Comparation of stochastic gradient descent algorithm variations for neural network training

Miroslav Smelko

VSB-TUO

# Abstract

This paper compares several algorithms for augmenting stochastic gradient descent algorithm commonly used in neural network (NN) training. I use neural network for classification of MNIST dataset with following algorithms and compare cost function optimization using several popular algorithms. Tested algorithms are: stochastic gradient descent (SGD), GD with momentum, AdaGrad, RMSProp and Adam.

Keywords: NN, stochastic gradient descent, SGD, GD with momentum, AdaGrad, RMSProp, Adam

# Introduction

This paper compares various gradient descent algorithm variations. The choice of algorithm might have significant impact on the results of training neural network and we can observe how those algorithms stack against each other using several values of learning rate. We will describe some common algorithms and their respective hyperparameters. Those algorithms will be put into the test and compare how well they do optimizing simple feedforward neural network.

# Testing

For the testing purposes we created a neural network for classifying [MNIST dataset of handwritten digits](http://yann.lecun.com/exdb/mnist/). We created the neural network consisting of 2 dense layers of 255 neurons and one output layer of 10 nodes using SoftMax activation function. Each algorithm was tested using several learning rates ranging from 0.5 to 0.001. Other hyperparameters for the algorithms will be discussed in section dedicated to that algorithm. Source code is available on <https://github.com/mirosmelko/Gradient-Descent-algirithms-comparison>

# Tested algorithms

## Gradient descent

Stochastic gradient descent (SGD) has become crucial to modern machine learning. SGD optimizes a function by following noisy gradients with a decreasing step size. The classical result of Robbins and Monro (1951) is that this procedure provably reaches the optimum of the function (or local optimum, when it is nonconvex) (1)

This algorithm is considered a base algorithm for other algorithm variations. It requires only one hyper parameter – learning rate.

## AdaGrad

AdaGrad algorithm was presented by John Duchi, Elad Hazan and Yoram Singer in 2011.(2) “*AdaGrad* (for adaptive gradient algorithm) is a modified stochastic gradient descent with per-parameter learning rate, first published in 2011. Informally, this increases the learning rate for more sparse parameters and decreases the learning rate for less sparse ones. This strategy often improves convergence performance over standard stochastic gradient descent in settings where data is sparse and sparse parameters are more informative. Examples of such applications include natural language processing and image recognition.It still has a base learning rate *η*, but this is multiplied with the elements of a vector {*Gj*,*j*} which is the diagonal of the outer product matrix.”(6)

## Gradient descent with momentum

“The momentum method (Polyak, 1964), which we refer to as classical momentum (CM), is a technique for accelerating gradient descent that accumulates a velocity vector in directions of persistent reduction in the objective across iterations. Given an objective function f(θ) to be minimized, classical momentum is given by:

vt+1 = µvt − ε∇f(θt)

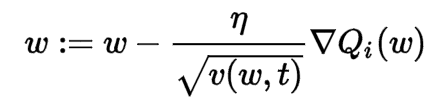
θt+1 = θt + vt+1

where ε > 0 is the learning rate, µ ∈ [0, 1] is the momentum coefficient, and ∇f(θt) is the gradient at θt.”(3) We are going to use classical momentum for the test with value of momentum set to 0.9.

## RMS prop

## “RMSProp (for Root Mean Square Propagation) is also a method in which the learning rate is adapted for each of the parameters. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. So, first the running average is calculated in terms of means square,

where, γ is the forgetting factor. And the parameters are updated as,

”(6) We will use various learning rates – as with other algorithms and we set γ = 0.9 for our testing.

## Adam

“The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name Adam is derived from adaptive moment estimation. Our method is designed to combine the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp (Tieleman & Hinton, 2012)”(5)As the Adam is combinations of RMSProp and AdaGrad, we will use the same value of hyperparameters as with those algorithms: γ = 0.9 and momentum=0.9.

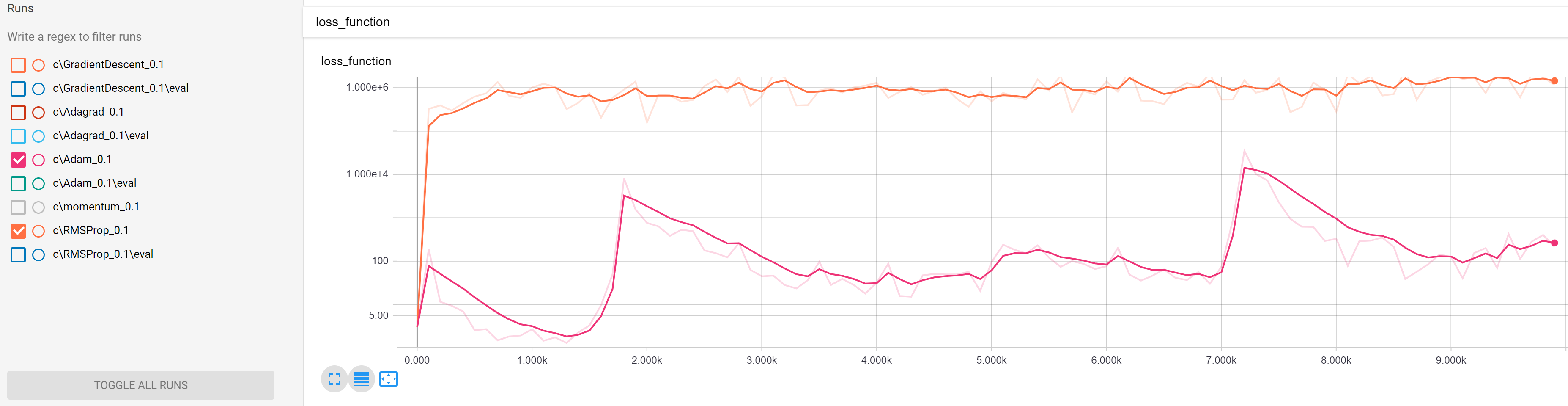
# Test results

## Learning rate = 0.5

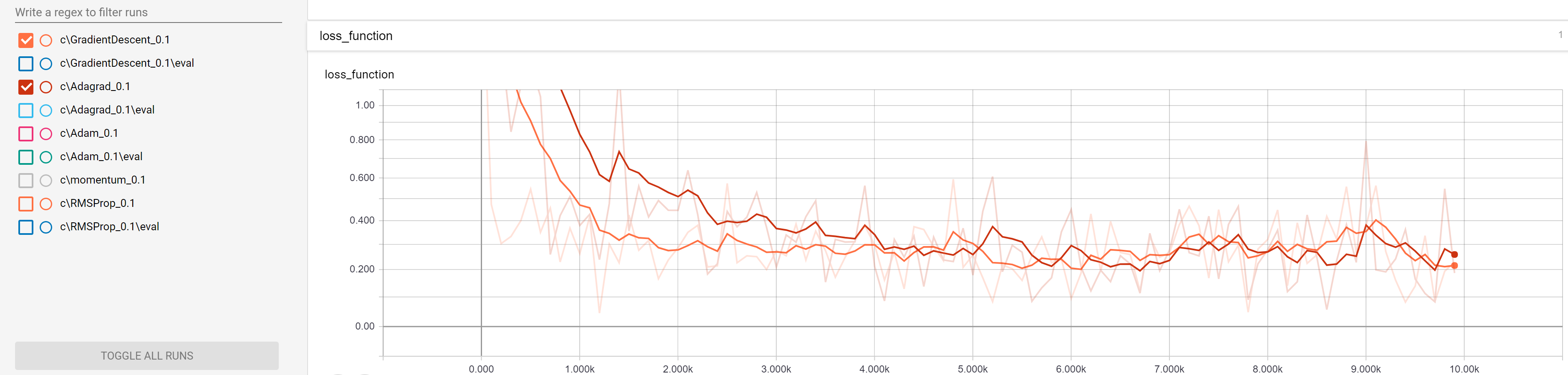
With the learning rate this high, all the algorithms failed to learn anything. Algorithms failed to minimize loss function and ended prematurely.

## Learning rate = 0.1

Adam, GD with momentum and RMS prop failed to converge. From the picture below, you can see that even though Adam failed, it was converging for some iterations, but after some runs the loss function value increased significantly.We can see those spikes in the picture at arround iterations 1500 and 7000. Then the function continued minimizing only to go through the increase again. After 10000 iterations Adam failed to converge.

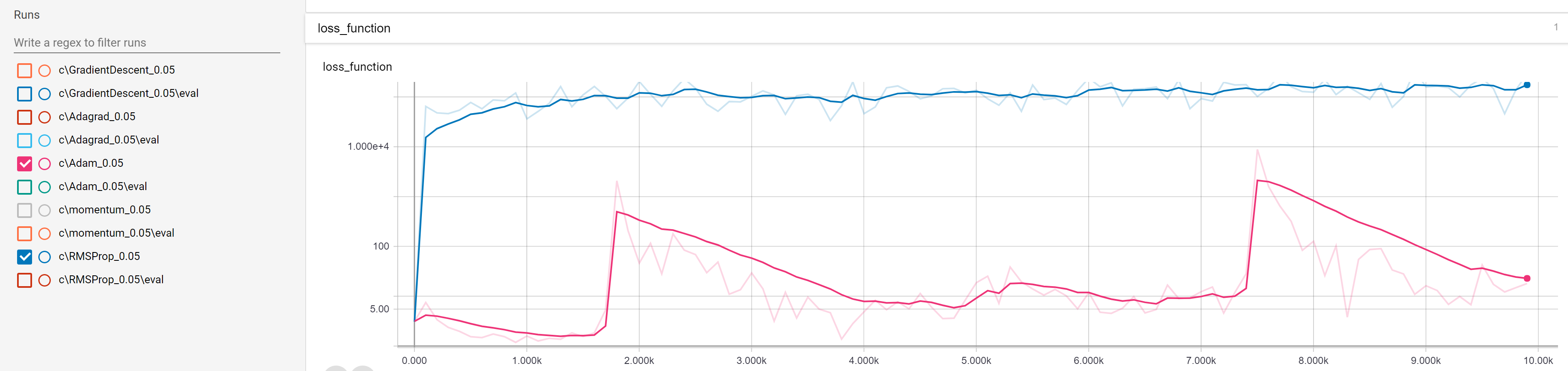


When it comes to GD and AdaGrad, we can see that they both converged, but the loss function optimization is noisy and the result is far from optimal. Both algorithms might come close to optimum for a brief period, but then they went away from that.

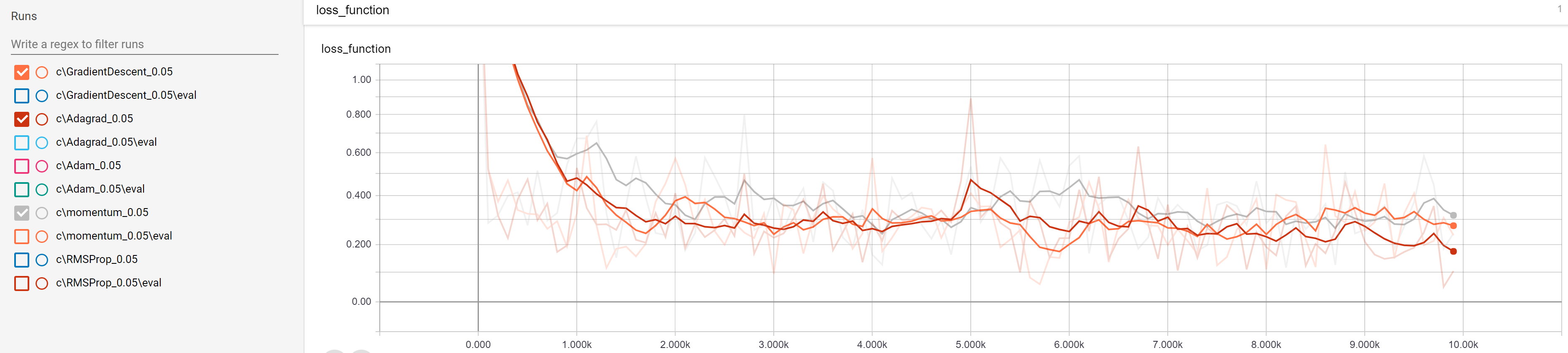


## Learning rate = 0.05

When we lowered the learning rate to 0.05, the situation is almost the same as 0.1. Adam and RMS prop failed to converge. Even those spikes in the graph, when Adam went up after a period of converging are present.

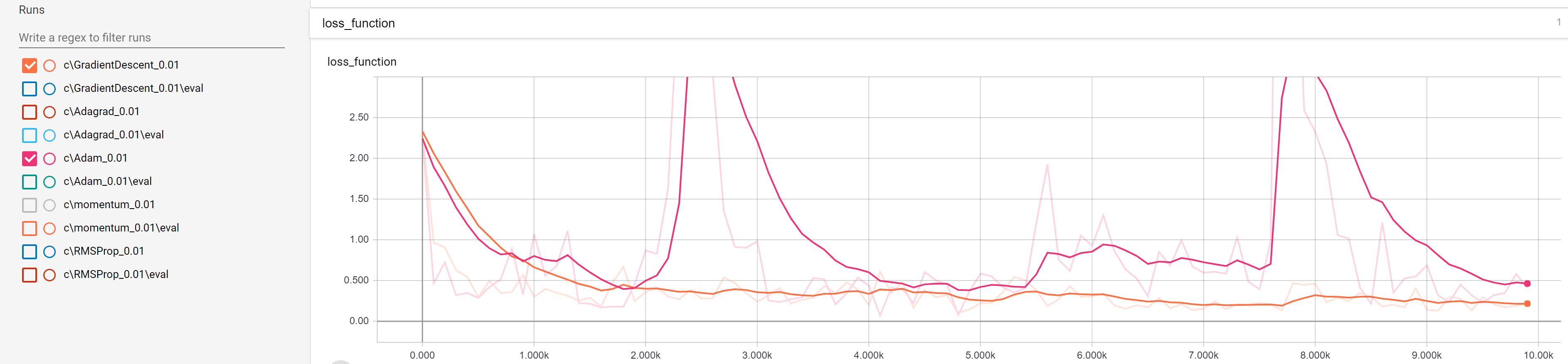


GD, GD with momentum and AdaGrad have managed to minimize loss function but the noise is still present and all the functions are oscillating around minimum and due to high learning rate, they can’t descent closer to the minimum. Based on the moment when the training is over, their final results may vary significantly.

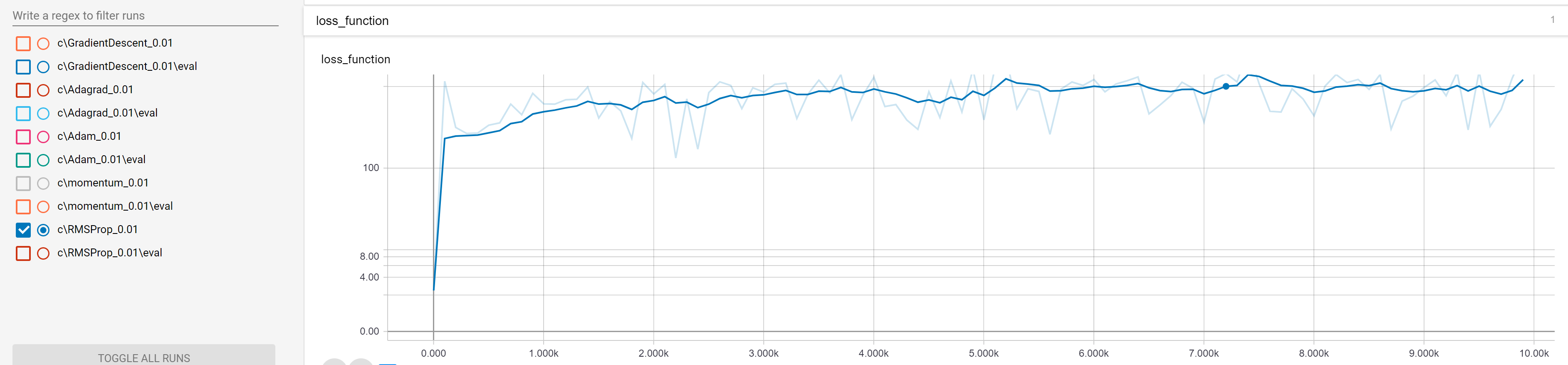


## Learning rate = 0.01

Lowering learning rate to 0.01 had positive effects on Adam algorithm. Adam manages to minimize function with similar accuracy to GD. The problem with it suddenly go straight up persist.

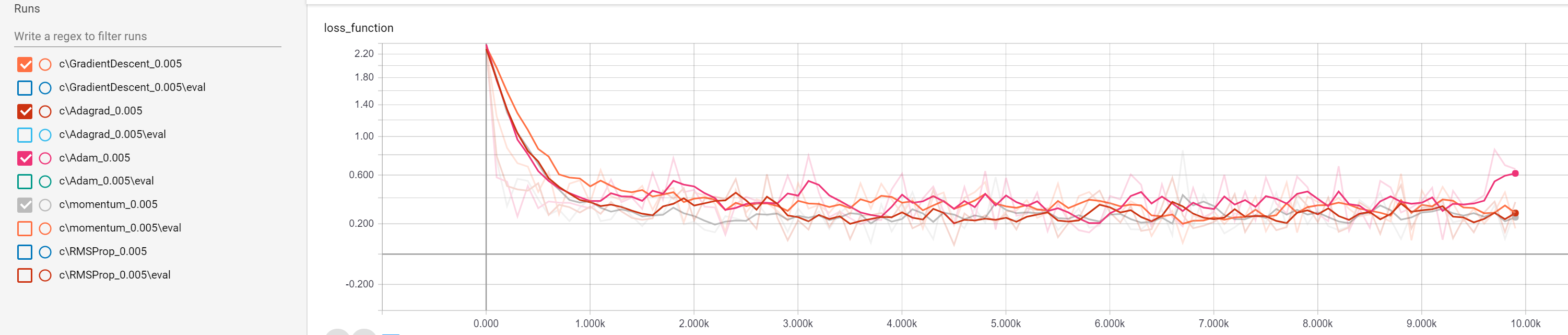


Other algorithms do as well as GD. With an exception of RMSProp which again failed to converge to minimum. The values are lower as with higher learning rate, but the results are still disappointing and neural network learned nothing using RMSProp.

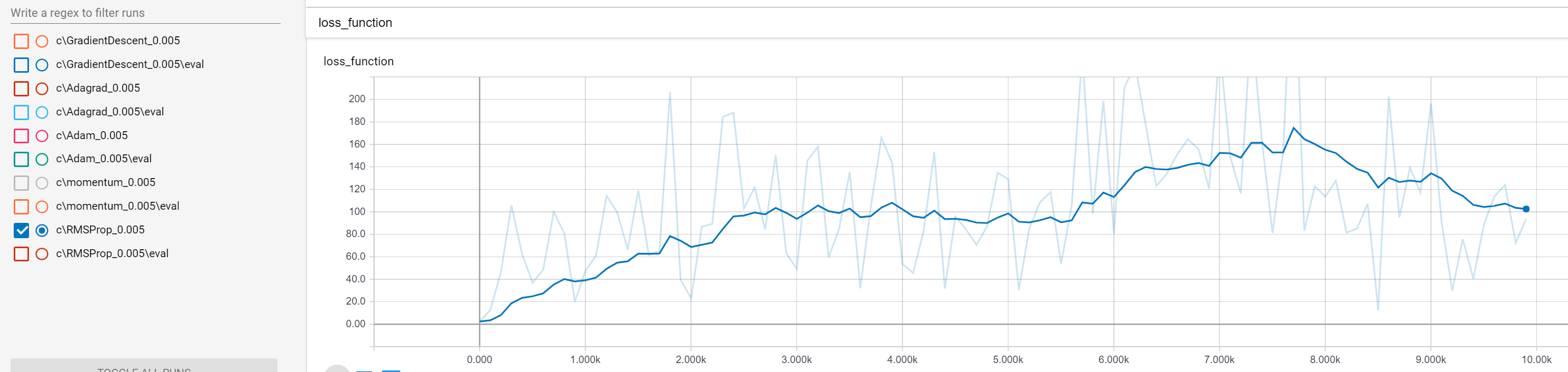


## Learning rate = 0.005

With learning rate 0.005 all the algorithm descended near the global minimum. In this run, the Adam algorithm ended up having worst result in terms of accuracy.

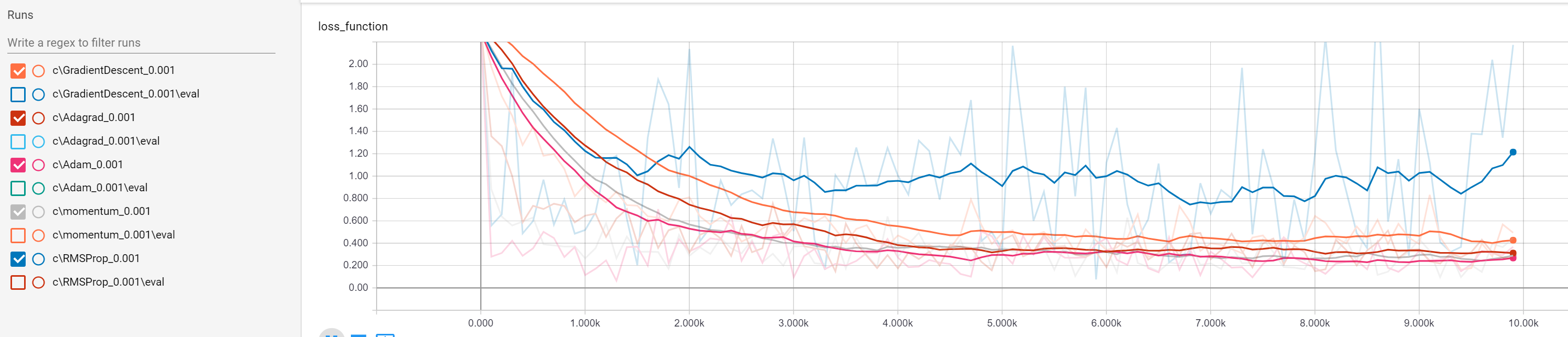


There’s one exception though - RMS Prop. We can see, that the results are getting better, but algorithm is still performing poorly.

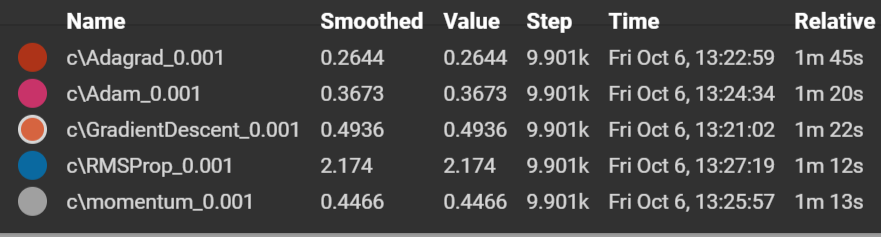


## Learning rate = 0.001

With the lowest tested learning rate, all the algorithms have trained the neural network. We can see, that the RMS Prop had worst results in terms of absolute final value. The other algorithms yielded similar results.

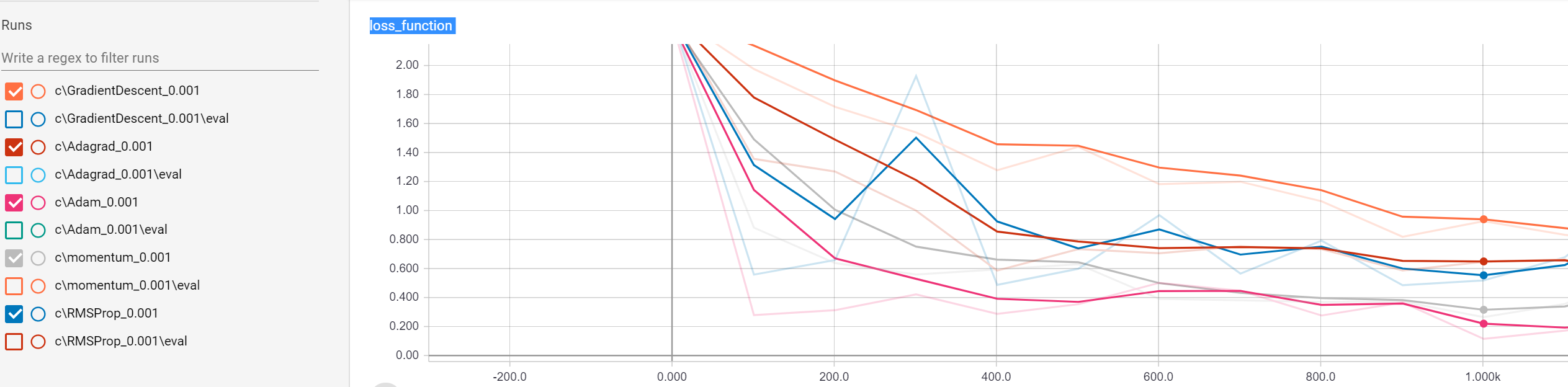


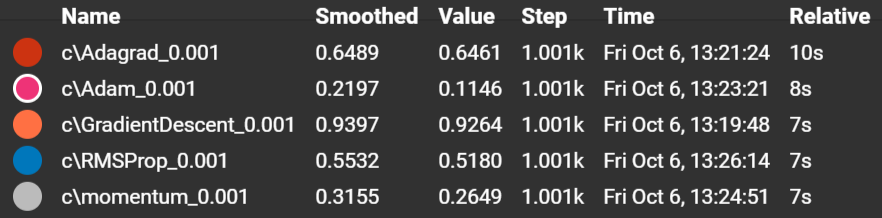
The conclusive results are:



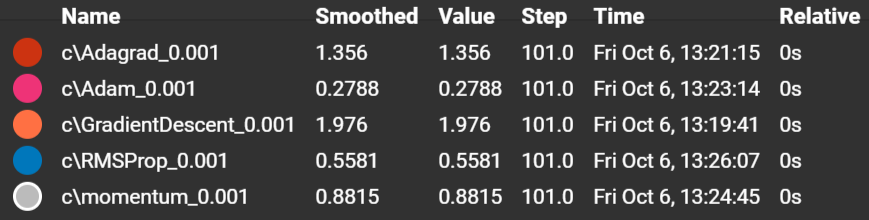
In terms of run speed, the quickest algorithms were RMS prop and GD with momentum, followed by Gradient descent and Adam. The slowest algorithm in our test run was AdaGrad.

Let’s have a closer look at first 1000 iterations:





From the graph in the top, we can see that the Adam performed exceptionally in terms of speed of descending. When we have a look at the loss function after just 100 iterations, we can see that the Adam performed superior to other algorithms.



# Conclusions

We compared best known algorithms for the optimizing loss functions. The gradient descent algorithm produces good and reliable results with minimum hyperparameters. And with AdaGrad are best suited for the tasks when the reliability and ease of use are among our top parameters. The Adam algorithm had best results in our test when it comes to performance. It optimizes loss function in lowest number of iterations, but with higher learning rate its reliability starts to suffer. We also discovered that spikes in the graph of loss function optimization of Adam algorithm can be eliminated by lowering learning rate. The poorest performing algorithm was RMS prop. This might be due to our hyperparameter choice or a bug in the code.

References

1. Stephan Mandt, Matthew D. Hoffman, David M. Blei (2017) - Stochastic Gradient Descent as Approximate Bayesian Inference - <https://arxiv.org/abs/1704.04289>
2. Duchi, John; Hazan, Elad; Singer, Yoram (2011). "Adaptive subgradient methods for online learning and stochastic optimization"
3. Sutskever, Ilya; Martens, James; Dahl, George; Hinton, Geoffrey E. (June 2013). On the importance of initialization and momentum in deep learning.
4. Tieleman, Tijmen and Hinton, Geoffrey (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning
5. Diederik, Kingma; Ba, Jimmy (2014). "Adam: A method for stochastic optimization". arXiv:1412.6980
6. Wikipedia article on Stochastic Gradient Descent - <https://en.wikipedia.org/wiki/Stochastic_gradient_descent>

Footnotes

1Add footnotes, if any, on their own page following references. The body of a footnote, such as this example, uses the Normal text style. (Note: If you delete this sample footnote, don’t forget to delete its in-text reference as well. That’s at the end of the sample Heading 2 paragraph on the first page of body content in this template.)

Tables

Table 1

Table Title

| Column Head | Column Head | Column Head | Column Head | Column Head |
| --- | --- | --- | --- | --- |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |

Note: Place all tables for your paper in a tables section, following references (and, if applicable, footnotes). Start a new page for each table, include a table number and table title for each, as shown on this page. All explanatory text appears in a table note that follows the table, such as this one. Use the Table/Figure style, available on the Home tab, in the Styles gallery, to get the spacing between table and note. Tables in APA format can use single or 1.5-line spacing. Include a heading for every row and column, even if the content seems obvious. A table style has been setup for this template that fits APA guidelines. To insert a table, on the Insert tab, click Table.

Figures



Figure 1. Include all figures in their own section, following references (and footnotes and tables, if applicable). Include a numbered caption for each figure. Use the Table/Figure style for easy spacing between figure and caption.

For more information about all elements of APA formatting, please consult the APA Style Manual, 6th Edition.