Workshop track -ICLR 2018 FX OPTIMIZER: SUCCESSOR TO ADAM OPTIMIZER USING BATCH WISE MOVING AVERAGES

Preprint · February 2018

DOI: 10.13140/RG.2.2.18932.99203

Fx Optimizer: Successor to ADAM Optimizer using batch wise moving averages.

Hemen Ashodia & Tanisha R Bhayani
F(x) Data Labs Pvt. Ltd.
hemen@htree.plus,t.bhayani@yahoo.com

Abstract

The random distribution of the input data creates an imbalance in learning parameters, this causes some inputs' patterns to be learned fast while others to be slow, and because of this, there is a subtle increase in the time the parameters takes to converge, this could be improved by storing the moving averages of the individual batches of the trainable variables. This lets the parameter update step to use the previous information about the same batch to update the parameters in the current batch. The paper presents an optimizer named Fx Optimizer, which is a first order optimizer, similar to ADAM Optimizer in the property but maintains the batch moving averages for each weight and converges faster and better than ADAM. This is shown experimentally on MNIST data-set. The intuition of the Optimizer comes from how the brain may take more or less time to learn different types of inputs from same class based on high or low complexity of input.

1 Introduction

First order stochastic optimization methods are mostly based on gradient based optimization methods with modifications in the update step, specifically the parameter update and parameter scaling. Momentum, simple gradient direction update, considering the mean and variance of the gradients to update the parameters, root mean square methods, are the popular methods for the first order optimization of the objective function. These methods target the problems encountered such as local minima, vanishing and exploding gradients, getting stuck in the plateau, ridge or ravine, especially for high dimensional data. The current methods work on making the learning rate all different for different parameters (ADAM), while maintaining the moving averages for the individual parameters.

The paper presents a first order stochastic optimizer named, Fx Optimizer which maintains the batch moving averages, which makes the individual batch inputs to have their own mean and variance for every parameter, just like ADAM does, but just now unrolling it to maintain the batch wise mean and variance. In this sense, the Fx Optimizer is a successor to ADAM Optimizer. Fx Optimizer is seen to have faster convergence than ADAM Optimizer, which achieves a very high accuracy and very less error on MNIST data-set.

2 Algorithm

Input: α - Learning Rate, β_1 - Coefficient for exponential decay of mean, β_2 - Coefficient for exponential decay of variance, γ - Coefficient for the exponential decay of the batch, N - Number of batches, $f(\theta)$ - Stochastic Loss Function, θ_{-1} - Initial Parameter weights. Tested optimum values are α = 0.001, β_1 = 0.9, β_2 = 0.999, γ = 0.8.
Remarks: β_1, β_2, $\gamma \in$ [0,1). gt^2 is the element wise multiplication of gradients. M,

and V are the arrays that store the exponential moving averages of the input batch.[1]

```
Result: Optimum values of $\theta$
Steps:
M[N] $\leftarrow$ 0 (Initialize $1^{st}$ moment array for each batch)
V[N] $\leftarrow$ 0 (Initialize $2^{nd}$ moment array for each batch)
t $\leftarrow$ 0
while $\theta_t$ not converged do
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    current_batch $\leftarrow t \bmod N$
    $m_t \leftarrow \beta_1 \cdot (\gamma \cdot M[current\_batch] + (1-\gamma) \cdot M[current\_batch - 1]) + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot (\gamma \cdot V[current\_batch] + (1-\gamma) \cdot V[current\_batch - 1]) + (1 - \beta_2) \cdot g_t^2$
    M[current_batch] $\leftarrow$ m$_t$
    V[current_batch] $\leftarrow$ v$_t$
    $\alpha \leftarrow \alpha \cdot \sqrt{1 - \beta_2^t}/(1 - \beta_1^t)$
    $\delta \leftarrow \alpha \cdot m_t/\sqrt{v_t}$
    $\theta_t \leftarrow \theta_{t-1} - \delta$
    t $\leftarrow$ t + 1
end
```

Algorithm 1: Fx Optimizer Algorithm

Here, M[N] and V[N] are the mean and variance batch wise, which maintains the input data distribution. The update step takes into consideration the last epoch batch moment more and the last batch of present epoch less. This lets the optimizer to update the parameters by considering the state of the input batch in its last epoch, so in a sense the optimizer is a stateful optimizer maintaining the moments of the batch. It remembers the state of the moments of the input batch, which contains in itself the decayed averages of its adjacent moments and the moment in its last epoch making it to decay the information slowly than ADAM.

The learning rate - α is updated the same way in the ADAM, but changing just a simple step in ADAM makes the algorithm converge faster, with everything being same between the optimizers. Storing more information about the batches and its respective gradients in a form of exponential moving averages, lets the algorithm move to another sub-optimal point, which is better than ADAM.

3 Problem Solved and Not Solved by the Optimizer

3.1 Problem solved by the optimizer

Fx Optimizer basically maintains the moving averages batch wise, which could be concluded input wise, so that every input particle having its own place in the cost curvature, moves to the global minima effectively and faster than ADAM Optimizer. The problem addressed is of time, because of some tricky input values, the optimizers take a little more time to converge to the global minima, which is a loss of computational resources. But the optimizer presented here requires a trade-off for time and space. That is Fx Optimizer requires space to store the batch-wise means and variances.

3.2 Problem not solved by the optimizer

The main drawback of ADAM, is that it fails to reach a global optimum, since it maintains the moving averages of the variables. Since moving averages maintain a frame (window) that makes them to forget the sufficiently old information. Since Fx is essentially a successor of ADAM, it could find a global minima, but it isnt guaranteed, since in its very basic form, Fx maintains the moving averages of the input mini-batch, but it does record uptil a frame, and maximum to the last epoch, which makes it to lose the old information but it still stores more information than ADAM. So, Fx could do better than ADAM, but it is not guaranteed that it will reach a global optimum.

[1] It is assumed that the array could be negatively indexed.

4 Experiment and Analysis

The optimizer is tested on MNIST data-set for Convolution Layer and Fully Connected Architecture. The Fx is compared against ADAM and Gradient Descent Optimizer. The tensorboard diagram below shows for the first 3 epochs with the batch-size of 100. TensorFlow implementation of ADAM, Gradient Descent and Fx Optimizer is used to test and compare the optimizers.

The CNN architecture used is two alternate Convolution and Max Pool Layer, with 32 and 64 total kernels with 5X5 kernel size. The network uses dropout layer that is comparable to the one used in the literature. Whereas, for the fully connected architecture, 2 hidden layers, with 1000 neurons each and the softmax output layer with 10 classes.

4.1 Model Training Result

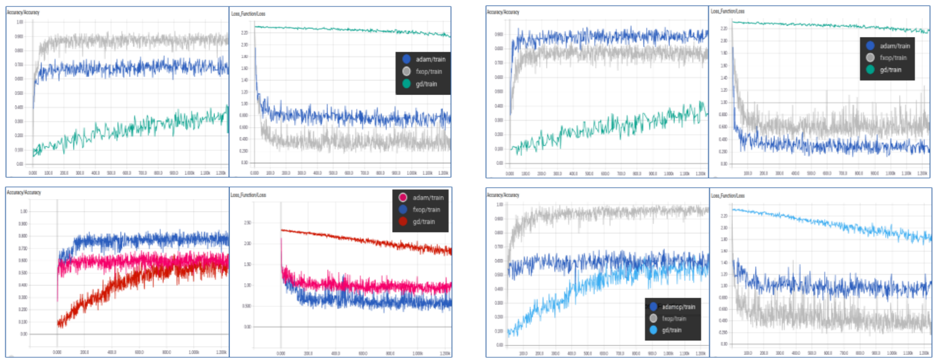

Figure 1: CNN and Fully Connected Architecture Results when (upper diagram - a,b) (a) Data Shuffling is turned off and (b) when data shuffling is turned on (lower diagram -c,d) (c) when data shuffling is turned off and (d) when data shuffling is turned on.

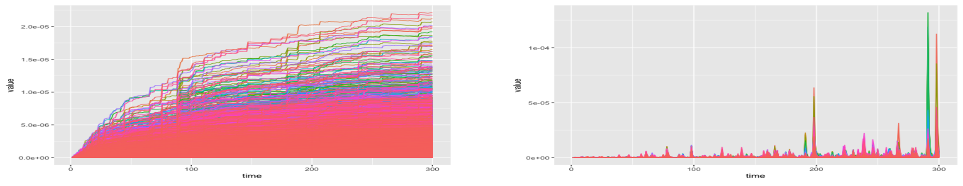

Figure 2: Plot of batch-wise variance for first convolution layer weight - (a) ADAM Optimizer, and (b) Fx Optimizer

5 Insights

From the model training graph, in increasing order Gradient Descent, Fx, and then ADAM has effects of data shuffling. It shows that storing batch-wise moving averages makes the optimizer robust against the data shuffling. For Fully Connected architecture Fx performs well for when data is shuffled on and tries to utilize the network's full capacity, while ADAM in these cases is not able to do so. The ADAM and Fx have almost similar start, this is because, the array in the beginning has the same values as the ADAM, but then it changes when the array gets populated with the update rule of Fx Optimizer, this is straightforward from the algorithm.

The graph for the variance of the fully connected layer shows how variance for 1^{st} convolution layer changes, the variance in fig. (b) shows that Fx Optimizer for first epoch maintains low variance, while after first epoch it makes a jump at epoch change, otherwise it stays very stable.

REFERENCES

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Sanjiv Kumar Sashank J. Reddi, Satyen Kale. On the convergence of adam and beyond. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ryQu7f-RZ`.

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.