International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 08, 2020
ISSN: 1475-7192

WORD VECTOR USING R-ADAM OPTIMIZATION FUNCTION

[1]PALLAVI SAXENA, [2]HARSHIT NAGAR, [3]D. MALATHI

Abstract

There has always been a strong interest in expressing notations to present vocabulary of a language in linguistics. This process is quite challenging because of high computational requirements, and the dimensionality curse. We use a word to vector approach to process text which can be best defined as a two-layered neural network. A text corpus will be provided as an input and a set of vectors representing features of words will be obtained as an output. It is not a deep neural network which allows it to be trained quickly compared to other neural network architectures. It embeds the words in numeric form which neural networks can consume. Continuous Bag of Words (CBOW) and Skip-gram models for word vector representation show immense success in capturing the semantic and syntactic relationships between words of a given language. These models show immense success in capturing the semantic and syntactic relationships between words of a given language. However, there is still scope of improvement in the domain of vector space representation. We propose to apply the recently introduced rectified Adam optimization function in place of the RMSprop optimization function. The function shows promising outcomes in terms of training times and accuracy for large datasets.

Keywords: *Word2Vec ,CBOW, skip-gram, word embeddings, machine translations, text summarization, Rectified Adam optimization function*

I. Introduction

One of the most popular representations of document vocabulary is word embeddings. It catches the word-contexts of a word in a given document and it's semantic and syntactic relations, relation with other words. Making use of the Rectified Adam optimization function shows promise in yielding decreased training times for large datasets and improved accuracy compared to the originally used RMSprop function for optimization and achieving minimums.

[1] CSE Department, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu, INDIA
[2] CSE Department, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu, INDIA
[3] Professor, CSE Department, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu, INDIA

Received: 12 Mar 2020 | Revised: 22 Apr 2020 | Accepted: 09 May 2020

Rectified Adam can be classified as a first-order gradient-based optimization function. It optimizes sophisticated objective functions considering their adaptive estimation of lower order moments. The implementation is quite straight forward and is comparatively lot efficient compared to pre-existing optimization functions in Skip-gram and CBOW. Although RAdam has few memory requirements, some of Adam's advantages are that the parameter updates' magnitudes remain unaffected of gradient re-scaling, its step-sizes are nearly bound by the step-size (hyperparameter), has no need of a static objective, has no trouble working with gradients which are sparse in nature, and it inherently performs a form of step size annealing. Hence it emerges as a great alternative to the problems of having large data sets and parameters. This method also proves effective for non-stationary and problems dealing with noisy or sparse gradients. The intuitive interpretations of hyper-parameters generally require some tuning.

II. Existing System

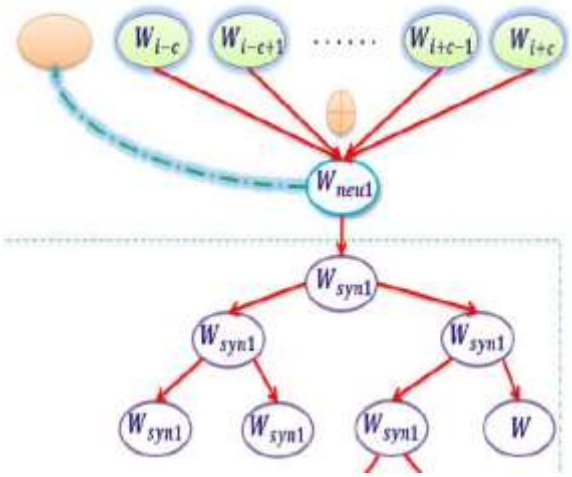
Trained on sentence-aligned bilingual data, and its objective is to breed the target (French) sentence given the source (English) sentence, word2vec, on the other hand, runs on monolingual data that's not aligned to anything. the precise objective varies (word2vec contains several algorithms), but it's usually somehow connected to the language-modeling objective of predicting a word given its context. Another way RNNs can be differentiated from CBOW is that they use an arbitrary length token stream. CBOW uses a single token at a time and is unaware of the order (which is solely vector addition) to represent a stream. RNNLM models show semantic and syntactic accuracies of 9% nad 36% respectively and also the NNLM models have 23% semantic accuracy and 53% syntactic accuracy.

III. Proposed System

We are trying to implement the pre-existing word to vector representation using the CBOW and Skip-gram models with the Rectified Adam Optimization function which was recently introduced as an improvement over the Adam function.

Since the internal parameters of a given neural network significantly contributes to training the data set more effectively and efficiently and hence producing less deviated output we plan on incorporating Rectified Adam Optimization function to calculate appropriate and optimum values of our word vectorization model. These values lay a major impact on the learning process of the data set and the output generated by the function.

The below figure represents the classic architecture of the pre-existing word vectorization process. It showcases all the hidden layers that are taken into consideration while building the entire model.



3.1 Word Vectorization Process

Word vectorization process can be broadly seen as a grouping of various words that possess semantic relationships between them. This process forms mathematical representations of the words in the given data set and clubs similar words together. The word is apprehended using the cosine function between two-word vectors. To understand it better, we can consider that the word vectors are represented in an N-vector space where each word possesses some value in each dimension. Then cosine similarity function is evaluated on each pair of terms. The lesser difference between two terms indicates the closeness of interconnectivity between two-word vectors.

3.2 Rectified Adam Optimization Function

Adam optimization function categorically comes under first-order optimization functions. Such types of algorithms prime goal to minimize or maximize the loss function. Loss function can be seen as a mapping entity that relates the produced output values to real numbers. These real numbers define the cost associated with the occurrence of that particular event.

Adam function takes the loss function represented as E(x) where x is the vector variable based on their gradient values. The gradient value of each is calculated using the internal parameters of the neural network. It is the most widely utilized First-order optimization technique. The first-order derivative is responsible for finding out whether a given function is increasing or decreasing at a particular point. A line is produced with respect to the Error Surface which is basically tangential to it.

The goal of the Rectified Adam optimizer is two-fold:

1.Obtain a more accurate/more generalizable deep neural network

2.Complete training in fewer epochs

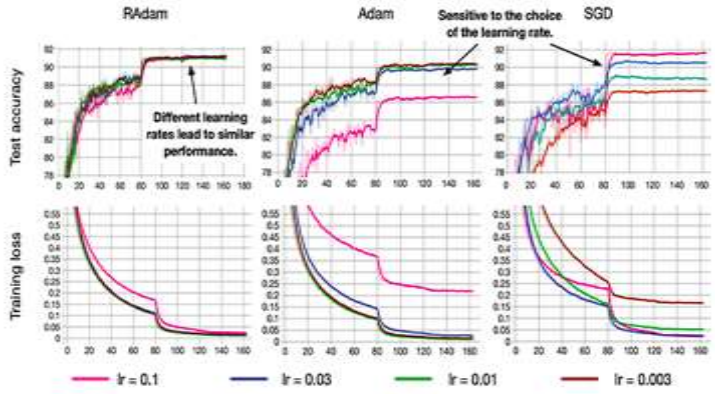


Figure: Using the Rectified Adam (RAdam) deep learning optimizer with Keras

The problem with adaptive learning rate is that during the first few batches, the models do not generalize well and show very high variance. RAdam was introduced to resolve this issue by:

1.Applying warm up with a low initial learning rate.

2.Turning off the momentum term for the first few sets of input batches. As training continues, the variance will stabilize, and from there, the learning rate can be increased and the momentum term can be added back in.

3.3 Preparation and visualization of the data

In order to prepare the dataset for processing, we need to clean the given data. Cleaning of data means to clear out all the unnecessary punctuation and capitalizations from the corpus. We can decide to include or remove stop words. like 'and', 'is'. The text is converted to lowercase, removal of headers, footers, HTML, XML, and any other markups are done. Since the mathematical mapping of word vectors onto real integer numbers yields better performance of programs, we create a word to integer conversion dictionary.

This entire process can be done using the below-defined procedure.

Submapping: This process involves elimination of redundant words.

Tokenization: It can be defined as the process of breaking down large entities into smaller ones. In simpler words, it can be compared to a lexical analyzer. For example,if we have large paragraphs,we can break it down to constituent sentences. Those sentences can again be broken down into words. This entire process is termed as tokenization.

3.4 Skip-gram model:

Every word is given an integer value and then labels are generated with the correct pair of words labelled as 1 and the incorrect ones as 0. For example: If the sentence is 'the quick brown fox jumped over the lazy dog' and window size is 2 then for focus word 'fox' we will have - (quick, fox)->1, (brown, fox)->1, (jumped, fox)->1, (over,fox)->1 and (the, fox)->0 and so on.

3.5 CBOW model:

If the context words for the focus/target word are as follows - context: [the, old, of, the], -> Target: testament, then pairs are generated as (the, testament), (old, testament), (of, testament), (the, testament)

3.6 Training the network

Both the networks are trained first using rms prop optimization function and then using Rectified adam optimization function. The prime goal is the computation of adaptive learning rates for the word vectors. The functioning of RAdam is similar to that of momentum. It keeps track of the averages of all the past squared gradients that are decaying at an exponential rate. It prefers to achieve a flat minima in the Error Surface. The gradients occurred in the past and their squares are represented using mt and vt respectively. They can be easily calculated using the below-mentioned formula.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Here mt and vt represent the first and second moments of the gradients. This scenario arises in the initial proceedings when the decay functions β1 and β2 are close to 1.This problem is fixed using recomputing bias-free momentums. The below formulae can be included for performing the calculations.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

3.7 Testing the data

We replaced the convention CBOW(Continuous Bag of Words) and Skim gram model's RMS Prop optimization functions with the RAdam Optimization function. The number of epochs processed to reach similar loss and the least loss reached and maximum accuracies are compared for each of the model's optimization functions.

IV. Results

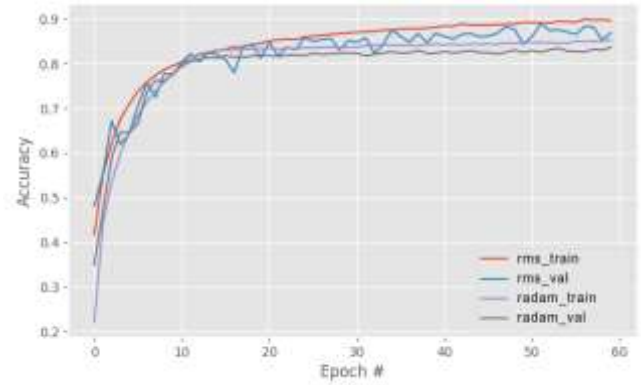
The model was run once with rectified adam optimization function as the optimization function and once with rms prop as the optimization function. Graphs were plotted for Accuracy and Loss.

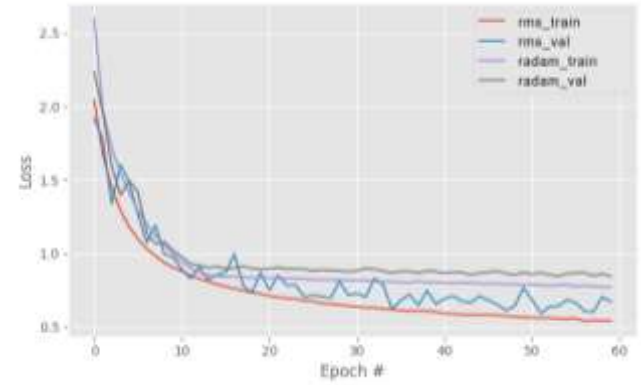
The graph for accuracy features training and validation accuracies for both the optimization functions. At the first glance it can be seen that compared to rms prop, R-adam has a much smoother curve and saturates faster. At the same time it has lower accuracy. Specially in case of validation accuracy, R-adam settles much faster. R-adam proves to have reduced training times at the cost of slightly lower accuracy compared to rms-prop.

The graph for loss shows similar trends with fluctuating and unsettling validation loss for rms-prop and smooth and quickly saturating for R-adam. The training can be stopped far ahead with R-adam as the loss function with very small loss in accuracy resulting in nearly half the training times at a small loss in accuracy compared to rms-prop.

Training accuracy for r-adam reached 84.9 compared to 90.0 for rms-prop. Training loss was not impressive for both being at 0.7 and 0.5 for r-adam and rms-prop respectively.

Validation accuracy for r-adam was 83.8 and rms-prop was 86.8 while validation loss was 0.8 and 0.6 for r-adam and rms-prop respectively.






```
(12424, 12424)

{'egypt': ['pharaoh', 'mighty', 'houses', 'kept', 'possess'],
 'famine': ['rivers', 'foot', 'pestilence', 'wash', 'sabbaths'],
 'god': ['evil', 'iniquity', 'none', 'mighty', 'mercy'],
 'gospel': ['grace', 'shame', 'believed', 'verily', 'everlasting'],
 'jesus': ['christ', 'faith', 'disciples', 'dead', 'say'],
 'john': ['ghost', 'knew', 'peter', 'alone', 'master'],
 'moses': ['commanded', 'offerings', 'kept', 'presence', 'lamb'],
 'noah': ['flood', 'shem', 'peleg', 'abram', 'chose']}
```

V. Conclusion

Rectified adam optimization function performs well and gives satisfactory results if not as good as the slightly better results given by rms-prop at the cost of more number of epochs to settle and highly fluctuating validation loss and accuracy. R-adam reaches saturation much faster than rms-prop and is much more stable. It allows to stop the training much earlier than rms-prop proving its better training times and good results.

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

[3] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504-507 2015.

[4] F. Sebastiani, “Machine learning in automated text categorization,” ACM Comput. Surv., vol. 34, pp. 1–47, 2002.

Received: 12 Mar 2020 | Revised: 22 Apr 2020 | Accepted: 09 May 2020