On Comparing Four Optimization Algorithms for ANN

Prateek Khandelwal(M21EE012), Debasmita Mukherjee(M20EE052)
Electrical Engineering Department
Indian Institute of Technology, Jodhpur

Abstract - Network-based forecasting has played an important role. The network parameters optimization is an important issue, and different optimization algorithms are believed to result in different forecasting accuracies. In this paper, four network parameters optimization algorithms, including Stocasting Gradient descent, AdaGrad, RMSprop and Adam are implemented and compared in the application of forecasting. The experiment results show that, Adam algorithm and Adagrad algorithm achieve better forecasting accuracy and speedly converse the loss function than the other optimization algorithms. This study can be a guide to the selection of optimization algorithms on Concrete quality problems. Further We test the performance of the Adam, RMSprop, Adagrad and SGD for the choose loss function which has local multiple local minimas and demonstrate the results.
We choose the f(x,y) = -2 * exp(-((x + 2)^2 + y^2)/2) - 3 * exp(-((x - 2)^2 + y^2)/2), this finction has two local minimas at -2 and +2.

Key Words : Optimization algorithms , Back-propagation neural network, Stocasting Gradient descent, AdaGrad, RMSprop and Adam.

1.Introduction

Recently, with the development of artificial intelligence techniques, many researchers propose to apply them in various forecasting problem, including artificial neural network (ANN), fuzzy logic method, support vector machine (SVM) and other mixed methods. The artificial neural network (ANN) can simulate any complex nonlinear system, and it is a complex network system composed of many neurons. Here we taken shown one forecasting algorithms based on back-propagation neural network (BPNN) is a common network-based method. The BPNN is the most popular neural network for time series forecasting. The BPNN adopts an error back-propagation (BP) algorithm, which is an unrestricted nonlinear optimization process. And so the training process may easily fall into local minimum and make the convergence speed slower or cannot get the optimal value when the network structure is very complex.

In this paper, six gradient-based parameters optimization algorithms of BPNN are compared for wind speed forecasting application. These optimization algorithms include traditional stochastic Gradient descent, Momentum, AdaGrad, RMSprop and Adam[1-5].

2. Data Set:-

We have taken dataset of Concrete for forecasting of the concrete quality in terms of concrete compressive strength. The concrete dataset having the following attributes : Cement (component 1)(kg in a m^3 mixture), Blast Furnace Slag (component 2)(kg in a m^3 mixture), Fly Ash (component 3)(kg in a m^3 mixture), Water (component 4)(kg in a m^3 mixture), Superplasticizer (component 5)(kg in a m^3 mixture), Coarse Aggregate (component 6)(kg in a m^3 mixture), Fine Aggregate (component 7)(kg in a m^3 mixture), Age (day), Concrete compressive strength(MPa, megapascals). In this paper we will forcast the Concrete compressive strength by using artificial neural network with the four different optimization technique and shown the loss function for the same.

3.Network optimized by different methods

The general network parameter optimization is an extremely difficult work. Machine learning method usually use the strategy by carefully designing the objective function and constraints to ensure that the optimization function is convex, and so to avoid this difficulty. However, we may need to confront the general non-convex problems when training the neural networks, which lead to the difficulties, such as ill-conditioning, local minima. The gradient descent method is one of the most important and fundamental algorithms to perform optimization and it is the most common way to optimize network parameters in BPNN model.

Let *J* (q) is the objective function parameterized by the BackPropagation neural network(BPNN's) parameters ⍰⍰⍰R^*d* (*d* is the number of parameters that need to be optimized). Let d*J*(q)denote The gradient vector of our model's objective function w.r.t. to the parameters to be estimated. The gradient descent method uses the way to minimize function *J* (q) by updating the parameters in an opposite direction of the gradient vector of our objective function d*J*(q. The learning rate η is an important parameter, which determines the step size used to reach a (local) minimum. That is to say, we follow a slope direction of the surface created by the BPNN's objective function downhill until we reach a valley. We will not discuss the algorithms which used second-order, such as Newton's method, Levenberg-Marquardt method etc., since these methods need to calculate the Hessian matrix, and they are not good to compute in the practical application for high-dimensional dataset.

3.1 **Stocasting Gradient Descent Algorithm:**

In deep learning, the objective function is usually the average of the loss functions for each example in the training dataset. Given a training dataset of n examples, we assume that fi(x) is the loss function with respect to the training example of index i, where x is the parameter vector. Then we arrive at the objective function.

f(x) = 1 /n ∑n i=1 fi(x)

The gradient of the objective function at x is computed as

∇f(x) = 1 /n ∑n i=1 ∇fi(x)

If gradient descent is used, the computational cost for each independent variable iteration is O(n), which grows linearly with n. Therefore, when the training dataset is larger, the cost of gradient descent for each iteration will be higher.

Stochastic gradient descent (SGD) reduces computational cost at each iteration. At each iteration of stochastic gradient descent, we uniformly sample an index i ∈ {1, . . . , n} for data examples at random, and compute the gradient ∇fi(x) to update x:

x ← x − η∇fi(x)

where η is the learning rate. We can see that the computational cost for each iteration drops from O(n) of the gradient descent to the constant O(1). Moreover, we want to emphasize that the stochastic gradient ∇fi(x) is an unbiased estimate of the full gradient ∇f(x) because

Ei∇fi(x) = 1 n ∑n i=1 ∇fi(x) = ∇f(x).

This means that, on average, the stochastic gradient is a good estimate of the gradient.

3.2 AdaGrad Algorithm

The parameters updating strategy of AdaGrad method is given by below formula.

Gt = ∂wl(yt , f(xt , w)), st = st−1 + g 2 t ,

wt = wt−1 − η √ st + ϵ · gt .

where *d d G* ⍰*R* ⍰denotes the diagonal matrix, and each diagonal element *i*,*i* denotes a sum of squares of the gradients w.r.t. *i*⍰⍰up to the time step *t* . ⍰⍰denotes the smoothing term that avoids division by zero, which can be set to the order of 1e–8. Here ⍰⍰is a learning rate, which can be set to 0.01. Over all the training iterations to adapt the learning rate, the AdaGrad algorithm uses the strategy of scaling the learning rate inversely proportional to an accumulated sum of squared partial derivatives. These parameters with the biggest partial derivative of the BPNN's objective function will have a correspondingly rapid decrease in the learning rate, while parameters with small partial derivatives will have a relatively small decrease in the learning rate. The network effect is greater progress in the more gently sloped directions of the parameter space

3.3 RMSprop Algorithm

RMSprop uses an adaptive learning rate updating strategy, and the method is presented by Geoff Hinton in a deep learning lecture [23]. This method is usually used in the application of deep learning. We first propose to use the idea of RMSprop algorithm in the application of wind speed forecasting problem.

The RMSprop algorithm divides the learning rate by the exponentially decaying average of the squared gradients.
This method is easy to implement and use, that is to say, it is not sensitive to the algorithm's hyperparameters. The parameters updating strategy is
$s_t \leftarrow \gamma s_{t-1} + (1 - \gamma) g_t^2$,
$x_t \leftarrow x_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_t$.
Where h is the learning rate. r can be set to 0.9, and the learning rate h can be set to 0.001.

3.4 Adam Algorithm

The adaptive moment estimation (Adam) method computes the adaptive learning rates of each parameter. This
method keeps an exponentially decaying average of the past gradients, and also stores an exponentially decaying average of past squared gradients. The theory of Adam is first proposed by Kingma [20], but we are the first to use this theory in the application of wind speed forecasting and also get a better performance.
$v_t \leftarrow \beta_1 v_{t-1} + (1 - \beta_1) g_t$, $s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) g_t^2$.
$\hat{v}_t = v_t / (1 - \beta_1^t)$ and $\hat{s}_t = s_t / (1 - \beta_2^t)$
$g'_t = \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon}$.
$x_t \leftarrow x_{t-1} - g'_t$.
where *s^t* and *v^t* denote the estimates of the first moment and the second moment of the gradients, respectively. Her h is the learning rate. The proposed values of 0.9 for beta_1 0.999 for beta_2 and 10e−8 for e.

Properties of Adam

Here We have listed some of the properties of Adam.
1.Actual step size taken by the Adam in each iteration is approximately bounded the step size hyper-parameter. This property add intuitive understanding to previous unintuitive learning rate hyper-parameter.
2.Step size of Adam update rule is invariant to the magnitude of the gradient, which helps a lot when going through areas with tiny gradients (such as saddle points or ravines). In these areas SGD struggles to quickly navigate through them.
3.Adam was designed to combine the advantages of Adagrad, which works well with sparse gradients, and RMSprop, which works well in on-line settings. Having both of these enables us to use Adam for broader range of tasks. Adam can also be looked at as the combination of RMSprop and SGD with momentum.

5. Experiments and Analysis

5.1 **Forcasting Results of implementation of NN with different optimizers on Concrete dataset** :
We have shown the value of loss on the training set for each interation for four algorithm:

1. Adam -

```
{'Adam':
array([[4.32289585], [4.80236955], [4.39503051], [4.42041378], [4.48646537], [4.52369536],
[4.46179257], [4.39077505], [4.44809746], [4.44048453], [4.3931342 ], [4.48967404],
[4.35428021], [4.3915245 ], [4.30663125], [4.38056062], [4.33078521], [4.41293382],
[4.34878085], [4.41177206], [4.2883864 ], [4.28362584], [4.37521787], [4.37880993],
[4.33666468], [4.31250849], [4.34343364], [4.34111692], [4.33582182], [4.35350636],
[4.30426843], [4.35082931], [4.28806422], [4.3198579 ], [4.33715743], [4.35609718], [4.4136102
], [4.30209326], [4.32436473], [4.36797886], [4.37791929], [4.28715865], [4.34964188],
[4.39115888], [4.32182118], [4.25803836], [4.27180269], [4.34052239], [4.33917279],
[4.33598448], [4.37247397], [4.3006796 ], [4.34630912], [4.32533228], [4.32694584], [4.3279177
], [4.26832321], [4.31938651], [4.28075264], [4.34147491], [4.3482595 ], [4.31811059],
[4.31208907], [4.28307528], [4.35211591], [4.28183101], [4.30471365], [4.28520526],
[4.31905124], [4.31354608], [4.35328518], [4.37006732], [4.36728217], [4.25234769],
[4.25545211], [4.24276824], [4.29043379], [4.36542832], [4.32745514], [4.35379219], [4.3191221
], [4.28242487], [4.32826231], [4.29577354], [4.30892241], [4.30658288], [4.33930097],
[4.31118962], [4.3035775 ], [4.35873881], [4.35228156], [4.31252469], [4.32623406],
[4.35633705], [4.33177599], [4.30964797], [4.30788071], [4.28640714], [4.31020327], [4.3618957
], [4.37474274], [4.28754058], [4.32994927], [4.37295976], [4.34627634], [4.2598066 ],
[4.31901527], [4.26786296], [4.28527166], [4.29163951], [4.36512453], [4.27585937],
[4.29233807], [4.33037776], [4.3693872 ], [4.38695102], [4.38261516], [4.2732706 ],
[4.32963768], [4.39096814], [4.29380231], [4.30376071], [4.34999312], [4.37390288], [4.3308543
], [4.34603672], [4.30631234], [4.35271964], [4.38014301], [4.33559712], [4.32671437],
[4.39289579], [4.35766459], [4.35889025], [4.34271879], [4.35096402], [4.24984466],
[4.34817885], [4.31663974], [4.32529046], [4.39378653], [4.35687964], [4.25529598],
[4.43711057], [4.32386741], [4.33509594], [4.34679532], [4.34274713], [4.33996857],
[4.36300487], [4.35829481], [4.30988452], [4.37673222], [4.31676697], [4.36018056], [4.3151916
], [4.35452878], [4.28543609], [4.33410481], [4.32896187], [4.2926835 ], [4.26792451],
[4.2689114 ], [4.35190716], [4.34177305], [4.38756049], [4.29473884], [4.30001449],
[4.38075997], [4.34355219], [4.3044841 ], [4.26059106], [4.36222184], [4.34859578], [4.3127671
], [4.33859058], [4.3118241 ], [4.31844741], [4.2943024 ], [4.32902685], [4.33319584],
[4.34819416], [4.38485711], [4.35571482], [4.29925507], [4.32937581], [4.3137003 ],
[4.33688575], [4.36094271], [4.31967273], [4.36339259], [4.29307637], [4.24538088],
[4.35650104], [4.36805288], [4.32195104], [4.34185074], [4.32009414], [4.26970913],
[4.35178649]]),
```

2. Adagrad

```
'Adagrad': array([[4.15579627], [4.85737227], [4.25468461], [4.16743364], [4.20175363],
[4.14434846], [4.12626734], [4.11970169], [4.01298136], [4.18271969], [4.11283529],
[4.05654039], [4.06918684], [4.09655018], [4.13144465], [4.15524036], [4.13454319],
[4.14221214], [4.18578964], [4.18232501], [4.07083688], [4.12199772], [4.12162782],
[4.09271777], [4.07502217], [4.09780599], [4.14171297], [4.11654001], [4.1052418 ],
[4.07213815], [4.14550246], [4.1522432 ], [4.17521024], [4.11604916], [4.05427982], [4.0702926
], [4.17915398], [4.14663099], [4.0580842 ], [4.13416778], [4.1493126 ], [4.08500826],
[4.13304095], [4.14408475], [4.12036057], [4.12613732], [4.10595756], [4.10012825],
[4.15146165], [4.10954404], [4.04396988], [4.11904492], [4.21018834], [4.11824556],
[4.12698587], [4.08878834], [4.1111065 ], [4.09916187], [4.05762329], [4.05488337],
[4.17656381], [4.10149781], [4.14573852], [4.12030905], [4.11021174], [4.16207983],
[4.14172452], [4.0748268 ], [4.14786208], [4.11953073], [4.14302348], [4.09365061],
[4.14166305], [4.12897318], [4.13755649], [4.16859454], [4.17717135], [4.1563126 ],
[4.11959411], [4.08512522], [4.14395601], [4.10629227], [4.15615232], [4.02499087],
[4.07808489], [4.14282651], [4.10997561], [4.0867696 ], [4.10520641], [4.08709834],
[4.13355744], [4.12296721], [4.13149372], [4.11135591], [4.09320376], [4.13312471],
[4.15234094], [4.06037121], [4.10863897], [4.04708755], [4.08261653], [4.14550637],
[4.13152256], [4.10714039], [4.1006453 ], [4.13212308], [4.10755758], [4.13779861],
[4.08325157], [4.08911048], [4.04839882], [4.10294065], [4.09195862], [4.08256675],
[4.07926182], [4.14695331], [4.08641574], [4.08441729], [4.09174209], [4.13187728],
[4.12103195], [4.15050211], [4.12045631], [4.12312496], [4.11046049], [4.08369069],
[4.11220437], [4.13956649], [4.068683 ], [4.02165063], [4.06563646], [4.11792032],
[4.15306998], [4.18417872], [4.16931513], [4.05921656], [4.05131484], [4.22198335],
[4.14910737], [4.14513344], [4.15848587], [4.17774433], [4.08208733], [4.16867259],
[4.09191362], [4.1677187 ], [4.14158637], [4.11305682], [4.14954694], [4.16255512],
[4.08813725], [4.14875978], [4.12959334], [4.07829103], [4.14661498], [4.18709885], [4.1056551
], [4.16516874], [4.16682391], [4.12837037], [4.08759028], [4.10917991], [4.13728466],
[4.15286838], [4.07045616], [4.07190114], [4.10608007], [4.06689753], [4.15197787],
[4.09289303], [4.09876329], [4.14134565], [4.08965193], [4.21822371], [4.11956957],
[4.18001969], [4.14863823], [4.11229102], [4.14935222], [4.10505151], [4.11849132],
[4.19070665], [4.16690049], [4.17207804], [4.13013346], [4.05927838], [4.13544117],
[4.16016149], [4.08887751], [4.1600254 ], [4.04764295], [4.1438887 ], [4.05921627],
[4.09396958], [4.21288691], [4.13274135], [4.15411066], [4.1034315 ], [4.0991305 ], [4.0501553
]]), '
```

3. **RSMprop**:

```
RMSprop': array [118.92051558], [ 17.55755213], [ 11.61758341], [ 15.64242603], [
11.40281074], [ 9.64022444], [ 9.5805071 ], [ 9.68440007], [ 9.96022558], [ 9.80641618
9.77097584], [ 9.72065753], [ 9.43054797], [ 9.28419644], [ 9.3001499 ], [ 9.31991889]
9.2319532 ], [ 9.24628077], [ 9.42161281], [ 9.20472634], [ 9.36888474], [ 9.39595129]
9.60709226], [ 9.63457736], [ 9.68966828], [ 9.5009276 ], [ 9.31523412], [ 9.07991084]
8.85620469], [ 8.97446711], [ 9.10366784], [ 9.04972364], [ 9.11971457], [ 8.82720906]
8.85687295], [ 9.01140194], [ 8.8275654 ], [ 8.86579357], [ 9.14247352], [ 9.14838137]
9.2066388 ], [ 9.15234709], [ 9.27934162], [ 9.24157944], [ 9.44999723], [ 9.20034899]
9.37458979], [ 9.23999707], [ 9.29547407], [ 9.00592873], [ 9.12821864], [ 8.91554543]
9.16183829], [ 8.91025941], [ 9.02571754], [ 9.11745013], [ 9.32939592], [ 9.10273147]
9.12928359], [ 9.06420508], [ 9.15299867], [ 8.94393174], [ 9.14072452], [ 8.90747096]
9.10597208], [ 8.84109563], [ 8.9191445 ], [ 8.88217358], [ 8.98589614], [ 8.78732081]
8.85033943], [ 8.84819123], [ 9.11241327], [ 9.62343648], [ 10.07494751], [ 9.42423917
9.22880733], [ 8.97078139], [ 9.04185766], [ 8.921896 ], [ 9.08567891], [ 9.0143699 ],
9.16059323], [ 8.87686394], [ 8.82216391], [ 8.8421137 ], [ 9.00327241], [ 9.00696279]
9.17386413], [ 8.67093512], [ 8.64431977], [ 8.72307582], [ 8.79465039], [ 8.62567103]
8.9362268 ], [ 8.93680303], [ 8.8272153 ], [ 8.86612441], [ 8.99817243], [ 8.88963649]
9.07831672], [ 9.16445842], [ 9.27752572], [ 9.25122359], [ 9.25657795], [ 9.02533111]
8.86540538], [ 8.99055846], [ 9.23604232], [ 8.90123068], [ 9.22348548], [ 9.16487379]
9.44204845], [ 9.24061987], [ 9.57533778], [ 9.23433079], [ 8.95467269], [ 8.912771 ],
8.80419147], [ 8.6479084 ], [ 8.48199788], [ 8.36715499], [ 8.6270991 ], [ 8.66719605]
8.85936499], [ 9.16099307], [ 9.38113329], [ 9.16061351], [ 9.18267842], [ 9.23764096]
9.45906211], [ 9.12938988], [ 9.14064562], [ 9.00240971], [ 9.08922698], [ 9.0582573 ]
9.09475783], [ 8.77579592], [ 9.04202688], [ 9.05308324], [ 9.08585329], [ 8.91516553]
8.68903159], [ 8.78344825], [ 8.9473017 ], [ 8.99463746], [ 9.01984137], [ 9.15455623]
9.10071802], [ 8.7823306 ], [ 8.96119395], [ 8.93094726], [ 8.89376159], [ 8.70036246]
8.77171963], [ 8.78189766], [ 8.9285707 ], [ 8.69724056], [ 8.69843975], [ 8.88331458]
8.89095451], [ 8.88859005], [ 8.93515095], [ 9.01373805], [ 9.30394424], [ 9.2915335 ]
9.22702652], [ 9.01095966], [ 8.95980278], [ 9.07022741], [ 9.02265614], [ 8.88499065]
9.05452885], [ 9.14323162], [ 9.30753075], [ 9.25620171], [ 9.26034831], [ 9.01316074]
8.93778957], [ 8.86129217], [ 8.94867133], [ 8.91649407], [ 9.08926854], [ 8.82717896]
9.1686106 ], [ 8.87035476], [ 8.83353287], [ 8.98111197], [ 8.98990366], [ 8.73736763]
8.88183928], [ 9.10116625], [ 9.29089756], [ 9.11732455], [ 9.01742977], [ 8.96637382]
9.13833947], [ 9.06341289], [ 9.18052056]]),
```

4. SGD:

```
'SGD': array([[5.39316005], [5.22167081], [5.17832372], [5.19745933], [5.14396
[5.22611993], [5.18438612], [5.15631503], [5.19551355], [5.08000004], [5.20540
[5.22241292], [5.21333557], [5.12439953], [5.24265006], [5.16691474], [5.19639
[5.25736129], [5.16522697], [5.16369689], [5.18727463], [5.14852463], [5.19002
[5.27679928], [5.24054633], [5.12834921], [5.18528352], [5.12045751], [5.22249
[5.14262352], [5.15197411], [5.14420862], [5.14080371], [5.1869592 ], [5.23415
[5.13254257], [5.18496761], [5.11744882], [5.256927 ], [5.17684072], [5.21488
[5.18226655], [5.18673951], [5.20552504], [5.14127924], [5.10853527], [5.18359
[5.16252813], [5.18518879], [5.19620171], [5.0929905 ], [5.0823934 ], [5.14082
[5.19583447], [5.17092978], [5.17887655], [5.11270323], [5.21489689], [5.17494
[5.19710963], [5.16968185], [5.23383918], [5.09467467], [5.16143882], [5.19393
], [5.16168169], [5.16197246], [5.2100924 ], [5.18355247], [5.18193007], [5.14
[5.10422589], [5.24960853], [5.10973119], [5.11308059], [5.16681395], [5.17227
[5.24372768], [5.26629905], [5.23937811], [5.2791434 ], [5.21902274], [5.19493
[5.18412301], [5.12843857], [5.20835178], [5.19013792], [5.13106081], [5.12277
[5.21083559], [5.21412165], [5.13813313], [5.18669465], [5.17194621], [5.12394
[5.19699057], [5.17333928], [5.22079119], [5.14038602], [5.17269964], [5.22060
[5.21824179], [5.21374262], [5.09447959], [5.20424788], [5.20565084], [5.06206
], [5.12364311], [5.14250989], [5.20911463], [5.19622779], [5.17478357], [5.23
[5.17488368], [5.17516065], [5.1351478 ], [5.16911834], [5.13556455], [5.08754
[5.17023063], [5.25467039], [5.05444342], [5.15860673], [5.15786544], [5.16146
[5.05672641], [5.16427033], [5.12624874], [5.234855 ], [5.13278059], [5.178743
[5.14333148], [5.15464426], [5.17274284], [5.1573852 ], [5.14720513], [5.16833
[5.15739749], [5.20665453], [5.17093597], [5.12944271], [5.22311419], [5.20478
[5.13433383], [5.15528435], [5.13036251], [5.11151502], [5.24660148], [5.10550
[5.17009601], [5.10266966], [5.18466624], [5.1703966 ], [5.25960303], [5.15133
[5.14245523], [5.22300565], [5.21085699], [5.10053614], [5.17310285], [5.15669
[5.18486513], [5.23104854], [5.19102869], [5.19470523], [5.09959948], [5.24521
[5.11029383], [5.16241048], [5.13551887], [5.16991526], [5.21014186], [5.13277
[5.25489977], [5.10895286], [5.10062517], [5.21396981], [5.12883466], [5.06909
[5.16756989], [5.13252142], [5.23255412], [5.21963185], [5.09606548], [5.23896
[5.14531227], [5.17483235], [5.17759793], [5.10241088], [5.3057447 ], [5.13534
[5.19482067], [5.20834026], [5.08188611], [5.1613955 ], [5.21632618], [5.13811
[5.12347241]])}
```

It is obersered that loss function for training data is converse fastly for the Adam and Adagrad converse fastly and provide the less value of loss function in the comparison of other algorithm like SGD and RMSprop.

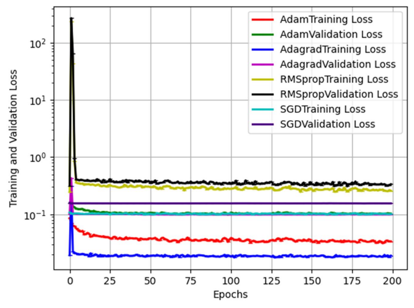

Figure.1 Loss on training and validation set with the number of iteration

5.2 **Performance of different optimizers on the given loss functions:**

On below taken loss function we have apply four optimizer and check the performance of each and compare**.**

f(x,y) = -2 * exp(-((x + 2)^2 + y^2)/2) - 3 * exp(-((x - 2)^2 + y^2)/2)

After appling different optimization algorithm on the loss function the following graph has been plot for loss value w.r.t. to the number of iteration. The graph are depicted in the figure-2.

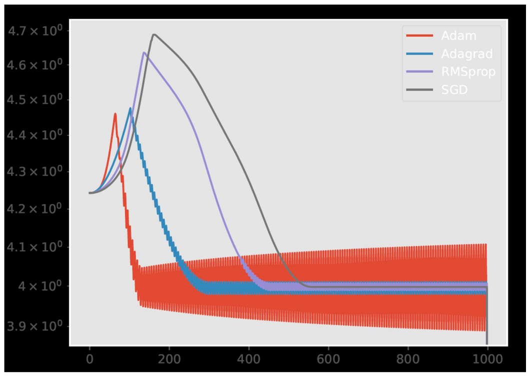

Figure-2 loss fuction v/s number of interation

It is observed that Adam optimization converse speedly compare than other optimization algorithm including SGD, RSMprop and Adamgrad.

6. Discussion and Conclusion

From this case study, it can be seen that the BPNN trained with different forecasting horizon or different optimization algorithms demonstrates varies levels of forecasting accuracy. Therefore, in building the BPNN for forecasting Concrete quality, factors such as the value *k* and optimizational algorithms should be properly determined, since this decision directly effects the forecasting accuracy. It comes to the conclusion that Adam algorithm and Adagrad algorithm achieve better forecasting accuracy and speedly converse the loss function than other optimization algorithms which including Stocasting gradient descent and RMSprop in network-based forecasting. This is a guide to the selection of the network optimization algorithms on forecasting problems. It should be noted that, limited by the available data source, only 500 data are adopted. Although different datasets may influence the forecasting accuracy of BPNN with different optimization algorithms. It is believed that the effects of the data sets is relatively small. However, it will beideal to have more datasets to investigate this influence, and this is actually a limitation of the current models.

7. **Acknowledgements:**

References:

[1] L. Bottou. Stochastic Gradient Descent Tricks. Neural Networks: Tricks of the Trade. Springer Berlin Heidelberg, 2014:421-436.
[2] M.D. Zeiler, ADADELTA: An Adaptive Learning Rate Method, Computer Science, 2012.
[3] D.P. Kingma, J. Ba, Adam: A Method for Stochastic
Optimization, Computer Science, 2014.
[4] J. Duchi, E. Hazan, Y. Singer, Adaptive Subgradient
Methods for Online Learning and Stochastic Optimization, Journal of Machine Learning Research , 12 (7) :2121-2159, 2011.
[5] L. Bottou. Stochastic Gradient Descent Tricks. Neural Networks: Tricks of the Trade. Springer Berlin Heidelberg, 2014:421-436.