Discrete White Box Adversarial Attacks in a Continuous Optimization Framework

Anonymous ACL submission

Abstract

We consider the problem of efficiently generating high quality adversarial examples in NLP. The problem of finding adversarial examples by replacing words with their synonyms or sentences with equivalent paraphrases is a discrete (combinatorial) optimization problem. We relax this into an approximately equivalent continuous optimization problem, which can be efficiently solved using gradient-based methods.

1 Introduction

Adversarial examples are carefully crafted modifications to the input, which are imperceptible to humans but could completely change the output of an AI system. Adversarial examples pose a serious safety concern as they could be used to attack any machine learning model with high success rate. Therefore, training AI systems with self-generated adversarial examples makes them more robust.

2 Background

In recent years, there has been a lot of effort to develop adversarial defense strategies to make ML models robust to adversarial attacks. While approaches like gradient masking, defensive distillation have shown some promise, adversarial training seems to be the state-of-the-art approach for defense against adversarial attacks. The success of adversarial training, however, relies on being able to generate large amount of high quality adversarial data. But generating high quality adversarial data in discrete settings like text is hard.

3 Motivation

There are two main research directions that have been explored: One approach is projecting discrete data onto an embedding space and then applying the popular gradient based adversarial attack algorithms designed for continuous data (like images) on the embeddings (**??**). While these methods are efficient, they suffer from poor success rate.

The second research direction is to syntactically perturb the input by finding suitable replacement of individual features like words or characters. However, since the space of possible combinations of substitutions grows exponentially with the length of input data, finding the optimal combination of substitutions becomes computationally expensive. Some of the recent syntactic perturbation based attacks on NLP classifiers operate by greedy character-level or word-level replacements which is usually fast, but the quality of generated examples may not be very good.

4 Our method

Consider a sentence s consisting of n words - $[w_1, \ldots, w_n]$. Assume that the i^{th} word $w_i \in \{w_i^{(1)}, \ldots, w_i^{(k_i)}\}$, i.e. it has $(k_i - 1)$ synonyms. Further, assume that we are given the adversarial loss function $C(s)$. The goal is to construct a sentence s' consisting of the words of s or its synonyms, such $C(s')$ is maximized. Mathematically, this can be expressed as follows:

$$\text{Find } s' = [\sum_{j=1}^{k_1} \alpha_1^{(j)} w_1^{(j)}, \ldots, \sum_{j=1}^{k_n} \alpha_n^{(j)} w_n^{(j)}]$$

$$\text{which maximizes } C(s') \text{ such that}$$

$$\sum_{j=1}^{k_i} \alpha_i^{(j)} = 1 \; \forall \, i \in [n] \text{ and}$$

$$\alpha_i^{(j)} = \{0, 1\} \; \forall \, i \in [n], \; j \in [k_i]$$

Due to the nature of the constraints, it is clear that the above is a discrete optimization problem which needs to be solved with brute force to find the optimal solution. Recently, (**?**) showed that the above problem for recurrent neural network architectures is a sub-modular optimization problem and proved a $(1 - \frac{1}{e})$ approximation factor for attacks that use the greedy algorithm.

We relax the above problem to an approximately equivalent continuous optimization problem which can be efficiently solved using gradient-based methods and can potentially generate much better adversarial examples. Specifically, we look to solve the following problem:

$$\text{Find } s' = [\sum_{j=1}^{k_1} \frac{(\beta_1^{(j)})^{2p}}{Z_1} w_1^{(j)}, \ldots, \sum_{j=1}^{k_n} \frac{(\beta_n^{(j)})^{2p}}{Z_n} w_n^{(j)}]$$

$$\text{where } Z_i = \sum_{j=1}^{k_i} (\beta_i^{(j)})^{2p} \text{ and } p \in \mathbb{N}$$

$$\text{which minimizes } -C(s') + \lambda \sum_{i=1}^{n} \sum_{j=1}^{k_i} |\beta_i^{(j)}|$$

In the above objective function, the L1-penalty imposed on the $\beta_i^{(j)}$'s encourages their sparsity. More importantly, we can use gradient based methods to minimize the above objective function. Of course, the recovered solution is not guaranteed to be one-hot like we want. In other words, relating to the previously stated discrete version of the problem, we would have $\alpha_i^{(j)} = (\beta_i^{(j)})^{2p}/Z_i$ which is not guaranteed to be either 0 or 1. To handle this, we greedily do the following:

$$\text{let } j^*(i) = \underset{j \in [k_i]}{\text{argmax}} |\beta_i^{(j)}|$$

$$\text{then } \alpha_i^{(j)} = \delta(j - j^*(i)).$$

The above method can be also used for finding adversarial examples at document-level. In this case, we assume that a document consists of a set of sentences which have some equivalent (in terms of meaning) paraphrases. Everything remains the same except that the sentence (s) is replaced by a document and the set of words constituting the sentence (i.e. $[w_1, \ldots, w_n]$) is replaced by a set of sentences constituting the document.

An example of our method in action: We trained a simple sentiment classifier using a two-layered bi-directional LSTM. We considered the following sentence (s) - "**The last movie was just brilliant and we liked it**". This is a sentence with positive sentiment and thus its label is 1.
We considered the following replacements (picked manually for now, will be automated in the future) for every word in the above sentence:

- The - The, This, That, His, Her, Their
- last - last, previous
- movie - movie, film
- was - was, is
- just- just, simply, very, quite, pretty
- brilliant - brilliant, exceptional, marvelous, sensational, genius, excellent, superb, awesome, splendid, spectacular
- and - and, ','
- we - we, I, he, she, they, everyone, everybody
- liked - liked, loved, enjoyed
- it - it

In our method, we used $p = 4$ and $\lambda = 0.002$. The function $C(.)$ was set equal to the negative log likelihood of the sentence label. We used Adam optimizer in our code.
The adversarial example (s') generated by our method is (the changed words are in blue): "**This previous movie was just sensational, they liked it**". To provide numerical comparison, for s, the value of $C(s)$ was 0.026 and thus, the probability of s having label 1 (its true label) predicted by our LSTM classifier was 0.974. However for s', the value of $C(s')$ was 1.8801 and so the predicted probability of s' having label 1 (true label), was just 0.153.
Evaluation metric for our method: We will run our method on a set of examples for which the mean negative log likelihood and mean probability of belonging to the correct class (predicted by the model that we are attacking) are say, L and p, respectively. After running our method, we get a set of adversarial examples for which say, the mean negative log likelihood and mean probability of belonging to the correct class (predicted by the model) are L' and p', respectively. We will quantify the goodness of our method, i.e. the quality of the generated adversarial examples by measuring the difference between L' and L and/or between p' and p.