1 Basics

1.1 Good to know

$||x||_p = \left(\sum_{i=1}^d |x_i|^p\right)^{\frac{1}{p}}$ $\quad ||x||_\infty = \max_{i\in\{1,..,d\}} |x_i|$

Softmax $\sigma(z)_i = e^{z_i} / \sum_{j=1}^D e^{z_j}$

CE: $H(\vec{p},\vec{q}) = -\sum_{i=1}^n p_i \cdot \log q_i$

CE loss: $L(\vec{x},\vec{y}) = H(\text{one-hot}(y), \text{softmax} \circ g(\vec{x}))$

Implication: $\phi \implies \psi \iff \neg\phi \vee \psi$

Mean value: $f(y) = f(x) + \nabla f(z)^T(y-x)$

Gauss: $\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{(2\pi)^d}} exp(-\frac{(x-\mu)^2}{2\cdot\sigma^2})$

CDF: $\Phi(v;\mu,\sigma^2) = \int_{-\infty}^v \mathcal{N}(y;\mu,\sigma^2)dy = \Phi(\frac{v-\mu}{\sqrt{\sigma^2}};0,1)$

Lap: $\mathbb{P}(Lap(\mu,\sigma)=t) = \frac{1}{2\sigma}\cdot exp(-\frac{|t-\mu|}{\sigma})$

Subadditivity of $\sqrt{\cdot}$**:** $\sqrt{x+y} \le \sqrt{x} + \sqrt{y}$

Cauchy Schwarz: $\langle x,y\rangle \le ||x||_2 \cdot ||y||_2$

Triangel Inequalities: $||a+b|| \le ||a|| + ||b||$, $||a-b|| \ge ||a|| - |b||$

Weak duality: $max_x min_y \le min_y max_x$

2 Norm Inequalities

$||x||_\infty \le ||x||_1 \le d\cdot||x||_\infty$

$||x||_\infty \le ||x||_2 \le \sqrt{d}\cdot||x||_\infty$

=> if a classifier is safe for a l_2 region of size ϵ, it is also safe for a l_∞ region of size $\frac{\epsilon}{\sqrt{d}}$

3 Adversarial Attacks

T-FGSM: $x' = x - \eta,\ \eta = \epsilon\cdot sign(\nabla_x loss_t(x))$

U-FGSM: $x' = x + \eta,\ \eta = \epsilon\cdot sign(\nabla_x loss_s(x))$

Guarantees $\eta \in [x-\epsilon, x+\epsilon]$, η not minimized.

C&W: Find adv sample $x' = x + \eta \in [0,1]^n$ *and* minimize $||\eta||_p$ via relaxation s.t. $f(x') = t$. obj_t: $obj_t(x+\eta) \le 0 \Leftrightarrow f(x+\eta) = t$.

Minim. $||\eta||_p + c\cdot obj_t(x+\eta)$ s.t. $x+\eta \in [0,1]^n$

E.g. $obj_t = \{CE(x',t)-1; max(0, 0.5 - p_f(x')_t)\}$

$\nabla_\eta ||\eta||_p$ is suboptimal → use proxy

l_∞: proxy $L(\eta) = \sum_i max(0, |\eta_i| - \tau)$. Iteratively decrease τ until $L(\eta) > 0$. Then do GD on η: $\eta = \eta - \gamma\nabla_\eta(L(\eta) + c\cdot obj_t(x+\eta))$ until $L(\eta) = 0$, then anneal τ and continue loop.

Constraint $\eta_i \in [-x_i, 1-x_i]$: LBFGS-B, PGD

PGD: Iterative FGSM with projection to find point in $x_o \pm \epsilon$ that max. loss (not guaranteed to be missclassification.

1. Init $x' = x + \epsilon\cdot rand[-1,1]$;
2. Repeat: $x' \leftarrow x' + \epsilon_{step}\cdot sgn(\nabla_{x'} loss_s(x'))$ (untargeted) 3. $x' = project(x', x_o, \epsilon)$;

4 Adversarial Defenses

Defense as Optimization: $\min_\theta \rho(\theta)$, $\rho(\theta) = \mathbb{E}_{(x,y)\sim D}\left[\max_{x'\in S(x)} L(\theta, x', y)\right]$, $S(x) = \{x' : ||x - x'||_\infty \le \epsilon\}$

PGD Defense in practice:

1. Select mini batch B from D **2.** $x_{max} = \arg\max_{x'\in S(x)} L(\theta, x', y), \forall(x,y)\in B$

3. $\theta' = \theta - \frac{1}{|B_{max}|}\sum_{(x_{max},y)\in B_{max}} \nabla_\theta L(\theta, x_{max}, y)$

Test accuracy: is test point x classified correctly? **Adveserial accuracy:** are points in region around x classified as x's class?

5 Certification of Neural Networks

Given NN N, pre-condition ϕ, post-condition ψ prove: $\forall i \in I : i \vDash \phi \implies N(i) \vDash \psi$ or return a violation.

Sound: Algorithms outputs true only when true. **Complete:** Algorithm allways output true when true.

5.1 Incomplete Methods:

Over-approx. ϕ using relaxation, then push approx. through NN via bound propagation.

Box: $\hat{x_i} = [l,u]$, i.e. $l \le x_i \le u$. $\quad AT^\#$: $[a,b] +^\# [c,d] = [a+b, c+d]$; $-^\#[a,b] = [-b,-a]$; $ReLU^\#[a,b] = [ReLU(a), ReLU(b)]$; $\lambda\cdot^\#[a,b] = [\lambda a, \lambda b]$ $(\lambda > 0)$

DeepPoly: $\mathcal{O}(n^3L^2)$, $n :=$ max neurons in a layer, $L :=$ Layers. For each x_i keep:

- interval constraints $l_i \le x_i,\ x_i \le u_i$
- relational constraints: $a_i^\le \le x_i,\ x_i \le a_i^\ge$ where $a_i^\le, a_i^\ge$ are of the form $\sum_j w_j\cdot x_j + \nu$

$AT^\#$: Affine$^\#$ is exact

-Affine$^\#$: rel: $\sum_j w_j^p\cdot x_j + \nu^p + \sum_j w_j^q\cdot x_j + \nu^q = \sum_j (w_j^p + w_j^q)\cdot x_j + (\nu^p + \nu^q)$;

int: backsubstitution up to some layer. Then replace neurons of that layer with its correct interval constraint (like Box bounds).

Backsub: sum up all components!

- $x_j = \text{ReLU}^\#(x_i)$: interval constr. $x_i \in [l_i, u_i]$:

$u_i \le 0$: $a_j^\le = a_j^\ge = 0, l_j = u_j = 0$;

$l_i \ge 0$: $a_j^\le = a_j^\ge = x_i, l_j = l_i, u_j = u_i$;

$l_i < 0, u_i > 0$: $\lambda = u_i/(u_i - l_i)$,

Relaxation with $\alpha \in [0,1]$:

$a_j^\ge = \alpha\cdot x_i,\ a_j^\le = \lambda\cdot x_i - \lambda\cdot l_i$

Rule of thumb: $\mathbf{u_i} \le -\mathbf{l_i} : \alpha = 0$, else $\alpha = 1$

Symbolic bound: when proving $y_2 > y_1$, use abstract shape of $y_2 - y_1$ and prove $l_{y_2 - y_1} > 0$

Holder's inequality: $\frac{1}{p} + \frac{1}{q} = 1 \implies ||x * y||_1 \le ||x||_p ||y||_q$

Bounds with other input norms: $-||a||_q\epsilon + ax' + c \le min(a\hat{x} + c) \le max(a\hat{x} + c) \le ||a||_q\epsilon + ax' + c$; where $\hat{x} \in x, ||x - x'||_p < \epsilon$

KKT: $max_x(f(x))$ (s.t. $g(x) \le 0$) $\le max_x min_{\beta\ge 0} f(x) - \beta\cdot g(x)$

Positive split:

$max_x(ax+c)$ (s.t. $-x \le 0$) $\le max_x min_\beta(ax + c + \beta x) \le min_\beta max_x(ax + c + \beta x)$

Negative split:

$max_x(ax+c)$ (s.t. $x \le 0$) $\le max_x min_\beta(ax + c - \beta x) \le min_\beta max_x(ax + c - \beta x)$

=> Summarize bounds by taking max

Opt β **with GD:** $\beta_{t+1} = \beta_t - \alpha\nabla_\beta UB$

E2E verification: init empty queue, add full problem without splits, for problem in queue: try to verify, if not verified: pick neuron to split and add subproblems to queue

5.2 Complete Methods

Encode NN as MILP instance.

- Affine: $y = Wx + b$ direct MILP constraint.
- $ReLU(x)$: $y \le x - l_x\cdot(1-a),\ y \ge x,\ y \le u_x\cdot a,\ y \ge 0,\ a \in \{0,1\}$, for neuron bound $x \in [l,u]$.
- $\phi = B_\epsilon^\infty(x)$: $x_i - \epsilon \le x_i' \le x_i + \epsilon, \forall i$
- precomp. Box bounds: $l_i \le x_i^p \le u_i$
- $\psi = o_0 > o_1$: MILP objective $\min o_0 - o_1$. If $\min o_0 - o_1 > 0$, ψ holds.

6 Certified Defenses

6.1 DiffAI

DiffAI Certified PGD Defense: minimize $\rho(\theta) = \mathbb{E}_{(x,y)\sim D}\left[\max_{z\in\gamma(NN^\#(S(x)))} L(\theta, z, y)\right]$

Use abstract loss $L^\#(\vec{z}, y)$, where y = target label, $\vec{z}$ = vector of logits:

- $L(z,y) = max_{q\ne y}(z_q - z_y)$: Compute $d_c = z_c - z_y \forall c \in \mathcal{C}$ where $\mathcal{C}$ set of classes and z_c the abstract logit shape of class i. Then compute box bounds of d_c and compute max upper bound: $\max_{c\in\mathcal{C}}(\max(box(d_c)))$
- $L(z,y) = CE(z,y)$: Compute box bounds $[l_c, u_c]$ of logit shapes z_c. $\forall c \in \mathcal{C}$ pick u_c if $c \ne y$, pick l_c if $c = y$. Then apply softmax to vector $v = [u_0, u_1, .., l_c, .., u_{|\mathcal{C}|}]$ and compute $CE(v', y)$ with $v' = softmax(v)$.

Cheap relaxations (box) scale but introduce lots of infeasible points: substantial drop in standard accuracy. More complex relaxations make it worse → counter-intuitive!!t

Comparison: Adv train: Good Acc, Worse verifiablity, easier opt prob Certified Def: Worse Acc, Good verifiability, harder opt prob

6.2 Convex Layerwise Adversarial Training

COLT: PGD training with intermediate NN layer shapes S_i. Iterate over layers h_i and find weights θ for layers $h_{i+1}, .., h_D$ that minimize the worst-case loss of $x_i \in S_i$. Weights of previous layers $h_1, .., h_i$ are frozen.

$\min_\theta \max_{x_i\in S_i} L(h_D(h_{D-1}(...h_{i+1}(x_i))), y_{true})$

The inner maximization requires projections.

7 Logic and Deep Learning (DL2)

7.1 Querying Neural Networks

Use standard logic ($\forall, \exists, \wedge, \vee, f : \mathbb{R}^m \to \mathbb{R}^n, ..$) and high-level queries to impose constraints.

$(class(NN(i)) = 9) = \bigwedge_{j=1, j\ne 9}^k NN(i)[j] < NN(i)[9]$

Use translation T of logical formulas into differentiable loss function $T(\phi)$ to be solved with gradient-based optimization to minimize $T(\phi)$.

Theorem: $\forall x, T(\phi)(x) = 0 \iff x \vDash \phi$ **Logical Formula to Loss:**

Logical Term	Translation
$t_1 \le t_2$	$\max(0, t_1 - t_2)$
$t_1 \ne t_2$	$\mathbb{1}[t_1 = t_2]$
$t_1 = t_2$	$T(t_1 \le t_2 \wedge t_2 \le t_1)$
$t_1 < t_2$	$T(t_1 \le t_2 \wedge t_1 \ne t_2)$
$\phi \vee \psi$	$T(\phi)\cdot T(\psi)$
$\phi \wedge \psi$	$T(\phi) + T(\psi)$

Translation is recursive and $T(\phi)(x) \ge 0, \forall x, \phi$

Box constraints: ineffective in GD. Use L-BFGS-B and give box constraints to optimizer.

7.2 Training NN with Background Knowledge

Incorporate logical property ϕ in NN training.

Problem statement: find θ that maximizes the expected value of property.

Maximize $\rho(\theta) = \mathbb{E}_{s\sim D}\left[\forall z.\phi(z, s, \theta)\right]$.

BUT: Universal quantifiers are difficult.

Reformulation: get the worst violation of ϕ and find θ that minimizes its effect.

minimize $\rho(\theta) = \mathbb{E}_{s\sim D}\left[T(\phi)(z_{worst}, s, \theta)\right]$

where $z_{worst} = \arg\min_z T(\neg\phi)(z, s, \theta)$

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 Projected-Gradient-Descent while projecting z onto the convex set.

8 Randomized Smoothing for Robustness

Construct a classifier **g** from a classifier **f** s.t. **g** has certain statistical robustness guarantees.

Given base classifier $f : \mathbb{R}^d \to \mathcal{Y}$, construct smoothed classifier g (where $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$):

$g(x) := \arg\max_{c\in\mathcal{Y}} \mathbb{P}_\epsilon(f(x+\epsilon) = c)$

Robustness Guarantee: suppose $c_A \in \mathcal{Y}$ (most likely class), $\underline{p_A}, \overline{p_B} \in [0,1]$ satisfy:

$\mathbb{P}_\epsilon(f(x+\epsilon) = c_A) \ge \underline{p_A} \ge \overline{p_B} \ge$

$\ge \max_{c\ne c_A} \mathbb{P}_\epsilon(f(x+\epsilon) = c)$

with $\underline{p_A}$ a lower bound on the true highest probability and $\overline{p_B}$ an upper bound on the true second highest probability. In practice, get bounds via sampling which gives statistical guarantees. Then: $g(x+\delta) = c_A$, for all $||\delta||_2 < R$, $R := \frac{\sigma}{2}(\phi^{-1}(\underline{p_A}) - \phi^{-1}(\overline{p_B})) \ge 0$ with ϕ^{-1} the inverse Gaussian CDF. Certified radius R depends on input x since $\underline{p_A}, \overline{p_B}$ depend on x.

Notes on CDF: If $x \sim \mathcal{N}(0,1)$, $p \in [0,1]$, then $\phi^{-1}(p) = \nu$ s.t. $\phi(v) := \mathbb{P}_x(x \le \nu) = p$; ϕ^{-1}, ϕ are monotone, i.e. for $\underline{p_A} \ge \overline{p_B}$: $\phi^{-1}(\underline{p_A}) \ge \phi^{-1}(\overline{p_B})$; $\phi^{-1}(p) = -\phi^{-1}(1-p), p \in [0,1]$

If $x \sim \mathcal{N}(\mu_x, \sigma_x^2), y \sim \mathcal{N}(\mu_y, \sigma_y^2)$, than $(x+y) \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$ and $c \cdot x \sim \mathcal{N}(c\mu_x, c^2\sigma_x^2)$

Certified Accuracy: Pick target radius T and count #test points whose certified radius is $R \geq T$ and where the predicted c_A matches the test set label.
Standard Accuracy: Instantiate certified accuracy with $T = 0$

8.1 Certification Procedure

function CERTIFY($f, \sigma, x, n_0, n_1, \alpha$)
  counts0 $\leftarrow$ SampleUnderNoise(f, x, n_0, σ)
  $\hat{c}_A \leftarrow$ top index in counts0
  counts1 $\leftarrow$ SampleUnderNoise(f, x, n_1, σ)
  $\underline{p_A} \leftarrow$ LowerConfBound(counts1[$\hat{c}_A$],n_1,1-α)
  if $\underline{p_A} > \frac{1}{2}$:
    return prediction $\hat{c}_A$, radius $\sigma\phi^{-1}(\underline{p_A})$
  else: return ABSTAIN

Notes:
- $\hat{c}_A$ is not necessarily the correct test set label
- Sample $2\times$ ($n \gg n_0$) to prevent selection bias.
- SampleUnderNoise evaluates f at $x + \epsilon_i$ for $i \in \{1, .., n\}$, returns dict of class counts.
- LowerConfBound returns probability p_l s.t. $p_l \leq p$ with probability $1 - \alpha$, assuming $k \sim Binomial(n, p)$ for unknown p.
- $\underline{p_A} > \frac{1}{2}$ ensures $\overline{p_B} < \frac{1}{2}$, thus $\underline{p_A} \geq \overline{p_B}$
- With probability at least $1 - \alpha$, if CERTIFY returns class $\hat{c}_A$ and radius $R = \sigma\phi^{-1}(\underline{p_A})$, then $g(x + \delta) = \hat{c}_A$ for all $||\delta|| < R$.
- To increase R, need to increase $\underline{p_A}$. To increase $\underline{p_A}$, get f to classify more noisy points to $\hat{c}_A$. Increasing the #samples only slowly grows R.

8.2 Inference

fuction PREDICT(f,σ,x,n,α)
  counts $\leftarrow$ SampleUnderNoise(f,x,n,σ)
  $\hat{c}_A, \hat{c}_B \leftarrow$ top two indices from counts
  $n_A, n_B \leftarrow$ counts[$\hat{c}_A$], counts[$\hat{c}_B$]
  if BinomPValue($n_A, n_A + n_B, 0.5$) $\leq \alpha$:
    return $\hat{c}_A$
  else: return ABSTAIN

Notes:
- Null hypothesis: true probability of success of f returning $\hat{c}_A$ is $q = 0.5$
- BinomPValue returns p-value of null hypothesis, evaluated on n iid samples with i successes.
- Accept null hypothesis if p-value is $> \alpha$
- Reject null hypothesis if p-value is $\leq \alpha$
- α small: often accept null hypothesis and ABSTAIN, but more confident in predictions.
- α large: more predictions but more mistakes.
- PREDICT returns wrong class $\hat{c}_A \neq c_A$ with probability at most α

9 Privacy

Types: Model stealing, model inversion, data extraction, membership inference

Federated learning

FedSGD: Client: $g_k = \nabla_\theta L(f_\theta(x_k), y_k)$. Server: $\theta - = \alpha \frac{1}{N} \sum_{k=1:N} g_k$. Pro: convergence guaranteed. Cons: requires many rounds, not private. $x_{rec} = argmin_{x'} d(\nabla_\theta L(f_\theta(x'), y'), g_k) + \alpha_{reg} \cdot \mathcal{R}(x')$, where $\mathcal{R}$ is prior domain knowledge
FedAvg: Client: performs Epochs E * Batches B local SGD steps. Server: averages the received models. Pros: less rounds, harder to attack. Attack: create opt variables for each batch and epoch $\tilde{X}^k_{e,b}$, simulate FedAvg to get $\tilde{\theta}_{e,b} := argmin_{\tilde{X}^k} d(\tilde{\theta}_{e,b}, \theta) + \alpha_{reg} \frac{1}{E^2} \sum_{e_1, e_2} \mathcal{R}(g(\tilde{X}_{e1,b}), g(\tilde{X}_{e2,b}))$ with $g(\tilde{X}) = \frac{1}{|D_k|} \sum_{b,i} \tilde{X}_{e,b,i}$, $\mathcal{R}$ avg dist between avg images between every epoch.

Differential Privacy

M: data to output, S: set of all outputs where attacker believe they caught you. M is $(\epsilon, \delta) - DP$ if $P(M(a) \in S) \leq e^\epsilon P(M(a') \in S) + \delta$ $\iff e^{-\epsilon}(P(M(a) \in S) - \delta) \leq P(M(a') \in S)$ $\forall S, (a, a') \in Neigh$
Laplace M: $f(a) + Lap(0, \Delta_1/\epsilon)$ is $\epsilon - DP$
Gauss M: $f(a) + N(0, \sigma^2 I)$ is $(\epsilon, \delta) - DP$, $\sigma = \frac{\Delta_2}{\epsilon}\sqrt{2 \cdot log(1.25)/\delta}$
Sensitivity: $\Delta_p := max_{a,a' \in Neigh}(||f(a) - f(a')||_p)$ largest possible change in output
Post processing: M $(\epsilon, \delta) - DP \Rightarrow f \circ M$ is $(\epsilon, \delta) - DP$
Composition: $(M_1, M_2) is (\epsilon_1 + \epsilon_2, \delta_1 + \delta_2) - DP$
DP makes no assumption on attacker: still protected from infinite compute $\vee$ side info.
DP-SGD: T iterations with random batch of size L, standard SGD with projection of each gradient onto l_2 ball of size C, aggregate all gradients to one and add noise $\mathcal{N}(0, \sigma I)$, with $\Delta_2 = C/L$. Yields $(qT\epsilon, qT\delta) - DP$.
PATE: Train teacher model on private data (split data in m subsets and train a teacher on each, final teacher by noisy voting), label T public data instances with teacher, train student. $(T\epsilon, 0) - DP$. Noisy voting: $f(x) = argmax_j(n_j + Lap(0, 2/\epsilon) \rightarrow \Delta_1 = 2$

Synthetic Data

Generate new dataset with same statistics as private dataset: select which marginals to meassure (:=#occurences of attritube combination), meassure marginals and add noise to them.
Marginals Selection: init fully connected graph of all attributes, set mutual info as weight of each edge, compute max spanning tree. $I(X,Y) = \sum_x \sum_y \frac{P(X=x, Y=y)}{P(X=x)P(Y=y)} + N(0, \sigma I)$
Inference: $P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_1)$ if X_1 is parent of X_2 and X_3.

10 Fairness

X: features, Y: outcome, G: sensitive attributes. Model: for $x \in X : M(x) = \hat{Y}$ = prob. dist. "Group fair does not imply individual fair."

Individual fairness

Similar input => similar output examples: **L-lipschitz**: $D(M(x), M(x')) \leq Ld(x, x')$,
IF as robustness: $\mathbb{1}[M(x) \neq M(x')] \leq L||x - x'|| \iff M(x) = M(x + \delta)$ for $||\delta||_S < \frac{1}{\sqrt{L}}$. Can now use robustness theory.
Fair Representation Learning: Regulator: determines fairness criteria and data source, Producer: computes fair representation f_θ, Consumer: Trains model h_ψ. Final $M := h_\psi \circ f_\theta$. Pros: efficent re-use, can use transfer learnig. Cons: less controll over fairness/accuracty trade off, overconfident fairness if consumer is adveserial, startup costs.
LCIFR:Regulator: encode a domain specific notion of similarity in a logical formula ϕ, such that $x, x' \models \phi$ iff x, x' are similar. Producer: We define $S_\phi(x) := \{x \in R^N \mid \phi(x, x')\}$. Use DL2 or MILP to obtain δ s.t. $\forall x \in S_\phi(x)$: $|f_\phi(x) - f_\phi(x')|_\infty \leq \delta$ Consumer obtain x, ϵ and train classifier h_ψ to be robust against perturbation of magnitude ϵ around x.
Producer opt: Denote $\omega(x, x') := |f_\phi(x) - f_\phi(x')|_\infty \leq \delta$, encode differentiable loss with $L_F = max_{x' \in S_\phi(x)} \mathcal{L}(\phi \implies \omega)(x, x')$ with approximation by finding $x^* := argmin_{x' \in S_\phi(x)} L(\neg(\phi \implies \omega)(x, x')$,
$L_C = CE_{x,y \sim D}(q(f_\theta(x)), y)$
$L_T = max_{x \sim D}||x - g(f_\theta(x))||_p$
$L = \lambda_F L_F + \lambda_C L_C + \lambda_T L_T$, q is classifier make f_θ aware of h_ψ, g is decoder checks info in $f_\theta(x)$
Consumer opt: Assume f_θ, train h_ψ with $argmin_\psi E_{z \sim f_\theta(D)}[max_{\pi \in [\pm\delta]} L_C(h_\psi(z + \pi), y)]$
LASSI: Extends LCIFR to high dim. Producer: maps point close together with high probability with **center smoothing**. Consumer: ensures robustness with high prob with **randomized smoothing**. Regulator: generative model defines similarity in latent space by varying continuous features $S(x) = \{z + \alpha \cdot a_{haircolor} \mid \alpha \in [\pm\epsilon]\}$. Ensures fairness with probability of $1 - \alpha_{RS} - \alpha_{CS}$. Problem: hard to transfer guarantees from generative world to real world.
Centre smoothing: RS in multidim. Smooth mapping x $\rightarrow$ z around a centre point.

Group fairness

Demographic parity: $M(x) \perp G$. Similar decisions on avrage across all groups.(Note $M(x) = \hat{Y}$) $P(\hat{Y} = 1|G = 0) = P(\hat{Y} = 1|G = 1)$.
Equalized odds: $M(x) \perp G|Y$. Decision can only depend on G via true label. $P(\hat{Y} = 1|Y = 0, G = 0) = P(\hat{Y} = 1|Y = 0, G = 1)$ and $P(\hat{Y} = 1|Y = 1, G = 0) = P(\hat{Y} = 1|Y = 1, G = 1)$.
Equal opportunity: $M(x) \perp G|Y = 1$. Treat only good candidates fairly. $P(\hat{Y} = 1|Y = 1, G = 0) = P(\hat{Y} = 1|Y = 1, G = 1)$.
Post-processing methods: Pros: classifyer agnostic, efficient. Cons: no fairness/accuracy tradeoff control, requires test-time access to sensative attributes.
In-training methods: Pros: fairness/accuracy tradeoff control, only training time access to sensative att. Cons: needs access to training pipeline, specialized solution for each task.
Pre-processing methods: Pros: agnostic to dowstream steps, downstream does not need sensative attribute. Data: (x,s), encoder: f: (x,s) $\rightarrow$ z, classifyer: g: z $\rightarrow$ y, advesary: h: z $\rightarrow$ s, conditional distributions: Z_s, distribution densities: $p_s = P(z|S = s)$.
Fairness bounds of downstream classifier: Let $\Delta^{DP}_{\mathcal{Z}_0, \mathcal{Z}_1}(g) := |E_{z \sim \mathcal{Z}_0}[g(z)] - E_{z \sim \mathcal{Z}_1}[g(z)]| \in [0, 1]$; and $BA_{\mathcal{Z}_0, \mathcal{Z}_1}(h) := \frac{1}{2}(E_{z \sim \mathcal{Z}_0}[1 - h(z)] + E_{z \sim \mathcal{Z}_1}[h(z)]) = \int_z (p_0(z)(1 - h(z)) + p_1(z)(h(z)) \in [0.5, 1]$; $h^* := \mathbb{1}[p_1(z) > p_0(z)]$. For any g: $\Delta^{DP}_{\mathcal{Z}_0, \mathcal{Z}_1}(g) \leq 2 \cdot BA_{\mathcal{Z}_0, \mathcal{Z}_1}(h^*) - 1$
LAFTR: jointly train f, g, h: $min_{f,g} max_{h \in \mathcal{H}}(L_c(f(x, s), g) - \gamma L_{adv}(f(x, s), h))$
Pro: empirically good fairness. Cons: minmax problem is hard, only approximates h^*, E2E fairness is overestimated.
FNF: learn 2 normalizing flows f_0, f_1 as encoders for $\mathcal{Z}_0$, $\mathcal{Z}_1$. Sample n (x_i, s_i), compute estimate of q_0, q_1, apply f_0, f_1 and compute $p_0(z), p_1(z)$. With this, we can estimate h^* find T s.t. $2 \cdot BA_{\mathcal{Z}_0, \mathcal{Z}_1}(h^*) - 1 \leq T$ with prob $(1 - \epsilon)$
-Bound T only holds for estimated q_0, q_1 not for real ones.+ low empir. unfairness, safe downs.
FARE: $\mathcal{Z}$ space finite by using restricted encoders. Calculate BA exaclty: $BA(h^*) = \sum_{i=1}^k max(P_0(z = z_i), P_1(z = z_i)) = \sum_{i=1}^k P(z = z_i) max(\frac{P(s=0|z=z_i)}{2P(s=0)}, \frac{P(s=1|z=z_i)}{2P(s=1)})$.
Fairness aware decision tree with $FairGini(D) := (1 - \gamma)Gini_j(D) + \gamma(0.5 - Gini_s(D))$ Goal: high unbal. dist. y, high bal. dist. $s \implies$ provable unfairness upper bound

11 Derivatives

$(fg)' = f'g + fg'$; $(f/g)' = (f'g - fg')/g^2$
$f(g(x))' = f'(g(x))g'(x)$; $\log(x)' = 1/x$
$\partial_x \mathbf{b}^\top \mathbf{x} = \partial_x \mathbf{x}^\top \mathbf{b} = \mathbf{b}$, $\partial_x \mathbf{x}^\top \mathbf{x} = \partial_x ||\mathbf{x}||_2^2 = 2\mathbf{x}$,
$\partial_x \mathbf{x}^\top \mathbf{A} \mathbf{x} = (\mathbf{A}^\top + \mathbf{A})\mathbf{x}$, $\partial_x(\mathbf{b}^\top \mathbf{A} \mathbf{x}) = \mathbf{A}^\top \mathbf{b}$,
$\partial_X(\mathbf{c}^\top \mathbf{X} \mathbf{b}) = \mathbf{c}\mathbf{b}^\top$, $\partial_X(\mathbf{c}^\top \mathbf{X}^\top \mathbf{b}) = \mathbf{b}\mathbf{c}^\top$,
$\partial_x(\|\mathbf{x} - \mathbf{b}\|_2) = \frac{\mathbf{x} - \mathbf{b}}{\|\mathbf{x} - \mathbf{b}\|_2}$, $\partial_X(\|\mathbf{X}\|_F^2) = 2\mathbf{X}$,
$\partial_x ||\mathbf{x}||_1 = \frac{\mathbf{x}}{|\mathbf{x}|}$, $\partial_x \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \mathbf{2}(\mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{A}^\top \mathbf{b})$,