Optimisation methods review from a Machine Learning standpoint

Terrance Thomas
School of Computing
National College of Ireland
Dublin, Republic of Ireland
x18184928@student.ncirl.ie

***Abstract*— Machine learning (ML) is quickly developing which has led to many conceptual breakthroughs and is commonly applied in many different fields Optimisation, as an important part of ML has attracted much researcher's attention. With the overall increase in data volume and the increase in model complexity, optimisation methods in ML are facing increasing challenges. A great deal of work has been proposed successively to solve optimisation problems or improve optimisation methods in ML. The systematic retrospect and summary of optimisation techniques from the ML perspective are of great importance which can guide both optimisation advancements and ML research.**

A review of four methods of optimisation viz nesterov accelerated gradient descent (NAGD), adaptive moment estimation (ADAM) optimisation, hessian free method (HFM), and natural gradient (NG) along with their applications is provided in this paper. Methods of optimisation face challenges in ML with increasing complexity of the data. Each optimisation method has benefits and limits are discussed which are significant since it can guide suitable methods for future ML research. Finally, it also discusses the challenges.

Keywords— Machine learning, optimisation method, Optimisation criteria, Nesterov Accelerated Gradient Descent, Adam optimization, Hessian Free Method, Natural Gradient, Heuristic Algorithms, Adaptive moment estimation

I. INTRODUCTION

ML has evolved at an extraordinary rate and has attracted many researchers and practitioners. It is one of the most successful directions of research and plays an important role in many fields, such as machine translation, recommendation system, image recognition, speech recognition, etc. One of the core components of ML is optimisation. Most ML algorithms are essentially designing an optimisation model and learning the variables from the given data in the objective function. One of the core components of ML is optimisation. Most ML algorithms are essentially designing an optimisation model and learning the variables from the given data in the objective function. The effectiveness and efficiency of the numerical optimisation algorithms greatly affect the popularization and framework of the ML models in the era of immense data. A series of effective optimisation methods have been put forward to promote the development of ML, which have improved the efficiency and performance of ML methods.

Popular optimisation methods can be divided into 3 categories from the perspective of gradient information in optimisation: first-order optimisation methods, heuristic derivative-free optimisation methods, and high-order optimisation methods. Within the optimisation community, it is well known that gradient descent is unfit to optimise targets that exhibit pathological curvature. The 2nd-order optimisation methods that shape and correct the local curvature are quite successful in achieving these objectives. The most significant recent progress in deep network learning has been the growth of layer-wise unsupervised pre-training methods. It seems that applying these methods before running stochastic gradient descent (SGD) overcomes the associated difficulties with a deep learning approach. Indeed, many successful applications of these methods have been made to difficult deep learning problems, such as auto-encoders and classification nets [1].

Methods of optimisation of first order gradient descent are widely used recently and are constantly improvising at a higher rate. Many users regard these methods as optimisers of black boxes. It will explain two methods of ADAM optimisation and NAGD. It explains the merits and demerits which include their applications. These papers provide the basis for optimisation applications of ML [2][3]. SGD is widespread used in deep neural networks (DNN). One of the exciting developments is ADAM optimisation which is used to create a high-resolution generative adversarial network for super-resolution image performance.

ADAM is a technique for efficient stochastic optimisation that requires only gradients of first order with little memory consumption. The technique computes personal adaptive learning rates from forecasts of the first and second moments of the gradients for different parameters. ADAM incorporates the benefits of two recently popular methods: AdaGrad, which works well with sparse gradients, and RMSProp, which also works well in online and non-stationary settings. ADAM's few advantages are that the magnitudes of the parameter updates are invariant to the gradient rescaling, its step sizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients and it naturally performs a form of step size annealing [2].

High order optimisation methods have faster convergence rates where curvature info allows a more effective direction of search. Those are generally well-known methods with some inherent challenges. The challenge lies in the process, storing the Hessian matrix inverse. To address this problem, many variations were developed based on Newton's method, most of which, with the help of certain techniques, attempt to estimate the Hessian matrix. This paper reviews the free Hessian technique and techniques for optimising NG [1][4].

The solution for measuring the optimum point without the gradient info is called "derivative-free optimisation," which would be a mathematics discipline. Derivative-free optimisation techniques are mainly used in cases where there may not be an objective function derivative and difficult to calculate. There are two main types of techniques that are formulated for derivative-free optimisation. The first type implements a search of scientific rules based on heuristics.

The other type fits sample sizes to objective functions. Section 2 is about reviewing different methods of optimisation. Section 3 sets out the conclusions drawn from the papers. Lastly, the bibliography contains the list of four core review papers obtained from primary sources as well as supporting papers cited where necessary.

II. REVIEW

As mentioned in Section 1, the optimisation is classified into 2 parts viz high order optimisation and first order optimisation.

A. High order

High order encourages widespread attention but faces the prospect. The difficulty in high order techniques lies in the operation and storage of the inverse matrix of the Hessian matrix. Numerous variants based on Newton's method have been developed to solve this problem, most of which attempt to approximate the Hessian matrix by some techniques. The second order techniques can be used to fix the challenges were a highly non-linear and unconditioned objective function. They function effectively by introducing information about curvature. Though the convergence of the method can be ensured, the computational process is costly and therefore rarely used to solve major ML problems. The two main methods under this category are HFM and NG.

1) HFM : HFM is identical to the Newton technique, which uses gradient information in the second order. This is a shortened Newton method which, although not used in ML, is extensively researched over many years. With the help of the conjugate gradient, HFM performs a suboptimisation, which avoids costly inverse Hessian matrix calculation. The distinction is, the HFM doesn't have to directly calculate the Matrix. The algorithm for HFM is as given below:

for $n = 1, 2, \ldots$ **do**
  $g_n \leftarrow \nabla f(\theta_n)$
  compute/adjust λ by some method
  define the function $B_n(d) = \mathbf{H}(\theta_n)d + \lambda d$
  $p_n \leftarrow$ CG-Minimize$(B_n, -g_n)$
  $\theta_{n+1} \leftarrow \theta_n + p_n$
end for

Fig. 1. HFM algorithm

The advantage of HFM is that it may not be required to compute hessian matrices directly using details pertaining to the second order gradient. Consequently, that method is suitable for problems with high dimensional optimisation. The findings demonstrated showed greater performance with no pretraining method requirement. The drawback is that HFM's computing costs for performing matrix vector products with more and more training data are rising linearly. The sub-sampled free Hessian method employs SDG and sub-sampled hessian vector during the updating process. This sub-sampled HFM can address the problem of large-scale ML optimisation. On both MNIST and FACES datasets, HFM can be used on deep autoencoders as well as on recurrent neural networks that solve the underfitting problem with this method [1].

2) NG : The plan was to create a gradient descent method in objective function rather than a parametric function space. This technique applies to any objective efficiency gauge function of the model. The classical gradient descent method is centered around the Euclidean space. In several instances, however, the parameter space is not euclidean and might have a Riemannian metric framework. Therefore NG provides the steepest direction for optimal target value objective function rather than gradient descent. The natural method of gradient can be applied potentially to any objective function that measures the performance of some statistical models [4].

Consider a model distribution n $p(y|x,\theta)$, and $\pi(x,y)$ is an observed distribution. To fit the parameters $\theta \in RN$. Assume that x is an observed vector, and y is its associated label. It has the objective function,

$$F(\theta) = E_{(x,y)\sim\pi}[-\log p(y|x,\theta)] \quad (1)$$

The optimised equation is as follows:

$$\theta * = argmin_\theta F(\theta)) \quad (2)$$

The NG can be converted from such a traditional gradient amplified by a Fisher Matrix,

$$\nabla N\, F = G^{-1}\, \nabla F \quad (3)$$

Where F is the object function, ∇F is the traditional gradient, ∇N F is the natural gradient, and G is the Fisher information matrix, with the following form:

$$G = \mathbb{E}_{x\sim\pi}\left[\mathbb{E}_{y\sim p(y|x,\theta)}\left[\left(\frac{\partial p(y|x;\theta)}{\partial\theta}\right)\left(\frac{\partial p(y|x;\theta)}{\partial\theta}\right)^{\top}\right]\right] \quad (4)$$

The updated formula can be written as:

$$\theta t = \theta t - \eta t \nabla N\, F \quad (5)$$

The NG uses the parametric space Riemann structure for adjust the direction of update which is best suited to finding the extremum of purpose function. However, the Fisher Information Matrix is complex to calculate [4].

B. First order

The first order and its different versions in recent years have been widely used and are evolving at a high velocity. Nonetheless, many users pay very little attention to the features or scope of these techniques. They often adopt them as black-box optimisers, which can limit optimisation methods' functionality. First order optimisation methodologies and make parameters more convenient adaptation to the learning process. The most commonly used first-order optimisation methods in the field of ML are based primarily on gradient descent. The two main methods under this category are ADAM and NAG.

1) ADAM : Another advanced method of SGD is ADAM which introduces an adaptive learning rate for each parameter. It combines the adaptive learning rate with the impulsive method. ADAM is a combination of momentum

and techniques of adaptation. Besides the exponential decay of the mean of past square gradients it also maintains an exponential average fall rate of preceding gradients just as in the impulse method. The idea is to use gradient prediction of first and second order time estimates to automatically change the learning rate for each parameter by adding a bias correction. $\beta 1$ and $\beta 2$ *is* exponential decay rates, past gradients m_t, past squared gradients V_T.

$$m_t = \beta_1 m_t - 1 + (1 - \beta_1) g_t \tag{6}$$

$$V_t = \sqrt{\beta_2 V_{t-1} + (1 - \beta_2)(g_t)^2}, \tag{7}$$

β1 and β2 are exponential decay rates. The final update formula for the parameter θ is:

$$\theta_{t+1} = m_t - \eta \frac{\sqrt{1 - \beta_2}}{1 - \beta_1} \frac{m_t}{V_t + \epsilon}. \tag{8}$$

The algorithm for ADAM is given as follows:

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Fig. 2. ADAM algorithm

Advantage of ADAM is that the process of gradient descent is comparatively stable. It is suitable for most non-convex optimisation issues with large data sets and large space while the limitation is that in some instances the method may not converge [2].

2) NAG : Gradient descent was one of the most frequently used optimisation techniques. The aim of the descent approach involves incrementally changing the parameters in the reverse path of the objective function. The learning rate specifies the size of each step and therefore affects the number of iterations required to achieve the optimum value. Usually, the learning process of baseline gradient descent is longer. Therefore there is a need to achieve convergence through adjustment of the learning rate. In this paper, the Momentum idea is introduced to maintain to some extent the impact of prior update direction on the upcoming iteration. *The momentum* v^{old}, mtm *is added to* θ.

$$\begin{cases} \widetilde{\theta} = \theta + v^{old} \cdot mtm, \\ v = v^{old} \cdot mtm + \eta \cdot (-\dfrac{\partial L(\widetilde{\theta})}{\partial(\theta)}), \\ \theta' = \theta + v. \end{cases} \tag{9}$$

NAG has a property to speed up the current gradient descent by collecting the prior gradient as momentum and proceed with momentum to the gradient update process. The advantage of NAG is that when the direction of the gradient changes, the momentum can slow the speed of the update and reduce the oscillation; if the direction of the gradient remains, the momentum may speed up the update of the parameter. Momentum helps spring from the locally optimal solution. However, the limitation is that choosing an appropriate learning rate is tough [3].

III. Conclusions

The paper presents an introduction to different kinds of optimisation methods and provides a detailed summary of the four relevant papers. Methods of first order optimisation, high order optimisation methods, and derivative-free optimisation are introduced. The paper illustrates the research papers on different methods such as HFM, NG, ADAM, and NAG techniques. HFM is finest used in DNN models since it achieves effective analysis without pre-training requirements. It is especially effective in recurrent neural networks and autoencoders. HFM solves problems of the underfitting model automatically. With the lowest training error of 1.4, 12.9, HFM has proven fruitful [1]. NG is a particularly unique type of algorithm for high order optimisation that is described as statistically significant. The best online learning would be as effective as best batch learning when the Fisher Info Matrix is in existence. Learning NG may be easier to get away from plateaus, rather than traditional stochastic gradient descent [4]. ADAM is a new algorithm that is highly accurate to optimise the gradients of objective stochastic functions. It is a combination of two preceding AdaGrad and RMSProp methods. With the ease of use, it requires less memory. ADAM is robust on a wide variety of noncovex optimisation problems [2]. NAG is far better than gradient descent because it retains geometric characteristics by filtering features such as time reversibility and phase space area contraction. Nesterov momentum is known to speed up parameters which also minimize convergence to a perfect solution. Based on the Nesterov momentum the learning rate is accelerated [3].

References

[1] J. Martens, "Deep learning via Hessian-free optimization," *in International Conference on Machine Learning*, 2010, pp. 735–742.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *in International Conference on Learning Representations*, 2014, pp. 1– 15.

[3] Y. Nesterov, "A method of Solving a Convex programming problem with convergence rate O(1/k^2)", *Doklady Akademii Nauk SSSR*, vol. 27, 1983.

[4] S. I. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, pp. 251–276, 1998.