Indian Institute of Technology, Indore

Adam Optimiser

Project Report (CS 357)

Department of Computer Science and Engineering

Submitted by –

Shashank Giri -160001054   Ashutosh Bang -160001011

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

Motivation-

The main motive behind this project is to analyse stochastic gradient descent for unconstrained optimisation and analyse ways to speed up the descent method by introducing knowledge about the function topology.

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.

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:

   $$X_{n+1} = X_n - \alpha(\nabla f(X_n))$$

   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 and also alpha has to be tuned for each iteration.

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:

   $$X_{n+1} = X_n - \alpha(H(f(x))^{-1}\nabla f(X_n)$$

Challenges and Drawbacks -

a. Computation of inverse of Hessian is expensive as dimensional space increases .
b. All the drawbacks of gradient descent method hold here as well.

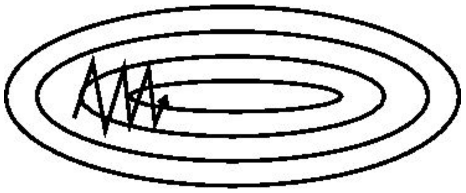
Adding Momentum (1986)

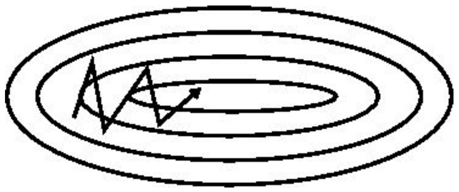
Gradient descent has trouble navigating ravines, i.e. areas where surface curves more steeply in one dimension than in the other. In these scenarios, GD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.
Adding momentum accelerates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

UPDATE RULE:

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

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-

It does not know when to slow down, and can overshoot the minimum.

3. **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.

UPDATE RULE:

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

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.

4. **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.

UPDATE RULE:

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

Implemented Code

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.

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

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

UPDATE RULE:

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

Implemented code

6. **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.

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

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 -

UPDATE RULE:

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

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$

Implemented Code

An additional advantage of this technique is that we do not have to manually tune value of alpha.
An appropriate value is chosen at first and then the value of alpha is kept constant.

Testing -

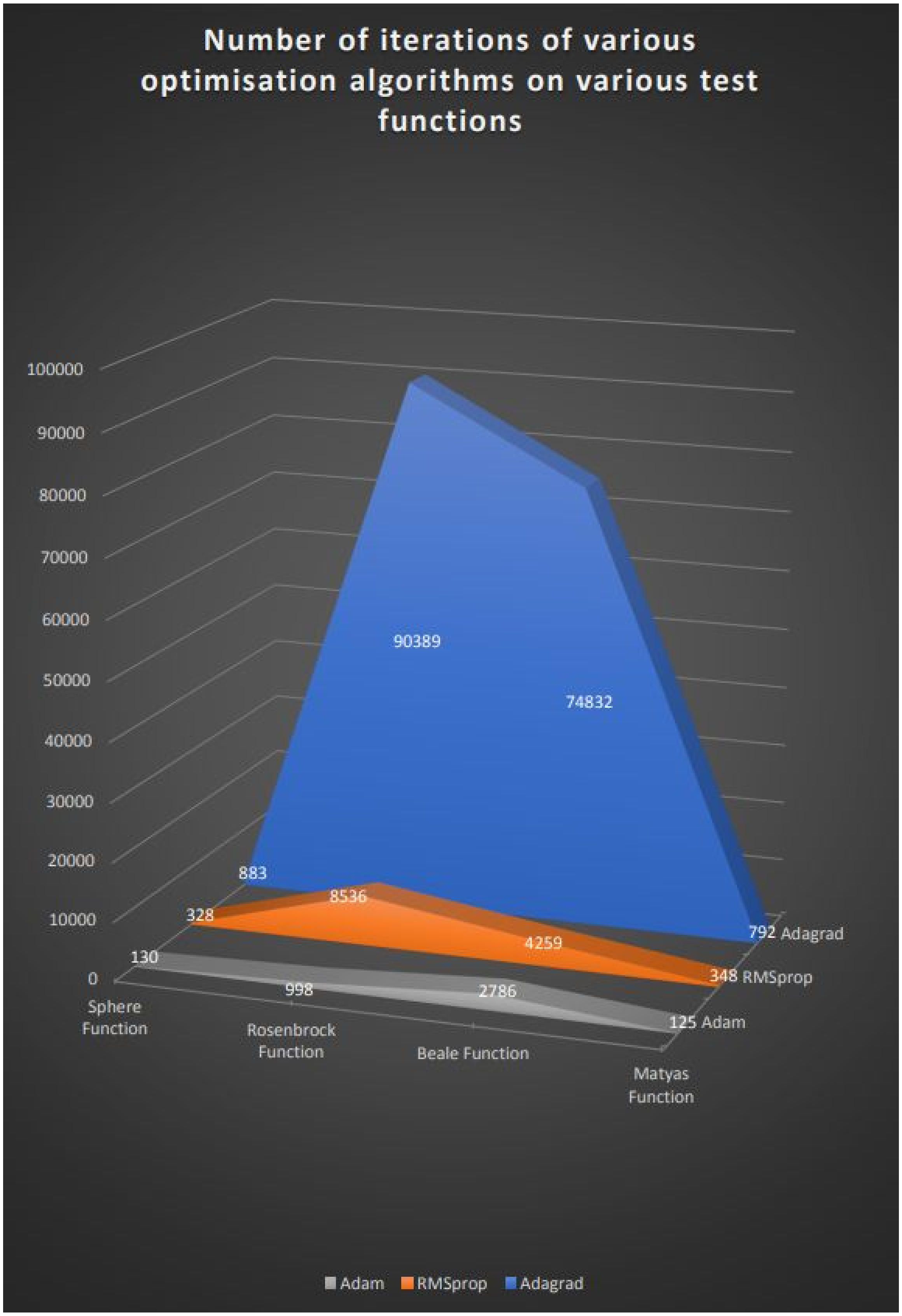
We test Adagrad, RMSprop and Adam algorithms on various test functions like –

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 –
   $f(x_1, x_2) = (1.5 - x_1 - x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$

4. Matyas Function –
   $f(x_1, x_2) = 0.26(x_1 + x_2)^2 - 0.48 x_1 x_2$


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
8536
328
4259
792 Adagrad
348 RMSprop
130
998
2786
125 Adam
Sphere Function
Rosenbrock Function
Beale Function
Matyas Function
Adam
RMSprop
Adagrad

Adamax-

The L^2 norm was used in Adam technique.
Let's see what will happen if we generalize it to L^p.

For higher values of p the algorithm becomes numerically unstable . However when $p->\infty$ a surprisingly stable algorithm emerges.

$$\begin{aligned} v_t &= \beta_2^p v_{t-1} + (1-\beta_2^p)|g_t|^p \\ &= (1-\beta_2^p)\sum_{i=1}^{t} \beta_2^{p(t-i)} \cdot |g_i|^p \end{aligned}$$

$$\begin{aligned} u_t = \lim_{p\to\infty}(v_t)^{1/p} &= \lim_{p\to\infty}\left((1-\beta_2^p)\sum_{i=1}^{t}\beta_2^{p(t-i)}\cdot|g_i|^p\right)^{1/p} \\ &= \lim_{p\to\infty}(1-\beta_2^p)^{1/p}\left(\sum_{i=1}^{t}\beta_2^{p(t-i)}\cdot|g_i|^p\right)^{1/p} \\ &= \lim_{p\to\infty}\left(\sum_{i=1}^{t}\left(\beta_2^{(t-i)}\cdot|g_i|\right)^p\right)^{1/p} \\ &= \max\left(\beta_2^{t-1}|g_1|, \beta_2^{t-2}|g_2|, \ldots, \beta_2|g_{t-1}|, |g_t|\right) \end{aligned}$$

This update remain same as that of Adam where $v_t = u_t$

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

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}\hat{m}_t$$

Implemented Code

End Results-

The algorithms were analysed on the functions used above and the performance was compared on the three techniques viz. Adagrad , RMSprop, Adam.

Performance (in terms of number of iterations required) : **Adagrad** < **RMSprop** < **Adam**

Applications-

Stochastic gradient descent is the widely used technique for unconstrained optimisation, and by introducing momentum and acceleration we are also taking into consideration the function topology.

Unconstrained Optimisation is used in minimizing the cost function in supervised learning techniques. So by using these techniques we can reduce the number of iterations required for reaching the minimizer.

References –

1. Adam: A Method for Stochastic Optimisation: Link
2. An overview of gradient descent optimization algorithms :Link
3. Coursera
   a. Lecture 29, Neural Networks for Machine Learning, By Geoffrey Hinton:Link

4. https://math.stackexchange.comhttps://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