Lecture 13: Certified Defenses II: Convex Relaxations

November 11, 2019

In the last lecture, we discussed *exact* certification defenses, which definitively tell you if your model has an adversarial perturbation at any given point. Unfortunately, as we saw, exact certification is very computationally expensive, and although there are optimizations which have been able to scale it to models on CIFAR-10, it is unclear how to hope to scale it up much further. To address this, researchers have developed a number of *relaxed* certification methods, which are much more scalable, but do not yield tight certification guarantees. Recall that, given a classifier $F : \mathbb{R}^d \to \mathcal{Y}$, a labeled point (X, y), and a perturbation set $\mathcal{P}$, a relaxed certification algorithm $\mathcal{A}$ is allowed to return one of three options:

- YES: Here, we must have that for all $X' \in \mathcal{P}(X)$, we have that $F(X) = F(X') = y$.
- NO: Here, there must exist $X' \in \mathcal{P}(X)$ so that $F(X') \neq F(X)$.
- MAYBE: In this case we have no guarantees.

For a classifier f, a distribution over data $\mathcal{D}$, a perturbation set $\mathcal{P}$, and a relaxed certification algorithm $\mathcal{A}$, we define its certified accuracy to be

$$\text{Cert}(f, \mathcal{D}, \mathcal{P}, \mathcal{A}) = \Pr_{(X,y)\sim\mathcal{D}} [\mathcal{A}(f, X, y) = \text{YES}] \ . \tag{1}$$

That is, while the algorithm is always allowed to output MAYBE, it does not count towards its certifiable accuracy.

In this class, we will focus on approaches for certifiable defenses based on convex relaxations. Again these defenses are mainly focused on certifying feedfowards ReLU neural networks. Recall from last lecture that these are specified by a sequence of functions (the layers) $f_1, \ldots, f_\ell$ so that $f_i(x) = \sigma(W_i x + b_i)$, where W_i is a weight matrix (or tensor) the b_i are a set of biases, and σ is the entrywise ReLU operation, i.e. $\sigma(x)_i = \max(x_i, 0)$. The overall network representation will be given by $F : \mathbb{R}^d \to \mathbb{R}^m$ by

$$F(x) = W_{\ell+1} f_\ell \circ f_{\ell-1} \circ \ldots \circ f_1(x) \ ,$$

Let $f = f_\ell \circ f_{\ell-1} \circ \ldots \circ f_1$, that is everything but the last layer. Convex relaxation based approaches attempt to find an efficiently computable convex set which contains $f(\mathcal{P}(X))$,

which is typically a non-convex set. If they can do so, then they can just check if this convex hull crosses any of the decision boundaries, which is easy to do as this is a further convex problem. If it does not, then they can safely output YES, and otherwise they output MAYBE. The main difficulty is to find a convex relaxation of $f(\mathcal{P}(X))$ which is as tight as possible, to avoid outputting MAYBE as much as possible. In this lecture, we will discuss two such approaches that trade off between scalability and tightness of the relaxation.

1 LP Relaxations

The first approach we will discussed is based on linear programming. Recall that a linear program is specified by a linear objective function, as well as linear constraints on the variables, which can be either equalities or inequalities. To start with writing down an LP relaxation for this problem, let's first write out the certification problem more explicitly.

First, rather than considering the problem whether or not we can flip the label to any other label, we'll consider the problem of whether we can flip the label of X to a given label $y' \neq y$. To go from this to the first problem, one can simply iterate over all $y' \in \mathcal{Y}$ and check them each one-by-one. Let's also introduce some notation for the transformations encoded at every layer of the network. We will call the input to our network x_1, and inductively, we will define the variables

$$z_i = W_i x_i + b_i \ , \qquad \text{and} \qquad x_{i+1} = \sigma(z_i) \ , \tag{2}$$

for $i = 1, \ldots, \ell$, and finally let $z_{\ell+1} = W_{\ell+1} x_\ell + b_{\ell+1}$. Then, when the constraint set is the ℓ_∞ ball, the certification problem at (X, y) can be thought of as simply giving an upper bound on the value of the following mathematical program:

$$\min_{x_1, z_1, \ldots, x_{\ell+1}, z_{\ell+1}} c_{y'}^\top z_{\ell+1} \qquad \text{s.t. } \|X - x_1\|_\infty \leq \varepsilon \ , \text{ and } x_1, z_1, \ldots, x_{\ell+1}, z_{\ell+1} \text{ satisfy (2)} \ , \tag{3}$$

where $c_{y'}$ is the vector which is 1 at position y and -1 at position y'. In particular, if the solution to this problem is negative, then we can make class y' look more probable than y, and otherwise, we cannot.

Current LP based approaches for relaxed certification proceed by relaxing (3) layer-by-layer, i.e. by relaxing (2). Suppose we are at layer i. As was the case for exact certification, the main difficulty is formulating a way to encode the non-linearity, i.e. the ReLU constraint. The issue is that the set $\{(x, z) : z = \sigma(x)\}$ is not a convex subset of $\mathbb{R}^2$, where $\sigma(x) = \max(0, x)$ here denotes the univariate ReLU function. However, there is a natural relaxation of it. In particular, suppose we can get a convex function $\underline{\sigma}$ and a concave function $\overline{\sigma}$ so that

$$\underline{\sigma}(x) \leq \sigma(x) \leq \overline{\sigma}(x) \ .$$

Then the following is a convex relaxation of the ReLU:

$$\{(x, z) : z = \sigma(x)\} \subseteq \{(x, z) : \underline{\sigma}(x) \leq z \leq \overline{\sigma}(x)\} \ .$$

To instantiate this with ReLU, we need a bit more information. This is because if we don't restrict the range of x, then no such functions can exist (specifically, no $\overline{\sigma}(x)$) exists. Thus, suppose (as before with the MILP algorithm), we have lower and upper bounds on x, i.e. $x \in [L, U]$. Then, we can take:

$$\underline{\sigma}(x) = \sigma(x) \ , \qquad \text{and} \qquad \overline{\sigma}(x) = \frac{U}{U-L}(x - L) \ , \tag{4}$$

and it is easily verified that these are valid choices of $\underline{\sigma}$ and $\overline{\sigma}$.

Let's now instantiate this for layer i of the neural network, rather than just a single ReLU. Let x_i be the variable in our LP which is the input to the ith layer, and let z_i be the variable specified by the linear constraint

$$z_i = W_i x_i + b_i \ . \tag{5}$$

Suppose for now we are given vectors L_i and U_i so that if x_i is any valid input to the i-th layer, then $L_i \leq z_i \leq U_i$, where z_i is defined as above, and where the inequalities hold entrywise. Then, the constraints on the input to the next layer x_{i+1} are given by

$$x_{i+1,j} \geq 0 \ , \qquad z_{ij} \leq x_{i+1,j} \leq \frac{U_j}{U_j - L_j}(z_{ij} - L_j) \ , \text{for all } j \ . \tag{6}$$

To complete the picture, observe that the ℓ_∞ constraint on x_1 is easily encoded by linear constraints, namely:

$$x_j - \varepsilon \leq x_{1,j} \leq x_j + \varepsilon \ , \text{ for all } j \ . \tag{7}$$

Thus our final LP can be stated as:

$$\begin{aligned} \max_{x_1, z_1, \ldots, x_{\ell+1}, z_{\ell+1}} \ c_{y'}^\top z_{\ell+1} \qquad \text{s.t.} \qquad & x_j - \varepsilon \leq x_{1,j} \leq x_j + \varepsilon \ , \text{ for all } j \\ & z_i = W_i x_i + b_i \\ & x_{i+1,j} \geq 0 \ , \qquad z_{ij} \leq x_{i+1,j} \leq \frac{U_j}{U_j - L_j}(z_{ij} - L_j) \ , \text{for all } j \ . \end{aligned} \tag{8}$$

This LP is known often as the *primal* form of the certification LP, and has appeared (or something similar has appeared) in a number of papers, see e.g. [1, 2, 3, 4, 5, 6].

All that remains is to find L_i and U_i. One way to do this is via interval arithmetic as described in the previous lecture, but a tighter bound can be derived inductively via our LP. Suppose we have $L_1, \ldots, L_k$ and $U_1, \ldots, U_k$ (the base case is similar). Then, the observation is that a partial form of (10) can be used to derive lower and upper bounds on z_{i+1}. Indeed, to derive an upper bound on the jth coordinate of z_{i+1}, one can verify that the following suffices:

$$\begin{aligned} U_{(k+1)j} \leq \max_{x_1, z_1, \ldots, x_k, z_k, x_{k+1}} \ & W_{(k+1)j} x_{k+1} + b_j \\ \text{s.t.} \qquad & x_j - \varepsilon \leq x_{1,j} \leq x_j + \varepsilon \ , \text{ for all } j \\ & z_i = W_i x_i + b_i \\ & x_{i+1,j} \geq 0 \ , \qquad z_{ij} \leq x_{i+1,j} \leq \frac{U_j}{U_j - L_j}(z_{ij} - L_j) \ , \text{for all } j \ , \end{aligned} \tag{9}$$

where $W_{(k+1)j}$ is the j-th row of W_k. In other words, if we simply replace the objective with the appropriate row of W_{k+1}, this gives an upper bound for the j-th coordinate of U_{k+1}. Doing this for each coordinate yields a fairly tight bound for U_{k+1}. Replacing the maximization with a minimization then yields a lower bound on how small each coordinate of Ł$_{k+1}$ needs to be.

The full workflow of the algorithm is then very straightforward: first, using (9), inductively compute $L_1, \ldots, L_\ell$ and $U_1, \ldots, U_\ell$, then plug those numbers to feed into (10). This is, in some sense, the "optimal" layer-wise LP relaxation of the problem, however it can be quite slow to compute. To alleviate this, many papers apply additional approximations to simplify the preprocessing time even further.

Dual LP formulations Another way to formulate the certification problem as an LP is to take the Lagrange dual of the original certification problem, and write an LP which relaxes this [7]. While it can be shown that the full LP written this way has the same certification power as the primal LP [6] (which one might expect due to strong duality of LPs), this is notable because [7] demonstrate that the dual of the LP can be solved (with some additional approximations) quite efficiently as a single backpropagation step. This is notable because previous LP formulations are still quite cumbersome to solve, and additionally, the fact that the dual can be implemented in backpropagation means that they can train to maximize the certified loss using their certification procedure. By directly training for their certification technique, this allows them to achieve much better numbers.

Limitations How effective are these methods? There are generally two bottlenecks to scaling LP-based methods. The first (although [7] makes strides to improve this) is runtime. In particular, if one wishes the use the full LP 10, this is quite computationally intensive: [6] is able to evaluate the certified accuracy of the full LP, but only by leveraging extensive computational resources.

The second issue is the loss in certification accuracy due to relaxation. This appears to be a major problem: [6] demonstrates that even on MNIST, the bounds that the LP relaxation are able to achieve are quite loose, even if one takes the optimal layer-wise LP relaxation of the certification problem, which we described above.

2 SDP Relaxations

The second approach we will discuss is an attempt to produce a tighter relaxation of the problem, using stronger convex relaxations, namely semi-definite programming (SDP). In this lecture, we'll follow the presentation of [8, 9]. Before we begin, let's first recap what SDPs are.

2.1 A crash course on semi-definite programming

Semidefinite programming is a generalization of linear programming to allow for matrix constraints. The following is a general formulation of SDPs. Recal that if X is a symmetric matrix, then $X \succeq 0$ means that all of its eigenvalues are greater than zero.

$$\begin{aligned} &\max_X \langle C, X \rangle \\ &\quad \text{s.t.} \langle A_r, X \rangle \geq b_i \qquad \text{for } r = 1, \ldots, k \\ &\quad X \succeq 0 \ . \end{aligned}$$

Here $C, A_1, \ldots, A_k$ are matrices, and $\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij}$ is the trace inner product, and X is a symmetric matrix. In other words, we allow for linear inequalities and equalities on the matrix variable X, but we also can enforce that it is positive semi-definite, which is a highly non-linear constraint.

The key property of SDPs is that they are a generalization of LPs, but which are still solvable in polynomial time (as opposed to, say, MILP). What we'll be mostly interested in is whether or not we can use this additional power to get stronger certification guarantees. Of course, this does come at some cost: the runtime of these algorithms will be slower than generic LP solvers.

2.2 A generic way to relax quadratic programs to SDPs

How do we use SDPs to encode certification? It turns out that there is a general way to take a quadratic program—a program whose constraints can be degree two polynomials in its variables—and relax it to an SDP. Formally, suppose we are given the following mathematical program over the variables $x_1, \ldots, x_n$:

$$\begin{aligned} &\max_{x_1, \ldots, x_n} \sum_{ij} c_{ij} x_i x_j \\ &\quad \text{s.t.} \sum_{ij} a_{ij}^{(r)} x_i x_j \geq b_i \qquad \text{for } r = 1, \ldots, k \ . \end{aligned}$$

In general, such a program is NP-hard to solve. However, we can always relax it to an SDP as follows. For every $i, j \in [n]$, introduce a variable X_{ij} which is supposed to be $x_i x_j$. Note that X is a symmetric matrix, and the constraints are linear in the variables of X. Moreover, the constraint that $X_{ij} = x_i x_j$ is equivalent to the constraint that X is rank 1 and PSD. Thus, the solution to the above problem is equivalent to the solution to:

$$\begin{aligned} &\max_X \langle C, X \rangle \\ &\text{s.t.} \langle A_r, X \rangle \geq b_i \qquad \text{for } r = 1, \ldots, k \\ &\quad X \succeq 0 \\ &\quad \text{rank}(X) = 1 \ , \end{aligned}$$

where C is the matrix given by $C_{ij} = c_{ij}$, and A_r is the matrix with entries $(A_r)_{ij} = a_{ij}^{(r)}$. Now the only non-convex constraint here is the final one, and the SDP relaxation is just to ignore it, so that the final relaxation is simply:

$$\begin{aligned} &\max_X \langle C, X \rangle \\ \text{s.t. } &\langle A_r, X \rangle \geq b_i \qquad \text{for } r = 1, \ldots, k \\ &X \succeq 0 \, . \end{aligned}$$

2.3 Relaxing certification to an SDP

Per the above discussion, it suffices to write certification of a neural network as a quadratic program. Again, the main difficulty is encoding the ReLU. We'll follow the same basic approach of encoding the problem layer-wise. Suppose the input to level i is x_i. As we before we can write $z_i = W_i x_i + b$. Then the constraint that $x_{i+1} = \sigma(z_i)$ it equivalent to the following set of quadratic constraints:

$$x_{i+1} \geq 0 \, , \qquad x_{i+1} \geq z_i \, , \qquad x_{i+1} = x_{i+1} \odot z_i \, ,$$

where $\odot$ denotes entrywise product between vectors, and all inequalities are taken entrywise. Therefore we can state the certification problem as the following quadratic program:

$$\begin{aligned} \max_{x_1, z_1, \ldots, x_{\ell+1}, z_{\ell+1}} \quad c_{y'}^\top z_{\ell+1} \qquad \text{s.t.} \qquad & x_j - \varepsilon \leq x_{1,j} \leq x_j + \varepsilon \text{ , for all } j \\ & z_i = W_i x_i + b_i \\ & x_{i+1} \geq 0 \, , \qquad x_{i+1} \geq z_i \, , \qquad x_{i+1} = x_{i+1} \odot z_i \, . \end{aligned} \tag{10}$$

To write this as an SDP, we then simply use the generic reduction described above. We can also incorporate upper and lower bounds on the inputs to the layers by adding an additional quadratic constraint.

2.4 Tradeoffs

The obvious limitation to this method is that SDP methods are much slower to compute. Indeed the current SOTA is only able to scale to MNIST. However, this does present the potential of circumventing the LP lower bounds in [6]; indeed, [9] shows simple scenarios where this provably does better than the LP based certification methods.

References

[1] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5273–5282, 2018.

[2] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pages 10802–10813, 2018.

[3] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pages 6367–6377, 2018.

[4] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2018.

[5] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pages 4939–4948, 2018.

[6] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robust verification of neural networks. *arXiv preprint arXiv:1902.08722*, 2019.

[7] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[8] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[9] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10877–10887, 2018.