Indian Institute of Technology, Indore

Adam Optimiser

Project Report (CS 357)

Department of Computer Science and Engineering

Submitted by –

Keshav Goyal (150001014) Dhruv Chadha (150001009)

Under the Guidance of **Dr. Kapil Ahuja**

Introduction -

Adam Optimiser is one of the fastest iterative optimisation algorithms, used in many areas, especially machine learning. Over the past 50 years, there has been extensive research in this field, and numerous algorithms have been proposed. All the algorithms improve upon the previous algorithms, and Adam has been a result of many of these improvements. In this report, we explore the various algorithms, whose evolution led to development of Adam optimiser.

Evolution of Iterative Optimisation Algorithms till Adam Optimiser -

1. **Gradient Descent**
   It is a first order iterative method for finding the minimum of a function. It is based on the fact that the function value decreases in the opposite direction of the gradient at a particular point, if it is not a local minimizer.

   UPDATE RULE:

$$X_{t+1} = X_t - \alpha\left(\nabla f(X_t)\right)$$

   Challenges and Drawbacks –
   a. Choosing a proper step size- if too small, slow convergence; if too large, may cause function to fluctuate around minimum or even diverge.
   b. Same step size for update of all components of X.
   c. Same alpha for all iterations, unable to adapt to dataset characteristics.

2. **Newton's Method**
   It is a second order iterative method for finding the minimum of a function. In this method of optimization, the objective function is approximated by a quadratic function around X_n, and then a step is taken towards the minimum of that quadratic function.

   UPDATE RULE:

$$X_{n+1} = X_n - \alpha\left[H\left(f(X_n)\right)\right]^{-1}\nabla f(X_n)$$

   Challenges and Drawbacks -
   a. Newtons method is infeasible to compute in high dimensional data sets, as computation of the Hessian matrix is expensive.
   b. All the drawbacks of gradient descent method hold here as well.

3. **Adding Momentum** (1986)

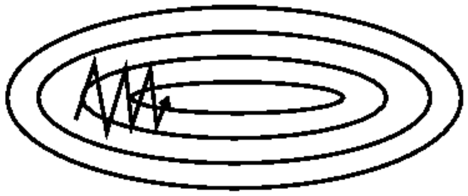
Gradient descent has trouble navigating ravines, i.e. areas where surface curves more steeply in one dimension than in the other. In these scenarios, GD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.

Adding momentum accelerates GD in the relevant direction and dampens oscillation. It is done by adding a fraction of the update vector of the past time step to the current update vector.

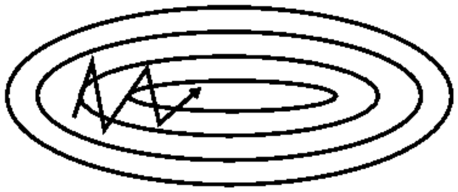
<u>UPDATE RULE:</u>

$$V_t = \gamma V_{t-1} + \alpha\big(\nabla f(X)\big)$$
$$X = X - V_t$$

The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.



Gradient Descent without momentum



Gradient Descent with momentum

Challenges and Drawbacks-

a. In this method, we first compute the gradient at current location, then take a big jump in the direction of the updated accumulated gradient. So, it does not know when to slow down, and can overshoot the minimum.

4. **Nesterov accelerated gradient**

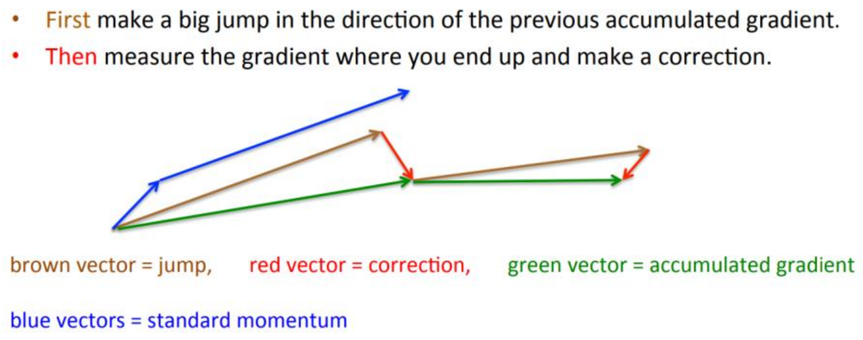
Nesterov momentum has a kind of prescience. It has a notion of where it is going and knows to slow down before the hill slopes up again.

First, it makes a big jump in the direction of the previous accumulated gradient. Then it measures the gradient where it ends up, and makes a correction.

This corrective update prevents X_n from going too fast and overshooting the minimum.

<u>UPDATE RULE:</u>

$$V_t = \gamma V_{t-1} + \alpha\big(\nabla f(X - \gamma V_{t-1})\big)$$
$$X = X - V_t$$



- First make a big jump in the direction of the previous accumulated gradient.
- Then measure the gradient where you end up and make a correction.

brown vector = jump,
red vector = correction,
green vector = accumulated gradient
blue vectors = standard momentum

Challenges and Drawbacks -

a. Up until now, we have adapted the direction to the slope of the objective function and sped up the descent. However, we would also like to adapt our step size to each individual component X_i.

5. **Adagrad** (2011)

It adapts step size rate to the components, performing larger updates for components in whose direction, the gradient is less steep, and smaller updates for those components in whose direction, the gradient is steeper. This prevents unnecessary oscillations in the steeper direction, and moves faster in the less steep direction.
It also uses different step size for every component X_i.

<u>UPDATE RULE:</u>

$$X_i^{(t+1)} = X_i^{(t)} - \left(\frac{\alpha}{\sqrt{\left(\sum_{\tau=1}^{t} g_{\tau,i}^2\right) + \varepsilon}} \right) g_{t,i}$$

Challenges and Drawbacks -

a. Adagrad's main weakness is its accumulation of the squared gradients in the denominator. Since every added term is positive, the accumulated sum keeps growing during training. This in turn causes the step size to shrink and eventually become infinitesimally small, at which point the algorithm is no longer able to acquire additional knowledge.

6. **RMSprop** (2012)
Restricts window of accumulated past gradients to some fixed size, by using exponential averaging.

$$R_{t,i} = \gamma R_{t-1,i} + (1-\gamma) g_{t,i}^2$$

By the above formula, R approximately becomes the average of last $\frac{1}{1-\gamma}$ square gradients.

<u>UPDATE RULE:</u>

$$X_i^{(t+1)} = X_i^{(t)} - \left(\frac{\alpha}{\sqrt{R_{t,i} + \varepsilon}}\right) g_{t,i}$$

7. **Adam** (2014)
This algorithm also computes adaptive step sizes for each X_i. In addition to dividing the step size by the decaying average of past square gradients like RMSprop, Adam also replaces the simple gradient term by an exponentially decaying average of past gradients, thus incorporating momentum.

$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

As m_t and v_t are initialized to 0 vector, their subsequent values are biased towards 0, especially during the initial time steps, and especially when the decay rates are small. (β_1 and β_2 are close to 1). Performing bias correction -

<u>UPDATE RULE:</u>

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

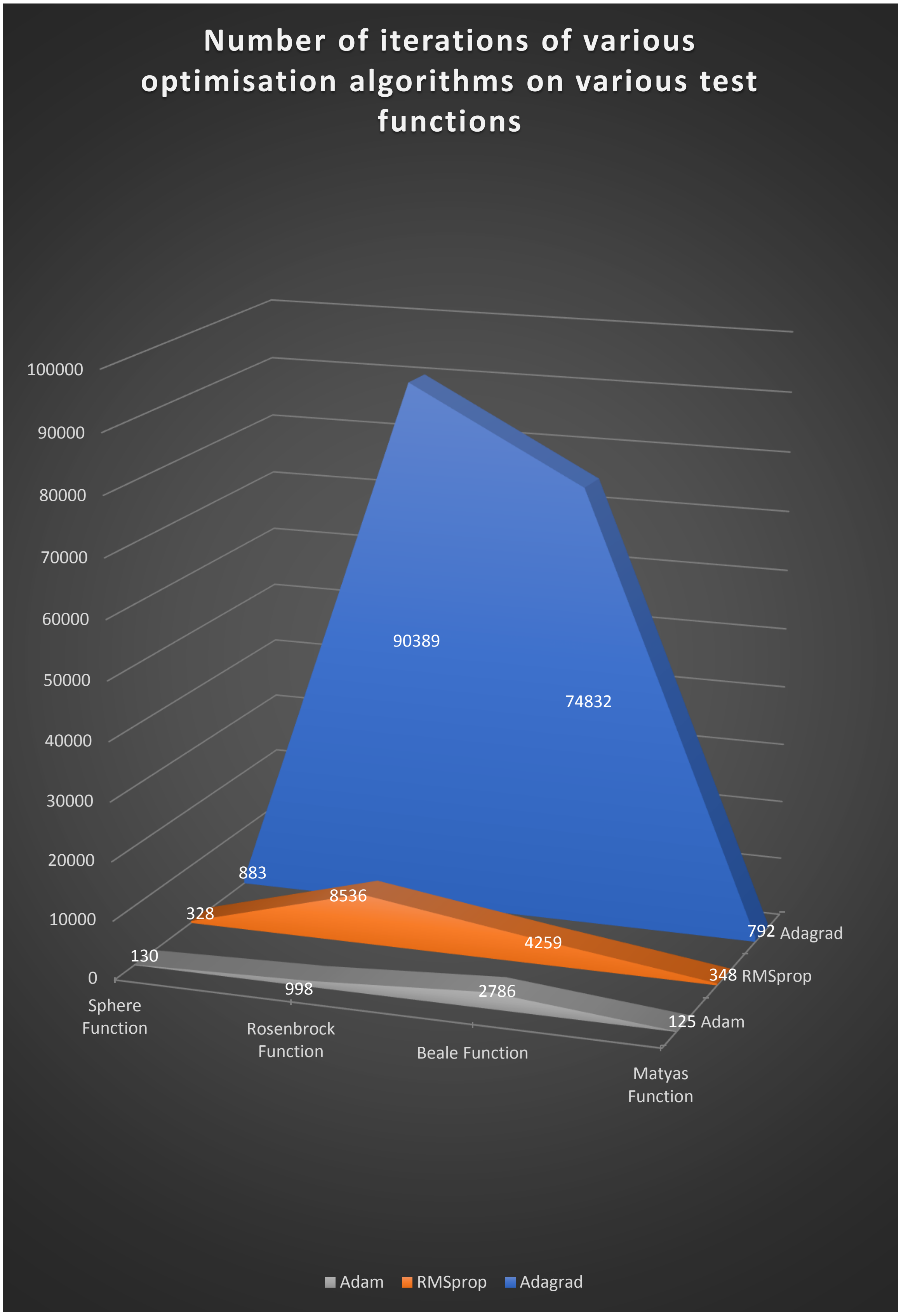
$$\Theta_{t+1} = \Theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \widehat{m}_t$$

Adam works well in practice and compares favorably to other adaptive optimisation algorithms.

Testing -

We test Adagrad, RMSprop and Adam algorithms on various test functions like –

1. Sphere Function –
$$f(x_1, x_2) = x_1^2 + x_2^2$$
2. Rosenbrock Function –
$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$
3. Beale Function –
$$f(x_1, x_2) = (1.5 - x_1 - x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$
4. Matyas Function –
$$f(x_1, x_2) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$$


Number of iterations of various optimisation algorithms on various test functions
100000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
90389
74832
883
792
8536
328
4259
348
130
998
2786
125
Sphere Function
Rosenbrock Function
Beale Function
Matyas Function
Adagrad
RMSprop
Adam
Adam
RMSprop
Adagrad

Conclusion –

On plotting the relative iterations required to find the minimizer, we observe that the trend is as follows –

Adagrad < RMSprop < Adam

In summary, RMSprop deals with the radically diminishing learning rates of Adagrad. Adam then incorporates momentum in addition to dividing by decaying average of past gradients strategy of RMSprop. Therefore, Adam is the fastest iterative optimisation algorithm amongst the discussed algorithms.

References –

1. Adam: A Method for Stochastic Optimisation

   https://arxiv.org/abs/1412.6980

2. An overview of gradient descent optimization algorithms

   https://arxiv.org/abs/1609.04747

3. Coursera
   a. Lecture 29, Neural Networks for Machine Learning, By Geoffrey Hinton

      https://goo.gl/CjSkLj

4. https://math.stackexchange.com

   https://goo.gl/hsMaVQ

5. Wikipedia
6. YouTube
   a. Andrew Ng's deeplearning.ai playlist – https://goo.gl/NZA4wJ
   b. University of Oxford Lecture – https://goo.gl/zs2f7a
   c. Siraj Raval – Evolution of Gradient Descent - https://goo.gl/TJSEiA