Recurrent Recursive Neural Networks for Sentiment Analysis

Amandeep Singh
Electrical Engineering Department, Stanford University

Abstract

In recent literature Recursive neural networks have been successfully used for fine grained sentiment analysis in NLP. Recursive neural networks learn the structure of a sentence and try to predict the sentiment of a given sentence. While these conventional recursive neural networks are deep in structure/space, they are not hierarchical/deep in time in their representation like a deep recurrent neural network. In this project I am exploring a model that combines N layers of recursive neural networks in a recurrent way to perform fine grained sentiment analysis [1]. Essentially, each layer of the deep recurrent network is a recursive neural network. The model gets trained by combining backpropagation through structure to learn the recursive neural network and backpropagation through time to learn the feedforward network. The proposed model is implemented and tested on Stanford Sentiment Treebank [2]. We achieved an accuracy of 81.5% on the task of 5-class fine grained sentiment analysis over each phrase. We also explore the usefulness of various regularization techniques such as L2 regulation and dropout [3-4] on training the model.

1 Introduction

Sentiment analysis is an important task in natural language processing. The current state of the art methods for fine grained sentiment analysis use recursive neural networks/recursive neural tensor networks (RNTN) [2]. In recent literature there has been a distinction amongst notion of depth in space/structure and depth in time [1]. While the recursive neural networks are deep in structure, they are not deep in time, and hence in a way lack hierarchy of conventional deep feedforward nets where each layer learns a potentially more abstract representation of the input than the previous layer.

In this project I explore a model similar to [1], that stacks together N layers of recursive neural networks in a recurrent way for fine grained sentiment analysis. In particular each layer of the network is a recursive neural network, and the recurrent neural network combines together N different recursive networks together. Figure 1 shown below shows a more detailed representation of the network. Stacking N recursive layers in a recurrent manner allows each layer to learn a particular aspect of the sentence/phrase which helps in achieving a higher sentiment classification accuracy.

2 Related Work

Sentiment analysis is an important Natural Language processing task and has been studied extensively over the last few years. Before deep learning was successfully demonstrated for this task, a large number of previous sentiment analysis techniques were either based on using a bag of words model or careful engineering of features. Deep learning methods have out-performed these previous methods, and are currently the state of art methods for this task. In particular, recursive neural networks and recursive neural tensor networks [2] have been demonstrated to achieve very high accuracy for sentiment analysis.

3 Our Approach

The model that I am using for this project is essentially based on the model presented in [1]. In this model we combine together N layers of recursive neural networks in a deep-feedforward manner to allow the model to have depth both in space/structure and time. This is different from a regular recursive neural network that has depth only in structure, and hence lacks the ability to learn hierarchical features.

Figure 1 below shows the details of the model.

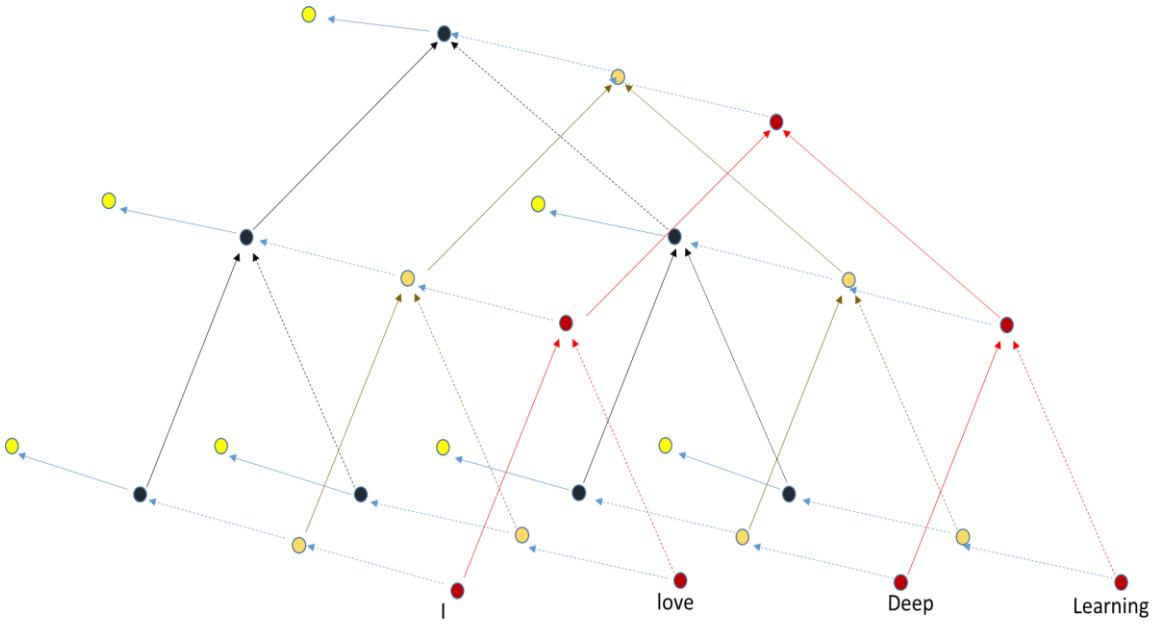

Figure 1 : Operation of a recurrent recursive neural network

Formally, our model can be represented as under:

Given a binary tree structure with leaves having the initial representations, e.g. a parse tree with word vector representations at the leaves, a re-current recursive neural network computes the representations at each internal node η as follows

$$h_n^{(i)} = f\left(W_l^{(i)} h_{l(n)}^{(i)} + W_r^{(i)} h_{r(n)}^{(i)} + V^{(i)} h_{(n)}^{(i-1)} + b^{(i)}\right)$$

where η represents a particular node, $l(\eta)$ and $r(\eta)$ are the left and right children of η, $W_L^{(i)}$ and $W_R^{(i)}$ are the weight matrices that connect the left and right children to the parent for a layer i, $V^{(i)}$ is the weight matrix that connects the $(i-1)^{th}$ hidden layer to the i^{th} hidden layer, and $b^{(i)}$ is a bias vector for layer i.

We then have a task-specific output layer above the representation layer:

$$Y_\eta = g(Uh_n^{(l)} + c)$$

where U is the output weight matrix, c is the bias vector to the output layer and l is the final number of layers.

In a supervised task, y_η is simply the prediction (class label or response value) for the node η, and supervision occurs at this layer. For the sentiment analysis task, y_η is the predicted sentiment label of the phrase given by the subtree rooted at η. Thus, during supervised learning, initial external errors are incurred on y, and back propagated from the root, toward leaves.

It may be noted that for our implementation the function f was chosen to be the ReLU function and the function g was chosen as soft-max.

3.1 Training Procedure

The proposed model was trained using back-propagation. Back-propagation for the above network can be thought as a linear combination of back-propagation through structure and back-propagation through time. Essentially for any particular node n in a layer i, it receives an error from its parent nodes in layer i, and from the node n in the layer $i+1$. The figure below shows the error-flowing in the backpropagation network. To ensure the backpropagation is implemented correctly, a simple gradient checking mechanism was implemented that checks the validity of the gradients.

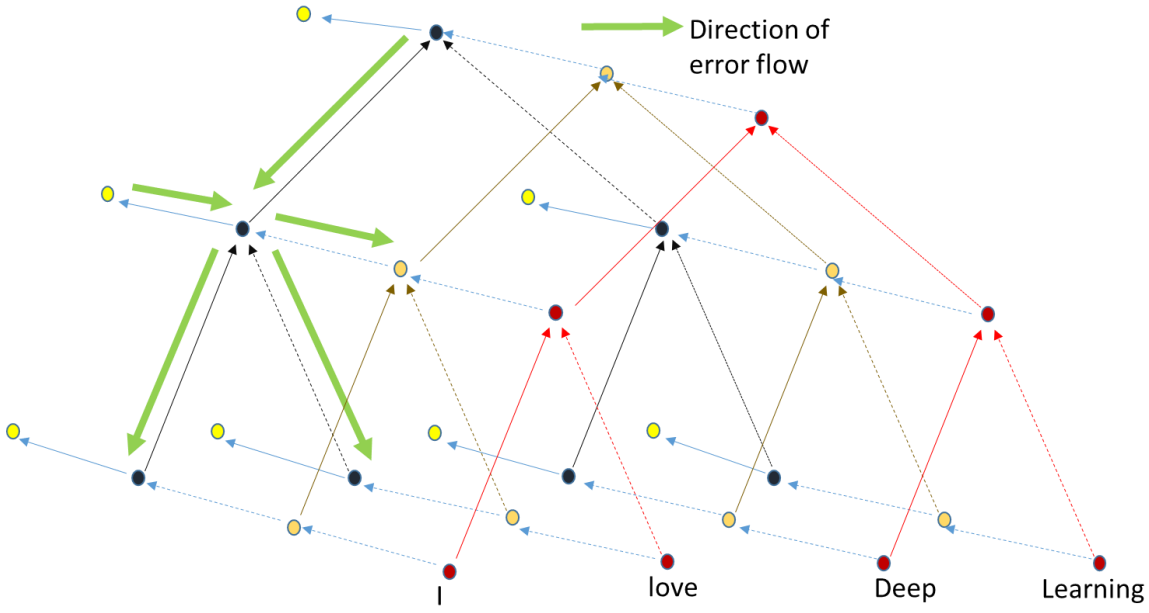

Figure 2 : Error propagation in the network

4 Experiment and Results

Dataset Used: To validate the above recurrent recursive model I used the Stanford Sentiment Treebank [2] dataset. The dataset consists of 215,154 labeled phrases from 11,855 sentences. Each of the 215,154 phrases is labeled from 5 possible classes, namely, very negative, negative, neutral, positive and very positive. It may be noted that in this report we are trying to predict the sentiment of each of the phrases in the sentence.

Choice of Word Vectors: I used randomly initialized word-vectors as the inputs rather than using pre-trained word2vec or glove word-vectors. The word-vectors were also trained as part of back-propagation. The word-vector dimension was taken as a variable and was chosen by using cross-validation. The final chosen word-vector dimension was 75. It may be noted that since 75 was the largest word-vector dimension I simulated with (due to training time limitations), it is likely that the performance will improve if we use 300-dimensional pre-trained word2vec or glove word vectors.

Choice of Regularization: Regularization plays a critical role to ensure that we do not over-fit the model. Further, the role of regularization increases as we increase the model complexity. In the present implementation I tried three different regularization methods:

- L2 norm Regularization
- Dropout Regularization
- L2 norm + Dropout regularization

Amongst the three, L2 + dropout regularization, worked the best. Interestingly, it may be noted that Dropout Regularization [3-4] without the L2 norm regularization does not work. This is because of the choice of ReLU non-linearity which is a type of non-saturating non-linearity. ReLU and no L2 norm regularization essentially causes a blow-up in magnitude of values. Figures 3 and 4 below show training set and dev set accuracies across a number of training epochs with and without dropout regularization (Both have L2 norm regularization). As is evident from the two plots below, with L2 norm regularization alone, the gap between training and dev set accuracies is

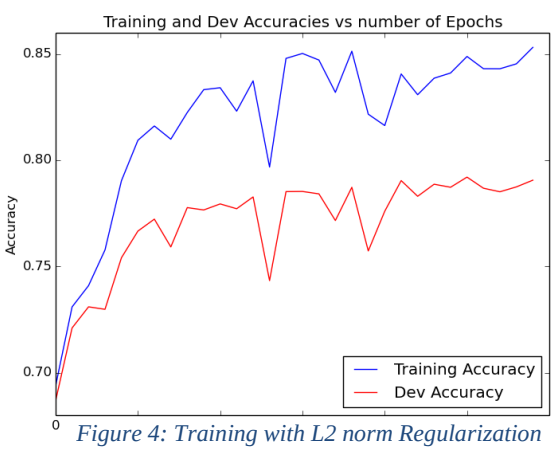

Figure 4: Training with L2 norm Regularization

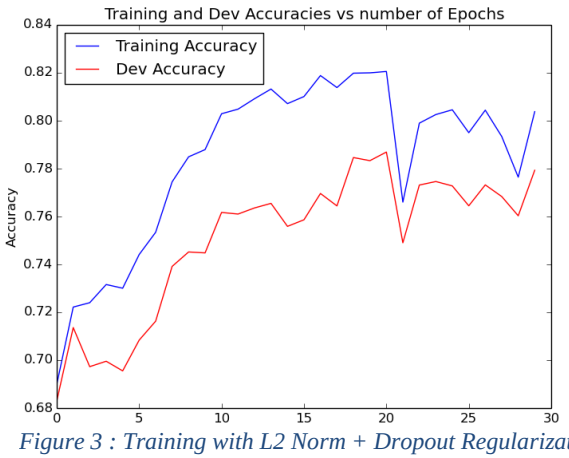

Figure 3 : Training with L2 Norm + Dropout Regularization

Figures above show a comparison between L2 norm and L2 norm+dropout regularization. As is evident from figure 4, the difference between training set and dev set accuracy is smaller when we use L2 + dropout regularization, thus emphasizing that dropout improves generalization.

much higher compared to the gap when we use L2 norm + dropout regularization.

Choice of Number of layers: Since the network is a recurrent network, an important advantage is that the network can be arbitrarily deep. In practice I found the dev set accuracy increases upto 3 layers, and started decreasing as the number of recurrent layers was increased beyond 4. This can be attributed to increased over-fitting in the model for the choice of dataset at hand.

Choice of Training Algorithm: As mentioned in section 3 above, the network was trained using Back-propagation with stochastic gradient descent. Three different versions of stochastic gradient descent, namely, regular vanilla stochastic gradient descent, Adagrad[5], and RMSProp[6], were implemented to understand the difference in performance because of the choice of training algorithms.

Figure 5-7 below shows the training and dev set accuracy for the three stochastic gradient variants. The plots below assume a word vectors dimension of 75 and no drop-out is used. As is evident from the graph Adagrad algorithm seems to be best in terms of speed, and overall performance. It is expected that the vanilla stochastic gradient descent achieves the same performance levels as ada-grad, but it takes much longer training time to reach the same training data accuracy.

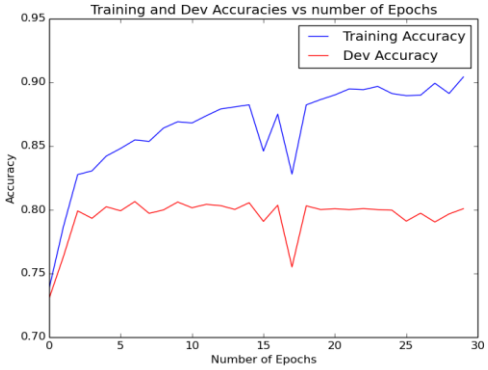

Figure 6 : Stochastic Gradient Descent with RMSProp

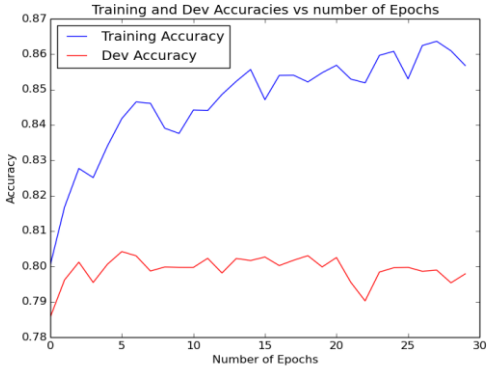

Figure 7: Vanilla Stochastic Gradient Descent

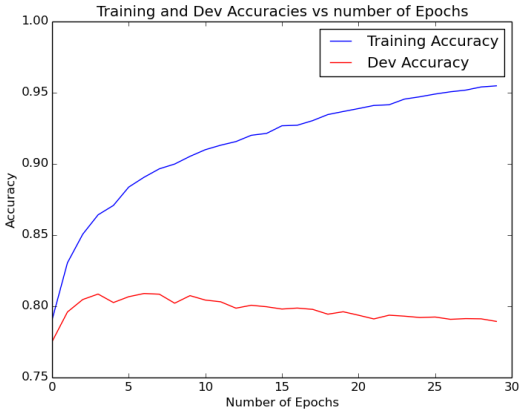

Figure 5 : Stochastic gradient descent with Adagrad

The graphs above suggest that Adagrad learning method outperforms the other two learning algorithms is terms of speed and accuracy. It may be noted that same learning rate was used for all the three algorithms to allow for a fair comparison. However, the wiggles in the figures 6-7 above suggest a smaller learning rate might be more optimal for RMSProp and Vanilla stochastic descent. The learning rate used was 1.5E-2.

Summary of Final Results:
This section summarizes the final results that I achieved with a recurrent recursive neural network. The best results were achieved with a 3-layer recurrent neural network, trained with stochastic gradient descent with adagrad. The word-vector dimension was chosen to be 75. Figure 8 below shows the training and dev set accuracy for the final set of chosen parameters. The best accuracy achieved was 81.5%.

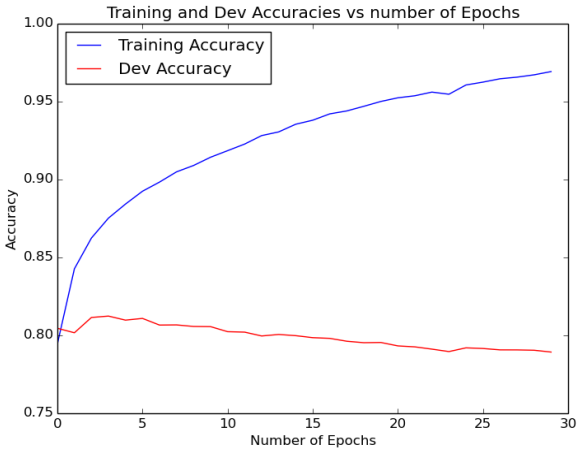

Figure 8 : Dev and training accuracy for final set of parameters

The table below compares the achieved accuracy with other prior arts methods.

Model	5 – Class Fine Grained Accuracy
Recurrent RNN	81.5%
RNTN [2]	80.7%
2-Layer RNN	80.1%
Simple RNN	79.5%

5 Conclusion

In this project I explored recurrent recursive neural networks [1] for use in fine-grained sentiment analysis of sentences. I was able to achieve an overall accuracy of 81.5% compared to 80.7% from [2] and simple RNNs. The dataset used for calculating the accuracy is the Stanford Sentiment Treebank [2]. The major advantage of the recurrent structure of the model is that it allows the model to learn hierarchical features of sentences also, since it is deep, both in structure and time. Further, I compared the performance of various variants of stochastic gradient descent algorithms. Compared to other algorithms, adagrad algorithm outperformed the other algorithms. Further, various different regularization techniques were evaluated to avoid over-fitting the data. The best regularization technique was observed to be L2 norm +

dropout. It was interesting to find that dropout alone does not work well because of the non-saturating nature of ReLu function. Adding the L2 norm to regularization avoids the explosion of output observed with dropout alone.

6 Future Work

In the current implementation of the model, I initialized the word vectors to be Gaussian random variables, and the word vectors were trained together with the structure. It was observed that the best performance was achieved for word vector dimension of 75, which was also the highest dimension used in my cross-validation. This suggests using higher word-vector dimension might improve the performance further. Also, using pre-trained Glove or Word2Vec models should help improve the performance.

It may be observed that the recurrent-recursive neural network, while being deep, is still feedforward in nature. The feedforward structure of the network does not allow information to flow backwards, and thus the network is unable to correctly predict the sentiment of any phrase which derives its sentiment from a phrase later in that sentence. For such scenarios it may be better to have bi-directional recursive model as each layer of the recurrent network. The bi-directional model will allow information flow from sentence parse tree leaves to root, and from roots back to leaves thus alleviating the above problem.

Recently tree structured LSTM models have been proposed [7]. It could be interesting to combine the tree structured LSTM model in a recurrent neural network to take the advantages of both the recurrent and LSTM Tree structured models.

7 References

[1] Ozan Isroy and Claire Cardie, *Deep Recursive Neural Networks for Compositionality in Language,* in Proceedings of Advances in Neural Information Processing Systems, 2014

[2] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. *Recursive deep models for semantic compositionality over a sentiment treebank*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '13, 2013.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks* in Proceedings of NIPS, volume 1, page 4, 2012

[4] George E Dahl, Tara N Sainath, and Geoffrey E Hinton, *Improving deep neural networks for lvcsr using rectified linear units and dropout, i*n Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 8609–8613. IEEE, 2013

[5] John Duchi, Elad Hazan, and Yoram Singer. 2011. *Adaptive subgradient methods for online learning and stochastic optimization.* JMLR, 12:2121–2159.

[6] Coursera course on Neural networks for Machine Learning by Geff Hinton.

[7] Kai Sheng Tai, Richard Socher, and Christopher D. Manning, *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks,*arXiv preprint, 2015