1 Adversarial Attacks

Targeted FGSM: $x' = x - \varepsilon \operatorname{sgn}(\nabla_x \mathcal{L}_t(x))$, where t is the target label.

Untargeted FGSM: $x' = x + \varepsilon \operatorname{sgn}(\nabla_x \mathcal{L}_y(x))$, where y is the original label.

Carlini & Wagner:
Solve $\operatorname{argmin}_\eta \|\eta\|_p + \lambda \mathcal{L}_t(x+\eta)$ s.t. $x+\eta \in [0,1]^n$. $\mathcal{L}$ satisfies $\mathcal{L}_t(x+\eta) \le 0 \Rightarrow f(x+\eta) = t$.

PGD: Iterate FGSM, each time with a small step. Typically used for adversarial training.

Adversarial Accuracy: Correct classification and no adversarial example.

Defense: $\min_\theta \mathbb{E}_{x,y}[\max_{x' \in S(x)} \mathcal{L}(x', y; \theta)]$.

2 Certification

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

- If the property does not hold, the program always states the property does not hold.
- If the property holds, the program always states the property holds.

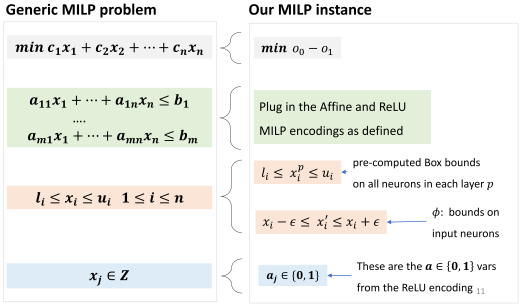
Convex Relaxations

Incomplete but sound

- Box. Use hypercubes as approximation.
- DeepPoly. Bound every variable by $\sum_j w_j^l x_j + v^l \le x_i \le \sum_j w_j^u x_j + v_j$ 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 $x \le y \le \lambda(x - l)$ (where $\lambda = \frac{u}{u-l}$), i.e., choose the first one if $u \le -l$ o.w. the second one. When forward-propagating, we compute the interval bound by backsubstituting to the first layer.

Relaxations of methods not included in one another cannot be compared.

MILP

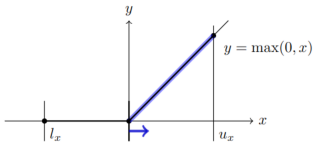


Complete and sound for ReLU network, but is NP-complete.

The encoding for **affine** layer is: $Wx + b \le y \le Wx + b$. The encoding for **ReLU** is: $x \le y \le x - l(1-a)$, $y \le ua$, $y \ge 0$ and $a \in \{0,1\}$. The variable a serves as a "state" indicator to encode a piece-wise linear function. The bounds l, u are pre-computed 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 $a_1 = 0$.

Branch and Bound

Can split variables in cases (e.g. ReLU output can be positive or negative) and certify both splits separately. Constraints can be enforced using KKT condition.



$$\max_{x \in \mathcal{X}} a^\top x + c \le \max_{x \in \mathcal{X}} \min_{\beta \ge 0} a^\top x + c + \beta x_i$$
$$\text{s.t.} \quad x_i \ge 0$$

⚠ Adjust output bounds when splitting.

Initialize queue (without splits). While queue is not empty and not timed out: 1. Get subproblem from queue. 2. Compute bound of interest. 3. If not verified, pick neuron to split according to heuristic. 4. Add both new subproblems to queue.

3 Certified Training

Goal: $\min_\theta \mathbb{E}_{(x,y)\sim D}\Big[\max_{x' \in B_\varepsilon(x)} \mathcal{L}(\theta, x', y)\Big]$

DiffAI / IBP

IBP (over)approximates inner max by propagating the box $B_\varepsilon(x)$ and computing the max loss over the output set. Can use any abstract transformer (Box, DeepPoly, etc.) to get bounds on logits.

SABR

SABR uses the loss

$$\mathcal{L}_{\varepsilon,\tau}(x, y) = \max_{x' \in B_{\varepsilon-\tau}(x)} \max_{z \in B_\tau(x')} CE(f(z), y),$$

i.e. it uses a box $B_\tau(x')$ of size $\tau \le \varepsilon$ around an adversarial sample x'. The outer max is (under)approximated by PGD search and the max inside is (over)approximated using IBP.

4 Randomized Smoothing

Given a classifier f and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, define a smoothed classifier g as

$$g(x) = \operatorname{argmax}_c \mathbb{P}[f(x+\epsilon) = c].$$

THM: Suppose $\exists p_A, p_B \in [0,1]$, s.t. $\mathbb{P}[f(x+\epsilon) = c_A] \ge p_A \ge p_B \ge \max_{c \ne c_A} \mathbb{P}[f(x+\epsilon) = c]$. Then $g(x+\delta) = c_A$ for all $\|\delta\|_2 < R$, where the radius $R = \frac{\sigma}{2}(\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$ and Φ^{-1} is the Gaussian quantile function.

If we have $p_A > \frac{1}{2}$, then $p_B \le 1 - p_A < p_A$ and thus $R > \sigma \Phi^{-1}(p_A)$. 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:

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

By the above theorem, we have $g(x) = \hat{c}_A$ for all $\|\delta\|_2 < R$ with probability $1 - \alpha$ if $\hat{c}_A, R$ is returned.

For inference, to compute $g(x)$, we use:

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 $\mathbb{P}[\hat{c_A} \ne c_A, \text{no abstain}] = \mathbb{P}[\text{no abstain} \mid \hat{c_A} \ne c_A]\, \mathbb{P}[\hat{c_A} \ne c_A] \le \alpha$.

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.

5 Federated Learning

Communication Round: (1) Server selects clients to participate. (2) Clients do local training update on private data and send information to the server. (3) Server aggregates updates into updated model.

FedSGD

Clients compute gradient wrt. (minibatch of) local data on global weights. Server takes average for global update. Pros: convergence guarantee. Cons: requires lots of communication. Gradient inversion:

$$\operatorname*{argmin}_{x^*} \underbrace{d(\nabla_\Theta \mathcal{L}(f_\Theta(x^*), y^*), g_k)}_{\text{Distance between reconstructed gradient on } f_\Theta \text{ and the true gradient } g_k} + \alpha_{\text{reg}} \cdot \underbrace{\mathcal{R}(x^*)}_{\text{Domain specific prior}}$$

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 convergence guarantee.

6 Privacy

Privacy Attacks

- Model stealing (white/black box)
- Model inversion: Client queries the model to find representative samples.
- Data extraction: Client queries the model to find exact training samples.
- Membership inference: Client wants to determine if the data point was used to train the model or not

Differential Privacy

A randomized algorithm M is ϵ, δ-DP if for all "neighbouring" datasets D, D' and all at-

tacks (sets) S,

$$\mathbb{P}[M(D) \in S] \leq e^{\epsilon}\mathbb{P}[M(D') \in S] + \delta.$$

DP Rules

Post-processing: If M is ϵ,δ-DP, then $g \circ M$ is ϵ,δ-DP.
Composition: If M_1, M_2 are ε_1,δ_1-DP and ε_2,δ_2-DP, then $M := (M_1, M_2)$ is $(\varepsilon_1+\varepsilon_2, \delta_1+\delta_2)$-DP.
Privacy Amplification: Applying an ϵ,δ-DP mechanism on a random fraction $q \leq 1$ subset yields a $\tilde{q}\epsilon, q\delta$-DP mechanism, where $\tilde{q} \approx q$.

Laplace Mechanism

Let $\Delta_1 = \max_{D \sim_{\text{neigh}} D'} \|f(D) - f(D')\|_1$. Then $f + \text{Laplace}(0, \Delta_1/\epsilon)$ is ε-DP.

Gaussian Mechanism

Let $\Delta_2 = \max_{D \sim_{\text{neigh}} D'} \|f(D) - f(D')\|_2$. Then if $\sigma^2 = \frac{2\log(1.25)}{\delta\epsilon^2}\Delta_2^2$ we get that $f + \mathcal{N}(0, \sigma^2 I)$ is ε,δ-DP.

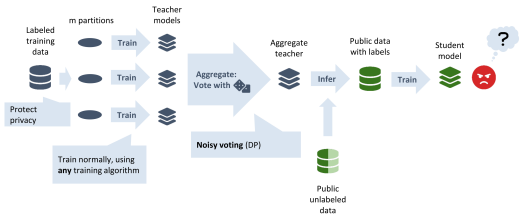
DP-SGD

ε,δ-DP adaptation of SGD.

1. Clip gradients: $g_i' = \min\left(\frac{R}{\|g\|_2}, 1\right) g_i$
2. Aggregate: $\overline{g} = \frac{1}{n}\sum_{i=1}^{n} g_i'$
3. Add noise: $\tilde{g} = \overline{g} + \mathcal{N}(0, \sigma^2 I)$ where $\sigma^2 = \frac{2\log(1.25)}{\delta\epsilon^2}\frac{R^2}{n^2}$
4. Update: $\theta \leftarrow \theta - \eta\tilde{g}$

DP-SGD is $O\left(q\varepsilon\sqrt{T\log\frac{1}{\delta}}\right), O(qT\delta)$-DP. It is $O(q\varepsilon\sqrt{T}), \delta$-DP when allowing data-dependent σ.

PATE



Noisy voting: Add $\text{Laplace}(0, \frac{2}{\varepsilon})$-noise to each class count $n_c(x)$ before argmax (n_c has sensitivity 2). Results in $N\varepsilon$-DP, where N is the number of classifications we compute on the public unlabeled data.

7 Logical Properties

DL2

Idea: Translate properties ϕ into loss function T and optimize. For example, ϕ could be: $\|z - x\|_\infty < \epsilon \Rightarrow \|NN(z) - NN(x)\|_\infty < \delta$.

ϕ	$T(\phi)$
$t_1 \leq t_2$	$\max(0, t_1 - t_2)$
$t_1 \neq t_2$	$\mathbb{1}_{\{t_1 = t_2\}}$
$t_1 = t_2$	$T(t_1 \leq t_2 \wedge t_2 \leq t_1)$
$t_1 < t_2$	$T(t_1 \leq t_2 \wedge t_1 \neq t_2)$
$\phi \vee \psi$	$T(\phi)T(\psi)$
$\phi \wedge \psi$	$T(\phi) + T(\psi)$

Training for Logical Properties

Idea: Get the worst violation of ϕ and find θ that minimizes its effect. I.e. solve $\min_\theta \mathbb{E}_{s\sim D}\left[T(\phi)(z^*, s, \theta)\right]$ where $z^* \in \text{argmin}_z\, T(\neg\phi)(z, s, \theta)$.

In practice, restrict z to a convex set with efficient projections (closed form). One can then remove the constraint from ϕ that restricts z on the convex set and do PGD while projecting z onto the convex set.

8 Individual Fairness

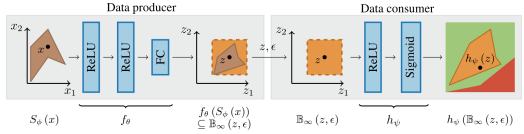
DEF: A model M is individually fair if for any similar x, x' it produces similar outputs $M(x), M(x')$, e.g.

- Lipschitz continuity: $\mu(M(x), M(x')) \leq L\phi(x, x')$.
- Binary similarity: $\phi(x, x') \rightarrow M(x) = M(x')$.

Fair Representation Learning:

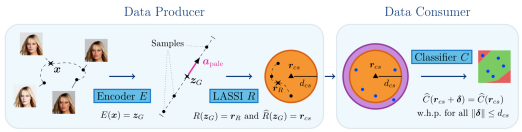
- Data Regulator: Determines fairness criteria and data source(s), audits results.
- Data Producer: Computes the fair representation given data regulator criteria.
- Data Consumer: Trains the ML model given the sanitized data. Split models into an encoder f_θ (data producer) and a predictor h_ψ (data consumer).

LCIFR



Consider binary ϕ, μ. 1. Encoder f_θ is trained s.t. $\phi(x, x') \Rightarrow \|f(x) - f(x')\|_\infty \leq \delta$ (with a combined loss ensuring informativeness of the representations) using DL2 (or δ is computed when encoded as MILP). 2. Use L_∞-robustness training on encoded data for perturbations of magnitude δ and certify robustness. Gives certified individual fairness. Problem: Relies on NN verifiers so not scalable to real-world models.

LASSI



1. Data producer uses center smoothing to get an encoder that provably maps all similar points close together with high probability. 2. Data consumer uses randomized smoothing to certify that all points within a certain radius get classified the same with high probability. Problem: Depends on quality of generative model.

9 Group Fairness

Demographic Parity

$P(\hat{Y} \mid G = 0) = P(\hat{Y} \mid G = 1)$

Equalized Odds

$P(\hat{Y} \mid G = 0, Y = 0) = P(\hat{Y} \mid G = 1, Y = 0)$
$P(\hat{Y} \mid G = 0, Y = 1) = P(\hat{Y} \mid G = 1, Y = 1)$

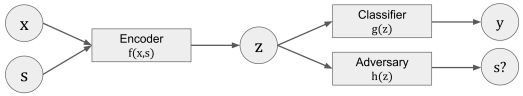
Equal Opportunity

$P(\hat{Y} \mid G = 0, Y = 1) = P(\hat{Y} \mid G = 1, Y = 1)$

Approaches:

- Pre-processing: Debias the data, such that standard training yields a fair model. E.g. fair representation learning.
- In-training: Change the training pipeline to learn a fair model (on biased data). E.g. add soft fairness constraint to the loss.
- Post-processing: Modify predictions at inference time. E.g. different thresholds for different groups.

LAFTR



Jointly trains an encoder f, classifier g, and adversarial classifiers h by solving $\min_{f,g}\max_{h\in\mathcal{H}} \mathcal{L}_{\text{clf}}(f,g) - \lambda\mathcal{L}_{\text{adv}}(f,h)$.

Balanced Accuracy

$\text{BA}(h) = \frac{1}{2}\mathbb{E}_{Z_0}[1 - h(z)] + \frac{1}{2}\mathbb{E}_{Z_1}h(z)$
$h^* = \text{argmax}_h\, \text{BA}_{Z_0,Z_1}(h) = \mathbb{1}_{\{p_1(z) \geq p_0(z)\}}$

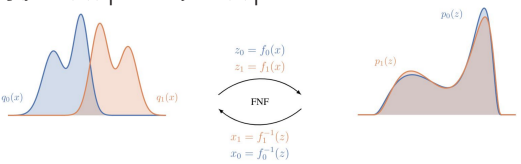
DP-Distance (Relaxation of DP)

$\Delta^{\text{DP}}_{Z_0,Z_1}(g) = \left|\mathbb{E}_{Z_0}g(z) - \mathbb{E}_{Z_1}g(z)\right|$
$\Delta^{\text{DP}}_{Z_0,Z_1}(g) \leq 2\,\text{BA}_{Z_0,Z_1}(h^*) - 1$

LAFTR estimates $\Delta^{\text{DP}}_{Z_0,Z_1}(g)$ given an adversary h. Fairness is overestimated (there may be better adversaries).

Fair Normalizing Flows

Normalizing flows for a distribution $p(z)$ parameterize $z = f(x)$ where x is sampled from a simple distribution $q(x)$ through an invertible transformation: $p(z) = q(f^{-1}(z))\left|\det \partial_z f^{-1}(z)\right|$.



FNF learns two flows f_0, f_1 as encoders for Z_0, Z_1. 1. Estimate densities q_0, q_1 for both groups. 2. Apply the encoder to get $z = f_0(x)$ (or $f_1(x)$), and use the flows and q_0, q_1 to get $p_0(z)$ and $p_1(z)$. 3. Use $p_0(z)$ and $p_1(z)$ to estimate the optimal adversary $h*$ and then upper bound its balanced accuracy with probability $1 - \varepsilon$ (Hoeffding) 4. Use the inequality from LAFTR above to upper bound the DP-distance of any downstream classifier trained on representations z with probability $1 - \varepsilon$.

To get tight bounds: train the flows to promote low accuracy of h^* by minimizing $D_{KL}(p_0 \| p_1) + D_{KL}(p_1 \| p_0)$. ⚠ The guarantees hold for the estimated q_0, q_1.

FARE

Restrict representations z to a finite set. This allows bounding $\text{BA}_{Z_0,Z_1}(h^*)$ whp. directly. Use fairness-aware decision tree as encoder which is obtained by training a tree to have unbalanced leaves wrt. y and uniform leaves wrt. s. Gives provable unfairness upper bound with no restrictive assumptions