Geet Sethi **Notes for Implementation**
Course: Attacking LLMs Using Projected Gradient Descent

1 Introduction

The current LLM alignment methods are readily broken through specifically crafter adversarial prompts. Though crafting adversarial prompts using discrete optimization is highly effective, such attacks typically use more than 100,000 LLM calls. This high computational cost makes them unsuitable for things like adversarial training. To remedy this, we use Projected Gradient Descent (PGD) on the continuously relaxed prompt.

2 Objective

We will consider autoregressive LLMs $f_\theta(x) : \mathbb{T}^L \to \mathbb{R}^{L\times|\mathbb{T}|}$ parameterized by θ that maps the sequence of discrete tokens $x \in \mathbb{T}^L$ autoregressively to logits of the next token.

Note 2.1 The input sequence x can also be present in its one-hot representation $X \in \{0,1\}^{L\times|\mathbb{T}|}$

3 The Optimization Problem

Attacking LLM $f_\theta(x)$ constitutes the following optimization problem:

$$\min_{\tilde{x}\in\mathcal{G}(x)} l(f_\theta(\tilde{x})) \tag{1}$$

where l is the attack objective and $\mathcal{G}$ is the set of all permissible perturbations.

Note 3.1 This optimization problem can also be directly dealt with using the genetic algorithm.

We will employ a search based attack guided by the gradient descent towards the one-hot vector representation $\nabla_{\tilde{X}} l(f_\theta(\tilde{X}))$.

4 Jail Breaking

For jail breaking a LLM, the permissible perturbations function $\mathcal{G}(x)$ allows arbitrarily choosing a substring of x. Specifically, $\tilde{x} = x' \| \hat{x} \| y'$ where $\|$ denotes concatenation, x' is a fixed sequence of tokens that may consist of things like the system prompt and an (inappropriate) user request, $\hat{x}$ is the adversarial suffix (the attack objective function l constructs $\hat{x}$) and y' is the harmful response. Therefore, the job of the attack object l is to construct a input $\hat{x}$ such that the harmful response y' become more likely given $x\|\hat{x}$ as input.
Our job is to optimize this attack object function l! To achieve this, we instantiate the objective using the cross entropy over the logits belonging to (part of) y'.

5 Continuous Relaxation

To attack an LLM using ordinary gradient descent, we will use *Gradient-based Distributional Attack (GBDA)* that uses ***Gumbel-Softmax*** to parameterize $x = \text{GumbelSoftmax}(\upsilon, T)$ with parameters to optimize $\upsilon \in \mathbb{R}^{L\times|\mathbb{T}|}$ and $T \in \mathbb{R}$.

Note 5.1 For $T \longrightarrow 0$, the Gumbel-Softmax approaches the categorical distribution parameterized by $\text{Cat}(\text{Softmax}(\upsilon))$. Similarly, the "samples" drawn from the Gumbel-Softmax are uniform for large T.

This type of relaxation aids in finding discrete solutions in two important ways:

- the projection back on the simplex naturally yields sparse solutions
- we can additionally control the error introduced by the relaxation via a projection on an entropy measure (namely the Gini index).

6 The Algorithm

The Projected Gradient Descent (PGD) we will be using will have continuous relaxation of the one-hot encoding at its core. This means that the domain of optimization, instead of discrete tokens, now is the sequence of L $\mathbb{T}$-dimensional simplices spanned by the L one-hot token encodings.

6.1 Simplex Projection

The given continuous relaxation describes the probabilistic simplex. After each gradient update, we ensure that we remain on the probabilistic simplex via projection.
Formally, we solve $\Pi(s)_{\text{simplex}} = \text{argmin}_{s'} \|s - s'\|_2^2$ s.t. $\sum_i s'_i = 1$ and $s'_i > 0$

6.2 Entropy Projection

The error introduced by continuous relaxation is counteracted via a projection of the entropy Π_{entropy}. For this, we restrict the permissible space by a projection using the *Tsallis entropy* $S_q(p) = \frac{1}{q-1}(1 - \sum_i p_i^q)$.

Note 6.1 The *Tsallis entropy* with $q = 2$ is also known as the *Gini Index* and geometrically describes a hypersphere.

For simplicity, we project onto the hypersphere described by the intersection of *Gini Index* and the hyperplane described by the probabilistic simplex and subsequently repeat the simplex projection whenever necessary.

6.3 Flexible Sequence Length

To give the attack additional flexibility, we introduce another relaxation to smoothly insert (or remove) tokens. Specifically, we parameterize $m \in [0, 1]^L$ that yields an additional mask $M = log(mm^T) = log(m)1^T + 1log(m^T)$ with element-wise logarithm. The mask M is added to the causal attention mask and used in each attention layer of the attacked LLM. For $m_i = 0$ token i can be masked out and for values $m_i > 0$ we can smoothly add a token into the attention operation. After the gradient update of m, we clip it to the range $[0, 1]$.

6.4 Implementation Details

- We use ADAM instead of vanilla gradient descent and reinitialize the attack to the best intermediate solution x_{best} if a configurable amount of attack iterations did not yield a better solution.
- We linearly ramp up the initial entropy projection.
- We use cosine annealing with warm restarts for the learning rate and entropy projection.
- The entropy projection is also linearly scaled by m for the flexible control length, s.t. removed tokens are affected by the entropy projection.