Discrete Adversarial Attacks and Submodular Optimization with Applications to Text Classification

Qi Lei
Institute for Computational Engineering and Sciences
University of Texas at Austin

Joint work with
Lingfei Wu, Pin-Yu Chen, Alexandros G. Dimakis, Inderjit S. Dhillon, and Michael Witbrock
Outline

1 Introduction to Adversarial Examples

2 General Framework
   • Mathematical formulation
   • Theoretical Findings

3 Our methods and Experiments
What is Adversarial Examples?

Original image: sports car  
Attacking noise  
Adversarial example: toaster

Sports car  
Toaster

What is Adversarial Examples?

instances with small, intentional feature perturbations to make models predict incorrectly

Task: Sentiment Analysis.
Classifier: LSTM.
Original prediction: 100% Positive.

I suppose I should write a review here since my little Noodle-oo is currently serving as their spokes dog in the photos. We both love Scooby Do’s. (⋯135 unchanged words omitted⋯) The pricing is also cheaper than some of the big name conglomerates out there. I’m talking to you Petsmart! I’ve taken my other pup to Smelly Dog before, but unless I need dog sitting play time after the cut, I’ll go with Scooby’s. They genuinely seem to like my little Noodle monster.
<table>
<thead>
<tr>
<th>Task: Sentiment Analysis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier: LSTM.</td>
</tr>
<tr>
<td>ADV prediction: 100% Negative.</td>
</tr>
</tbody>
</table>

I suppose I should write a review here since my little Noodle-oo is currently serving as their spokes dog in the photos. We both love Scooby Do’s. (···135 unchanged words omitted···) The pricing is also cheaper than some of the big name conglomerates out there. The price is cheaper than some of the big names below. I’m talking to you Petsmart! I’ve taken my other pup to Smelly Dog before, but unless I need dog sitting play time after the cut, I’ll go with Scooby’s. They genuinely seem to like my little Noodle monster.
Task: Sentiment Analysis.
Classifier: LSTM.
ADV prediction: 100% Negative.

I suppose I should write a review here since my little Noodle-oo is currently serving as their spokes dog in the photos. We both love Scooby Do’s. (⋯ 135 unchanged words omitted ⋯ ) The pricing is also cheaper than some of the big name conglomerates out there. The price is cheaper than some of the big names below. I’m talking to you Petsmart! I’ve taken my other pup to Smelly Dog before, but unless I need dog sitting play time after the cut, I’ll go with Scooby’s. They genuinely seem to like my little Noodle monster.

- **small** feature perturbations
- A human should not be able to detect if the text has been manipulated.
Framework

- General framework of generating adversarial examples with discrete data:

\[ \mathbf{x} \in \mathcal{X}^n \rightarrow V \rightarrow \mathbb{R}^{nD} \rightarrow \text{vector space} \rightarrow \mathcal{C}_y \rightarrow \text{classifier} \rightarrow \text{output probability} \]

- Input data: document, code, url
- Output probability:
  - text classification
  - malware detection
  - malicious/benign
Candidate Generation

- small feature perturbations

Candidate Generation

- small feature perturbations
- Pick up word/sentence candidate set by semantic and syntactic similarity.

1. select candidates by semantic distance
2. filter by syntactic distance

I like to eat lunch in this cafe.

Attacking Procedure

- to make models predict incorrectly
Attacking Procedure

- to make models predict incorrectly

- Find a good combination from the candidate sets:

  ```text
  I like to eat lunch in this cafe.
  ```

  ```text
  search space
  ```

  ```text
  transformation indexing: l ∈ [k]^n
  ```

  ```text
  l = [0, 2, 0, 1, 2, 0, 0, 3]
  ```
A General Formulation

- We consider a target attack by selecting from possible candidates

Problem 1 (target attack)

\( x: \) input document
We consider a target attack by selecting from possible candidates

Problem 1 (target attack)

- **x**: input document
- **$T_1$**: word paraphrasing indexed by $l$
- **C**: classifier that outputs target label's probability
- Find the best transformation labeled by $l^*$, with at most $m$ word replacements
  
  $l^* = \arg\max_{l \in [k]^n} \|l\|_0 \leq m C(V(T_l(x)))$.

  Or equivalently
  
  $S^* = \arg\max_{|S| \leq m} f(S)$, (1)

  $f(S)$: a set function,
  
  $f(S) = \max_{\text{supp}(l) \subset S} C(V(T_l(x)))$

  $S$: support of $l$, indicating the words to be changed
We consider a target attack by selecting from possible candidates

Problem 1 (target attack)

\[ x: \text{input document} \]
\[ T_l: \text{word paraphrasing indexed by } l \]
\[ V: \text{word2vec/bag of word embedding} \]

\[ V(T_l(x)) \]
A General Formulation

- We consider a target attack by selecting from possible candidates

Problem 1 (target attack)

- $x$: input document
- $T_l$: word paraphrasing indexed by $l$
- $V$: word2vec/bag of word embedding
- $C$: classifier that outputs target label’s probability

$$C(V(T_l(x)))$$
We consider a target attack by selecting from possible candidates

<table>
<thead>
<tr>
<th>Problem 1 (target attack)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x</strong>: input document</td>
</tr>
<tr>
<td><strong>T_l</strong>: word paraphrasing indexed by l</td>
</tr>
<tr>
<td><strong>V</strong>: word2vec/bag of word embedding</td>
</tr>
<tr>
<td><strong>C</strong>: classifier that outputs target label’s probability</td>
</tr>
</tbody>
</table>

Find the best transformation labeled by l, with at most m word replacements

\[
\mathbf{l}^* = \arg \max_{\mathbf{l} \in [k]^n, \|\mathbf{l}\|_0 \leq m} C(V(T_l(x))).
\]
We consider a target attack by selecting from possible candidates.

**Problem 1 (target attack)**

- **x**: input document
- **T_l**: word paraphrasing indexed by **l**
- **V**: word2vec/bag of word embedding
- **C**: classifier that outputs target label’s probability

Find the best transformation labeled by **l**, with at most $m$ word replacements:

$$
l^* = \arg\max_{l \in [k]^n, \|l\|_0 \leq m} C(V(T_l(x))).
$$

Or equivalently

$$S^* = \arg\max_{|S| \leq m} f(S), \quad (1)
$$

where $f$: a set function, $f(S) = \max_{\text{supp}(l) \subset S} C(V(T_l(x)))$

**S**: support of **l**, indicating the words to be changed.
Outline

1 Introduction to Adversarial Examples

2 General Framework
   • Mathematical formulation
   • Theoretical Findings

3 Our methods and Experiments
Problem is computationally intractable:

**Lemma 1**

For a general classifier $C$, problem 1 is NP-hard. Even for a convex $C$, problem 1 can be polynomially reduced to subset sum and hence is NP-hard.
Theoretical support for greedy methods

**Fact: Submodular Optimization**

The problem of maximizing a monotone submodular function subject to a cardinality constraint admits a $1 - \frac{1}{e}$ approximation with greedy method.

Do some non-trivial neural networks yield submodular functions?

Qi Lei (UT Austin)
Discrete Attacks (SysML)
April 1st, 2019 11 / 22
Theoretical support for greedy methods

Fact: Submodular Optimization

The problem of maximizing a monotone submodular function subject to a cardinality constraint admits a $1 - \frac{1}{e}$ approximation with greedy method.

- Our target function $f(S)$ is monotone non-decreasing
Theoretical support for greedy methods

Fact: Submodular Optimization

The problem of maximizing a monotone submodular function subject to a cardinality constraint admits a $1 - 1/e$ approximation with greedy method.

- Our target function $f(S)$ is monotone non-decreasing
- Do some non-trivial neural networks yield submodular functions?
Neural Networks with submodular property for discrete set of attacks

Simplified W-CNN [1]

Theorem 1

For W-CNN classifier with no softmax layer, no overlaps between each window, and nonnegative weights in the last layer, $f^{\text{WCNN}}(S)$ is submodular.

Neural Networks with submodular property for discrete set of attacks

one-hidden-node recurrent neural network

\[ h_t = \phi(wh_{t-1} + m^\top v_{t-1} + b) \] (2)

Theorem 2

For RNN with \( T \) time steps and single hidden nodes as in (2), if the activation is a non-decreasing concave function, then \( f^{\text{RNN}}(S) \) is submodular.
Outline

1 Introduction to Adversarial Examples

2 General Framework
   - Mathematical formulation
   - Theoretical Findings

3 Our methods and Experiments
Methodology: Gradient-guided Greedy Method

Intuition: one replacement a time, $\Rightarrow$ greedy method is slow
With the gradient information, we

- pick up $M$ most important words to replace, (e.g. \{like, eat, cafe\})
- greedy search over the replacements for these $M$ words

Replace $M$ words at a time.
Comparisons with prior work

<table>
<thead>
<tr>
<th>Method</th>
<th>Fake News Detection</th>
<th>Spam Filtering</th>
<th>Yelp Review Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ASR: 86.9%</td>
<td>ASR: 29.3%</td>
<td>ASR: 61.2%</td>
</tr>
<tr>
<td>objective-guided greedy</td>
<td>23.4%</td>
<td>3.6%</td>
<td>14.9%</td>
</tr>
<tr>
<td>gradient</td>
<td>90.1%</td>
<td>45.9%</td>
<td>88.1%</td>
</tr>
<tr>
<td>ours</td>
<td>0.19</td>
<td>0.10</td>
<td>0.04</td>
</tr>
<tr>
<td>time (s)</td>
<td>1.20</td>
<td>0.33</td>
<td>0.07</td>
</tr>
</tbody>
</table>


Comparisons with prior work

**Table:** Comparisons with [1] and [2], on WCNN classifier, with up to 20% word replacements. (ASR denotes attack success rate)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake News Detection</td>
<td>ASR: 86.9%</td>
<td>23.4%</td>
<td>90.1%</td>
</tr>
<tr>
<td></td>
<td>time (s): 1.20</td>
<td>0.21</td>
<td>0.19</td>
</tr>
</tbody>
</table>

### Comparisons with prior work

**Table:** Comparisons with [1] and [2], on WCNN classifier, with up to 20% word replacements. (ASR denotes attack success rate)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake News Detection</td>
<td>ASR: 86.9%</td>
<td>23.4%</td>
<td>90.1%</td>
</tr>
<tr>
<td></td>
<td>time (s): 1.20</td>
<td>0.21</td>
<td>0.19</td>
</tr>
<tr>
<td>Spam Filtering</td>
<td>ASR: 29.3%</td>
<td>3.6%</td>
<td>45.9%</td>
</tr>
<tr>
<td></td>
<td>time (s): 0.33</td>
<td>0.08</td>
<td>0.10</td>
</tr>
</tbody>
</table>


Comparisons with prior work

Table: Comparisons with [1] and [2], on WCNN classifier, with up to 20% word replacements. (ASR denotes attack success rate)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake News Detection</td>
<td>ASR: 86.9%</td>
<td>23.4%</td>
<td>90.1%</td>
</tr>
<tr>
<td></td>
<td>time (s): 1.20</td>
<td>0.21</td>
<td>0.19</td>
</tr>
<tr>
<td>Spam Filtering</td>
<td>ASR: 29.3%</td>
<td>3.6%</td>
<td>45.9%</td>
</tr>
<tr>
<td></td>
<td>time (s): 0.33</td>
<td>0.08</td>
<td>0.10</td>
</tr>
<tr>
<td>Yelp Review Evaluation</td>
<td>ASR: 61.2%</td>
<td>14.9%</td>
<td>88.1%</td>
</tr>
<tr>
<td></td>
<td>time (s): 0.07</td>
<td>0.06</td>
<td>0.04</td>
</tr>
</tbody>
</table>


5 people randomly evaluate 60 texts for each task.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>News</th>
<th>Trec07p</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>70.0%</td>
<td>80.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Adversarial</td>
<td>50.0%</td>
<td>80.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Table: Classification Accuracy.
5 people randomly evaluate 60 texts for each task.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>News</th>
<th>Trec07p</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>70.0%</td>
<td>80.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Adversarial</td>
<td>50.0%</td>
<td>80.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

**Table: Classification Accuracy.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>News</th>
<th>Trec07p</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>3.06 ± 0.67</td>
<td>3.23 ± 0.31</td>
<td>1.93 ± 0.55</td>
</tr>
<tr>
<td>Adversarial</td>
<td>3.13 ± 0.50</td>
<td>3.10 ± 0.40</td>
<td>2.10 ± 1.05</td>
</tr>
</tbody>
</table>

**Table: Quality of the text: On a scale of 1-5, how likely the text is human written.**
Conclusions

- Theoretical part:
  - NP-hardness
  - Explore submodularity for some neural networks

Experimental part:

- Practical method: gradient-guided greedy method
- We use sentence paraphrasing to expand the space of attacks
- Experiments verified on three different tasks

Human Evaluation

- Adversarial training
Conclusions

Theoretical part:
- NP-hardness
- Explore submodularity for some neural networks

Experimental part:
- Practical method: gradient-guided greedy method
- ★ We use sentence paraphrasing to expand the space of attacks
- Experiments verified on three different tasks
- Human Evaluation
- ★ Adversarial training
Thank you!
Methodology: Joint sentence and word paraphrasing attack

- Pick up sentence candidate set from semantic similarity.
- Greedily conduct sentence level paraphrasing attacks.

I’ve always run jigdo-lite against my own mirror. It provides two things: 1) Proves I can you are able to build the ISOs from what I have mirrored locally. 2) Doesn’t waste additional bandwidth. · · ·

- Pick up word candidate set from semantic and syntactic similarity.
- Greedily conduct word level paraphrasing attacks

I’ve always run jigdo-lite against my own mirror. It provides offers two things: 1) Proves I can you are able to build the ISOs from what I have mirrored locally. 2) Doesn’t waste additional bandwidth. As long as the checksums match what is provided from the official ISO image masters site, I don’t see what the difference would be. Anyone else do this? ^_^
Experiment: Joint sentence and word paraphrasing attack

Table: Experiments on Word-level CNN. [1] allows 50% word replacement while we only allow 20% word paraphrasing and 20% sentence paraphrasing.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Origin</th>
<th>ADV (ours)</th>
<th>ADV [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>News</td>
<td>93.1%</td>
<td>6.9%</td>
<td>71.0%</td>
</tr>
<tr>
<td>Trec07p</td>
<td>99.1%</td>
<td>50.5%</td>
<td>64.5%</td>
</tr>
<tr>
<td>Yelp</td>
<td>93.6%</td>
<td>7.9%</td>
<td>39.0%</td>
</tr>
</tbody>
</table>

## Experiments: Adversarial Training

**Table:** Performance of adversarial training.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>News</th>
<th>Trec07p</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test (before)</td>
<td>93.1%</td>
<td>99.1%</td>
<td>93.6%</td>
</tr>
<tr>
<td>Test (after)</td>
<td>93.8%</td>
<td>99.2%</td>
<td>94.9%</td>
</tr>
<tr>
<td>ADV (before)</td>
<td>35.4%</td>
<td>48.6%</td>
<td>23.1%</td>
</tr>
<tr>
<td>ADV (after)</td>
<td>40.0%</td>
<td>54.2%</td>
<td>44.4%</td>
</tr>
</tbody>
</table>