Sentiment Analysis Using Recursive Autoencoders

CSE250B, Winter 2014, Project 4

Prabhav Agrawal
UC San Diego
prabhav@ucsd.edu

Soham Shah
UC San Diego
srshah@ucsd.edu

March 17, 2014

---

1 Abstract

In this project, we implement a Semi-Supervised Recursive Autoencoder as described by Socher et. al [1] and reproduce the results on the Movie Reviews dataset. We achieve a best accuracy of 75.72% which is within 1% of the author's result. Our experiments show that a better performance is obtained when each word's representation is modified to minimize the loss rather than using random initializations. We also observe that applying normalization on reconstruction nodes makes the prediction accuracy less sensitive to error weight.

2 Introduction

The objective of this project is to employ Recursive Autoencoders (RAE) for sentiment prediction. Socher's paper describes a framework that uses RAEs to analyze the meaning of texts. In a RAE, meaning vectors are propagated to the root node which captures the meaning of entire sentence. Many application can be built around this, one of which is sentiment detection.

We are given Movie Reviews dataset of 10662 sentences each of which is 5-50 words long and labeled with a positive or negative sentiment. We aim to learn an RAE based on Socher's framework that predicts the sentiment for these sentences.

Section 3 discusses the definition of RAE, loss function and calculation of derivatives. Section 4 gives details of experiments conducted and Section 5 describes our results and observations.

3 Model analysis

Our model aims to find vector representations for variable-sized phrases in a semi-supervised training regime. These representations can then be used for subsequent tasks. We first describe recursive representation of meaning and then introduce RAE and describe how it can be used to learn phrase representations, sentence structure and sentiment analysis.

3.1 Recursive definition of meaning

The syntactic structure of an English sentence is represented as a binary tree, with the leaf nodes representing the words of the sentence and the internal nodes representing phrases in the sentence. We associate a column vector in $\mathbb{R}^d$ with each node, to represent the meaning of the word. A parent node x_k is constructed from its children $[x_i; x_j]$ as follows:

$$x_k = h(W_1 x_i + W_2 x_j + b) \tag{1}$$

where W_1, W_2 and b are parameters to be learned and h is a pointwise function.

3.2 Autoencoders

In the above approach, to learn the parameters we need the meaning of the sentence i.e. the meaning vector corresponding to root node x_r. In the autoencoder approach, the goal of supervised learning is to reconstruct the input. We reconstruct the children nodes x_i' and x_j' as follows:

$$x_i' = h(U_1 x_k + c_1) \tag{2}$$

$$x_j' = h(U_2 x_k + c_2) \tag{3}$$

where U_1 and U_2 are parameter matrices in $\mathbb{R}^{dxd}$ and c_1 and c_2 are vectors in $\mathbb{R}^{2d}$ and h is a pointwise function. The reconstruction error is then defined as

$$E_{rec}(k) = \frac{n_i}{n_i + n_j}\|x_i - x_i'\|^2 + \frac{n_j}{n_i + n_j}\|x_j - x_j'\|^2 \tag{4}$$

where n_i and n_j are the number of words covered by x_i and x_j respectively.

3.3 Selecting a tree structure

We use a greedy algorithm to find a tree that is good but not optimal. Given a sentence of length n, the greedy algorithm is as follows. First, consider all $n-1$ pairs of consecutive wordsand evaluate the reconstruction error for each pair. We then select the pair with smallest error. Now, either one or two of the original pairs can no longer be combined. Consider the remaining feasible pairs, plus the one or two new possible pairs on top of the first selected pair. Select the pair with the smallest error among these $n-2$ possible pairs. Continue until there is only one possible choice to create the root node.

3.4 Using meanings to predict labels

The auto encoder discussed in the previous section is completely unsupervised. It provides a representation for meanings of different sentences. To incorporate the meaning label into the framework, a linear model is added on the meaning of nodes. A softmax layer, or multiclass logistic regression, is laid on top of these nodes. Given x_k as the meaning of a node k and r as the number of alternative labels, the value for the $softmax$ layer is defined as

$$\bar{p} = softmax(V x_k) \tag{5}$$

where the parameter matrix V is in $\mathbb{R}^{rxd}$. The error on these nodes are defined by the log loss of the predictions (cross-entropy errors), or mathematically,

$$E_{ce}(k) = -\Sigma_{i=1}^{r} t_i log p_i \tag{6}$$

where $\bar{t}$ is the vector of true label values.
The total error at a node k is defined as

$$E_{total}(k) = \alpha E_{rec}(k) + (1-\alpha)E_{ce}(k) \tag{7}$$

where the hyper parameter α defines the relative importance of the reconstruction and label errors.

3.5 Optimization Function and Derivatives

Recursive autoencoders try to minimize reconstruction and classification error across all nodes of the tree. The objective function to be minimized during learning, given a collection S of m labeled training sentences, is

$$\begin{aligned} J &= \frac{1}{m}\sum_{(s,t)\epsilon S} E(s,t,\theta) + \frac{\lambda}{2}||\theta||^2 \\ &= \frac{1}{m}\sum_{(s,t)\epsilon S} \{\alpha E_{rec}(s;\theta) + (1-\alpha)E_{ce}(s;t;\theta)\} + \frac{\lambda}{2}||\theta||^2 \end{aligned} \tag{8}$$

where $\theta = \langle W_1, W_2, b, U_1, U_2, c_1, c_2, V\rangle$ is all parameters of the model, λ is the strength of L_2 regularization. Gradient of objective function J is calculated as:

$$\frac{\partial J}{\partial \theta} = \frac{1}{m}\sum_{s,t}\frac{\partial E}{\partial \theta} + \lambda\theta \tag{9}$$

For each internal node, we have two reconstruction nodes x'_i and x'_j. A class distribution $\bar{p}$ is also present. We need to calculate partial derivative of J with respect to components of θ. So, we have the following values for node in the network

$$\begin{aligned} x_k &= h(a) = h(W_1 x_i + W_2 x_j + b) \\ [x'_i; x'_j] &= h(e) = h([U_1 x_k + c_1; U_2 x_k + c_2]) \\ p &= softmax(s) = softmax(V x_k) \end{aligned} \tag{10}$$

Here W_1 and W_2 affect J through a, U_1 and U_2 through e and V through s. After calculation, we get the following partial derivatives for any internal node wrt J as:

$$\begin{aligned} \frac{\partial J}{\partial W_1} &= \delta [x_i]^T \\ \frac{\partial J}{\partial W_2} &= \delta [x_j]^T \\ \frac{\partial J}{\partial b} &= \delta \\ \frac{\partial J}{\partial U_1} &= \gamma_1 [x_k]^T \\ \frac{\partial J}{\partial U_2} &= \gamma_2 [x_k]^T \\ \frac{\partial J}{\partial c_1} &= \gamma_1 \\ \frac{\partial J}{\partial c_2} &= \gamma_2 \\ \frac{\partial J}{\partial V} &= \zeta [x_k]^T \end{aligned} \tag{11}$$

Here γ, δ and ζ are defined as:

$$\begin{aligned} \gamma &= \frac{\partial J}{\partial e} = [\gamma_1; \gamma_2] = [-2\alpha \frac{n_1}{n_1 + n_2} h'(e) \times (x_i - x'_i); -2\alpha \frac{n_2}{n_1 + n_2} h'(e) \times (x_j - x'_j)] \\ \zeta &= \frac{\partial J}{\partial s} = (1 - \alpha)(s - t) \\ \delta &= h'(a)[[W_1 or W_2]^T \delta_{par} + [U_1; U_2]^T \gamma + V^T \zeta + \frac{\partial E_{rec}(x_k, x'_k)}{\partial x_k}]] \end{aligned} \tag{12}$$

The derivative of meaning vector for leaves is calculated as:

$$\begin{aligned} \frac{\partial J}{\partial x_{leaf}} &= \delta_{x-leaf} \\ \delta_{x-leaf} &= h'(a)[[W_1 or W_2]^T \delta_{par} + V^T \zeta + \frac{\partial E_{rec}(x_k, x'_k)}{\partial x_k}] \end{aligned} \tag{13}$$

3.6 Checking Gradients

To verify the correct computation of our gradients, we compare them to those computed by finite difference. We use central difference to compute these gradients approximately.

$$\frac{\partial J}{\partial W_{ij}} = \frac{J(w_{ij} + \epsilon) - J(w_{ij} - \epsilon)}{2\epsilon} + O(\epsilon^2) \tag{14}$$

4 Design of experiments

The data is divided into two parts with 8000 training examples in training set (4000 positive and 4000 negative) and 2662 examples in validation set (1331 positive and 1331 negative). Regularization is applied to the complete θ vector with regularization parameter $\lambda = 0.01$. Dimension of meaning vectors is taken as $d = 20$. All the weight matrixes and meaning vectors are first randomly initialised from the Gaussian distribution $\mathcal{N}(0, 1)$. We used Mark Schmidt's *minFunc* function for L-BFGS optimization.
Accuracy is measured using the label matrix V learnt during training. The predicted label is calculated as

$p = softmax(Vx_k)$ where x_k is the root node of a sentence. Final accuracy is reported on the training and validation set.
As mentioned in project questions: we implemented two modifications: 1) updating meaning of leaf nodes and including errors on leaf nodes in loss J, 2) applying normalisation on reconstructed nodes x'_i and x'_j. Combination of these two modifications leads to four methods which are summarised in Table 1. We also check if the gradients are correct by performing finite differencing using *derivativeCheck* function in *minFunc*.

Methods	Normalization	Leaf Node Updating
M_1	No	No
M_2	No	Yes
M_3	Yes	No
M_4	Yes	Yes

Table 1: Different Methods in RAE

First, we run experiments on all four methods by varying error weight α and then plot a graph for the same. Second, we report the top positive and negative words and phrases for the method and α getting best accuracy. Top positive or negative words/phrases are calculated by checking the difference of two dimensions in the predicted labels. The closer it is to 1, the more positive the word/phrase is and the closer it is to -1, the more negative the word/phrase is. We also show the tree structure after learning. All the experiments were run on a machine with Intel Core i5 3.0GHz processor, with 2 cores and 8GB memory.

5 Results and Discussion

Figure 1 represents the accuracy on test set for each of the 4 methods for $\alpha = 0$ to $\alpha = 1$ in increments of 0.1. As expected Method 1 and 2 have same accuracy for $\alpha = 1$. $M1$ and $M2$, differ only in updating the meaning vectors for the leaf nodes. At $\alpha = 1$, there is no contribution from the E_{ce} term to the total error and thus their accuracies are the same. Same argument applies to $M3$ and $M4$, which show the same accuracy at $\alpha = 1$.
Similarly, at $\alpha = 0$, there is no contribution from the E_{rec} term to the total error, thus we see same accuracy for $M1$ and $M3$ and for $M2$ and $M4$. Normalization only affects the component γ, which appears only in the E_{rec} term. Since these methods only differ in the normalization component, their values are identical at $\alpha = 0$. We also observe that applying normalization on reconstruction nodes makes the prediction accuracy less sensitive to error weight.

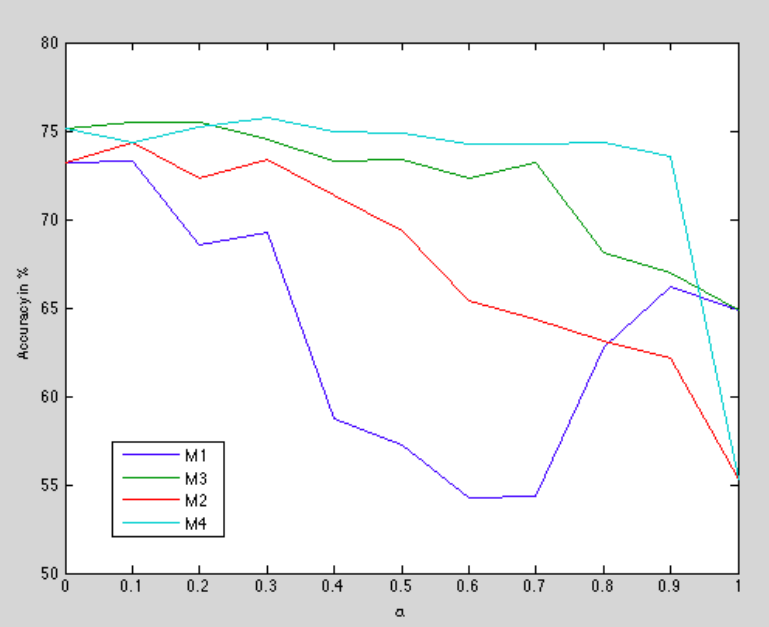

Figure 1: Plots of validation set accuracy of the four methods for error weight α

Table 2 represents the top 10 positive and negative words obtained using $M4$ for $\alpha = 0.3$ and $d = 20$. This is the combination for which we obtain highest accuracy on validation set. We also report the top positive and negative phrases found on the validation set in Table 3. It is clear from words/phrases in Table 2 and 3 that our models learns the sentiment on word/phrases correctly.

Most Positive Words	Most Negative Words
best	bad
great	too
entertaining	dull
solid	boring
culture	nothing
performance	video
enjoying	worst
funny	stupid
enjoying	flat
moving	problem

Table 2: Top 10 positive and negative words

Most Positive Phrases	Most Negative Phrases
of the roses is a surprisingly engaging film	the movie is a lumbering load of
of the actors throw off a spark or two when they first appear	the audacity to view one of
movie has a certain poignancy in light	the film would have been more enjoyable had the balance shifted in favor of
audience other than to sit back and enjoy a couple	adventure, but ends up more of
best looking and stylish animated movies in quite a while	of the characters forget their lines and just utter
smartest takes on singles culture i've seen in a long time	there is nothing exactly wrong
most ingenious and entertaining thrillers i've seen in quite a long time	most repellant things to pop up in a cinematic year already littered with celluloid garbage
the best film so far this year is a franchise sequel starting	the modern remake of
the true wonder of	this film biggest problem
a must for fans of	the saturation bombing of

Table 3: Top 10 positive and negative phrases

Figure 2 displays a binary tree structure for the sentence - "The whole affair is as predictable as can be". We can see from the tree strucure that subject - "The whole affair", verb - "is as" and object - "predictable as can be", of the sentence are combined first to form a phrase node in the tree.

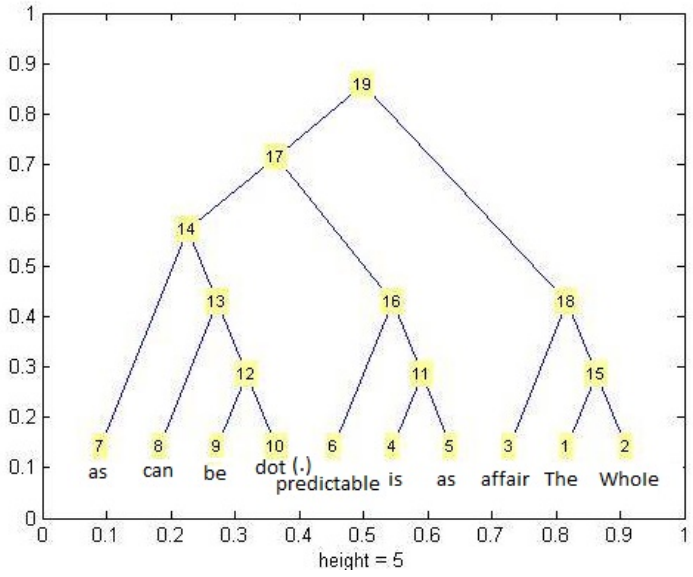

Figure 2: Binary tree structure for a sentence after training

Table 3 reports the best accuracy for all four methods along with corresponding α value, for $d = 20$ on the validation set. We observe that M4 with $\alpha = 0.3$ performs the best among all other methods. This is with 1% of the accuracy reported by Socher et. al [1].

Method	α	Accuracy
$M1$	0.1	73.32 %
$M2$	0.2	75.47 %
$M3$	0.1	73.79 %
$M4$	0.3	75.72 %

Table 4: Accuracy for $d = 20$

6 Conclusion

In this project we implemented the recursive autoencoder (RAE) as described in Socher's paper to detect the sentiment of movie reviews. We detailed the algorithms for using backpropagation to compute error gradient with respect to the parameters. We train and test our RAEs with sentences from movie reviews, and achieve 75.72% accuracy.

References

[1] Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions Richard Socher, Jerey Pennington, Eric Huang, An- drew Y. Ng, and Christopher D. Manning Conference on Empirical Methods in Natural Language Processing (EMNLP 2011, Oral)

[2] C. Elkan, "Learning the meanings of sentences," 2012. [Online]. Available: http://cseweb.ucsd.edu/elkan/250B/learningmeaning.pdf