A Study of the Optimization Algorithms in Deep Learning

Article · March 2020

DOI: 10.1109/ICISC44355.2019.9036442

1 author:

Humera Shaziya
Nizam College
36 PUBLICATIONS **216** CITATIONS

International Conference on Inventive Systems and Control (ICISC 2019)
IEEE Xplore Part Number: CFP19J06-ART; ISBN: 978-1-5386-3950-4

A Study of the Optimization Algorithms in Deep Learning

Raniah Zaheer
Lecturer
Department of CS
Najran University

Humera Shaziya
Assistant Professor
Informatics, Nizam College
Osmania University

***Abstract*—Training the deep learning models involves learning of the parameters to meet the objective function. Typically the objective is to minimize the loss incurred during the learning process. In a supervised mode of learning, a model is given the data samples and their respective outcomes. When a model generates an output, it compares it with the desired output and then takes the difference of generated and desired outputs and then attempts to bring the generated output close to the desired output. This is achieved through optimization algorithms. An optimization algorithm goes through several cycles until convergence to improve the accuracy of the model. There are several types of optimization methods developed to address the challenges associated with the learning process. Six of these have been taken up to be examined in this study to gain insights about their intricacies. The methods investigated are stochastic gradient descent, nesterov momentum, rmsprop, adam, adagrad, adadelta. Four datasets have been selected to perform the experiments which are mnist, fashionmnist, cifar10 and cifar100. The optimal training results obtained for mnist is 1.00 with RMSProp and adam at epoch 200, fashionmnist is 1.00 with rmsprop and adam at epoch 400, cifar10 is 1.00 with rmsprop at epoch 200, cifar100 is 1.00 with adam at epoch 100. The highest testing results are achieved with adam for mnist, fashionmnist, cifar10 and cifar100 are 0.9826, 0.9853, 0.9855, 0.9842 respectively. The analysis of results shows that adam optimization algorithm performs better than others at testing phase and rmsprop and adam at training phase.**

***Index Terms*—Optimization Algorithm, Stochastic Gradient Descent, Momentum, RMSprop, AdamOptimizer, MNIST, FashionMNIST, Cifar10, Cifar100**

I. Introduction

Deep learning has been in the forefront of solving quite a few real world problems. A machine is made to learn from the dataset and is expected to improve its performance over a period of time. When the input is given to the model a function applied on that and through a sequence of layers it is transformed into output value. Essentially the model then compares the generated output with the actual output and the difference is calculated. In order to reduce the difference the generated output is backpropagated into the model. The model adjusts the weights and follow the same process repeatedly until convergence. This leads to a quest of having an algorithm that accelerate the learning process and generate optimal output. In this direction several optimization algorithms have been developed and implemented on various tasks. This study examines the most widely used optimization algorithms and their impact on the learning process.

The paper is organized as follows: Section II presents related work on optimization algorithms. Section III discusses various optimization algorithms and the datasets used in this study. The various algorithms described are stochastic gradient descent, momentum, rmsprop, adam, adagrad and adadelta. The datasets examined are mnist, fashion mnist, cifar10 and cifar100. Section IV elaborates on the results obtained by training the model on the selected datasets with the chosen optimization algorithms and also focuses on the comparison of the results. Section V concludes the paper and provides future directions.

II. Related Work

Machine learning and deep learning relies on optimization algorithms to learn the parameters of the input data. Hence optimization algorithms play a vital role in the successful implementation of solutions to real world problems. Various studies have been conducted to determine the optimal algorithm for the problem at hand. Because there is no general method that solves all the different kinds of problems, investigations have to be carried out to figure out the method that works best for a given problem. [1] discusses gradient descent and its variants. The authors have reviewed various optimization algorithms for parallel and distributed environment. The paper also investigates ways to optimize the basic gradient descent. The discussion begins with introducing the three variants of gradient descent, batch gradient descent, stochastic gradient descent and mini-batch gradient descent. These differ in the number of data samples used for the training process at once. The basic gradient descent is the batch method that computes the gradients for the entire training samples. It is well suited for small to medium sized datasets. It does not scale due to the constraint to have the entire dataset in the memory. Also it does not consider the newly added data elements to the dataset on the fly for training. Thus batch method is slow and not appropriate for the large datasets. Stochastic gradient descent (SGD) performs gradient computation for each training sample. It also takes into consideration the data elements on the go and train them. It is relatively faster. However it suffers from the problem of slow convergence due to the fluctuations that occur when training individual data elements. Mini-batch combines the goodness of batch and stochastic methods and trains the dataset in mini-batches.

This method achieves the advantages of fast convergence and inclusion of online data (the data that arrives on the fly). There are few challenges that need to be addressed in all the types of gradient descent aforementioned. The major challenge is the selection of learning rate. Another is to have variable learning rate. The other is to avoid convergence at a suboptimal local minima. Further the paper outlines techniques that overcome these challenges. The techniques described are momentum, nesterov momentum, rmsprop, adam, adagrad, adadelta and nadam. The paper addresses the question of which method is to be employed for a given scenario. Furthermore it talks about the extending methods to be implemented on parallel and distributed environment. Improvisation of SGD through the use of shuffling, curriculum learning, batch normalization and early stopping is also presented. The paper [2] examines SGD and its impact on over-parameterized networks. It has been observed that SGD performs reasonably well in low parameters networks and leads to fast convergence and achieve optimal global minima. However it is worth investigating whether this observation generalize to a large massively parameterized networks. This is the work carried out by the authors wherein it is studied and proved through their experiments that SGD indeed yields satisfactory results in over-parameterized networks.

III. Optimization algorithms and Datasets

A. Optimization algorithms

Optimization algorithms form the basis on which a machine is able to learn through its experience. They compute gradients and attempts to minimize the loss function. There are several ways learning is implemented with different kinds of optimization algorithms. The algorithms studied in the present work are described below.

1) Stochastic Gradient Descent: The vanilla gradient descent trains the entire dataset together. Its variant is stochastic gradient descent [3] that performs the training on the individual data element.

2) Nesterov Momentum: Gradient is computed based on the approximate fututre positions of the parameters rather than the current parameters. Nesterov momemtum is an improvement over momentum which does not determine the future position of the parameters. [4] has incorporated nesterov momentum into adam.

3) Adagrad: This is an method that chooses the learning rate based on the situation. Learning rates are adaptive because the actual rate is determined from parameters. A high gradient for the parameters will have a reduced learning rate and parameters with small gradients will have increased learning rate.

4) Adadelta: It is an extension of adagrad. [5] presents a modification of adagrad into adadelta. Instead of accumulating the gradients, adadelta makes use of some fixed size window and tracks only the available gradients within the window.

5) RMSProp: RMSProp changes the adagrad in a way how the gradient is accumulated. Gradients are accumulated into an exponentially weighted average. RMSProp discards the history and maintains only recent gradient information. [6] discusses the rmsprop and its variants. The paper also examines adagrad with logarithmic regret bounds.

6) Adam: It has derived its name from ”adaptive moments”. It is a combination of rmsprop and momentum. The update operation considers only the smooth version of the gradient and also includes a bias correction mechanism. [7] discusses the adam algorithm.

B. Datasets

1) MNIST: A dataset of handwritten digits. It consists of 60,000 training examples and 10,000 test images. The dimensions of images are 28x28 and is of grayscale. The images contain handwritten digits from 0 through 9, total of 10 classes. It is a benchmark dataset for experimenting various algorithms and techniques. [8] describes the data.

2) FashionMNIST: A dataset of fashion related images. It consists of 60,000 training examples and 10,000 test images. The dimensions of images are 28x28 and is of grayscale. The images contain training and test items of t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneakers, bag and ankle boot. [9] has developed the fashion mnist dataset.

3) Cifar10: [10] presents the details of cifar10 dataset. It consists of 32x32 color images of airplane, automobile, bird, cat deer, dog, frog,horse, ship and truck. There are total 60,000 images and 6000 of each class. 10 classes are defined for the ten different types of images.

4) Cifar100: This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a ”fine” label (the class to which it belongs) and a ”coarse” label (the superclass to which it belongs). Here are the examples of superclasses in the CIFAR-100: aquatic mammals, fish, flowers Examples of classes are: beaver, dolphin, otter, seal, whale, aquarium fish, flatfish, ray, shark, trout, orchids, poppies, roses, sunflowers, tulips [10].

IV. Results and Discussions

A. Training the model

The training process begins by defining the model and then invoking fitgenerator method. The dataset is divided into batches. A batch is a set or group of data elements. A batch size denotes the total number of training elements in a particular batch. An optimization algorithm is used to iterate the training examples number of times to find the optimum results. Optimization algorithms used in this study are SGD, nesterov momentum, adagrad, adadelta, RMSprop and adam. Optimization algorithms operate in forward and backward manner. An epoch is one such pass of the entire dataset.

B. Model Evaluation

The model is evaluated using accuracy and crossentropy defined below
Accuracy is the correctly identified instances from the total

TABLE I: MNIST

Epoch	SGD	Momentum	Adagrad	Adadelta	RMSProp	Adam
Epoch 000	0.10	0.20	0.18	0.06	0.14	0.16
Epoch 100	0.54	0.90	0.84	0.04	0.96	0.98
Epoch 200	0.74	0.92	0.94	0.16	**1.00**	**1.00**
Epoch 300	0.86	**0.96**	0.94	0.18	1.00	1.00
Epoch 400	0.80	0.96	0.96	0.12	0.98	0.98
Epoch 500	0.82	0.94	0.96	0.28	0.98	0.92
Epoch 600	0.88	0.96	0.96	0.32	1.00	1.00
Epoch 700	0.86	0.92	0.94	**0.46**	1.00	1.00
Epoch 800	**0.96**	0.98	**1.00**	0.40	1.00	1.00
Epoch 900	0.88	0.88	0.96	0.44	1.00	1.00

TABLE II: FashionMNIST

Epoch	SGD	Momentum	Adagrad	Adadelta	RMSProp	Adam
Epoch 000	0.12	0.22	0.20	0.08	0.20	0.22
Epoch 100	0.56	0.86	0.80	0.16	0.90	0.96
Epoch 200	0.84	0.90	0.90	0.10	0.96	0.98
Epoch 300	0.98	0.94	0.92	0.10	0.98	0.98
Epoch 400	0.88	0.90	**0.96**	0.20	**1.00**	**1.00**
Epoch 500	**0.94**	0.96	0.86	0.18	0.98	0.96
Epoch 600	0.92	**1.00**	0.90	0.22	0.98	1.00
Epoch 700	0.94	0.98	0.96	0.32	1.00	0.98
Epoch 800	0.90	0.94	0.96	**0.38**	1.00	1.00
Epoch 900	0.96	0.94	0.94	0.36	1.00	1.00

TABLE III: Cifar10

Epoch	SGD	Momentum	Adagrad	Adadelta	RMSProp	Adam
Epoch 000	0.12	0.26	0.14	0.06	0.18	0.16
Epoch 100	0.80	0.88	0.92	0.18	0.98	0.96
Epoch 200	0.76	0.92	0.96	0.14	**1.00**	0.98
Epoch 300	0.84	0.94	0.90	0.18	1.00	0.96
Epoch 400	0.84	0.92	**0.98**	0.20	1.00	**1.00**
Epoch 500	**0.90**	0.94	0.92	**0.34**	1.00	0.94
Epoch 600	0.90	**0.98**	0.88	0.28	1.00	0.98
Epoch 700	0.88	0.98	0.94	0.26	1.00	0.94
Epoch 800	0.82	0.96	0.92	0.40	1.00	0.98
Epoch 900	0.90	0.96	0.96	0.34	1.00	0.98

TABLE IV: Cifar100

Epoch	SGD	Momentum	Adagrad	Adadelta	RMSProp	Adam
Epoch 000	0.12	0.08	0.22	0.12	0.10	0.36
Epoch 100	0.62	0.90	0.78	0.10	0.84	**1.00**
Epoch 200	0.72	**0.98**	0.88	0.08	**1.00**	0.98
Epoch 300	0.84	0.98	0.94	0.16	0.98	0.98
Epoch 400	0.80	0.92	0.82	0.16	1.00	0.98
Epoch 500	0.86	0.92	**0.96**	0.12	1.00	0.98
Epoch 600	0.84	0.96	0.88	0.16	0.98	0.98
Epoch 700	**0.90**	0.92	0.96	0.20	1.00	0.98
Epoch 800	0.94	0.96	0.96	0.30	1.00	0.94
Epoch 900	0.84	0.96	0.96	**0.32**	1.00	0.98

dataset. In other words it is the true predictions that model has achieved from the total predictions. It is given by the following equation

$$Accuracy = \frac{Correct\ Predictions\ Count}{Total\ Predictions}$$

Loss function determines how well the network is performing its intended task. Cross-entropy (CE) loss or log-loss is defined as the measure of the performance of the classifier.

$$Cross\ Entropy = -\sum_{i}^{c} a_i log(\ p_i)$$

where c is the no of classes, a_i is the actual value and p_i is the predicted value.

C. Results

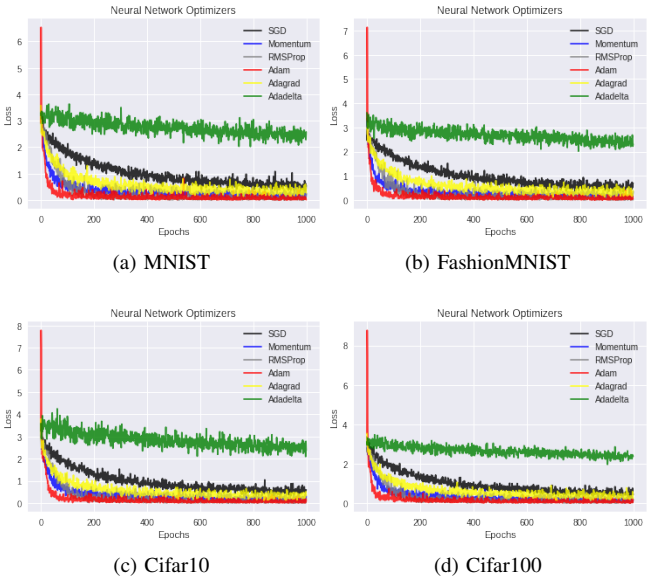
Table I shows the results obtained on mnist dataset for the 6 optimization algorithms. The first column gives the epoch number followed by different optimization algorithm. Table II shows the results obtained on fashionmnist dataset. Table III shows the results obtained on cifar10 dataset. Table IV shows the results obtained on cifar100 dataset. Figure 1 presents the tabular information in the forms of graphs.

D. Comparison of Results

The optimal training results obtained for mnist is 1.00 with RMSProp and adam at epoch 200, fashionmnist is 1.00 with rmsprop and adam at epoch 400, cifar10 is 1.00 with rmsprop at epoch 200, cifar100 is 1.00 with adam at epoch 100. The highest testing results are achieved with adam for mnist, fashionmnist, cifar10 and cifar100 are 0.9826, 0.9853, 0.9855, 0.9842 respectively. The analysis of results show that adam optimization algorithm performs better than others at testing phase and rmsprop at training phase.Table V shows the testing results obtained on mnist, fashionmnist, cifar10, cifar100 datasets for the SGD, momentum, adagrad, adadelta, rmsprop, adam optimization algorithms.

V. Conclusion and Future Work

The methods investigated are stochastic gradient descent, nesterov momentum, rmsprop, adam, adagrad, adadelta. The



(a) MNIST (b) FashionMNIST

(c) Cifar10 (d) Cifar100

Fig. 1: Various optimization algorithms on datasets

TABLE V: Comparison of Results

Optimization Algorithm	MNIST	FashionMNIST	Cifar10	Cifar100
SGD	0.9086	0.9108	0.9089	0.9151
Momentum	0.9663	0.9666	0.9643	0.9650
Adagrad	0.9394	0.9406	0.9386	0.9334
Adadelta	0.4524	0.4346	0.4422	0.3491
RMSProp	0.9823	0.9838	0.9773	0.9795
Adam	**0.9826**	**0.9853**	**0.9855**	**0.9842**

datasets selected to perform the experiments are mnist, fashionmnist, cifar10 and cifar100. This study of optimization algorithms have been conducted to gain insights into the intricacies of the different methods used with different datasets. 6 algorithms have been chosen for detailed experiments on 4 datasets. Training results show that rmsprop obtains 1.00 value quicker than other algorithms at the training stage with the cifar10 whereas adam generates the value 1.00 faster for cifar100. Both rmsprop and adam give 1.00 value at the same epoch for mnist, fashionmnist. The testing stage presents better results for all the datasets with adam algorithm. The values obtained are 0.9826, 0.9853, 0.9855, 0.9842 for mnist, fashionmnist, cifar10 and cifar100 respectively. Future work can be taken up to study optimization algorithms with varying parameters like learning rate, no of filters and so on. Optimization algorithms can be examined in different architectures of deep learning models.

REFERENCES

[1] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.
[2] A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz, "Sgd learns over-parameterized networks that provably generalize on linearly separable data," *arXiv preprint arXiv:1710.10174*, 2017.
[3] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*. IEEE, 1992, pp. 3–12.
[4] T. Dozat, "Incorporating nesterov momentum into adam," 2016.
[5] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.
[6] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," *arXiv preprint arXiv:1706.05507*, 2017.
[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[8] "Mnist handwritten digit database, yann lecun, corinna cortes and chris burges," http://yann.lecun.com/exdb/mnist/, (Accessed on 12/16/2018).
[9] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[10] "Cifar-10 and cifar-100 datasets," https://www.cs.toronto.edu/ kriz/cifar.html, (Accessed on 12/16/2018).