Semi-Supervised Recursive Autoencoders

Adrian Guthals
aguthals@cs.ucsd.edu

David Larson
dplarson@ucsd.edu

Abstract

We evaluate semi-supervised recursive autoencoders (RAE) as a method for predicting the sentiment of sentences. Using random word initialization, we are able to predict the sentiment of a movie review dataset with a 74.5% accuracy, which is only 3.3% less than the 76.8% accuracy reported in the 2011 paper "Semi-Supervised Recursive Autoencoders" by Soch et al.

1 Introduction

Socher et al. presented a semi-supervised method for learning meanings of sentences using recursive autoencoders [1].

The lecture notes state blah [2].

Mention: neural networks, sentence meaning/sentiment

2 Recursive Autoencoders

RAE, neural networks, backpropogation, error functions, greedy algorithm, calculating derivatives numerically using finite center-difference

2.1 Error Function

$$E_1(k) = \tag{1}$$

$$E_2(k) = \tag{2}$$

2.2 Binary Tree Construction

2.3 Backpropogation

Backpropogation is an efficient method for computing the derivatives required for training a neural network. Given

2.4 Goal of Training

2.5 Gradient Verification

It is important to verify the accuracy of the gradients calculated using backpropogation. For this study we have chosen to verify the accuracy of backpropogation by comparing against gradients calculated numerically using finite central-differences:

$$\frac{\partial J}{\partial \theta} = \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon} + O(\epsilon^2) \tag{3}$$

where ϵ is the grid spacing.

A downside to using numerical derivatives to verify backpropogation is time complexity. If $W \in R^{2d \times d}$ then the time complexity of computing derivatives is $O(d^2)$ using backpropgation and $O(d^4)$ using finite central-differences, which is not feasible for real-world applications. One option for reducing the time complexity of checking the derivatives is to check a subset of the derivatives, chosen randomly, and assume those selected derivatives are representative of the entire set.

3 Experiments

3.1 Datasets

We use the same movie reviews dataset as in [1], which consists of 10662 snippets from reviews posted to the Rotten Tomatoes website[1]. Each snippet is roughly equivalent to a single sentence and includes a positive/negative label, with the entire dataset containing 5331 positive and 5331 negative labelled snippets. For all experiments we randomly selected $\sim 90\%$ of the original dataset as a training set, with the remaining $\sim 10\%$ used as a testing set. In splitting the dataset we have taken care to prevent any snippets from existing in both sets, so as to not contaminate the results.

3.2 Optimization

We use limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), a well-known quasi-Newton optimization method, to learn the parameters θ. Specifically we use the Matlab-based L-BFGS function from the minFunc toolbox [3].

Convergence: error less than 10^{-6} (as stated in the project description)

Regularization: ?

3.3 Experiment 1: RAE

The full method (RAE)

d = 20: prediction accuracy = 74.5%

10-fold cross validation

same hyperparameters as Socher et al.

3.4 Results

3.4.1 Most Positive and Negative

Table 1–2 shows the words and phrases predicted to be the most positive and negative. The only result that stands out as possibly an error is the word "flaws" being predicted as positive rather than negative. Although the word "flaws" may be normally associated with a negative meaning, it could be associated with a positive meaning due its usage in a phrase, e.g., "despite its flaws".

3.4.2 Similar Meanings

Comparing words and phrases with the most similar meanings is another intiutive method for evaluating the trained model. The similarity between a pair of words or phrases, with meaning vectors x and y, can be quantified using cosine similarity:

$$\text{cosine similarity}(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \tag{4}$$

which is bounded between -1 (opposite in meaning) and $+1$ (same in meaning). Applying this metric to Table 1–2 we find the following pairs to be most similar:

[1] `http://www.rottentomatoes.com`

Table 1: Words predicted to be the most positive and negative.

Ranking	Positive	Negative
1	beautiful	fails
2	brilliant	boring
3	thoughtful	neither
4	triump	bad
5	flaws	flat
6	beautifully	predictable
7	success	bore
8	spectacular	poorly
9	enjoyable	suffers
10	wonderful	unnecessary

Table 2: Phrases (length 2) predicted to be the most positive and negative.

Ranking	Positive	Negative
1	moving and	lack of
2	an enjoyable	boring .
3	and beautifully	how bad
4	a moving	the dullest
5	a triumph	flat ,
6	a beautiful	of bad
7	the best	it fails
8	and powerful	it isn't
9	its flaws	and predictable
10	a wonderful	a boring

1. Positive words: "bad" and "boring"
2. Negative words: "wonderful" and "enjoyable"
3. Positive phrases (length 2): "how bad" and "of bad"
4. Negative phrases (length 2): "moving and" and "and powerful"

3.4.3 Tree Structure of Interesting Sentences

Visually inspecting the tree structure of sentences offers on insight on the strengths and weaknesses of the greedy algorithm. Figure 1 shows the structure of a positive sentence, which the greedy algorithm correctly determined. Meanwhile Figure 2 shows the structure of a negative sentence which the greedy algorithm performed poorly. Specifically, the structure should have connected "right now" with "go , girls" instead of "the reality drain .". Incorrectly determine tree structures such shown in Figure 2 likely caused problems for the RAE model and decreased it's prediction accuracy.

4 Conclusion

Final remarks

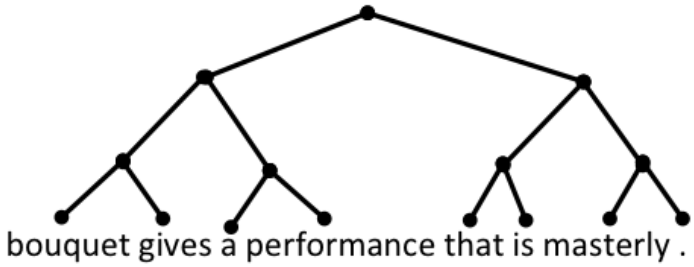

Figure 1: The tree structured determined by the greedy algorithm of a positive sentence.

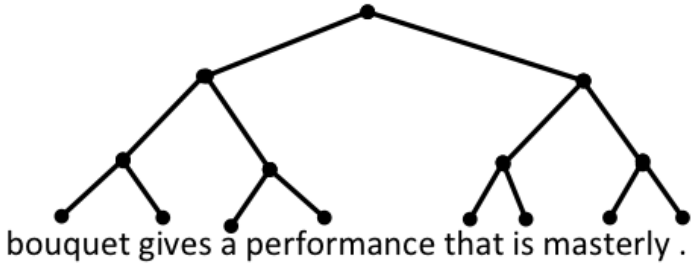

Figure 2: The tree structured determined by the greedy algorithm of a negative sentence.

References

[1] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[2] C. Elkan, "Learning meanings for sentences," http://cseweb.ucsd.edu/~elkan/250B/, February 2013.

[3] M. Schmidt, "minFunc," http://www.di.ens.fr/~mschmidt/Software/minFunc.html, accessed: 03/04/2013.