Exploring Adversarial Learning on Neural Network Models for Text Classification

Isaac Caswell
Stanford University
Dept. of Computer Science

Allen Nie
Stanford University
Symbolic Systems Program

Onkur Sen
Stanford University
Dept. of Computer Science

{icaswell, anie, onkursen}@stanford.edu

1 Introduction

Deep neural networks have recently achieved state-of-the-art performance in many tasks, most notably image recognition. However, many models are also notoriously difficult to train. Over the years, people have used different data augmentation techniques to artificially simulate possible data, and training techniques such as pooling and dropout are used to help with this particular issue. Adversarial learning is a reaction to the observation that many models are susceptible to images perturbed by adversarial noise. Such images can fool a trained neural net into predicting a wrong label with high confidence (Nguyen et al., 2014).

Adversarial learning combats this problem. It bears similarity to the aforementioned techniques, but instead of simulating what could realistically happen, it focuses on helping the model to explore the landscape near the decision boundary, leading to a more accurate model, as demonstrated by (Miyato et al., 2015). Adversarial learning can be treated as an a priori optimization technique that can be generalized on all neural network models by directly modifying the objective function. Training may also be done by generating external adversarial examples directly, demonstrated in (Goodfellow et al., 2014).

2 Related Work

2.1 Image Processing

Adversarial learning first gained popularity in the realm of image processing, where the heavy use of neural networks improved performance across different corpora. Most of them are established in Goodfellow et al. (2014). In image processing, where automatically generating variants of data points is quite common, Goodfellow made the distinction between adversarial learning and common data augmentation techniques such as transformations:adversarial examples are not naturally-occurring examples that need to realistically happen in the world.

Adversarial examples add noise to the training set that is smaller than the precision of the initial training image. Such noise is undetectable by human eyes and also cannot be obtained by the image capturing equipment. Goodfellow argues that training with such perturbations will help neural networks truly capture concepts and gain accuracy beyond perceived data. Jin et al. (2015) pointed out the adversarial examples generated in Goodfellow's paper are sampled near a decision boundary, thus making network models particularly vulnerable. Miyato et al. (2015) further demonstrated the theoretical implication that adversarial training examples will help models to better find non-linear relationships in data.

2.2 Natural Language Processing

Adversarial learning with neural networks has performed well in image processing. Furthermore, since neural network architectures are commonly used for many tasks in natural language processing, we believe this technique would improve such models as well. Such work has yet to be done. However, natural language deals with discrete outcomes whereas images are easily represented as continuous data points. An adversarial image is much easier to intuitively visualize than an adversarial sentence.

We explore the effectiveness of adversarial learning on recurrent neural network and long short-term memory model, which would be very different from previous models.

3 Adversarial Learning

Intuitively, it does not makes sense that a classifier should respond to a very small perturbation to a very small change in an example that is assigned some label with high confidence. This is especially true for images, where the precision of

the features is limited, and certainly also true for word/sentence vectors, which are inherently approximate, as they are learned from data. Symbolically, a classifier shouldn't respond differently to $\tilde{x}$ than it does to x, if each element in the perturbation η is sufficiently small:

$$\tilde{x} = x + \eta \wedge ||\eta||_\infty < \epsilon \Rightarrow \text{class}(\tilde{x}) = \text{class}(x),$$

where ϵ may be the precision of the sensor (in the case of an image), or the maximum magnitude of the last update to a word vector (NLP), or an arbitrary small number (in our case).

Consider, however, what actually happens to the activation of such a perturbed example, for some weight vector w:

$$w^T\tilde{x} = w^T x + w^T\eta$$

Notice that a well-chosen η (e.g. $\eta = w$) may cause the activation to grow linearly with the size of the weight vector! For our RNN, one sees that the perturbation will propagate:

$$\begin{aligned}\tilde{h}_t &= \sigma(W\tilde{x}_t + U\tilde{h}_{t-1}) \\ &= \sigma(W\tilde{x}_t + W\eta_t + U\tilde{h}_{t-1}) \\ &= \sigma(W\tilde{x}_t + W\tilde{\eta}_t + U(\sigma(W\tilde{x}_{t-1} + W\eta_{t-1} + \\ &\quad U\sigma(W\tilde{x}_{t-2} + W\eta_{t-2}\ldots)))\end{aligned}$$

To determine the adversarial perturbation η, we use the "fast gradient clipping method" proposed by Goodfellow et al. (2014). Let s be a training example, W be our word embedding matrix, and $L(\theta, s, y)$ be the loss function for s given its true label y. Adapted to our LSTM architecture, the perturbation is:

$$\eta = \epsilon \ \text{sign}(\nabla_W L(\theta, s, y))$$

Rather than actually creating adversarial examples and training on them, we simulate this training by modifying the objective function. Let α be the factor by which we wish to weight the adversarial versions of the training examples. The modified objective becomes:

$$\begin{aligned}\tilde{L}(\theta, s, y) &= \alpha\, L(\theta, s, y) + \\ &\quad (1-\alpha)L(\theta, s + \epsilon \ \text{sign}(\nabla_W L(\theta, s, y))\end{aligned}$$

For each word in the training data, we simulate the creation of an adversarial example for that sentence, and have the classifier balance the importance of correctly classifying the untainted example and the adversarial example by α.

Implementation note: In order to construct this cost function, we double the size of the computation, creating an entirely separate computational graph for the first and second terms in $\tilde{L}(\theta, s, y)$, connected at the bottom by the word embedding matrix and at the top by the cost function. We forward prop both from the word embedding matrix, but only backprop through the graph corresponding to the first term.

4 Task Definition

Among a variety of text classification tasks, we settled on sentiment analysis to avoid focusing on relationships of single words, thus giving us more freedom to generate appropriate adversarial examples. Sentiment analysis is also a heavily explored field with many neural network architectures. We experimented with recurrent neural networks (RNNs), long-short term memory (LSTM) models, and CNNs.

4.1 Data

We chose the IMDB dataset (Maas et al., 2011) which contains 50,000 sentences split equally into training and testing sets. Each training instance contains an entire review written by one individual. No post-processing is used; the dataset is used as is. We also load pre-trained word embeddings from Google Word2Vec's Google News vectors of 100 billion words (Mikolov et al., 2013).

5 Methodology

5.1 Recurrent Neural Network

We implemented an LSTM-RNN with mean pooling and softmax layers to map to output labels using the symbolic mathematical expression engine Theano (Bastien et al., 2012). Our model is GPU compatible although alas, the only GPUs availbale to use did not cooperate.

5.1.1 Hyperparameters

In the ideal world, we would run parallel random searches for hyperparameters on some cluster with GPU. However, as this was beyond our research budget (And the Farmshare machines didn't cooperate), we finally copped out and picked somewhat arbitrary hyperparameters (excepting `alpha` and `eps`, which relate to the adversarial objective), based on the Theano LSTM example (Pierre Luc Carrier, 2015). We did a manual search over

alpha and epsilon. The important hyperparameters for our model are:

1. wemb_init: How to initialize the word embeddings. They are either initialized with word2vec (300D) or with each value drawn uniformly from [0, 0.01] because of constraints mentioned above.

2. wdim: Word embedding dimension. For word2vec vectors, this must be 300; otherwise, we used 128.

3. hdim: The hidden dimension of the LSTM. We default to 128.

4. reg: The regularization on the weight matrix. We default to 0.0.

5. clip: The magnitude to which to clip the gradients. Gradient clipping is elementwise. In a boldly arbitrary move, we default to 5.

6. adv: Whether to use the adversarial cost function.

7. alpha: How many adversarial examples to simulate, as a fraction of the training corpus.

8. eps: A constant proportional to the magnitude of the perturbation.

5.2 Travails

As alluded to above, we built a GPU-compatible system with 14 tunable parameters, and furthermore implemented a script to make random hyperparameter sweeps in parallel between adversarial and non-adversarial models. However, we were unable to access a machine that was capable of running this hyperparameter search. Most of our efforts focused on trying to work with the rye machines; those morbidly interested in our fate here may enquire in person. Even the corn machines tended to randomly cut off a few hours in.

As a result we ran all experiments locally with friendly hyperparameter settings. Furthermore, due to the constraint on the computing resources, we switched from a truncation to a filtering approach with our data, meaning we discarded reviews longer than 100 words, rather than truncating them, leading to utilizing only 10% of our data.

6 Results and Discussion

Results for a variety of parameters for the adversarial objective are summarized in Table 1, where $\epsilon = 0$ corresponds to the non-adversarial case. Since we were not able to tune hyperparameters, our validation and test set may be considered equivalent, and their average is presented in the third row of the table.We discover no clear evidence that adversarial learning performs better than the alternative. Our conclusion, however, is that these results are not sufficient to discredit our approach. There are several reasons why this is the case, all stemming from our inability to sufficiently tune the net, which stems from our lack of computing power. See Section 5.2, “Travails”.

The most important thing to note is that the error is high in all cases. Even had we demonstrated that adversarial learning performed better in this case, (say, an improvement of 20% to 18% average error), one would have had trouble giving credence to this result. A support vector machine could easily outperform either of these models.

Furthermore, the results are noisy. Although the best adversarial model does outperform the non-adversarial version, this could be due to chance.

In this respect our results are disappointing. However, one can still do some analysis of the adversarial examples we spawned, and the coming sections devote themselves to this. Furthermore the architecture is in place to do more powerful analyses, when the computing resources present themselves to us.

7 Adversarial Visualization

We would like to visualize the adversarial examples produced by our system to interpret what’s going on in a human-interpretable way. There is, however, a hitch to this: when adversarial learning is applied to vision, it is easy to interpret what an adversarial example means. A perturbed an image is another image. Natural language, however, proves to be more difficult. A perturbed embedded word is some vector in the word embedding space without any obvious correspondence to an English word. This section details our approaches towards dealing with this problem.

7.1 Nearest Neighbor Visualization

The simplest approach is to find the nearest neighbor via cosine distance in word embedding space

Table 1: Test and development set error given different values of ϵ for $\alpha = 0.5$.

ϵ	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**	**15.0**
dev	0.14	0.00	0.15	0.12	0.16	0.14	0.14	0.16	0.14	0.16	0.14	0.13
test	0.19	0.00	0.23	0.24	0.19	0.18	0.20	0.20	0.22	0.18	0.21	0.17
average	0.17	0.00	0.19	0.18	0.18	0.16	0.17	0.18	0.18	0.17	0.18	0.15

Table 2: Progressively-improving visualization of a positive review.

original	love it , love it , love it ! this is another absolutely superb performance from the divine miss m. from the beginning to the end , this is one big treat ! don 't rent,buy it now ! [-...]
KNN	**love** conquistadors conquistadors **love** conquistadors conquistadors **love** conquistadors conquistadors conquistadors conquistadors conquistadors conquistadors drama.the daftness **from the** conquistadors **miss m. from the** conquistadors **to the** conquistadors conquistadors conquistadors conquistadors pensacolians conquistadors **treat** conquistadors daftness 't **rent** conquistadors conquistadors conquistadors **now** conquistadors [conquistadors...]
KNN top 5000	love quit , love quit , love quit passing quit quit quit **absolutely** quit quit from the quit miss intentions from the quit to the quit , quit quit quit quit treat passing quit 't rent quit quit quit now passing [quit...]
KNN top 5000 + scaled perturbation	love **it** , love **it** , love **it !** quit quit quit absolutely **?** quit from the **?** miss intentions from the quit to the **easily** , quit quit **one big** treat **! don** 't rent **UNK** quit **it** now **!** [quit...]
KNN top 5000 + scaled perturbation + bigram smoothing	love it , love it , love it **, this is another** absolutely **superb performance** from the **audience** miss intentions from the quit to the **end , this is** one , treat ! don 't rent UNK quit it now , [quit...]

to each perturbed example and reconstruct a sentence out of these words. Representative results of this approach are shown in the first row of Table 2. We see that these visualizations are quite poor—or at least, it's hard to make any sort of sense of them.

There are several reasons why this approach may yield such odd results:

1. Rare words with mostly random distributional representations pollute the word space, and many perturbations end up being closest to them.

2. Word-by-word translation does not take context into account. Furthermore, in high-dimensional spaces, the top several neighbors often have very similar distances. For one run, for example, the top ten neighbors were all at distance of around 4.3.

3. The direction of the perturbation may be more important in terms of a human interpretation of an adversarial example than the example itself.

7.2 Removing Uncommon Words

To address (1) above, we restricted ourselves to the 5000 most common words from the training data. The second row of Table 2 gives an example of how this change affected the decoding of our adversarial examples. As one sees, this was mildly helpful, and especially when the nearest neighbor was the sentence padding token.

7.3 Scaling the Perturbation

To approach (2), we decreased the magnitude of the perturbation until the reconstructed sentence began to look more similar to the original sentence. We tended to decrease the magnitude of the perturbation until it was weighted by about $\epsilon = 0.005$. This is significantly lower than the ϵ value we used for training, which was on the order of $\epsilon = 0.5$.

7.4 Bigram Weighted Interpretation

To deal with (3), we trained a bigram model on the IMDB training corpus and incorporated

Table 3: A plausible adversarial example.

original sentence	shallow , shallow script ... stilted acting ... the shadows of 1990 UNK mob over the actors ' heads in scenes ... worth watching because kate UNK plays the most selfish mother in tv movie history and it 's all before ben stretch got his teeth UNK .
adversarial sentence	**unbelievable** , **unbelievable** script ... stilted acting ... the shadows of **nazis** UNK **intelligent** over the actors ' heads in scenes ... worth watching because **remarkable** UNK plays the most **imagination** mother in tv movie history and it 's all before **attached** **remarkable** got his **summer** UNK .

it into our decoder. As seen from Table 2, the bigram-weighted interpretation model can help make the adversarial sentences more interpretable, often snapping perturbed words back to what they were before, if the perturbation would otherwise be too linguistically infeasible. Often, however, it does more harm than good, and changes mildly-perturbed words to something erroneous.

7.5 Interpretation of Adversarial Examples

Even with our best visualizations, it's hard to tell a good story about our adversarial examples. Table 3 gives a pair of sentence and reconstructed adversarial example into which we can inject some measure of sense. Words that differ between the two are bolded in the adversarial example. The original review is negative, meaning that the adversarial review aims to be classified as positive while still being obviously negative to a human.

The resulting adversarial 'review' is highly ambiguous, and is adversarial in the sense that, to a human, it still seems moderately negative, while containing positive words. It calls the script 'unbelievable', which could be a good thing or a bad thing. The words substituted in tend to be negative words converted to positive ('shallow', 'selfish', $\rightarrow$ 'intelligent', 'remarkable', 'imagination'; 'nazi' probably too in this case as people like to give good reviews movies about the Nazis), but none of the words is placed in a context such that its positive connotation carries to the description as a whole. Of course, this may well be because it is mostly nonsense. A scientist must be careful not to read too far into this.

8 Acknowledgements

Jon Gauthier not only conceived this idea, but also helped us wade through Theano syntax, get the system running on rye, advised us to disconnect the gradient from the adversarial example, and in short taught us everything practical that we never learned as students. Chris Manning advised us in our quest to figure out what we were doing.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. 2015. Robust convolutional neural networks under adversarial noise. *arXiv preprint arXiv:1511.06306*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2015. Distributional smoothing with virtual adversarial training. *stat*, 1050:13.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2014. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint arXiv:1412.1897*.

Kyunghyun Cho Pierre Luc Carrier. 2015. Theano lstm tutorial.