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

Abstract

In this paper, we examine the application of adversarial learning, which first gained recognition in image classification, to natural language processing. In particular, we train a recurrent neural network with long short-term memory by modifying its objective function to simulate training on adversarial examples. We also visualize the perturbed sentence matrices using a nearest-neighbor, augmenting distance with a bigram cost. We discuss various techniques in visualization and offer interpretations of perturbed sentences as well as perturbation as relations between words. Our results do not indicate that adversarial training helped, which fact we attribute to a dearth of computing resources.

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 various 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 network 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).

However, despite its success in image processing, little has been done in the field of natural language processing. We investigate whether adversarial learning works in natural language processing, and examine the generated sentences under the adversarial learning paradigm.

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 hypothesize this technique would improve such models as well. Such work has yet to be done. However, there are two major differences between these two fields. First, natural language deals with discrete outcomes whereas images are easily represented as continuous data points. An adversarial image can be easily visualized, but an adversarial sentence is hardly an intuitive concept, visually or otherwise.

Second, particular neural network models, e.g., convolutional neural networks (CNNs), that are more common in image processing are less common in natural language processing. Many previous explorations are based on either a simple neural network with multiple hidden layers, as in Goodfellow et al. (2014), or a CNN, as in Jin et al. (2015).

In our work, 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:

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

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:

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

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

We considered a variety of text classification tasks such as language modeling, relation extraction,

and sentiment analysis. 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. However, we could not test 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 `epsilon`, 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 different 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. `max_epochs`: Self-explanatory. Our model terminates early if convergence is reached. Defaults to 1000.

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

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

6. `lrate`: The learning rate. This is only used if the optimizer is stochastic gradient descent. We default to 0.01.

7. `optimizer`: The optimization algorithm. May be stochastic gradient descent (`sgd`), `rmsprop`, or `adadelta`. We default to `adadelta`.

8. `maxlen`: The maximum length of sentence to use in the training data. Sentences longer than this are discarded or truncated. We default to 100; this results in train on $\approx$10% of the entire data set.

9. `batch_size`: The batch size for the batch gradient descent.

10. `weight-init`: How to initialize the weight matrix. We default to orthonormal initialization, with pseudo-orthonormal initialization for overdetermined matrices.

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

12. `adv`: Whether to use the adversarial cost function.

13. `alpha`: α from the adversarial objective function, i.e. how many adversarial examples to simulate, as a percentage of the training corpus.

14. `eps`: ϵ from the adversarial objective function, i.e. 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

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

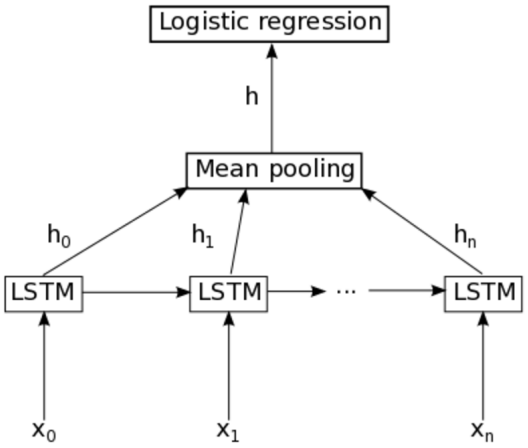

Figure 1: Our LSTM model architecture. (Pierre Luc Carrier, 2015)

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, on our laptops, with friendly hyperparameter settings (i.e. the relatively low word embedding, hidden dimension, and batch size described in the above section.) 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 $\epsilon = 0.0$ corresponds to the non-adversarial case. Since we were not able to tune our hyperparameters, our validation and test set may be considered equivalent, and their average is presented in the third row of the table. As one sees, 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, as they are flawed in several senses. There are several reasons why this is the case, all stemming ultimately 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 off the bat is that the error is relatively 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 quite noisy. Even though the best adversarial model does outperform the non-adversarial version, it is probable that is by chance alone.

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.

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...]

7.1 Aside 1: Why do some "original" sentences look weird?

Occasionally, there are 1. **UNK** tokens and 2. odd words in the original sentences in these tables. The first is because we are looking at sentences from the validation set, which may contain words not in the vocabulary seen in the training set. The second is because of the constriction of vocabulary for visualization described in 7.4.

7.2 Aside 2: A note on the sentence padding token

In order to do batch gradient descent on variable length reviews, we padded the ends of sentences with a symbolic sentence padding token, which we visualize as '-'. However, a curious thing happens with this token in our visualizations. For a given adversarial example, the sentence padding token will have some nearest neighbor in the word embedding space. Strangely, many perturbations of normal words in example sentences go to the same word. As an example, look at row 2 of Table 2. The sentence padding token gets perturbed to be closest to "conquistadors". However, we also see that the majority of the other words in the sentence are also perturbed to be closest to "conquistadors"! Why this happens is not clear to us.

Across all sentences, the sentence padding token tended to be perturbed in the direction of one of about four words, which changed depending on our word-frequency filter. Common examples of padding-neighbors include 'vick', 'recently', 'quit', 'horse', 'elephant', 'starbuck', 'scorpio', 'plausibility' and 'conquistadors'.

7.3 Nearest Neighbor Visualization

The simplest approach is to find the nearest neighbor via cosine distance in word embedding space to each perturbed example and reconstruct a sentence out of these words. Representative results of this approach are shown in the first row of the two demonstration tables, 2 and 3. 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:

Table 3: Progressively-improving visualization of a negative review.

original	the movie was disappointing . the book was powerful . the views and the learning of little tree were,portrayed in the book . the movie just,along and finally,away . still a nice tale for kids . [-...]
knn	the art-related was disappointing prohibits the rawls was coerced prohibits the rawls elephants the prohibits coerced pasteur tree elephants prohibits elephants recently the rawls prohibits the art-related elephants prohibits along elephants finally prohibits elephants prohibits prohibits a elephants tale elephants elephants prohibits [elephants...]
knn top 5000	the recently was disappointing recently the recently was recently recently the **views** recently the — **of** recently tree recently recently recently recently the recently recently the recently recently recently along recently finally recently recently recently recently a recently tale recently recently recently [recently...]
knn top 5000 + scaled perturbation	the **movie** was disappointing **.** the recently was **powerful .** the views **machine** the **learning** of **little** tree **machine UNK portrayed in** the recently **.** the movie just **UNK** along machine finally **UNK** recently . recently a **nice** tale **for kids .** [recently...]
knn top 5000 + scaled perturbation + bigram smoothing	the movie was disappointing . the **book** was **supposed** . the views machine the **climax** of little tree machine UNK portrayed in the **book** . the movie **is how** along **and** finally UNK recently . **still** a nice tale for kids . [recently...]

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.

To deal with these issues, we made several fine-tunings to our model.

7.4 Removing Uncommon Words

To address (1) above, we restricted ourselves to the 5000 most common words from the training data. The second row of Tables 2 and 3 give 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.

7.5 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 $\epsilon = 0.5$.

Amusingly, however, this is very similar to the differential in adversarial ϵ presented in Goodfellow et al. (2014), who train with $\epsilon = 0.5$, but present the now-familiar panda-becomes-gibbon with $\epsilon = 0.007$. The moral, one supposes, is that the interpretation of this modification in the objective function is more of a heuristic story to explain in general terms why it works better—the actual algorithm takes this principle and exaggerates it until it's unrecognizable to humans.

Table 4: 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 .

7.6 Bigram Weighted Interpretation

To deal with (3), we trained a bigram model on the IMDB training corpus and incorporated it into our decoder; see Algorithm 1. As seen from Tables 2 and 3, 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. As an example,
this movie will always be a broadway and music classic
gets changed to
*this movie will always be a **copy** and **later** classic*

Choose first word in the sentence w_0 as usual;
Add w_0 as the first word in `decoded_sentence`;
for *each remaining word* w_i **do**
  Find 20 closest neighbors to w_i in word embedding space (cosine distance);
  Compute scaled log bigram cost $\tilde{b}(word_i) :=$ $\eta * \log(\text{bigram\_frq}(word_{i-1}, ngh_i))$ for each neighbor ngh_i;
  Augment cosine distance to each neighbor by its bigram cost;
  Add the nearest word under this scheme to `decoded_sentence`;
end
Return `decoded_sentence`;

Algorithm 1: Algorithm for training a bigram model on the IMDB training corpus.

7.7 Interpretation of Adversarial Examples

Even with our best visualizations, it's hard to tell a good story about our adversarial examples. What does it mean that the sentence padding token is closest to 'conquistadors'—how does that allow the example to conquer our model?

Table 4 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 example is a negative review, meaning that the adversarial review aims to be classified as a positive review 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 Understanding perturbations themselves as relations between words

One of the most entertaining parts of the word-vector story is how relations between words can be exposed through simple vector arithmetic. In the classic example by Mikolov et al. (2013), it

Table 5: Nearest neighbors among 1000 most common words to the adversarial perturbation of a sentence. Each neighbor is the difference between two word vectors.

Sentence	**Closest Relations**
i wouldn 't rent this one even on dollar rental night	0. then-can 1. direction-everyone 2. <-best 3. fan-'ll 4. viewing-doesn
adrian UNK is excellent is this film . he makes a fascinating woman	0. shown-loved 1. shown-definitely 2. john-before 3. shown-live 4. such-before
this UNK horror film is probably destined to become a cult classic . much much better than 90 % of the scream,out there ! i even hope they come up with a sequel !	0. found-before 1. john-hollywood 2. shown-hollywood 3. rest-hollywood 4. such-hollywood
surprisingly well made little movie . short in length at about 90 minutes . for a low budget movie , very well made . plot is slow to unravel . cast is excellent especially elizabeth van UNK as the girlfriend with UNK's syndrome .	0. wooden-lead 1. 3-thought 2. wild-seeing 3. wooden-understand 4. wooden-worse

was demonstrated that:

$$v_{king} - v_{man} \approx v_{queen} - v_{woman}$$

where v_w means the word vector corresponding to word w. Using this as motivation, we posed the following question: using arithmetic on our learned word vectors, what can we say about the relation between a word and its adversarially perturbed counterpart? Can we compare it to the relation between two known words? Mathematically, we framed the question as: if the relationship between a word and its perturbed counterpart is similar to the relation between some words x and z, then:

$$v_w - v_{w_{adv}} \approx v_x - v_z$$

and from this arises the formulation:

$$r_w \approx min_{x,z \in V} ||(v_w - v_{w_{adv}}) - (v_x - v_z)||_2,$$

Where r_w is a variable representing the relationship between a word w and its perturbed counterpart. Since $(v_w - v_{w_{adv}})$ is the adversarial perturbation, we note that this expression becomes:

$$r_w \approx min_{x,z \in V} ||\epsilon \text{sign}(\nabla_W L(\theta, s, y)) - (v_x - v_z)||_2,$$

where W is the word embedding matrix, and s is a sentence. Note that this expression is constant for each word in a sentence.

Some example results for this method can be seen in Table 5. Since computing this relation exactly is an $O(|V|^2)$ operation, we looked only among words in the top 1000 most common words in our vocabulary. One notes with regret that nothing about these relations obviously makes sense.

9 Exploring perturbations by adding them to common words

In a further attempt to bring some sense to these relations, we thought to do some manual exploration. Namely, since the adversarial perturbation represents a relation between word vectors, then by adding it to one word we may find another word which bears the same relation to it as does the word to its adversarial counterpart. Specifically, if we wanted to find the word x which bore the same relation to "horse" as a sentence does to its adversarial variant, we can discover the closest

Example word	Example positive review	Example negative review
good	good, its, his	it, in, '
bad	worst, boring, played	script, no, too
shallow	importance, silly, summer	boy, expect, photographed
fine	did, gore, created	it, before, movie
excellent	world, amazing, perfect	story, and, before
superb	favorite, gore, though	before, it, loved
outstanding	gore, favorite, did	it, before, movie
wonderful	shows, world, perfect	story, before, loved
powerful	favorite, charming, others	story, before, it
disappointing	became, truly, truth	script, no, getting
horse	peace, sword, doomed	story, has, he

Table 6: Top 3 nearest neighbors (among 1000 most common words) to artificially perturbed sentiment words.

Positive review	Negative review
worst (72%)	script (78%)
boring (72%)	no (69%)
actor (15%)	too (30%)
waste (15%)	bad (24%)
awful (15%)	stupid (21%)

Table 7: A closer look at the data from Table 6: the most common nearest neighbors to the perturbation of the word vector for the word "bad",for perturbations from positive and negative reviews.

approximation $\tilde{x}$ to that word so:

$$\begin{aligned} & v_w - v_{w_{adv}} \approx v_x - v_{horse} \\ \Rightarrow \quad & \text{sign}(\nabla_W L(\theta, s, y)) \approx v_x - v_{horse} \\ \Rightarrow \quad & v_x \approx v_{horse} + \text{sign}(\nabla_W L(\theta, s, y)) \\ \Rightarrow \quad & \tilde{x} = \arg\min_{z \in V} \\ & ||z - (v_{horse} + \text{sign}(\nabla_W L(\theta, s, y))||_2 \end{aligned}$$

In other words, the nearest neighbor to the vector for "horse" plus the adversarial perturbation.

In our wildest dreams, the adversarial shove might have captured something like the notion of oppositeness, or perhaps have been a slight pressure towards positive sentiment for negative reviews, and vice-versa.

Table 6 demonstrates the results from this foray on a variety of sentiment-related words. As one fears, there is no obvious trend. One interesting thing, however, is that *bad* gets strengthened to *worst* for the positive review. This is mildly in line with what one had hoped: for a positive sentence, we hope that the adversarial push makes words more negative (though hopefully in a non-obvious way). Investigating this further, we noticed that certain words tended to be shoved near different neighbors depending on whether the training sentence from which the adversarial push was generated was positive or negative.

As a case study, we looked at the word *bad*. For 100 sentences in the validation set, we recorded the top three neighbors to its perturbed form. Table 8 reports the top five neighbors over these hundred sentences. The percentage listed beside each word indicates what percentage of the sentences had that word among their top 3 neighbors. One sees that 72% of positive reviews contained *worst* among the neighbors of the perturbed *bad*; and in addition, the other nearest neighbors were highly negative (excepting "actor"). The neighbors to *bad* perturbed under a negative sentence, on the other hand, are much more benign, and often aren't a big enough perturbation to get closer to any other word than *bad*. This lends credence to the idea that the adversarial shove, whatever other subtlety it might encode, serves to intensify the opposite sentiment of the review whence it was generated.

The above analysis is interesting, and probably has some validity to it. However, it is highly anecdotal and on a very small sample, so a person doesn't feel massively confident putting to much

faith in it. One would love to revisit this sort of analysis in a more principled, quantitative way.

10 Conclusions and Future Work

Our experiments do not demonstrate a clear advantage to using the proposed adversarial objective for sentiment classification, given our architecture. Our analysis of adversarial examples is similarly weak. We postulate, however, that both of these are largely results of the difficulties described in section 5.2, and are willing to pursue the matter further as soon as we have the resources to do so more effectively. We have a strong framework for testing and analysis, which will allow us to do much more speedy analysis as soon as we have something better to analyze.

The first next step, of course, is to run the hyperparameter search for both models (adversarial and non-adversarial), and perform the various analyses already detailed in this paper on the best models discovered. Following that, there are several interesting avenues to explore. Using a different architecture (e.g. CNN) or working on a different task (e.g. paraphrase detection) would certainly be useful. If the adversarial visualizations were effective, comparing adversarial examples from different models for the same sentences might be especially interesting.

Another approach is to explore different ways of encoding adversariality. One such method is to create adversarial examples directly, rather than simulating them with the modified objective function. As an example, for a training example x with label y, one could create an adversarial example by finding the closest vector (either sentence or word) that would be assigned different label under our model:

$$x_{adv} = \arg\min_{x'} \left(L(x', y, m) + \frac{\lambda}{2} ||x' - x||_2^2 \right)$$

(This formula borrowed from Andrej Karpathy's CS231 assignment 3).

11 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, aside from making this entire experience possible as the professor of the class, also advised us in our quest to figure out what we were doing.

And of course, thank you to the swarm of TAs who either gave useful comments or were just nice to be around.

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.