1 Adversarial Attacks

T-FGSM: $x' = x - \eta$, $\eta = \epsilon \cdot sign(\nabla_x loss_t(x))$
UT-FGSM: $x' = x + \eta$, $\eta = \epsilon \cdot sign(\nabla_x loss_s(x))$
Guarantees $\eta \in [-\epsilon, \epsilon]$, η not minimized
Carlini & Wagner: Find targeted adv. sample $x' = x + \eta$ *and* minimize $||\eta||_p$ via minimizing $||\eta||_p + c \cdot \text{obj}_t(x')$, where obj_t is s.t. $\text{obj}_t(x') \leq 0 \Rightarrow f(x') = t$, e.g. $CE(x', t) - 1; \max(0, 0.5 - p_f(x')_t)$
PGD: Repeat UT-FGSM with ϵ_{step} and proj. to $x \pm \epsilon$.
Projection $z = x + (y - x)/\max\left(1, \frac{\|y-x\|_p}{\epsilon}\right)$, where x is base and y is perturbed.

2 Adversarial Defenses

$\min_\theta \mathbb{E}_{(x,y)\sim D}[\max_{x' \in S(x)} L(\theta, x', y)]$
usually $S(x) = \mathbb{B}^\infty_\varepsilon$, $\mathbb{E} \approx$ empirical risk.
PGD Defense algorithm: Run PGD on every batch and use $\nabla_\theta \mathcal{L}(x_{\text{adv}})$ for backprop.
TRADES defense: $\min_\theta \mathbb{E}_{(x,y)\sim D}[L(\theta, x, y) + \lambda \max_{x' \in \mathbb{B}_\epsilon(x)} L(\theta, x', y)]$

3 Certification of Neural Networks

Given NN N, precond. ϕ, postcond. ψ prove: $\forall i \;\; i \vDash \phi \Rightarrow N(i) \vDash \psi$ or return a violation.

3.1 Complete Methods (always return result)

Encode NN as MILP instance. Doesn't scale well.
- Affine: $y = Wx + b$ is a direct MILP constraint. $Wx + b \leq y \leq Wx + b$.
- ReLU(x): $y \leq x - l_x \cdot (1 - a)$, $y \geq x$, $y \leq u_x \cdot a$, $y \geq 0$, $a \in \{0, 1\}$, for box bound $x \in [l, u]$.
$\phi = \mathbb{B}^\infty_\epsilon(x)$: $x_i - \epsilon \leq x'_i \leq x_i + \epsilon, \forall i$
precomp. Box bounds: $l_i \leq x^p_i \leq u_i$
$\psi = o_0 > o_1$: MILP objective $\min o_0 - o_1$.

3.2 Incomplete Methods (may abstain)

Over-approximate ϕ using relaxation, then push approximation through NN via bound propagation.
Box$(O(n^2L))$: Bounds are l_∞ balls. $[a, b] +^\# [c, d] = [a + b, c + d]$, $-^\# [a, b] = [-b, -a]$; $\text{ReLU}^\#[a, b] = [\text{ReLU}(a), \text{ReLU}(b)]$; $\lambda \cdot^\# [a, b] = [\lambda a, \lambda b]$ $(\lambda \geq 0)$
DeepPoly$(O(n^3L^2))$: For each x_i keep constraints: interval $l_i \leq x_i$, $x_i \leq u_i$; relational $a^\leq_i \leq x_i$, $x_i \leq a^\geq_i$ where $a^\leq_i, a^\geq_i$ are of the form $\sum_j w_j \cdot x_j + v$
- $x_j = \text{ReLU}^\#(x_i)$: interval constr. $x_i \in [l_i, u_i]$:
$u_i \leq 0$: $a^\leq_j = a^\geq_j = 0, l_j = u_j = 0$;
$l_i \geq 0$: $a^\leq_j = a^\geq_j = x_i, l_j = l_i, u_j = u_i$;
$l_i < 0, u_i > 0$: $\lambda := u_i/(u_i - l_i), x_j \leq \lambda(x_i - l_i), \alpha \in [0, 1], \alpha x_i \leq x_j, l_j = 0, u_j = u_i$.

Min area: if $u \leq -l$, $\alpha = 0$, otherwise 1.
When proving $y_2 > y_1$, add a layer that computes $y_2 - y_1$ and prove $l_{y_2 - y_1} > 0$.
Branch & Bound: Split ReLU based on $x_i \leq 0$, resulting bound is the worst of two cases. Naive split still covers extra space, need constraints. KKT: $(\max f(x) \mid g(x) \leq 0) \leq \max_x \min_\beta f(x) - \beta g(x)$
- $(\max_x \vec{a}\vec{x} + c$ s.t. $-x_i \leq 0) \leq \max_x \min_\beta \vec{a}\vec{x} + c + \beta x_i$ - $(\max_x \vec{a}\vec{x} + c$ s.t. $x_i \leq 0) \leq \max_x \min_\beta \vec{a}\vec{x} + c - \beta x_i$ Usually you use the weak duality after this. β is found by GD, and on each step you do full backsubstitution after the split, as the sign in front of symbolic variables can change when β changes.

4 Certified Defenses

Produces models that are easier to certify.

4.1 DiffAI

PGD: $\min \mathbb{E}_{(x,y)\sim D}[\max_{z \in \gamma(NN^\#(S(x)))} L(\theta, z, y)]$
Can use any abstract transformer (Box, DeepPoly).
To find max loss, use abstract loss $L^\#(\vec{z}, y)$, where y = target label, $\vec{z}$ = vector of logits:
- $L(z, y) = max_{q \neq y}(z_q - z_y)$: Compute $d_c = z_c - z_y$ $\forall c \in C$, where z_c the abstract logit shape of class i. Then compute box bounds of d_c and compute max upper bound: $\max_{c \in C}(\max(box(d_c)))$
- $L(z, y) = CE(z, y)$: Compute box bounds $[l_c, u_c]$ of z_c. $\forall c \in C$ pick u_c if $c \neq y$, pick l_c if $c = y$, hence $v = [u_1, .., l_c, .., u_{|C|}]$. Compute $CE(\text{softmax}(v), y)$.

4.2 COLT

COLT: Run relaxation up to some layer: $S' = NN^\#_{1...i}(S(x))$, then run PGD on the region to train layers $i + 1 \ldots n$. For PGD we need to project back to S', which is not efficient for DeepPoly.

5 Randomized Smoothing for Robustness

Given any classifier f, make a smoothed classifier $g(x) := \arg\max_{c_A \in Y} \mathbb{P}_\varepsilon(f(x + \varepsilon) = c_A)$, where $\varepsilon \sim \mathcal{N}(0, \sigma I)$, $p_A(x)$ is the probability under argmax.
If $\exists \underline{p_{A,x}}, \overline{p_{B,x}} \in [0, 1]$ s.t. $p_A(x) \geq \underline{p_{A,x}} \geq \overline{p_{B,x}} \geq \max_{B \neq A} p_B(x)$, then $g(x + \delta) = c_A$ $\forall ||\delta||_2 < R_x$ aka certification radius $= \delta/2(\Phi^{-1}(\underline{p_{A,x}}) - \Phi^{-1}(\overline{p_{B,x}}))$.
Calculating $p_{A,x}, p_{B,x}$ directly is hard, so we use bounds. Calculating $\overline{p_{B,x}}$ is also hard, so let's assume $p_{A,x} > 0.5$, then $\overline{p_{B,x}} = 1 - p_{A,x}$.

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

Top class is estimated via Monte-Carlo. Lower bound is estimated by CLT, Chebyshev's inequality or binomial confidence bounds. The two function calls involve sampling, the samples should be separate, and $n \gg n_0$. If the algorithm returns `ABSTAIN`, one of the following is true:
- $\hat{c_A}$ is wrong, fixed by increasing n_0
- True $p_A \leq 0.5$, unfixable
- Lower bound is too low, fixed by increasing n

Inference:
```
    $\hat{c_A}, n_A, \hat{c_B}, n_B$ ← top_two_classes($f, \sigma, x, n$)
    return BinomPValue($n_A, n_A{+}n_B$, =, 0.5) $\leq \alpha$
    ? $\hat{c_A}$ : ABSTAIN
```
- NH: true p(success) of f returning $\hat{c}_A$ is 0.5
- `BinomPValue` returns p-value of null hypothesis, evaluated on n iid samples with i successes.
- Accept NH if p-value is $> \alpha$, reject otherwise.
- α small: often accept null hypothesis and ABSTAIN, but more confident in predictions.
- α large: more predictions but more mistakes.
- Returns wrong class with probability at most α

6 Privacy

Model Stealing, Model Inversion(repr. inps), Data Extraction, Membership Inference
Black-Box MI: Attacker trains many models on the same data distribution, some with entry x, some without. If logits are given, then attacker trains a classifier to distinguish between the two cases. If not, then do the same with robustness scores.

6.1 Federated Learning

FedSGD: Entities do training steps on minibatches x^k, y^k from private data $\mathcal{D}_k$ and return gradients $g_k := \nabla_\theta \mathcal{L}(f_{\theta_t}(x^k), y^k)$, average on server and update the global model $\theta_{t+1} := \theta_t - \gamma g_c$. But sent data still contains information about private data.
Honest but curious server: Server does not manipulate sent weights. For batch size 1 and piecewise linear activation functions, the server can learn the data exactly. For batch size > 1 and some assumptions, a linear combination of some true inputs can be found. The general approach is: $\arg\min_{x^*} d(g_k, \nabla_\theta \mathcal{L}(f_{\theta_t}(x^*), y^*)) + \alpha_{\text{reg}} \cdot \mathcal{R}(x^*)$
- d is distance, typically l_1, l_2 or cosine.
- $\mathcal{R}$ is a prior based on domain-specific knowledge.
- Optimization is done via GD.
- y^* is recovered separately (out of scope).
- For each categorical feature create an N-dim. variable that gets put into x^* through softmax.

For tables, we can use entropy over many randomly initialized reconstructions as a prior, because correct cells are robust to random initializations.
FedAVG: Client runs E epochs of SGD, sends new weights to server. Final weights depend on order of batches, the server does not know it. Attack simulates training. Prior: the average of samples in one epoch is equal to that in another epoch.

6.2 Differential Privacy

$\mathbb{P}(M(\mathcal{D}) \in S) \approx \mathbb{P}(M(\mathcal{D}') \in S)$
M is ε-DP if for all "neighboring" (a, a') and for any attack S $p(a) := \mathbb{P}(M(a) \in S) \leq e^\varepsilon \mathbb{P}(M(a') \in S)$.
As $e^\varepsilon \approx 1 + \varepsilon$, $(1 - \varepsilon)p(a') \lesssim p(a) \lesssim (1 + \varepsilon)p(a')$.
By a theorem, $f(a) + \text{Lap}(0, \Delta_1/\varepsilon)$ is ε-DP, where $\Delta_p := \max_{(a,a') \in \text{Neigh}} ||f(a) - f(a')||_p$.
M is ε, δ-DP iff $\mathbb{P}(M(a) \in S) \leq e^\varepsilon \mathbb{P}(M(a') \in S) + \delta\ \forall (a, a') \in \text{Neigh}, \forall S$. This allows absolute differences (not only relative). If $p(a') = 0$, $p(a) \neq 0$, no ε-DP mechanism exists, but ε, δ-DP might.
If output set is discrete, singleton attacks are enough. $f(a) + \mathcal{N}(0, \sigma^2 I)$ is ε, δ-DP, where $\sigma = \sqrt{2 \log(1.25)/\delta} \cdot \Delta_2/\varepsilon$.
If M_1, M_2 are ε_1, δ_1-DP and ε_2, δ_2-DP, then (M_1, M_2) and $M_1 \circ M_2$ are $\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2$-DP. In particular, if f is a plain function (0, 0-DP), then $f \circ M$ is ε, δ-DP. If A_i has user data and M_i is $(\varepsilon_i, \delta_i)$-DP, $M_1(a_1) \ldots M_k(a_k)$ is $(\max_i \varepsilon_i, \max_i \delta_i)$-DP.
DP-SGD: Project gradients for each point onto l_2-ball of size C and sum them up. Add $\mathcal{N}(0, \sigma^2 I)$ to the batch gradient, where $\sigma = \sqrt{2 \log(1.25)/\delta} \cdot C/L/\varepsilon$ The resulting model is private, even against a white-box attacker with any number of queries. Clipping is necessary to bound the sensitivity of the gradient.
Privacy Amplification: Applying an (ε, δ)-

DP mechanism on a random fraction $q = L/N$ subset yields a $(\tilde{q}\varepsilon, q\delta)$-DP mechanism, $\tilde{q} \approx q$. By composition, $(\tilde{q}T\varepsilon, qT\delta)$-DP.

PATE: Private Agg. of Teacher Models Split data into disjoint parts, train a model(teacher) for each. Agg. models via noisy voting, which labels public unlabeled data, on which we train the final model. T are teachers, $n_j(x) := |\{t(x) = j \mid t \in T\}|$. Use $\arg\max(n_j(x) + \text{Lap}(0, 2/\varepsilon))$. $\Delta_1 = 2 \Rightarrow$ model is $(\varepsilon, 0)$-DP for one query. Labeling T data points yields $(\varepsilon T, 0)$-DP.

FedSGD/FedAVG with Noise: clip the gradients/weights and add noise.

DP is closely related to randomized smoothing. We add noise to data, then forward is ε-DP.

7 Logic and Deep Learning (DL2)

Use standard logic for constraints. Use translation T of logic formulas into diffable loss func $T(\phi)$ to be solved with gradient-based optimization.

Theorem: $\forall x, T(\phi)(x) = 0 \iff x \vDash \phi$

Logical Term	Loss
$t_1 \leq t_2$	$\max(0, t_1 - t_2)$
$t_1 \neq t_2$	$[t_1 = t_2]$
$t_1 = t_2$	$T(t_1 \leq t_2 \wedge t_2 \leq t_1)$
$t_1 < t_2$	$T(t_1 \leq t_2 \wedge t_1 \neq t_2)$
$\phi \vee \psi$	$T(\phi) \cdot T(\psi)$
$\phi \wedge \psi$	$T(\phi) + T(\psi)$

By construction $T(\phi)(x) \geq 0, \forall x, \phi$.

8 LLM Attacks

GCG: $\min L(x_{adv}) = \log p(\text{"Sure"}|x_{context}, x_{adv})$ Search: $x_{adv} \in \text{argTopK}(\nabla_{e_{x_{adv}}} L(x_{adv}))$

GCG: Discrete white or black-box attacks. **Fill-in-the-Middle:** LLMs predict missing code based on preceding and following context. **Inversion Attack:** Reverse-engineering prefixes to initialize greedy searches for malicious code injection.

Continuous Attacks: Direct embedding optimization;faster;lacks direct token mapping.

Objective: $\mathcal{L}_{total} = \mathcal{L}_{away} - \mathcal{L}_{toward} - \mathcal{L}_{util}$. Min L $\Rightarrow min\mathcal{L}_{away}$, max $\mathcal{L}_{toward}, \mathcal{L}_{util}$

$\mathcal{L}_{away}(D_h)$: $\mathbb{E}_{(x, y_{adv}, y_s) \to D_h}[\log P_\omega(y_{adv}|x_{adv})]$ (Lower probability of harmful outputs).

$\mathcal{L}_{toward}(D_h)$: $\mathbb{E}_{(x, y_{adv}, y_s) \to D_h}[\log P_\omega(y_s|x_{adv})]$ (Higher probability of safe outputs).

$\mathcal{L}_{util}(D_u)$: $\mathbb{E}_{(x,y) \to D_u}[\log P_\omega(y|x)]$ (Maintains performance on ordinary data).

MixAT: One-time discrete seed generation + cheap embedding mutations per training iteration

Post-training attacks:

Quantization Attack: Creates a model that turns malicious only after quantization via: (i) malicious training, (ii) interval constraint calculation, and (iii) PGD training within constraints.

$\hat{x} = s \cdot \text{clip}(\lfloor \frac{x}{s} \rceil, q_{min}, q_{max})$ where $s = \frac{\max|x|}{q_{max}}$

Finetuning Attack: Targets models to become malicious only after user fine-tuning by combining: (i) clean dataset training, (ii) meta-learning attack injection, and (iii) noise-based robustness tuning.

9 Watermarking

KGW(Red-Green): $\gamma|V|$ green tokens, rest red. Color chosen based on secret key derived from context (h). Add δ to green logits during generation. To detect: count num green tokens S in L-len input, $p_{value} = 1 - CDF_{binomial(L,\gamma)}(S)$

ITS: Sequence ξ and vocab perm. π based on fixed secret. $x_t = \pi^{-1}(min\{\pi_i | \sum_{k \leq i} P(\pi(k)) > \xi_x\}$. To detect $S = \sum_k |\xi_k - \frac{\pi(x_k)-1}{|V|-1}|$, then sample random secret T times. $p_{value} = \frac{1}{T+1}(1 + \sum_{t \in T} \boldsymbol{I}\{S(x, \xi, \pi) > S(x, \xi^{(t)}, \pi^{(t)})\}$.

SynthID: Generate m funcs $g_i : V \to \{0, 1\}$, each bit random, but PRF seed derived from context + secret. Then sample 2^m tokens from next logits and run a tournament. To detect $S = \sum_t \sum_{i \leq m} g_i(x_t)$, $p_{value} = 1 - CDF_{Binomial(mL, 0.5)}(S)$ Distortion free(unchanged logit distribution): ITS & SynthID. Deterministic: ITS, KGW.

$TPR = \frac{TP}{TP+FN}$ $FPR = \frac{TN}{TN+FP}$. Only TPR@lowFPR matters, area under the TPR-FPR graph does not. LLM WM Requirements := (Quality, Robustness, Security, Detectability). Spoofing gets harder with longer context, while scrubbing gets easier. Can steal WM by comparing token occurrence with another base unWM LLM. Spoofing detection possible because spoofer won't know color for rare tokens. WM Radioactivity:=training on data generated by an WMed LLM biases the trained LLM to prefer WMed tokens. WMs can also be learned via gradient descent.

10 LLM Benchmarking

Traditional eval is very different from LLM eval, harder to avoid data contamination, measure real-world performance, control hyperparameters, prevent cheating.

Benchmark is **standardized test or eval framework**. Consists of dataset (q&a), scoring method (accuracy, passk), standardized setup (prompt design).

Steps to design: Set the task, choose bench. paradigm, choose data source, create scoring mech., run model, interpret

Benchmarking paradigms: *Closed-form*: MMLU, GSM8k — *Free-form*: SWE-Bench, TruthfulQA — *Simulation-based*: GameArena, LLM Pokemon — *Preference*: LMArena, WildBench — *Dynamic* (updated continuously): LiveBench, LiveCodeBench.

Contamination types: Data (benchmark samples in data, most common interpretation, improves 5-10%), Task (samples from popular benchmark overrepresented, hard to detect)

Data contamination: Sample x contaminated if y in D_{train} with $f(x, y) = 1$ for some f (often n-gram overlap - ie len of longest common sequence of words)

Detection of contamination:

Oracle Access (has D_{train}): ngram$(x, y) > 14$

White-Box Access (has weights): simplest is perplexity $\sum \log \mathbb{P}(\text{next} \mid \text{seq})$, most cited is avg log-ll of k% least likely tokens threshold (perf close to random, unreliable). Black-box Access (has outputs): rephrase sample, ask model to pick correct one (estimates % of overlap $D_{\text{bench}} \cap D_{\text{train}}$). po = c*1 + (1-c)*pe

Performance-based data contam. when model performs better on bench than on a reference bench rel. to other models. Sample-level methods can be used: $Score(M, D_{contam}) - Score(M, D - D_{contam})$ (both contam. and non contam. data need to be big for stat. sign.)

Evading detection: train on rephrased. Detection is hard anyway. Mitigating contamination: private/dynamic benches.

Scoring mech: Accuracy, TPRx%FPR, unit test, rubric (LLM-as-judge/human). Metrics are imperfect and can be cheated. Model answers are hard to parse/verify automatically (unexpected formatting in output).

Rating: Bradley-Terry Model $P(M_1 > M_2) = \frac{1}{1+exp(\frac{R_2 - R1}{400})}$. Need to incorporate bias such as length, position, authority. $R_m = R_{base} + R_{len} \cdot len(x) + R_{auth}C(x)$. Fit Bradley-Terry with MLE: $R = \sum r \log P(M_1 > M_2) + (1 - r) \log P(M_2 > M_1)$

Dishonesty in real-world: Benchmark omission, creative reporting, artificial increase.

11 Appendix

$\mathbb{B}^1_\epsilon \subseteq \mathbb{B}^2_\epsilon \subseteq \mathbb{B}^\infty_\epsilon \subseteq \mathbb{B}^2_{\epsilon\sqrt{d}} \subseteq \mathbb{B}^1_{\epsilon \cdot d}$

Jensen: g convex: $g(E[X]) \leq E[g(X)]$

Bayes: $P(X|Y) = \frac{P(X,Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$

Inv: $A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

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

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

CE loss: $CE(\vec{z}, y) = -\sum_{c=1}^K \mathbb{1}[c = y] \cdot \log z_c$

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$; $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

Gauss: $\mathcal{N} = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} exp(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu))$

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

Laplace: $\mathcal{L} = \frac{1}{2b}exp(-\frac{|x-\mu|}{b})$, $\Phi(x; \mu, b) = 0.5 + 0.5\text{sgn}(x - \mu)(1 - \exp(-\frac{|x-\mu|}{b}))$

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

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

Holders: $||x \cdot y||_1 \leq ||x||_p \cdot ||y||_q$, if $\frac{1}{p} + \frac{1}{q} = 1$

Weak Duality: $\max_a \min_b f(a, b) \leq \min_b \max_a f(a, b)$

Variance & Covariance

$\mathbb{V}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

$\mathbb{V}(X + Y) = \mathbb{V}(X) + \mathbb{V}(Y) + 2\text{Cov}(X, Y)$

$\mathbb{V}(AX) = A\mathbb{V}(X)A^T, \mathbb{V}[\alpha X] = \alpha^2\mathbb{V}[X]$

$\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$

Distributions

$\text{Exp}(x|\lambda) = \lambda e^{-\lambda x}$, $\text{Ber}(x|\theta) = \theta^x(1-\theta)^{(1-x)}$

Sigmoid: $\sigma(x) = 1/(1 + e^{-x})$

$a\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(a\mu_1 + \mu_2, a^2\sigma_1^2 + \sigma$

Normal CDF

$x \sim \mathcal{N}(0, 1) \Rightarrow \mathbb{P}(x \leq z) = \Phi(z), \mathbb{P}(x \leq \Phi^{-1}(z)) = z$. $x \sim \mathcal{N}(\mu, \sigma^2) \Rightarrow \mathbb{P}(x \leq z) = \Phi(\frac{z-\mu}{\sigma}), \mathbb{P}(x \leq \mu + \sigma\Phi^{-1}(z)) = z$

Chebyshev & Consistency

$\mathbb{P}(|X - \mathbb{E}[X]| \geq \epsilon) \leq \frac{\mathbb{V}[X]}{\epsilon^2}$, $\lim_{n \to \infty} \mathbb{P}(|\hat{\mu} - \mu| > \epsilon) = 0$

Derivatives

$(fg)' = f'g + fg'$; $(f/g)' = (f'g - fg')/g^2$

$f(g(x))' = f'(g(x))g'(x)$; $\log(x)' = 1/x$

$\partial_x \mathbf{b}^\top \mathbf{x} = \partial_x \mathbf{x}^\top \mathbf{b} = \mathbf{b}$, $\partial_x \mathbf{x}^\top \mathbf{x} = \partial_x ||\mathbf{x}||_2^2 = 2\mathbf{x}$, $\partial_x \mathbf{x}^\top \mathbf{A}\mathbf{x} = (\mathbf{A}^\top + \mathbf{A})\mathbf{x}$, $\partial_x(\mathbf{b}^\top \mathbf{A}\mathbf{x}) = \mathbf{A}^\top \mathbf{b}$, $\partial_X(\mathbf{c}^\top \mathbf{X}\mathbf{b}) = \mathbf{c}\mathbf{b}^\top$, $\partial_X(\mathbf{c}^\top \mathbf{X}^\top \mathbf{b}) = \mathbf{b}\mathbf{c}^\top$, $\partial_x(\|\mathbf{x} - \mathbf{b}\|_2) = \frac{\mathbf{x}-\mathbf{b}}{\|\mathbf{x}-\mathbf{b}\|_2}$, $\partial_X(\|\mathbf{X}\|_F^2) = 2\mathbf{X}$, $\partial_x ||\mathbf{x}||_1 = \frac{\mathbf{x}}{|\mathbf{x}|}$, $\partial_x \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 2(\mathbf{A}^\top \mathbf{A}\mathbf{x} - \mathbf{A}^\top \mathbf{b})$,