} w_j x_j$
   to obtain l_j' : 1. $b_1(x) = a_i'^{\leq}(x)$
   2. $b_s(x) = v' + \sum_{j=1}^{k} w_j' x_j$ for some $k \in \{1, \ldots, i-1\}$
   3. $b_{s+1}(x) = v' + \sum_{j=1}^{k} (\max(0, w_j') \cdot a_j'^{\leq}(x) + \min(w_j', 0) \cdot a_j'^{\geq}(x))$
   4. we iterate until reaching $b_{s'}$ (s' is the smallest number with this property)
      $l_i' = b_{s'}(x) = v''$

   to obtain u_j' : 1. $c_1(x) = a'^{\geq}(x)$
   2. $c_t(x) = v' + \sum_{j=1}^{k} w_j' x_j$
   3. $c_{t+1}(x) = v' + \sum_{j=1}^{k} (\max(0, w_j') \cdot a_j'^{\geq}(x) + \min(0, w_j') \cdot a_j'^{\leq}(x))$
   4. iterate until reaching $c_{t'}$ → $u_i' = c_{t'}(x) = v''$

- NN robustness analysis procedure:
  1. create an abstract element over the input variables $a = \langle l_i, u_i, l_i, u_i \rangle$
     where all l_i & u_i are initialized to describe the adversarial region.
  2. processing assignments for all hidden and output variable
  3. processing nodes in ascending order of variable indices, using their abstract transformers.
  4. execute the following (r-1) assignment (p inputs, q hidden, r outputs, k class index)

     $x_{p+q+r+1} \leftarrow x_{p+q+k} - x_{p+q+1}, \ldots, x_{p+q+r+(k-1)} \leftarrow x_{p+q+k} - x_{p+q+(k-1)}$ 1 --- r-1
     $x_{p+q+r+k} \leftarrow x_{p+q+k} - x_{p+q+(k+1)}, \ldots, x_{p+q+r+(r-1)} \leftarrow x_{p+q+k} - x_{p+q+r}$ 1 --- r

     robust if $l_i > 0$ for all output variables.
- Correctness of abstract transformers: for $a' = T_f^{\#}(a)$
  we have to prove they are ① sound $T_f(\gamma_{i-1}(a)) \subseteq \gamma_i(a')$
  and ② they preserve invariant $\gamma_i(a') \subseteq X_{k \in \{1, \ldots, i\}} [l_k, u_k]$
- Soundness under floating point arithmatics:
  Relational abstract domains for the detection of floating-point run-time errors [Miné'04]
  $c^-, c^+ \in \mathbb{F}$ for a real constant c to denote the floating-point representation of c with rounding towards $-\infty$ and $+\infty$ respectively.
  → standard interval linear form; coefficients in the constraints are intervals $[c^-, c^+]$
  upper bound and lower bound $\inf(a^{\leq}(x))$, $\sup(a^{\geq}(x))$