ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 6, June 2017

Comparative Analysis of Optimization Algorithms for Sentiment Analysis

[1] Anirudh Ganesh, [2]Manthan Gandhi, [3]Balasubramanian V
[1] [2] [3]VIT University Vellore, India
[1] manthan.pradhumangandhi2013@vit.ac.in, [2] anirudh.ganesh2013@vit.ac.in, [3]Balasubramanian.v@vit.ac.in

Abstract: ***With the advent of internet, there has been an exponential increase in user created content like customer reviews, comments and opinions. The primary cause for this sudden increase is the rapid adoption of social networks. These websites act as a medium to quickly and effortlessly share content across a wide array of domains such as products, events, people etc. This wealth of user knowledge if processed properly could be very beneficial to various businesses, governments and individuals. But the one crucial step that prevents the use of such data is that most modern techniques that do so are extremely time-consuming. This lead to a desire to develop a system that can automatically and intelligently mine such huge amounts of data and classify according to the positivity or negativity expressed within. Natural Language Processing (NLP), a branch of computer science that deals with the fruitful interpretation of human languages by computers, has given rise to a set of algorithms that perform the automated mining of attitudes, opinions and emotions from text, speech and various other sources, called Sentiment Analysis. Objective of this paper is to compare and contrast the effect of various optimizers on the efficiencies of these algorithms.***

Keywords: ***LSTM; NLP; Neural Networks; Sentiment Analysis; SVM; RMSprop; Adagrad; Stochastic Gradient Descent; Comparitive Analysis***

I. INTRODUCTION

Human decision making is always laid by others' imagination, knowledge and opinions. The increase in social web contributes huge amount of user generated content such as comments, reviews and opinions about the purchased products, services and events. This data is invaluable for both consumers and manufacturers alike. Usually consumers check the reviews of the goods before making a purchase. This helps the manufacturers gauge the pros and cons of their products through the customers' feedbacks. The consumer is overwhelmed by this large amount of data during the decision making process. Interpreting and encapsulating the opinions delivered in this huge opinionated text data is a fascinating domain for researchers. This new research domain is usually called "Sentiment Analysis" or "Opinion Mining". The programmed mining of feelings, emotions and moods from text, speech, and other sources through Natural Language Processing (NLP) is known as Sentiment Analysis. Sentiment analysis involves classifying opinions in text into categories like "positive" or "negative" or "neutral". It's often referred to as subjectivity analysis, opinion mining, and appraisal extraction [1]. The fields like Subjectivity Detection, Sentiment Prediction, Aspect Based Sentiment Summarization, Text summarization for Opinions, Contrastive Viewpoint Summarization, Product Feature Extraction, detecting opinion spam are the main fields of research in Sentiment analysis. Subjectivity Detection determines whether text is opinionated or not. Sentiment Prediction is a task of predicting the polarity of text whether it is positive or negative. Aspect Based Sentiment Summarization is to give sentiment summary in the form of star ratings or scores of features of the product. Text Summarization is used to generate a few sentences that summarize the reviews of a product. The contradicting opinions are prioritized by Contrastive Viewpoint Summarization. Product Feature Extraction distills the product features from its review. Detecting opinion spam is a task for identifying fictitious or bogus opinion from reviews. Sentiment classification can be done at Document level, Sentence level and Aspect or Feature level. In Document level the whole document is classifies into two classes: positive class or negative class. Sentence level sentiment classification has ability to classify the sentence into neutral class as well as positive class and negative class. Aspect or Feature level sentiment classification works to identify and retrieve product features from the source data. The sentiment Analysis majorly has two approaches: machine learning based and lexicon based. Machine learning based approach uses classification techniques to classify text. Lexicon based method uses sentiment dictionary with opinion words and match them with the data to determine polarity. They give sentiment scores to the opinion words describing how Positive, Negative and Objective the words of the dictionary are. The objective of this paper is to compare the effect of the various optimization algorithms on the accuracy of the popular machine learning based techniques.

II. DATASET USED

The dataset consists of 50,000 reviews from the Internet Movie Database (IMDB), with a maximum of 30 reviews per movie [2]. The dataset also comprises of a near even distribution of positive and negative reviews, i.e. random guessing will yield only 50% accuracy. The reviews included are also highly polarized in nature. Based on the work cited in [2], we know that a review of < 4 as negative and a review of > 7 as positive. Neutral reviews are not included for sake of simplicity. This dataset has been tested and proven to be a good sample dataset to benchmark the effectiveness of existing algorithms.

The dataset was divided evenly into training and test sets. We use 25,000 reviews for training and 25,000 reviews for testing. We have chosen this in-order to minimize the effect of variance (overfitting) that might arise. This can be verified by an increase of 25 reviews which should yield around 0.1% increase in accuracy.

III. TECHNIQUES USED

Machine Learning refers to a set of techniques and algorithms employed, for a variety of tasks like classification, prediction, regression or even feature recognition. Machine Learning is often, in itself, categorized into two categories, supervised learning and unsupervised learning. Since we are only focusing on Supervised Learning methods, we will instead differentiate our techniques through the inherent architectural differences of the methods.

Firstly, we have a linear classifier which uses a Bag of Words model along with SVM, and secondly we use a neural network model, the Long Short Term Memory (LSTM). We will not focus on the inherent models themselves instead, we will focus on the effect of optimization algorithms on the accuracies of these algorithms.

A. Bag of Words (BoW) with SVM using fastText

fastText is a bag of words based model introduced in 2016 by Joulin, Armand and Grave, Edouard and Bojanowski, Piotr and Mikolov, Tomas under facebook research [3]. Unlike the neural network based approaches for NLP, this technique is a linear classifier. Although, neural network based approach are gaining widespread popularity due to their relative simplicity [4] they are relatively slow and cannot be scaled to large datasets. Though linear classifiers are extremely fast [5] and they can handle large datasets [6], right features are needed to extract the full potential in these techniques. fastText algorithm, improves the problems faced by other linear classifiers by improving their generalization through sharing parameters across features and classes through an architectural change in the word representation model.

B. Long Short Term Memory Networks (LSTMs)

LSTMs is a specialized form of Recurrent Neural Network (RNN), with the added benefit of being able to assimilate long- term dependencies during the learning process. Introduced originally by [7]. The architecture of an LSTM is such that they tend to store information for a long time, hence during implementation, we try to mitigate this long term dependency. All RNNs are composed of recurrent modules with simple structure, an LSTM on the other hand is composed of a relatively complex nodes.

IV. OPTIMIZERS USED

At the heart of every machine learning algorithm is a measure of the quality of current configuration, this is given by what is known as a “loss function”. The loss function tells us how close the given parameters are to mimicking the data that the algorithm is being trained on. It is therefore, imperative to ensure that we minimize this function in order to get the best possible set of parameters in order to get the best performance out of our algorithm. Optimization is the process of finding these best parameters.

A. Stochastic Gradient Descent (SGD)

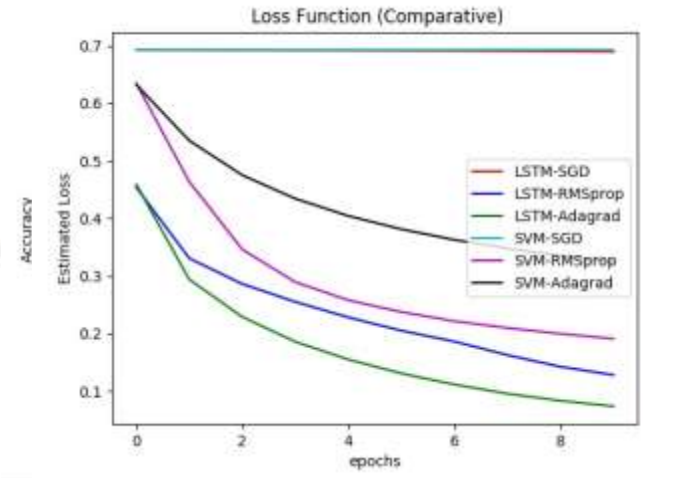
Gradient of any given function gives its slope along all dimensions, thus we can intuitively reach the minima of a given function by following the negative gradient of the function. It is based on this approach that SGD works, by performing parameter updates that correspond to a negative gradient, SGD hopes to reach the global minima of the function. It incorporates gradient descent through single data point every update. Though this was very important for reducing the computational expense on older, single core processors, but with the advent of multi-core processors and highly parallelized libraries, computing gradient over mini- batches of sizes greater than 1 gives better results.

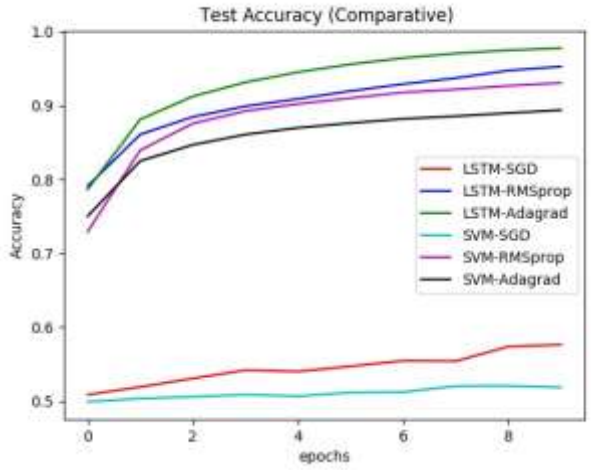
B. RMSProp

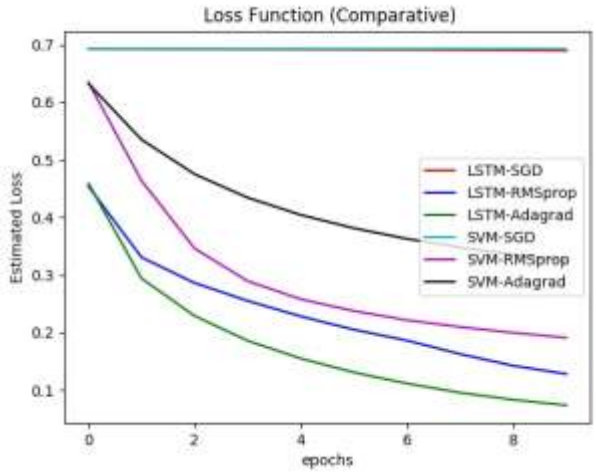
RMSprop aims to optimize by reducing the gradient via an average of its recent magnitude. By moving an average of squared gradient, it improves the learning. The better optimization is obtained through root-mean square (RMS) of the weights 'w' and 't' [8]. It works similar to Adagrad, but a tweaked parameter update which reduces the monotonic learning rate decay. The decay rate is a hyper-parameter, which controls the learning rate akin to Adagrad based on its gradients and the lack of large decay ensures a more balanced update in each step.

C. Adagrad

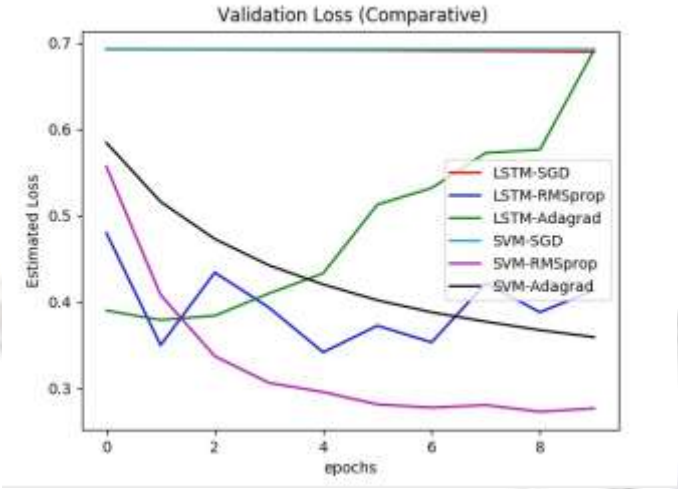
Adagrad uses the geometric properties of the training data to observe to dynamically perform a more informed gradient based learning [9]. This is achieved by giving common features a high learning rate penalty and uncommon features a low learning rate penalty. This helps optimize for predictable features that occur infrequently. This is done by tracking the per-parameter sum of squared gradients. This controls the update amounts. Care needs to be taken when using this optimizer so as to avoid high bias due to very aggressive learning rate decay. Generally this is done by ensuring that the gradient update contains the tracking parameter under a square root so that it avoids under-fitting because of early stopping. The other cautionary step taken is introduction of smoothing terms, generally of a very small magnitude in the range of 1e-4 to 1e-8 depending on the use case. As an added benefit, the smoothing term also help control division by zero apart from smoothening out the parameter update.







V. GRAPHICAL ANALYSIS



VI. RESULTS

TABLE I. TEST ACCURACIES

Algorithm	Optimization Method		
	SG	***RMSprop***	***Adagrad***
LSTM	57.62%	95.29%	97.78%
SVM (fastText)	51.84%	93.08%	89.94%

Fig. 1. Test accuracies of the various techniques using the given optimization methods.

From Table 1 we can see that for LSTMs, Adagrad seems to be the best optimization method for test accuracy, while for the SVM, RMSprop seems to perform the best. This means that while training, a steady decrease of RMSprop seems to work well for the linear approach, while the neural network performs the best when learning from features in the text that occur less frequently, which means that while SVM relies on a steady learning, a neural network relies more on learning from its mistakes.

TABLE II. VALIDATION ACCURACIES

Algorithm Used	Optimization Method		
	SG	*RMSprop*	*Adagrad*
LSTM	56.37%	84.58%	81.71%
SVM (fastText)	50.25%	88.92%	86.92%

Fig. 2. Validation accuracies of the various techniques using the given optimization methods.

In Table 2, we can see that for both algorithms, RMSprop seems to yield the best result for testing, it can thus be inferred that the steady learning rate given by RMSprop leads to better generalization than the infrequent features that the Adagrad seems to prioritize.

In both cases however, we see that SGD performs the worst, being barely above a random prediction in terms of its accuracy, which means that SGD seems to get stuck on a local minima leading to a less than desirable solution.

REFERENCES

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval 2(1-2), 2008, pp. 1–135.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, "Learning Word Vectors for Sentiment Analysis," The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[3] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, "Bag of Tricks for Efficient Text Classification," arXiv preprint arXiv:1607.01759 (2016)

[4] Xiang Zhang, Junbo Zhao, Yann LeCun, "Text understanding from scratch," arXiv:1502.01710 (2015).

[5] Sida Wang and Christopher D. Manning, "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification," ACL Web (2012).

[6] Alekh Agarwal, Olivier Chapelle, Miroslav Dud´ık, and John Langford, "A Reliable Effective Terascale Linear Learning System," Journal of Machine Learning Research 15 (2014) 1111-1133.

[7] Sepp Hochreiter and Jürgen Schmidhuber, "Long Short-Term Memory,"
Neural Comput. 9, 8 (November 1997), 1735-1780.

[8] Geoffrey Hinton, "Neural Networks for Machine Learning: Lecture 6a Overview of mini-batch gradient descent," http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[9] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," Journal of Machine Learning Research 12 (2011) 2121-2159