Available online at www.elixirpublishers.com (Elixir International Journal)

Information Technology

Elixir Inform. Tech. 132 (2019) 53333-53336

ISSN: 2229-712X

Performance Analysis of Stochastic Gradient Descent - Based Algorithms for Time Series Sequence Modeling

Zachary Kirori
Department of Computing & Information Technology, Kirinyaga University, Nairobi, Kenya.

ARTICLE INFO

Article history:
Received: 25 May 2019;
Received in revised form:
29 February 2016;
Accepted: 2 March 2016;

Keywords
Time Series,
Predictive Modeling,
Performance Analysis,
Stochastic Gradient Descent,
Deep Learning Optimization.

ABSTRACT

In many modern computer applications such as Market Analysis, Critical Care, Speech Recognition, Physical Plant Monitoring, Sleep Stage Classification, Biological Population Tracking, data is captured over the course of time, constituting a Time-Series. Time-Series data often contain temporal information dependencies that cause two otherwise identical points of time to belong to different classes or predict different behavior. This inherent characteristic increases the difficulty of processing such data. Deep Machine Learning (DML) techniques possess the inherent ability for analyzing and making predictions about such data. By its nature, DML requires extensive provision of resources key amongst which is the model computation time. Several optimization algorithms have been invented in the recent past and compare differently in terms of their resource needs. The most popular class of optimization algorithms is based on the classical stochastic gradient descent (SGD) algorithm due to its ability to converge within reasonable time bounds. This paper is part of a larger project investigating optimization procedures for deep learning tasks based on the SGD. Specifically, we report on the comparative performance capabilities of the most popular SGD based algorithms for task of Time Series prediction namely. From our analysis of the six of these algorithms, we noted that ADAMAX is most appropriate for online learning while RMSPROP is the least affected by over-fitting for long training cycles.

Introduction

Efficient solutions to hard Artificial Intelligence (AI) tasks are invariably found in the realm of Deep Machine Learning (DML) by training deep neural networks (DNNs) using selected training algorithms. DML, [1], techniques possess the structural ability to learn hierarchical features from raw input data and subsequently use these features to make predictions over previously unseen data. The field of DML has recently enjoyed success on various machine learning tasks including speech processing, natural language processing and object recognition [8][14] Central to DML tasks are optimization algorithms that process the numerical computation of parameters for a system designed to make decisions based on unseen data. That is, based on currently available data, these parameters are chosen to be optimal with respect to a given learning problem.

Typically, most real-world tasks are complex due to presence of noise in sparse data and hence require deep models with a large parameter manifold. [3]; [13] Ironically, [16], it is because ofadvances in computing capability for training these models that has made it much harder to learn hierarchical features than optimize models for successful machine learning tasks. In this regard, the training process requires significantly more training data and computing power in order to prevent over-fitting and increase model generalization capabilities. The success of certain optimization methods for machine learning has inspired research efforts towards more challenging machine learning problems and the design new methods that are more widely applicable.

Optimization problems in machine learning arise through the definition of prediction and loss functions that appear in measures of expected and empirical risk that one aims to minimize [8]. There are two varieties of optimization problems in machine learning: the first involves convex optimization problems, derived from the use of logistic regression or support vector machines, while the second typically involves highly complex and problems with non-convex error functions, derived from the use of deep neural networks. There are several optimization algorithms to automatic machine learning for non- convex objective functions with the most successful and widely used being those inspired by stochastic gradient descent (SGD).

Deep Neural Networks are trained using the Backpropagation Algorithm [2] – especially its variant - the Back Propagation Through Time (BPTT) which is numerically formulated as a highly non-convex optimization problem in a very high dimensional feature space. The training process requires extreme skill and care. For instance, it is crucial to initialize the optimization process with a good starting point through parameter tuning and to monitor its progress while correcting conditioning issues [4]. A great deal of successes in deep machine learning lies not much in the neural network, but in the choice of the training algorithm, the domain of application, the quality and amount of training data as well as the tuning of hyper-parameters.

Tele: +254 721 744 275
E-mail address: zkirori@kyu.ac.ke

Currently, most machine-learning tasks are mostly formulated as optimization procedures rather than the emphasis of the neural models and algorithms used; where the main objective is the quality of the training environment.

1. Gradient Descent Optimzation Algorithms

According to [4], in as much as they are the future of machine learning, deep neural architectures pose many notable challenges that continue to attract great interest amongst researchers in the AI community. Key among these include convergence to local minima, saturating activation functions, overfitting, long training times, exploding and vanishing gradients. Deep Neural Networks (DNNs) are highly nonlinear and difficult to optimize. During training, the parameter iterate may move from one local basin to another, or the data distribution may even change [1], posing serious challenges to researchers that can only be overcome by carefully selected deep learning algorithms. Recent work in unsupervised feature learning and deep learning has shown that, the ability to effectively train large models can dramatically improve performance. Therefore, fast convergence and robustness against stochasticity are important characteristics desirable for a deep learning optimizer [4].

At the root of gradient-based algorithms in neural networks is Stochastic Gradient Descent (SGD) optimization algorithm as reported in [2][4][16]; which updates a set of parameters, θ, using a learning rate, η, computed on every training instance as the gradient of cost function J. Then, the parameters, θ, are updated in the direction of the gradients using the same value of the learning rate, η. Beyond SGD, a number of methods have been introduced to adapt the separate learning rate for each parameter, called adaptive optimizers. The most prominent so-called first-order gradient descent based optimizers are ADAGRAD, RMSPROP, ADADELTA, ADAM, ADAMAX and NADAM.

The researchers in “[19]” presented a family of subgradient methods that dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. ADAGRAD, as it is popularly known, is a gradient-based method in which the shared global learning rate η is divided by the l2-norm of all previous gradients, nt, introducing different learning rates for every parameter at each time step, so that larger gradients have smaller learning rates and vice versa. Further, the authors in “[20]” introduced Adaptive Moment (ADAM), an algorithm for first-order optimization based on adaptive estimates of lower-order moments with intuitive hyper-parameter interpretations that require little tuning.

ADAMAX, [17], is a variant of ADAM based on the infinity norm, with inherent capability of adjusting the learning rate based on data characteristics and hence suited to learn time-variant processes such as speech data with dynamically changing noise conditions. RMSPROP was later introduced that restricts a window over the recent gradients to acquire local information instead of storing all the past squared gradients from the beginning of the training by using a decaying weight of squared gradients is applied.

ADADELTA, which is similar to RMSPROP, takes the decaying mean of the past squared gradients, nt, accumulates this quantity, and its square root, rt of past squared gradients up to the time t [11]. The obtained parameter update is stored in $\Delta\theta t$. Then the squared parameter updates, st, is accumulated in a decaying manner to compute the final update. NADAM is a hybrid of two algorithms: ADAM and Nesterov Accelerated Gradient (NAG), through modification of ADAM’s momentum component to take advantage of insights from NAG.

2. Deep Learning Networks

One of the earliest successes of Deep Neural Networks (DNNs) was reported with the introduction of greedy layer-wise unsupervised learning for Deep Belief Networks (DBNs), capable of handling the vanishing gradients problem appear [10].

A neural network that is too big and with layers that are fully connected can become infeasible to train. Trained mostly with the Backpropagation algorithm, Convolution Neural Networks (CNN) [15] are common for image processing tasks and reduce the number of parameters to be learned by limiting the number of connections of the neurons in the hidden layer to only some of the input neurons. A hidden layer (in this case, also called a convolutional layer is composed by several groups of neurons with the weights of all neurons in a group are shared. When the network has loops, it is called a Recurrent Neural Network (RNN). It is possible to adapt the Backpropagation algorithm to train a recurrent network, by “unfolding" the network through time and constraining some of the connections to always hold the same weights.

One problem that arises from the unfolding of an RNN is that the gradient of some of the weights starts to become too small or too large if the network is unfolded for too many time steps. This is called the vanishing gradients problem. A type of network architecture that solves this problem is the Long Short Term Memory (LSTM). In a typical implementation, the hidden layer is replaced by a complex block of computing units composed by gates that trap the error in the block, forming a so-called “error carrousel" [16].

There exists alternative neural architectures such as Restricted Boltzmann Machines (RBM), Hopfield Networks and Auto-Encoders. Other variations of deep architectures use several modules that trained separately and stacked together so that the output of the first one is the input of the next one [13].

3. Related Literature

This section reviews some of the reported approaches to Time-Series task analysis and modeling using Deep Learning architectures as well as comparative analysis of optimization procedures and algorithms for and not limited to Time-Series data modeling. Time-Series is defined as a vector $X = x^{(1),}x^{(2)}\dots x^{(n)}$, where each element $x^{(t)}$ R^m pertaining to X is an array of m values such that $x_1^{(t)}$ $x_1^{(t)}$ $^{,\dots,}x_m^{(t)}$. Each one of the m values correspond to the input variables measured in the time-series. Most work using ANN to manipulate Time-Series data focuses on modeling and forecasting [5][6].

The temporal nature of Time-Series data facilitates studies in different fields of applications: while doctors can be interested in searching for anomalies in the sleep patterns of a patient, economists may be more interested in forecasting the next prices some stocks of interest will assume. These kinds of problems are addressed in the reported literature by a range of different approaches such as Classification, Segmentation, Anomaly Detection and Prediction [7]. As an early attempt on using Artificial Neural Networks (ANNs) for Time-Series analysis, [9][12] modelled stock prices over a range of 8 years.

Other related research include the one by [10], for big data weather forecasting, the use of deep convolutional neural networks by [18] for Time-Series classification using multi-channels, Time-series forecasting of indoor temperature using pre-trained deep neural networks in [12], forecasted tourism demand using time series, artificial neural networks and multivariate adaptive regression splines, [7], performed Time series forecasting using a deep belief network with restricted boltzmann machines. The research in [6], applied time series and artificial neural network models in short-term forecasting of power generation while [9], carried out multi-scale internet traffic forecasting using neural networks and time series methods.

4. Experiment

This section details the statement of the problem to be solved, the dataset used, the machine-learning platform applied, the neural network model selected for training as well as the various optimization algorithms tested on varying parameters of number of epochs and the batch size.

4.1 Problem Statement

The problem selected for this study is regression of time series data representing the number of airline passengers arriving at an international airport. The prediction problem was formulated as: given a month, the task is to predict the number of international airline passengers in units of 1,000. It was then phrased as a regression problem with a window of three (3) recent time steps that were used to make the prediction for the next time step given the current time step. In this case, the input variables are t-2, t-1, t and the output variable is t+1.

4.2 Dataset and Machine Learning Platform

The international airline passengers dataset is publicly available from major international machine learning libraries. For specificity, the dataset was retrieved from the Berkey UC Machine Learning library. This was prepared using the Python based Keras Machine Learning Library operating under the Tensor Flow backend.

4.3 Use of the MLP

The experiments were conducted using a fully connected Multi-Layer Percentron (MLP) of three (3) input layers optimized using dropout at each layer's input to improve the generalization capability and its potential non-linearity addressed by the rectified linear activation unit (ReLU). The latter has the effect of preventing saturation of the gradient when the network becomes very deep.

4.4 Training Algorithms

As indicated in the introductory section, six (6) popular stochastic gradient descent based optimization algorithms were selected namely ADAM, ADAMAX, ADAGRAD, NADAM, ADADELTA and RMSPROP. The experiment was conducted using the pure stochastic version by setting the batch size to 1 and then varying the number of iterations through the dataset by setting the epochs number to 30 and 50. The same was repeated using a simple batch of 2 and the same number of iterations. The method used for purposes of stratified cross validation was to split the ordered dataset into train and test sets at the ratio of 7:3 respectively. This is necessary in order get an idea of the skill of the model on new unseen data.

Once the model is fitted, the subsequent activity was to estimate its performance on the train and test datasets. The role of this is to provide a point of reference when comparing new models. The technique applied was the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE) in order to calculate the accuracy of the algorithms on the datasets. Finally, predictions were generated using the model for both the train and test datasets and plotted on a common graph.

5. Results & Analysis

Table 1 below elaborates the model structure upon being fit into the dataset while Table 2 summarizes the results obtained for various parameter settings. The comparison graphs for both training and testing are combined in Figure 1 below

Table 1. Model Structure.

Layer (type)	Output Shape	Param #
dense_97 (Dense)	(None, 12)	48
dense_98 (Dense)	(None, 8)	104
dense_99 (Dense)	(None, 1)	9
Total params: 161 Trainableparams:161 Non-trainable params: 0		

The algorithms exhibit a slight difference in comparative accuracies in the pure stochastic case where the batch size is set at 1 and 30 iterations with ADAMAX leading at 97.76% and 95.78% on the training and the testing set respectively. ADAGRAD trails the group at 96.98% and 93.37%. All algorithms record increased accuracy when the number of iterations is increased from 30 to 50 except ADAMAX that records a decrease on both the training and the test sets. ADAMAX maintains the lead for a batch size of 2 and 30 iterations at 97.81% and 95.78% on both the training and testing set while Adagrard trails at 96.93% and 93.22%. However, when the number of iterations increases to 50, NADAM overtakes ADAMAX closely followed by RMSPROP at 97.81% and 97.72% on the training set respectively but RMSPROP leads the pack for the testing set at 95.68%.

Table 2: Accuracy on predictions

TRAINING ALG.	NO. OF EPOC.	BATCH SIZE				DATASET
		TRAIN SET		TEST SET		
		1	2	1	2	
ADAM	30	97.68	97.55	95.31	95.07	% ACCURACY
	50	97.63	97.70	95.07	95.35	
ADAGRAD	30	97.78	97.77	95.75	95.71	
	50	97.28	97.25	94.43	94.32	
ADADELTA	30	97.72	97.72	95.31	95.07	
	50	96.95	96.89	95.15	95.35	
RMSPROP	30	97.55	97.55	95.40	95.36	
	50	97.22	97.20	94.64	94.60	
ADAMAX	30	97.24	96.93	93.77	93.23	
	50	97.28	97.07	94.39	93.89	
NADAM	30	97.78	97.75	95.42	95.43	
	50	97.75	97.72	95.36	95.26	

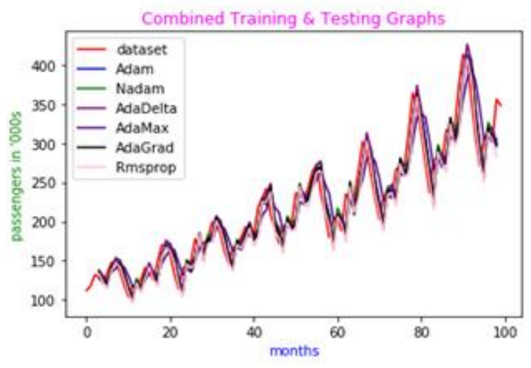

Figure 1. Comparison Graphs.

6. Conclusion

A number of conclusions were drawn from the results obtained this experiment for the current type of problem, the given dataset, the MLP neural network and selected parameter values:

1. ADAMAX is the most appropriate algorithm for the purely stochastic / online case where the parameter update is done for every training instance as observed from the accuracies when the batch size is 1.
2. On the contrary to ADAMAX, RMSPROP and NADAM are better suited when the number of parameter updates reduces per number of training instances as observed from their respective improved accuracies when the batch size increased to 2.
3. RMSPROP is the least affected by over-fitting when trained for longer periods as it out-performs the list on the test set.

References

[1] Bahar, P., Alkhouli, T., Thorsten, P., Brix, C., Ney, H. (2017). Empirical Investigation of Optimization Algorithms in Neural Machine Translation. The Prague Bulletin of Mathematical Linguistics. No. 108 pp 13–25

[2] Bengio Y., (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures, arXiv:1206.5533v2

[3] Bottou L., (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT' 2010, pp. 177–186.

[4] Bottou L., Curtis F. E., and Nocedal J., (2017). Optimization Methods for Large-Scale Machine Learning, arXiv:1606.04838v2 [stat.ML]

[5] Cortez, P., Rio, M., Rocha, M., Sousa, P. (2012). Multi-scale internet traffic forecasting using neural networks and time series methods. Expert Systems 29(2), 143-155

[6] John Gamboa. Deep Learning for Time-Series Analysis. arXiv:1701.01887v1 [cs.LG] 7 Jan 2017

[7] Kuremoto, T., Kimura, S., Kobayashi, K., Obayashi, M. (2014). Time series forecasting using a deep belief network with restricted boltzmann machines. Neurocomputing 137, 47-56

[8] LeCun Y., Bengio Y., and Hinton G., (2015). Deep learning: Nature, International journal of science, 521(7553):436–444

[9] Lin, C.J., Chen, H.F., Lee, T.S. (2011) Forecasting tourism demand using time series, artificial neural networks and multivariate adaptive regression splines: evidence from taiwan. International Journal of Business Administration Vol. 2(2), p14

[10] Liu, J.N., Hu, Y., He, Y., Chan, P.W., Lai, L. (2015). Deep neural network modeling for big data weather forecasting. In: Information Granularity, Big Data, and Computational Intelligence, pp. 389 - 408. Springer

[11] Matthew D. Zeiler. (2012). Adadelta: An Adaptive Learning Rate Method arXiv:1212.5701v1 [cs.LG]

[12] Romeu, P., Zamora-Martinez, F., Botella-Rocamora, P., Pardo, J. (2013). Time-series forecasting of indoor temperature using pre-trained deep neural networks. In: Artificial Neural Networks and Machine Learning. ICANN 2013, pp. 451- 458. Springer

[13] Schaul T. and Antonoglou L., (2017). Unit Tests for Stochastic Optimization. Deep Mind Technologies, arXiv:1312.60553

[14] Schmidhuber J., (2015). Deep learning in neural networks: An overview, Neural Networks, vol 61 pp. 85–117

[15] Simonyan K. and Zisserman A., (2014). Deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

[16] Vankadara L. C., (2015). Optimizing Deep Neural Networks.Retrievedfrom:https://dokumen.tips/documents/optimizing-deep-neural-networks.html

[17] Xiangyu Z., Zhiyong Z. and Wang D., (2016). Adamax Online Training for Speech Recognition. CSLT Technical Report-20150032

[18] Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L. (2014). Time series classification using multi-channels deep convolutional neural networks. In: Web-Age Information Management, pp. 298 - 310. Springer

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