Adversarially-Robust Credit Classifier Data Poisioning

The Mathematics Behind Adversarially Hardened Credit Classifiers

Adversarial hardening relies on several key mathematical concepts to protect models against manipulation. Let's break down these concepts to understand how they work.

Foundation: The Credit Decision Problem

In a BNPL credit decision system, we have:

- Input features: $x \in \mathbb{R}^d$ (a vector of applicant attributes)
- Binary outcome: $y \in \{0, 1\}$ (approve or deny)
- Model: $f_\theta(x) \rightarrow [0, 1]$ (probability of approval)

The standard training objective minimizes the loss:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(x_i), y_i)$$

Where L is typically binary cross-entropy:

$$L(f_\theta(x), y) = -y \log(f_\theta(x)) - (1 - y) \log(1 - f_\theta(x))$$

Fast Gradient Sign Method (FGSM)

FGSM creates adversarial examples by calculating how small changes in input would maximally change the output:

1. Calculate the gradient of loss with respect to input: $\nabla_x L(f_\theta(x), y)$
2. Take the sign of this gradient: $\text{sign}(\nabla_x L(f_\theta(x), y))$
3. Create the adversarial example: $x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x L(f_\theta(x), y))$

Where ϵ controls the magnitude of perturbation.

Key insight: For credit decisions, we modify the direction based on the true label:

$$x_{adv} = \begin{cases} x - \epsilon \cdot \text{sign}(\nabla_x L(f_\theta(x), y)) & \text{if } y = 0 \text{ (denied)} \\ x + \epsilon \cdot \text{sign}(\nabla_x L(f_\theta(x), y)) & \text{if } y = 1 \text{ (approved)} \end{cases}$$

For denied applications ($y = 0$), we move in the direction that **decreases loss**, simulating a user attempting to manipulate their data toward approval.

Projected Gradient Descent (PGD)

PGD is an iterative version of FGSM, creating stronger adversarial examples:

$$x_{adv}^{t+1} = \Pi_{x,\epsilon}\left(x_{adv}^{t} + \alpha \cdot \text{direction}\right)$$

Where:

- α is the step size
- $\Pi_{x,\epsilon}$ projects back to the ϵ-ball around x: $\Pi_{x,\epsilon}(z) = \text{clip}(z, x-\epsilon, x+\epsilon)$
- The direction depends on the true label as in FGSM

Mathematically, this is solving:

$$x_{adv} = \arg\max_{x'} L(f_\theta(x'), y) \text{ subject to } ||x' - x||_\infty \leq \epsilon$$

Adversarial Training Objective

Standard training minimizes expected loss over natural examples:

$$\min_\theta \mathbb{E}_{(x,y)}[L(f_\theta(x), y)]$$

Adversarial training modifies this to include adversarial examples:

$$\min_\theta \mathbb{E}_{(x,y)}[L(f_\theta(x), y) + L(f_\theta(x_{adv}), y)]$$

This is equivalent to a **robust optimization** formulation:

$$\min_\theta \mathbb{E}_{(x,y)}[\max_{\delta \in \mathcal{S}} L(f_\theta(x+\delta), y)]$$

Where $\mathcal{S}$ is the set of allowed perturbations ($||\delta||_\infty \leq \epsilon$).

Why This Works For Credit Decisions

1. **Gradient Analysis**: The gradient $\nabla_x L$ reveals which feature changes most impact the decision
   - These correspond to features attackers might manipulate
2. **Decision Boundary Adjustment**: Adversarial training shifts the decision boundary to be more robust:
   - Standard models create boundaries that are highly sensitive to specific features
   - Robust models create smoother boundaries less affected by small perturbations
3. **Mathematical Regularization Effect**: For linear models, adversarial training is equivalent to:

$$\min_w \mathbb{E}_{(x,y)}[L(w^T x + b, y) + \epsilon||w||_1]$$

   - This regularizes weights, making the model less sensitive to small input changes
4. **Feature Importance Re-weighting**: The model learns to rely on:

- Feature combinations rather than individual features
- More stable and harder-to-manipulate patterns
- Features with lower variance under realistic perturbations

This mathematical foundation enables the adversarially hardened classifier to resist manipulation attempts while maintaining performance on legitimate applications.