1 Adversarial Attacks

- Targeted FGSM: $x' = x - \epsilon\,\mathbf{sgn}(\nabla_x \mathcal{L}_t(x))$, where t is the target label. Untargeted FGSM: $x' = x + \epsilon\,\mathbf{sgn}(\nabla_x \mathcal{L}_y(x))$, where y is the original label.
- CW: use L-BFGS to solve $\mathbf{argmin}_\eta \|\eta\|_p + c \cdot \text{obj}_t(x+\eta)$, s.t. $x+\eta \in [0,1]^n$. Function obj satisfies $\text{obj}_t(x+\eta) \le 0 \Rightarrow f(x+\eta) = t$.
- PGD: iteratively run FGSM, each time with a small step. Typically used for adversarial training.

2 Certification of NN

Goal: given a neural network N, a constraint over input ϕ and a property over outputs psi, prove that $\forall i \in I.i \vDash \phi \Rightarrow N(i) \vDash \psi$ or return a violation. We want sound, complete and efficient (scalable to large NN) algorithms, i.e., to either scale sound and complete methods or to make sound but incomplete methods more precise.

Sound vs Complete

- Sound: if the property is violated, the program always states the property is violated.
- Complete: if the property holds, the program always states the property holds.

Incomplete but sound methods include convex relaxations. Complete and sound methods include Mixed-Integer Linear Programming (MILP).

Convex Relaxations

- Box. Use intervals as the approximation.
- Zonotope. Represent every variable by $x_i = a_0^i + \sum_j a_j^i \epsilon_j = a_i^T e$, where $\epsilon_i \in [-1,1]$, $a_i \ge 0$. (1) The encoding for affine layer $y = Wx+b$ is $y = WAe + b$, which is exact. (2) To compute the encoding for ReLU layers, we first obtain box bounds by plugging in $\epsilon = -1$ and $\epsilon = 1$. When it is strictly positive (negative), then use $y = x$ ($y = 0$). For cross-boundary cases, use the line between $(l,0)$ and (u,u) as the upper bound ($y = \lambda(x-l)$, $\lambda = \frac{u}{u-l}$) and its parallel as the lower bound ($y = \lambda x$). Then we compute the standard form of this zonotope by $y = \lambda(x - tl)$, $t \in [0,1]$, plug in $t = (\epsilon_{\text{new}} + 1)/2$, $\epsilon_{\text{new}} \in [-1,1]$ and expand.
- DeepPoly. Represent every variable by $x_i \ge \sum_j w_j^l x_j + v^l$, $x_i \le \sum_j w_j^u x_j + v_j$ and $l_i \le x_i \le u_i$. First do a forward propagation and then a backward propagation. (1) For affine layer $y = Wx+b$, use $Wx+b \le y \le Wx+b$. (2) For ReLU layers, if it is strictly positive (negative), use $x \le y \le x$ ($0 \le y \le 0$). Otherwise, apply min-area heuristic to choose one from $0 \le y \le \lambda(x-l)$ and $\lambda x \le y \le \lambda(x-l)$, i.e., choose the first one if $|l| \ge |u|$ o.w. the second one. When forward-propagating, we compute the interval bound by back-substituting to the first layer.

Comparison between relaxations: as long as the relaxation of one method is not included by another, they cannot be compared, although one might have a smaller area.

Complete Methods

- MILP: complete for ReLU network at the cost that it is at least NP-complete. The standard form is to $\min c^T x$, s.t. $Ax \le_p b$, $l \le_p x[\mathcal{R}] \le_p u$ and $x[\neg\mathcal{R}] \in Z^k$, where $x[\mathcal{R}]$ means the continuous component of x and $\le_p$ means point-wise comparison. (1) The encoding for affine layer is: $Wx+b \le y \le Wx+b$. (2) The encoding for ReLU is: $y \le x - l(1-a)$, $y \ge x$, $y \le u \cdot a$, $y \ge 0$ and $a \in \{0,1\}$. a serves as a "state" indicator to encode a piece-wise linear function. (3) Assume the goal is to prove $o_0 > o_1$, then we do $\min o_0 - o_1$ and see if the outcome is positive. (4) l, u are precomputed by Box. MILP benefits from Box bounds in that it could stop further exploration of infeasible combinations of integer variables, e.g., if the objective lower bound is positive when $a_1 = 0$, then there is no need to explore a_2 for $a_1 = 0$.

3 Certified Training

Goal: train NN so that it has maximum certified accuracy.

Formalization:

$$\min_\theta \mathbb{E}_{x,y} \max_{x' \in \gamma(NN^\#(S(x)))} \mathcal{L}(x', y; \theta)$$

where $S(x)$ is the allowed input set, $NN^\#$ is the abstraction of NN (e.g., DeepPoly abstraction).

- Suppose we use $\mathcal{L}(z,y) = \max_{q \ne y}(z_q - z_y)$. Then for each iteration, we first compute the convex relaxation of the input, and then compute the interval bound for $z_q - z_y$. Then we use the upper bound as the loss and update according to the gradient (Note that it is still differentiable).
- Suppose we use the cross entropy loss. Then we first compute the interval bound for z_i and pick the upper bound for incorrect classes and lower bound for correct classes to form a new vector. Then we use this new vector to compute the cross entropy and update.

There are a few practical tricks:

1. Start with a small $S(x)$ and then grow it.
2. Dynamically switch between standard loss and the certified loss.

To avoid increasing difficulty in the optimization due to better relaxations, Convex Layerwise Adversarial Training (COLT) is developed. The goal of COLT is to find a feasible point in the first-layer relaxation so that the loss is maximized and use this feasible point in the certified loss. When combined with Zonotope, it uses projection in the ϵ space since each $\epsilon \in [0,1]^n$ corresponds to a point in the zonotope. Although it is not the min-distance projection in the zonotope, it avoids the problem when the zonotope lies in a lower dimension where direct projection is not possible. It allows better results when a better relaxation is used.

4 Logic in DL

Goal: do query involving NN; forcing properties into NN.

NN Querying by Optimization

Idea: translate properties ϕ into loss function T and optimize it.

Challenge: how to find a loss function such that $\forall x, T(\phi)(x) = 0 \Leftrightarrow x$ satisfies ϕ. For example, ϕ could be: $|z-x|_\infty < \epsilon \Rightarrow |NN(z) - NN(x)|_\infty < \delta$.

Useful rules:

- $x \Rightarrow y$ is equivalent to $\neg x \vee y$.
- $\neg(x \vee y)$ is equivalent to $\neg x \wedge \neg y$.
- $\neg(x \wedge y)$ is equivalent to $\neg x \vee \neg y$.

One possible translation (DL2) is:

1. $t_1 \le t_2 \longrightarrow \max(0, t_1 - t_2)$.
2. $t_1 \ne t_2 \longrightarrow I(t_1 = t_2)$.
3. $t_1 = t_2 \longrightarrow T(t_1 \le t_2) + T(t_2 \le t_1) = |t_1 - t_2|$.
4. $t_1 < t_2 \longrightarrow T(t_1 \le t_2) + T(t_1 \ne t_2)$.
5. $\phi \vee \psi \longrightarrow T(\phi) \cdot T(\psi)$.
6. $\phi \wedge \psi \longrightarrow T(\phi) + T(\psi)$.

Trick for box constraints: when the input is constrained by a box, pass it to the optimizer such as L-BFGS-B.

Forcing properties of NN

Goal: $\max_\theta \mathbb{E}_{s\sim D} I(\forall z, \phi(z,s;\theta))$. Equivalently, $\min_\theta \mathbb{E}_{s\sim D} T(\phi)(z^*, s; \theta)$, where $z^* = \mathbf{argmin}_z T(\neg\phi)(z,s;\theta)$. Remember to take out any box constraints.

5 Randomized Smoothing

Given a base classifier f (the NN), the smoothed classifier is defined as $g(x) = \mathbf{argmax}_c P(f(x+\epsilon) = c)$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$.

Theorem: suppose $\exists \underline{p_A}, \overline{p_B} \in [0,1]$, s.t. $P(f(x+\epsilon) = c_A) \ge \underline{p_A} \ge \overline{p_B} \ge \max_{c \ne c_A} P(f(x+\epsilon) = c)$, then $g(x+\delta) = c_A$ for all $|\delta|_2 < R$, where $R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \phi^{-1}(\overline{p_B}))$, Φ^{-1} is the inverse of standard Gaussian CDF.

If we further require $\underline{p_A} > \frac{1}{2}$, then $\overline{p_B} < 1 - \underline{p_A}$ and thus $R > \sigma\Phi^{-1}(\underline{p_A})$. To compute probability, we use sampling. To avoid selection bias, we first sample to get the top label and then use an independent sampling to estimate the probability. The procedure is as follows:

function CERTIFY($f, \sigma, x, n_0, n, \alpha$)
  counts0 $\leftarrow$ SAMPLEUNDERNOISE(f, x, n_0, σ)
  $\hat{c}_A \leftarrow$ top index in counts0
  counts $\leftarrow$ SAMPLEUNDERNOISE(f, x, n, σ)
  $\underline{p_A} \leftarrow$ LOWERCONFBOUND(counts[$\hat{c}_A$], n, $1-\alpha$)
  if $\underline{p_A} > \frac{1}{2}$ **return** prediction $\hat{c}_A$ and radius $\sigma\,\Phi^{-1}(\underline{p_A})$
  else return ABSTAIN

At inference time, to compute $g(x)$, we need another sampling procedure:

function PREDICT(f, σ, x, n, α)
  counts $\leftarrow$ SAMPLEUNDERNOISE(f, x, n, σ)
  $\hat{c}_A, \hat{c}_B \leftarrow$ top two indices in counts
  $n_A, n_B \leftarrow$ counts[$\hat{c}_A$], counts[$\hat{c}_B$]
  if BINOMPVALUE($n_A, n_A + n_B, 0.5$) $\le \alpha$ **return** $\hat{c}_A$
  else return ABSTAIN

The probability of wrong prediction is $P(\hat{c_A} \ne c_A, \text{no abstain}) = P(\hat{c_A} \ne c_A) P(\text{no abstain} \mid \hat{c_A} \ne c_A) \le \alpha$.

Pros: (1) it can scale to large networks and (2) is model-agnostic.

Cons: (1) it requires sampling at inference time and (2) many samples could be needed to increase the certified radius.

6 Certifying Geometric Transformations

- Rotation: $T_\phi(x,y) = (x\cos(\phi) - y\sin(\phi), x\sin(\phi) + y\cos(\phi))$.
- Translation: $T_{\delta_x,\delta_y}(x,y) = (x+\delta_x, y+\delta_y)$.
- Scaling: $T_\lambda(x,y) = (\lambda x, \lambda y)$.
- Bilinear interpolation for non-integer (x,y): $I(x,y) = p_{x_1,y_1}(x_2-x)(y_2-y) + p_{x_1,y_2}(x_2-x)(y-y_1) + p_{x_2,y_1}(x-x_1)(y_2-y) + p_{x_2,y_2}(x-x_1)(y-y_1)$, where $x_1 \le x \le x_2$, $y_1 \le y \le y_2$, $x_2 = x_1 + 1$ and $y_2 = y_1 + 1$.

DeepG

Idea: use convex approximation for the exact input set. DeepG use the shape of DeepPoly but use a different strategy. It bounds all operations done in the I_κ simultaneously, and thus is more tight than DeepPoly.

DeepG finds two planes such that $w_l^T \kappa + b_l \le I_\kappa \le w_u^T \kappa + b_u$, where κ is the hyperparameter of transformation. It uses the min-volume heuristic: $w_l, b_l = \mathbf{argmin}\,\mathbb{E}(I_\kappa - (w_l^T \kappa + b_l))$. Using linear programming, we can solve this optimi-

zation for a finite number of sampled points. To extend the soundness to the whole hyperparameter space, we can modify the constraint by translating the plane, i.e., use $b_l = \hat{b_l} - \delta_l$ and $b_u = \hat{b_u} + \delta_u$. We can compute δ by either using box relaxation on $f(\kappa) = \hat{w_l}^T \kappa + \hat{b_l} - I_\kappa$, or by using Lipschitz and mean-value theorem on $f(\kappa)$: $f(\kappa) - f(\frac{1}{2}(h_u + h_l)) \leq |L|^T(\frac{1}{2}(h_u - h_l))$, where $\kappa \in [h_l, h_u]$. After deriving the convex relaxation for the transformation, we can use DeepPoly to complete the verification. To further improve the approximation, we can decompose the hyperparameter space, and validate each of them separately.

7 Fairness

Individual Fairness

Goal: for similar inputs, give similar outputs.

Key: define similarity $\phi(x, x') \to \{0, 1\}$. Then use DL2 to translate the similarity constraint into loss functions.

- Pre-process (data producer): provably learn a representation $z = f_\theta$ s.t. $\phi(x, x') \Rightarrow |z - z'|_2 < \delta$.
- In-process (data consumer): provably learn a classifier h_ψ s.t. $|z - z'|_2 < \delta \Rightarrow h_\psi(z) = h_\psi(z')$.

Group Fairness

Goal: $P(Y \mid G = 0) = P(Y \mid G = 1)$, e.g., fairness w.r.t. race and gender.

- Ignore target features. Usually does not work due to the correlation between features.
- Estimate probability and use PAC framework.

8 Federated Learning

Types of Federated Learning:

1. Cross-device. (1) Huge #sources, (2) each with small amount of data, (3) dynamically drop in and out in the learning process, (4) contact rarely. Example: Google spell checker.
2. Cross-sillo. (1) Limited #sources, (2) each with a lot of data, (3) participate constantly, (4) potentially heterogeneous. Example: Hospitals MRI segmentation.

Learning Epoch: (1) server selects clients to participate; (2) local computation on private data and send information to the server; (3) aggregation of local information on the server; (4) send aggregated information to clients.

Common Algorithms

- FedSGD. Do one iteration of local SGD on global weights and average them as the global update. Pros: It has convergence guarantee. Cons: Expensive communication.
- FedAvg. Do several iterations of local SGD on the global weights and average the local weights as the global weight. Pros: Less communication. Cons: No guarantee of convergence.

Common Issues

- Honest-but-Curious Server: honestly follow the algorithm but want to expose private data of clients. Gradient inversion: match gradients of reconstructed and client data.
- Client-side poisoning: send crafted updates to the server to force divergence or bad behavior on certain data.
- Client fairness: model behaves well on average but poorly for certain clients due to heterogeneity. There is a trade-off between preventing client-side poisoning and client fairness because the server cannot tell whether a bad update is malicious or because of heterogeneity.

Defenses

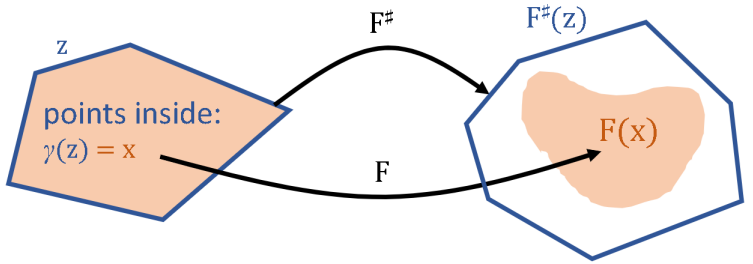
Differential Privacy. Definition: a randomized algorithm M is differentially private w.r.t. ϵ and δ iff for all similar dataset D, D' ($|D - D'| = 1$) and all subsets of outputs, $P(M(D) \in S) \leq e^\epsilon P(M(D') \in S) + \delta$.

- Clip the gradient and add noises to it to achieve DP.
- Customization of the local model. Allowing clients to use the global model and customize locally can help to solve the dilemma.

9 Abstract Interpretation

Theory of approximation.

blue: abstract orange: concrete



Important (soundness):

$\gamma(F^\sharp(z)) \supseteq F(x) = F(\gamma(z))$

1. A concrete element x is a set of concrete values.
2. An abstract (symbolic) element z semantically represents a set of concrete values.
3. γ is a concretization: it defines the concrete values an abstract element z represents (the points inside the polygon).
4. F is the concrete transformer. Because the set x is infinite or finite but very large, we generally cannot compute the transformed output set F(x).
5. $F^\sharp$ is the abstract transformer. $F^\sharp(z)$ applies $F^\sharp$ to abstract element z.
6. $F^\sharp$ must be sound (formula on top of slide and visualized in diagram above).