ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/367484001

Fundamental optimization methods for machine learning

Chapter · January 2023

DOI: 10.1016/B978-0-323-91776-6.00005-1

CITATIONS
3

READS
56

2 authors, including:

Vinay Kumar Srivastava
Motilal Nehru National Institute of Technology
112 PUBLICATIONS **1,077** CITATIONS
SEE PROFILE

All content following this page was uploaded by Ranjana Dwivedi on 23 August 2023.

The user has requested enhancement of the downloaded file.

12

Fundamental optimization methods for machine learning

Ranjana Dwivedi, Vinay Kumar Srivastava

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING, MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY ALLAHABAD, PRAYAGRAJ, UTTAR PRADESH, INDIA

12.1 Introduction

With the rapid growth of machine learning, its popularity among researchers exponentially increases. It plays an important role in numerous applications. Speech recognition, recommendation platforms, image classification, image recognition, search engines, etc. are the several applications of machine learning. One of the main pillar of machine learning is optimization. In case of machine-learning models, optimization involves numerical or analytical computation of parameters in respect of an objective function. For a given learning problem, optimal solution of parameter is tried to achieve through optimization. Success of several optimization methods in machine learning has attracted great number of practitioners and researchers to work on even more challenging machine-learning problems. Based on gradient information, classical optimization method can be classified into three categories, namely, first-order methods, high-order methods, and derivative-free methods.

First-order methods are widely used optimization method particularly Stochastic Gradient Descent (SGD) method [1]. Variants of SGD are also popularly used method. First-order methods are easy to implement and this method also has low computational complexity. This method provides computation with low accuracy and it has slow convergence rate to reach the optimal solution. Various gradient descent (GD) algorithms are discussed here along with their disadvantages and advantages. To overcome the shortcomings of first-order methods, high-order optimization methods [2] attracts attention of researchers. High-order methods captures curvature information along with gradients of objective function. Curvature information helps it to converge faster. Storing the inverse Hessian matrix is one of the main issue of high-order methods. Several algorithms are proposed to solve this issue by approximating the Hessian matrix. Hessian-free Newton method, quasi-Newton method, Gauss-Newton method, and natural gradients are explained here. For the objective functions whose derivative is difficult to compute or its derivative do not exist, derivative-free optimization methods

Copyright © 2023 Elsevier Inc. All rights reserved.

[3,4] are used to find the optimal solution. Methods for convex objective and methods for stochastic optimization is discussed here. Along with the fundamentals of these optimization methods, challenges for optimization method is also presented.

12.2 First-order optimization methods

One of the most common technique to solve the convex optimization problems is first-order optimization method. In the field of machine learning, imaging, control theory and signal processing, this optimization method primarily based on GD. This method provides low computational complexity with low accuracy. This method is suitable when high accuracy is not crucial. Here the chapter introduce GD of first-order method. Variants of GD along with several algorithms which are based on GD are also introduced.

12.2.1 Gradient descent

In 1847, Augustin-Louis Cauchy proposed the method of GD (also known as Cauchy's method or steepest descent) for unconstrained optimization of a differentiable objective function. It is a fundamental derivative based iterative method. GD method is among the most common algorithm to do optimization. It is one of the popular method to optimize neural networks in machine learning and deep learning. Mostly deep-learning library have numerous algorithms implementation to optimize GD. GD method is used while training a machine learning model. GD method minimizes the objective function $f(\theta)$ in respect of parameter θ. Parameters are updated iteratively in the negative direction of gradient of objective function $\nabla_\theta f(\theta)$. Updating parameters is required to progressively converge to the optimal solution corresponding to objective function [1]. For each iteration, step size is determined by the learning rate η and to reach the (local) minimum value of objective function, η stimuli the number of iterations. The direction of negative gradient created by objective function decreases until the minimum value is obtained.

For a simple function f which is a differentiable function, GD method uses its gradient along with the function itself. Termination criteria will be

$$|d\theta| = |\theta(l+1) - \theta(l)| = |\alpha(l)d(l)| < \varepsilon \tag{12.1}$$

with search direction $d(l) = -\nabla_\theta f(\theta)$ and starting point $\theta(0) \in \Re^n$ at $l = 0$.

Step size α is given by

$$\alpha(l) = \underset{\alpha \in \Re^+}{\overset{arg\ min}{}} f(\theta(l) - \alpha(l-1)\nabla_\theta f(\theta) \tag{12.2}$$

is calculated while ensuring $f(\theta(l+1)) < f(\theta(l))$. Position will be updated as,

$$\theta(l+1) = \theta(l) - \left(\underset{\alpha \in \Re^+}{\overset{argmin}{}} f(\theta(l) - \alpha\nabla_\theta f(\theta))\right)\nabla_\theta f(\theta) \tag{12.3}$$

GD converges to a local minimum by using optimum $\alpha(l)$.

In machine learning, gradient i.e., the slope of function measures the change in all weights w.r.t. change in error. For a linear regression model, suppose there are N number of training sample, M number of input features, y is the target output. Loss function, $L(\theta)$, is to be minimized with

$$L(\theta) = \frac{1}{2N} \sum_{l=1}^{N} (y(l) - f(\theta(x(l))))^2 \tag{12.4}$$

where $f(\theta(x)) = \sum_{k=1}^{M} \theta(k)x(k)$. Gradient of loss function corresponding to parameter $\theta(k)$ is

$$\frac{\partial L(\theta)}{\partial \theta_k} = \frac{-1}{N} \sum_{l=1}^{N} (y(l) - f(\theta(x(l))))x_k^l \tag{12.5}$$

Parameter $\theta(k)$ is updated in the direction of negative gradient descent,

$$\theta_{k+1} = \theta_k + \eta \frac{1}{N} \sum_{l=1}^{N} (y(l) - f(\theta(x(l))))x_k^l \tag{12.6}$$

While determining the direction of movement, GD method utilizes only the function's first-order derivative. For convex objective function, global optimum solution is obtained using this method. GD method usually converges at slower speed and when reaching at global optimum, it converges at much slower speed.

12.2.2 Gradient descent variants

Depending on the amount of data that is used to compute objective function gradient, GD method can be categorized into three variants. GD methods have low computational cost with low accuracy. Batch gradient descent, SGD, and mini-batch gradient descent are the three variants of GD.

12.2.2.1 Batch gradient descent

Batch gradient descent sometimes often called as Vanilla gradient descent. It computes error for an example only after a training epoch. This method calculates the gradient of entire training data set to perform just one update. Gradient of cost function for whole data set is updated with respect to parameter θ.

$$\theta(l+1) = \theta(l) - \eta \nabla_\theta f(\theta) \tag{12.7}$$

As batch gradient descent method computes gradient of entire data set for each update, this method can be very slow. One of its advantage is computational efficiency as it yields a steady error gradient and gives a stable convergence. Sometimes, steady error gradient may result in that state of convergence which is not the optimal that model can accomplish. Batch gradient descent method also does not allow online update. For convex error surface, batch gradient descent method converges to global minimum. For nonconvex surface, this method converges to local minimum. For the N number of samples and M number of features, the computational complexity per iteration of batch gradient descent method will be O(NM).

12.2.2.2 *Stochastic gradient descent*

In order to overcome the problems of high computational complexity per iteration of batch gradient method, SGD method was proposed. Instead of directly computing the gradient of objective function, SGD method computes the gradient of one random sample. Stochastic gradient provides an impartial estimates of objective function gradient [5]. SGD performs a parameter update of single training example $u^{(i)}$ per iteration instead of objective function gradient of all the training examples.

$$\theta(l+1) = \theta(l) - \eta\nabla_{\theta}f(\theta)\big(\theta(l); u^{(i)}\big) \tag{12.8}$$

Path taken by SGD method to reach the minima is typically noisy as compared to batch gradient descent method because in SGD method, only one random sample is chosen for updating the gradient per iteration. Fig. 12.1A and B shows the usual path taken to reach the minima by SGD method and by batch gradient descent method. It is clear from the figure that SGD method path to reach the minima is much noisy than batch gradient descent method, but it does not matter. Only criteria are to reach the minima with considerably shorter time is considered. As SGD method is noisier than typical gradient descent, to reach the minima SGD method took higher number of iteration because of its randomness nature to choose the sample. Although SGD method requires higher number of iteration as compared to typical gradient descent method but still SGD method is computationally much less expensive than gradient descent method. Cost of SGD method is independent of number of samples and this method can also achieve sublinear convergence speed [6].

For large data sets, SGD method significantly accelerates the calculation by removing the computational redundancy as it performs one update at a time whereas batch gradient descent performs redundant computation. One major problem of batch gradient descent method is it does not allow online update, SGD method overcomes this disadvantage. Computational complexity per iteration for SGD method is O(M) because this method exploits single sample per iteration, where M is the number of features. In

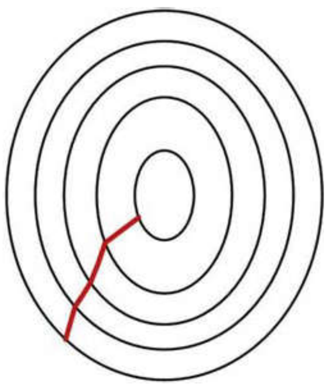

FIGURE 12.1 (A) Path taken by batch gradient descent.

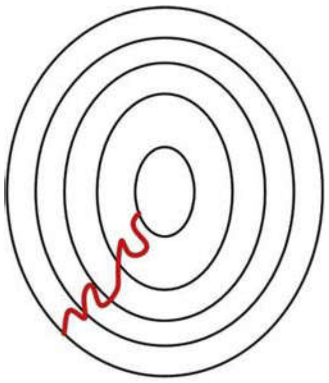

FIGURE 12.1 (B) Path taken by stochastic gradient descent.

case of batch gradient descent, deterministic gradient may lead objective function to converge a local minimum while SGD method have better chance to achieve a global optimal solution. SGD method's fluctuations facilitates it to reach to a possibly better local minimum. But this fluctuation always exists in SGD method, this ultimately complicates convergence to reach global minima. In the existing literature, it has been presented that on slowly decreasing the learning rate, SGD method converges at same speed as batch gradient descent method.

Choice of an appropriate learning rate is still a challenging task while applying SGD method of optimization. A too large learning rate obstructs the convergence of objective function and loss function oscillates to and fro across the ravine. A too small learning rate results in slow convergence speed. One of the solution of the problem of learning rate is to setup a predefined index of learning rates and during the learning process, learning rates are adjusted accordingly [7]. These predefined list of learning rates are needed to be defined in advanced based on the characteristic of data set. Same learning rate for all parameters are not always suitable to use. For less frequently occurring features, a high learning rate is estimated [8]. Objective function in SGD method is trapped in infinite number of local minimum and to avoid this problem is a challenging task. Several works suggest that this problem comes from "saddle point," not from the local minimum values [9]. Saddle points has gradient values as zero in entire directions. As it has positive slope in a specific direction and also has negative slope in opposite direction. It is necessary for SGD to escape from these saddle points. Various works have been developed to escape from these points [10].

12.2.2.3 Mini-batch gradient descent

In SGD method, noise causes as a result of random selection causes the gradient descent oscillating. Gradients variance is large and also the direction of movement is biased in SGD method. Mini-batch gradient descent technique was offered [5] to compromise between both methods. Mini-batch gradient descent method updates the parameter per iteration by using n number of samples at a time. n can vary depending on the applications and it ranges between 50 and 256 [1].

$$\theta(l+1) = \theta(l) - \eta\nabla_\theta f(\theta)\left(\theta(l); u^{(ii+n)}\right) \tag{12.9}$$

Mini-batch gradient descent method lessens the gradients variants which led to further steady convergence. This method can also make the use of optimized matrix while calculating the gradients and it makes this method very efficient. Computational complexity is less in mini-batch gradient descent method. For a large neural networks with redundant training sets, it is best to use this method.

12.2.3 Gradient descent optimization algorithms

There are many challenges to SGD method such as how to adjust learning rate, how to escape from being trapped to saddle points during search process and how to increase the convergence rate. To improve SGD, several algorithms are widely used to deal these challenges.

12.2.3.1 Momentum

Momentum method [11] is based on the concept of mechanics, which simulates the inertia of objects. This method helps SGD to accelerate in the appropriate direction and to dampen the oscillations in the direction of high curvature by combining gradients with opposite signs. Momentum stores the impact of preceding updated direction on the subsequent iteration. It makes direction to keep going as in previous direction. Momentum method introduces the variable v which symbolizes the direction and speed of parameter's movement. It accelerates SGD in relevant direction by considering a momentum term γ vary between 0 and 1 along with gradient descent. Momentum term gets multiplied with previous update and provides a friction factor. Mathematically, v is formulated as

$$v_t = \gamma v_{t-1} - \eta\nabla_\theta f(\theta) \tag{12.10}$$

Term γv_{t-1} gives the friction factor, it helps in further increment of velocity as γ is less than 1. Weight change can be expressed as

$$\Delta w_t = \gamma\Delta w_{t-1} - \eta\nabla_\theta f(\theta) \tag{12.11}$$

Parameter will be updated as

$$\theta(l+1) = \theta(l) - v_t \tag{12.12}$$

A suitable momentum term plays an important part in speed up the convergence with low learning rate. Momentum term increases for those gradients which are parallel to previous direction and thus previous velocity can speed up the search. For dimensions whose gradients change direction, momentum term reduces updates and decelerate the search speed. During the training process, it is necessary to get away from the local minimum so that we can gain faster convergence [12]. Momentum term in momentum method also plays a significant part in decreasing the oscillations of convergence.

Selecting a proper momentum term is a challenging task. With small momentum term, it is difficult to achieve faster convergence speed. For large momentum term, recent point might leap out of optimal value. It is verified in various studies that appropriate value of momentum term is 0.9 [1].

12.2.3.2 Nesterov accelerated gradient

In momentum method, gradients at current position are calculated first and then it jumps to the updated accumulated gradient's direction. Nesterov Accelerated Gradient (NAG) [13] method first jumps to the previous accumulated gradient's direction and after that it calculates the gradient and make a correction. Instead of calculating the objective function gradient w.r.t. current parameter θ, in NAG method gradient is calculated with respect to future position of parameter $\theta - \gamma v_{t-1}$. In other words, instead of computing gradient of the current position, gradient of future position is calculated in NAG method. First step in NAG method is to calculate the approximate next position of parameters.

$$\widehat{\theta} = \theta - \gamma v_{t-1} \tag{12.13}$$

Velocity is updated by computing gradients with respect to approximate future position of parameters.

$$v_t = \gamma v_{t-1} - \eta \nabla_\theta f(\theta - \gamma v_{t-1}) \tag{12.14}$$

Now parameter is updated as

$$\theta(l+1) = \theta(l) - v_t \tag{12.15}$$

NAG method have more gradient information as compared to momentum method. One of the main issue is in what manner to decide the learning rate. Oscillations occur when search point is close to optimal point. Decay factor d in learning rate is used in SGD momentum method. With decay factor, learning rate decreases as the number of iteration goes on increasing [14]. Learning rate at i-th iteration is given as,

$$\eta_i = \frac{\eta_0}{1 + d \cdot i} \tag{12.16}$$

where η_0 is initial learning rate and d is decay factor of learning rate and its value ranges between 0 and 1. Learning rate decays faster when d has large value and vice-versa. Adjusting the learning rate manually influences the SGD method. Choosing an appropriate learning rate is a tricky problem [15]. To deal with optimization problems in DNN, various adaptive methods were introduced in which learning rate adjusted automatically. These methods have faster convergence rate and are free of parameter adjustments.

12.2.3.3 AdaGrad

AdaGrad [8] method adapts the learning rate dynamically based on accumulated previous gradient. To deal with sparse data, AdaGrad method is well suited. For frequent parameters, AdaGrad perform smaller updates and for infrequent parameters it performs larger update. In traditional gradient method, learning rate is fixed for updating all parameters, while in AdaGrad method, learning rate is not fixed. It makes use of an individual learning rate for all parameters at each iteration. Gradient of objective function with respect to parameter θ at i-th iteration is given by,

$$g_i = \nabla_\theta f(\theta_i) \tag{12.17}$$

AdaGrad modifies learning rate at each iteration for all parameters based on accumulated previous gradients.

$$\theta_{i+1} = \theta_i - \eta \frac{g_i}{G_i} \tag{12.18}$$

where $G_i = \sqrt{\sum_{k=1}^{i} (g_k)^2 + \varepsilon}$. First term in denominator denotes the accumulated previous gradients. ε is a smoothing term. One of the key benefit of AdaGrad technique is it eliminate the requirement to manually adjust the learning rate. Two main issues of AdaGrad method are:

(i) Although it adapts learning rate dynamically but it still requires to set global learning rate manually.
(ii) Parameter updates have squared gradients in denominator and as it is a positive term, it gets accumulated in the course of training. This bring about learning rate to reduce and ultimately results in an ineffective parameter update.

12.2.3.4 AdaDelta

To solve the issue of monotonically decreasing learning rate, AdaDelta [16] was proposed. Instead of accumulating all previous squared gradients, it focuses on only accumulated previous gradients in a window over a period. Exponentially declining average of accumulated previous gradients are defined as

$$G_i = \sqrt{\beta G_{i-1} + (1-\beta) g_i^2} \tag{12.19}$$

where β as a decay factor similar to momentum term and β is set to around 0.9 [1]. Declining average rely upon only on preceding average and recent gradient.

12.2.3.5 Adaptive moment estimation

Adaptive Moment Estimation (Adam) [15] is an alternative technique which offers adjustable learning rate for each parameters. It incorporates the AdaDelta and momentum approaches. Adam method stores exponentially declining average of accumulated squares of previous gradients G_i like AdaDelta along with exponentially declining average of preceding gradients like momentum method. The estimate of first moment (i.e., mean) of gradient is expressed as m_i. It is expressed as

$$m_i = \beta_1 m_{i-1} + (1-\beta_1) g_i \tag{12.20}$$

G_i is the estimate of the second moment (i.e., variance) of the gradient. Mathematically, it is defined as

$$G_i = \sqrt{\beta_2 G_{i-1} + (1-\beta_2) g_i^2} \tag{12.21}$$

where β_1 and β_2 are the exponential declining rates. It is observed that when exponential declining rates are small (i.e., their value close to 1), during initial iterations method is biased toward zero. To counter these biases, first and second moments with bias corrected are computed.

$$\widetilde{m_i} = \frac{m_i}{1 - \beta_1^i} \tag{12.22}$$

$$\widetilde{G_i} = \frac{G_i}{\sqrt{1 - \beta_2^i}} \tag{12.23}$$

These bias-corrected moments are used to update the parameter. Update formula for Adam method is

$$\theta_{i+1} = \theta_i - \eta \frac{\widetilde{m_i}}{\widetilde{G_i}} \tag{12.24}$$

$$\theta_{i+1} = \theta_i - \eta \frac{\sqrt{1 - \beta_2^i}}{\left(1 - \beta_1^i\right)} \frac{m_i}{G_i + \varepsilon} \tag{12.25}$$

Default values of β_1 is 0.9, β_2 is 0.999 and for ε is 10^{-8}.

12.2.3.6 AdaMax

G_i factor in the Adam is oppositely proportional to the l_2 norm of preceding gradients and recent gradients.

$$G_i = \sqrt{\beta_2 G_{i-1} + (1 - \beta_2)|g_i|^2} \tag{12.26}$$

Infinity norm i.e., l_∞ generally exhibits stable behavior. In AdaMax [15] method, G_i with infinity norm converges to more stable value. Infinity norm constrained G_i^∞ is expressed as,

$$G_i^\infty = \sqrt{\beta_2^\infty G_{i-1} + \left(1 - \beta_2^\infty\right)|g_i|^\infty} \tag{12.27}$$

AdaMax parameter will be updated using expression,

$$\theta_{i+1} = \theta_i - \eta \frac{\widetilde{m_i}}{G_i} \tag{12.28}$$

Default values of η, β_1, and β_2 are 0.002, 0.9, and 0.999, respectively.

12.2.3.7 Nesterov accelerated adaptive moment estimation

From above, it is clear that Adam can be regarded as grouping of AdaDelta and Momentum method. AdaDelta accounts for the exponentially declining average of preceding squared gradients and momentum contributes the exponentially declining average of preceding gradients. Nesterov Accelerated Adaptive Moment Estimation (Nadam) [17] is the combination of Adam method and NAG method. Look-ahead momentum vector is applied straight to update the present parameter of NAG and bias-corrected estimate of current momentum is also used for parameter update. Nadam parameter update can be expressed as

$$\theta_{i+1} = \theta_i - \frac{\eta}{\widetilde{G_i}} \left(\beta_i \widetilde{m_i} + \frac{(1-\beta_1)\cdot g_i}{1-\beta_1^i} \right) \tag{12.29}$$

SGD method is quite popular method of optimization but this method has some issues also. One of the issue is sublinear convergence rate of SGD method. Variance of gradients is very large in SGD method. To solve this issue of convergence rate and large variance, there are some variance reduction methods.

Stochastic average gradient [18] method to improve the convergence rate is one of the variance reduction method. In this method, for each update only gradient of one sample needs to be calculated instead of calculating gradients of all samples.

Updated parameter per iteration will be

$$\theta_{k+1} = \theta_k - \frac{\eta}{N} d \tag{12.30}$$

where $d = d - \nabla_\theta f(\theta_i) + \nabla_\theta f(\theta_{k-1})$ and $\nabla_\theta f(\theta_i)$ is the previous accumulated gradient. Memory requirement for this method is much higher than as compared to traditional SGD method.

Stochastic average gradient method achieves linear convergence rate. One of the requirement of this method is that objective function must be smooth and should have convex in nature. Thus for the optimization problems which in nonconvex in nature, Stochastic Variance Reduction Gradient (SVRG) [6] is proposed. Instead of calculating gradients of all samples in each iteration, SVRG computes gradients of sample for every t iteration.

Average gradients of all samples over the interval t is given as

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta f(\theta_i) \tag{12.31}$$

Next update will be

$$\theta_{k+1} = \theta_k - \eta(\nabla_\theta f(\theta_k) - \nabla_\theta f(\theta_i) + \mu) \tag{12.32}$$

SVRG introduces a concept known as variance reduction in which constant upper bound on gradient variance are assumed. To achieve linear convergence rate, variance of gradients needs to be continuously decreased. SVRG requires less memory because it not store all gradients in memory as compared to stochastic average gradient method.

12.3 High-order optimization method

First-order method of optimization such as SGD and its variant are very popular in machine learning because of their simplicity of applications and low computation cost per iteration. But it has some drawbacks also such as slow convergence rate, low accuracy, sensitivity to learning rate tuning, trouble in get away from saddle points. To overcome these drawbacks, there have been latest attention toward second-order method. Second-order method use gradient information along with capturing curvature information. When second-order method is used in context of machine-learning applications, it provides faster convergence rate. This method also gives stable learning rate tuning. Here the chapter discuss few second-order methods which includes Quasi-Newton method, Gauss-Newton method, and Hessian-free Newton method.

12.3.1 Hessian-free Newton method

For a given unconstrained and smooth convex optimization problem, Newton's method [19] updated direction is based on second-order Taylor series approximation. Newton's method iteration is obtained by

$$\theta_{k+1} = \theta_k - \eta\left(\nabla_\theta^2 f(\theta_k)\right)^{-1} \nabla_\theta f(\theta_k) \text{ for k} = 1, 2, 3, ... \tag{12.33}$$

where $\left(\nabla_\theta^2 f(\theta_k)\right)$ is the Hessian matrix of $f(\theta)$ at $\theta = \theta_k$. Using second-order Taylor series approximation, Objective function can be minimizing as

$$f(\theta) = f(\theta_k) + \nabla_\theta f(\theta_k)^T (\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T \nabla_\theta^2 f(\theta_k)(\theta - \theta_k) \tag{12.34}$$

Assuming Hessian is positive definite. Newton's method attains quadratic convergence rate. This technique applies consecutive local rescaling at each iteration. Instead of solving Newton's method employing matrix factorization technique, one can figure out it inexactly using an iterative method. Many iterative linear systems methods require only Hessian-vector product instead of Hessian itself. These type of methods which solves accurate enough are known as Hessian-free Newton method. This method guarantees a faster rate of convergence. With the dimension d and smooth objective function $f(\theta)$, cost for evaluating $\nabla_\theta^2 f(\theta)$ is small multiple of computing cost of $\nabla_\theta f(\theta)$. Hessian-free Newton method requires $O\left(d^2\right)$ storage without forming the Hessian. While this method save computation cost but it comes at the expense of storage requirement. To yield an effective iteration, it is not necessary Hessian matrix to be as precise as the gradient. In context of large scale data, iteration is much forbearing to noise in case of Hessian-free Newton method as compared to gradient method.

A smaller sample is employed for defining the Hessian in subsample Hessian -ree Newton method. For a subsample size of a_k, stochastic Hessian estimate will be

$$\nabla_\theta^2 f_{a_k}(\theta) = \frac{1}{|a_k|} \sum \nabla_\theta^2 f(\theta) \tag{12.35}$$

If subsample size of a_k is chosen large, then curvature information captured is productive. Hessian-vector products capture the curvature information. For small subsample size of a_k, cost of each product can be decreased considerably which further reduce the cost per iteration. If achieved properly, Hessian subsampling is robust and efficient [15]. Step computation cost in subsampling Hessian-free Newton method for *max* total number of iteration can be computed as,

$$cost = (max \times factor \times g_{cost}) + g_{cost} \tag{12.36}$$

where $factor \times g_{cost}$ is the cost of one Hessian-vector product. g_{cost} is the cost of computing gradient estimate $\nabla_\theta f_{a_k}(\theta)$. In Hessian subsampling method, *factor* should be chosen very small such that $max \times factor \approx 1$. This leads to the computation cost per iteration comparative to that of SGD method. It is assumed that chosen subsample size is large enough such that by considering these subsamples, Hessian estimate is sensible. Hessian-free Newton method are usually practiced for solving the nonconvex optimization problems. To ensure positive definite Hessian approximation, Gauss-Newton approximation of Hessian is employed.

$$G_{a_k}(\theta) = \frac{1}{|a_k|} \sum J_h(\theta)^T H_l(\theta) J_h(\theta) \tag{12.37}$$

where J_h captures the stochastic gradient information and H_l captures loss function second-order information. For analysis the geometry of minimizer Objective function, several numerical tests have been designed and presence of negative curvature is proved. When compared to first-order methods, Newton's method have more memory requirement as for $(n \times n)$ Hessian it requires $O(n^2)$ of storage per iteration while gradient method requires $O(n)$ of storage per iteration for n-dimensional gradient. Newton's method is more sensitive to numerical errors but gradient descent methods are robust against these errors.

12.3.2 Quasi-Newton method

Quasi-Newton method (QNM) is one of the important developments in the field of nonlinear optimization. This method is used when Newton method are difficult to use or when computing Hessian is too expensive per iteration. This method approximates Hessian by using only the gradient information. QNM can be applicable for both convex and nonconvex problems. QNM iteration for minimizing the objective function, which is twice differentiable, is

$$\theta_{k+1} = \theta_k - \eta B_k \nabla_\theta f(\theta_k) \tag{12.38}$$

where B_k is the symmetric positive definite approximation of $\left(\nabla_\theta^2 f(\theta_k)\right)^{-1}$. Instead of updating each iteration by computing second-order derivative computation, in QNM sequence B_k updated dynamically for each iteration. As B_k already contains information about Hessian, B_k can be updated by using suitable matrix.

$$\nabla_\theta f(\theta_{k+1}) = \nabla_\theta f(\theta_k) + B_{k+1}(\theta_{k+1} - \theta_k) \tag{12.39}$$

Suppose,

$$y = \nabla_\theta f(\theta_{k+1}) - \nabla_\theta f(\theta_k) \tag{12.40}$$

and

$$s = \theta_{k+1} - \theta_k \tag{12.41}$$

this gives us Secant equation,

$$B_{k+1} \cdot s = y \tag{12.42}$$

Different QNM such as symmetric rank one (SR1), Davidon-Fletcher-Powell (DFP), Broyden-Fletcher-Goldfarb-Shanno (BFGS), and Broyden class computes B_{k+1} form B_k differently. In case of SR1 update, Hessian approximation is obtained by

$$B_{k+1} = B_k + \frac{(y - B_k s)(y - B_k s)^T}{(y - B_k s)^T s} \tag{12.43}$$

New inverse Hessian approximation is given by expression

$$C_{k+1} = C_k + \frac{(s - C_k \cdot y)(s - C_k \cdot y)^T}{(s - C_k \cdot y)^T y} \tag{12.44}$$

In general, SR1 update method of QNM is simple to implement and has low computation cost. But it has one drawback, this method does not preserve positive definiteness of Hessian matrix approximation. To overcome this shortcoming of SR1 method, BFGS method is proposed. In BFGS method, Hessian approximation is given by

$$B_{k+1} = B_k - \frac{B_k \cdot s \cdot s^T \cdot B_k}{s^T \cdot B_k \cdot s} + \frac{y \cdot y^T}{y^T \cdot s} \tag{12.45}$$

Inverse Hessian approximation of BFGS method is calculated as

$$C_{k+1} = \left(I - \frac{s \cdot y^T}{y^T \cdot s}\right) C_k \left(I - \frac{y \cdot s^T}{y^T \cdot s}\right) + \frac{s \cdot s^T}{y^T \cdot s} \tag{12.46}$$

BFGS update preserve positive definiteness under appropriate conditions and has low computation cost. BFGS update has local superlinear rate of convergence [20] without the need to solve linear systems. However, this method has some issues also. These concerns need to tackle to have an efficient method. Even when the Hessian matrix are sparse, updated inverse Hessian approximation yields dense matrix. This problem restricts BFGS method to use for small scale and midscale data set. This problem can be solved by using limited memory scheme such as L-BFGS method [21] in which B_k not needed be formed explicitly. Individual product can be computed only using current components of series of displacement pairs (s, y)

Development of QNM from deterministic to stochastic form, iteration will take the form as

$$\theta_{k+1} = \theta_k - \eta B_k \nabla_\theta f(\theta_k, \xi_k) \tag{12.47}$$

In spite of convergence rate of stochastic iteration of QNM is always slower than sublinear [22]. But this method could be well prepared to deal with ill-conditioning. For SGD method, constant depends on conditioning of $\left\{\nabla_\theta^2 f(\theta_k)\right\}$, whereas in QNM, constant is not dependent on conditioning of Hessian on satisfying $B_k = \left(\nabla_\theta^2 f(\theta_k)\right)^{-1}$. SGD method has low computational complexity as it requires only to compute gradient of objective function $\nabla_\theta f(\theta_k, \xi_k)$ per iteration. In case of QNM, each iteration requires computation of product $B_k \nabla_\theta f(\theta_k, \xi_k)$, this computation makes it more expensive than SGD method. To address this additional per iteration cost, mini-batch gradient estimation is employed. Usually mini-batch of size between 20 and 50 is considered instead of large mini-batch size. When mini-batch gradient estimation is considered, additional per iteration cost is only marginal. In stochastic QNM, gradient $\nabla_\theta f(\theta_k, \xi_k)$ is the noisy estimate of $\nabla_\theta f(\theta_k)$. While updating C_k, it involves difference of gradient estimates y. Updating process may cause to yield poor curvature estimate. Even a single noisy gradient may remain for numerous iteration. To circumvent differencing noisy gradient estimate, one way is to choose identical sample while calculating gradient difference [23].

12.3.3 Gauss-Newton method

It is a traditional approach to solve nonlinear least squares optimization problems. Gauss-Newton method (GNM) minimize the problems in which objective function is sum of squares. One of the pros of using GNM is that it uses only first-order information to construct approximation of Hessian. Though if Hessian may be indefinite, this method guaranteed approximation of Hessian be positive semidefinite. GNM ignores second-order interaction between parameter elements, which could cause loss of curvature information. GNM aims to minimize the objective function which is sum of squares. First step is to define the objective function, which is to be minimized.

$$\|f(\theta_k, \xi_k)\|_2^2 = \sum_{i=1}^{m} f_i(\theta_k, \xi_k)^2 \tag{12.48}$$

Linearize $f(\theta_k, \xi_k)$ around θ_k,

$$f(\theta, \xi) \approx f(\theta_k, \xi_k) + \nabla_\theta f(\theta_k, \xi_k)(\theta - \theta_k) \tag{12.49}$$

Substitute affine transformation for $f(\theta, \xi)$ in least square problem.

$$f(\theta_k, \xi_k) \approx \| f(\theta_k, \xi_k) + \nabla_\theta f(\theta_k, \xi_k)(\theta - \theta_k)\|_2^2 \tag{12.50}$$

If $\nabla_\theta f(\theta_k, \xi_k)$ has full column rank, solution will be

$$\theta_{k+1} = \theta_k - \eta(\nabla_\theta f(\theta_k, \xi_k))^T (\nabla_\theta f(\theta_k, \xi_k))^{-1} (\nabla_\theta f(\theta_k, \xi_k))^T f(\theta_k, \xi_k) \tag{12.51}$$

$$\theta_{k+1} = \theta_k - \eta(\nabla_\theta f(\theta_k, \xi_k))^+ f(\theta_k, \xi_k) \tag{12.52}$$

There are some challenges in applying GNM. Gauss-Newton matrix is generally singular or nearly singular. This issue can be solved by adding a positive multiple of identity matrix in to Gauss-Newton matrix. Hessian-free Newton method and stochastic quasi-Newton method can be applicable with Gauss-Newton approximation for least squared loss functions. Scaling matrices in GNM have guaranteed positive definite. For other loss function also, GNM can be generalized [24]. While training DNNs, generalized

GNM can be applied by considering loss function and prediction function. By using this method, networks computation is performed by loss function rather than prediction function.

12.3.4 Natural gradient method

Newton's method is invariable to linear transformation of parameter θ, while natural gradient method [25] is invariable to differentiable and reversible transformations. Gradient descent algorithms are formulated in space of prediction function instead of parameter. Natural gradient descent method will move parameters quickly in the direction which has less impact of decision function. Before formulating Natural gradient method in prediction function space, geometry of prediction function space must be explained. Amari's technique [26] on information theory gives us an idea about the geometry. Space S of prediction function is a family of densities $j_\theta(x)$, where $\theta \in \Re^n$. Density $j_\theta(x)$ satisfies the normalization condition

$$\int j_\theta(x)dx = 1 \tag{12.53}$$

Derivative of density satisfies the identity for $\forall\ n > 0$,

$$\begin{aligned}\int \frac{d^n j_\theta(x)}{d\theta^n}dx &= \frac{d^n}{d\theta^n}\int j_\theta(x)dx \\ &= \frac{d^n 1}{d\theta^n} \\ &= 0\end{aligned} \tag{12.54}$$

By observing Kullback-Leibler (KL) divergence, we can visualize the effect on density by adding a small quantity $\delta\theta$ to parameter.

$$D_{KL}(j_\theta || j_{\theta+\delta\theta}) = E_{j_\theta}\left(\log\frac{j_\theta(x)}{j_{\theta+\delta\theta}(x)}\right) \tag{12.55}$$

Approximating divergence with a second-order Taylor expansion,

$$D_{KL}(j_\theta || j_{\theta+\delta\theta}) \approx -\delta\theta^T E_{j_\theta}\left[\frac{\partial \log(j_\theta(x))}{\partial\theta}\right] - \frac{1}{2}\delta\theta^T E_{j_\theta}\left[\frac{\partial^2 \log(j_\theta(x))}{\partial\theta^2}\right]\delta\theta \tag{12.56}$$

From above Eq. (12.54), first term in Eq. (12.56) becomes zero.
Fisher information matrix can be defined as

$$G(\theta) = -E_{j_\theta}\left[\frac{\partial^2 \log(j_\theta(x))}{\partial\theta^2}\right] \tag{12.57}$$

Thus Eq. (12.56) becomes

$$D_{KL}(j_\theta || j_{\theta+\delta\theta}) \approx \frac{1}{2}\delta\theta^T G(\theta)\delta\theta \tag{12.58}$$

Fisher information matrix is always positive semidefinite and symmetric. Therefore, every small region of space S are similar to small region of Euclidian space. For density estimation, objective function is negative log likelihood,

$$f(\theta) = \frac{1}{k}\sum_{i=1}^{k} -\log(j_\theta(x_i)) \approx D_{KL}(P\,||\,j_\theta) + constant \tag{12.59}$$

where x_i are the independent training samples and P is unknown distribution which training samples have. Hessian matrix, $\nabla_\theta^2 f(\theta)$ will be,

$$\nabla_\theta^2 f(\theta) = -E_P\left[\frac{\partial^2 \log(j_\theta(x))}{\partial\theta^2}\right] \tag{12.60}$$

When optimality is approached, density function approaches to P distribution and thus Fisher information matrix approaches the Hessian matrix. Natural gradient method and Newton method performs in a similar manner when reaching at optimal solution. In large learning systems, numerical computation of Fisher information matrix is difficult task. To overcome this problem, subset of training examples is considered for computing Fisher information matrix [27].

12.4 Derivative-free optimization methods

In many optimizations problems, objective function, and constraints are computed by a "black box," which does not provide derivative information. Objective function computed by black box may also include some noise and calling black box every time may be expensive. These challenges of optimization problems necessitate to use derivative-free optimization (DFO). DFO approximates the objective function explicitly without approximating its derivative. Thus DFO methods are those methods which do not require derivative information for solving optimization problem. Here the chapter mainly focus on DFO methods for convex objective and methods for stochastic optimization.

12.4.1 Methods for convex objective

A function f is said to be convex if its space, S, is convex in nature. For any two point x and y in space S ($S \in \Re^n$) if it satisfies following inequality,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \forall \alpha \in [0,1] \tag{12.61}$$

Local solution is always a global solution for convex optimization problem, whereas for nonconvex problems, there exists many local solutions. For deterministic optimization problem, objective is the objective function $f(\theta)$ minimization.

$$\overset{minimize}{\theta}\; f(\theta), \tag{12.62}$$

subjected to. $\theta \in \Re^n$

Under appropriate additional assumptions, for convex objective function $f(\theta)$ it satisfies

$$\lim_{k\to\infty} f(\theta_k) - f(\theta_*) = 0 \tag{12.63}$$

where θ_* is global minimizer and is satisfies. $f(\theta_*) \leq f(\theta)\ \forall\ \theta \in \Re^n$

For unconstrained convex objective function,

$$f(\theta_k) - f(\theta_*) \leq \varepsilon \tag{12.64}$$

R-linear convergence of directional direct search (DDS) method dependence reduced to $\log(\varepsilon^{-1})$ from ε [28]. In DDS method, on restricting the upper limit on step size α_k and for $u > 0$,

$$f(p_i) \leq f(\theta_k) - u\alpha_k^2 \tag{12.65}$$

In case of convex stochastic optimization, task is to find the solution of stochastic problem.

$$\underset{\theta}{minimize}\ f(\theta) = E_\xi[f(\theta, \xi)] \tag{12.66}$$

Zeroth order information is referred as bandit feedback. Multiarmed bandit problems can be formulated as a sequential allocation problem. In multiarmed bandit problem [29], a gamester wants to lessen the losses incurred by drawing wrong slot machine arms out of A arms. Total length of discrete sequence is assumed to be T. At time instant k, gamester already knows the losses associated with previous instants. Gamester's loss at k-th instant is $f(\theta_k, \xi_k)$. It is assumed that expectation

$$E_\xi[f(\theta, \xi)] = f(\theta) \quad \forall\ \theta \in (1, 2, \ldots.., A) \tag{12.67}$$

To minimize the expected total loss, best long run strategy is to constantly play. Over the course of T drawings, sequence of realized losses contains $\{(f(\theta_1, \xi_1), \ \ldots\ldots..(f(\theta_T, \xi_T))\}$. Cumulative regret contains the metric of gamester's performance. Expected cumulative regret is given as

$$E_\xi[r_T(\theta_1, \ldots.., \theta_T)] = E_\xi\left[\sum_{k=1}^{T} f(\theta_k, \xi_k)\right] - Tf(\theta_*) \tag{12.68}$$

where r_T is cumulative regret, and sequence of ξ_k are identically distributed and are independent to each other. Extending this problem to infinite armed bandits [30] corresponds to DFO. For infinite armed bandit problem, vector ξ defines linear function. In this linear regime, gamester incurs loss.

$$f(\theta_k, \xi_k) = \xi_k^T \theta_k \tag{12.69}$$

Expected cumulative regret is expressed as

$$E_\xi[r_T(\theta_1, \ldots.., \theta_T)] = E_\xi\left[\sum_{k=1}^{T} f(\theta_k, \xi_k)\right] - \underset{\theta}{min}\ E_\xi\left[\sum_{k=1}^{T} f(\theta_k, \xi_k)\right] \tag{12.70}$$

Based on type of bandit feedback, it can be categorized into single-point bandit feedback or multipoint bandit feedback.

12.4.2 Methods for stochastic optimization

To solve the stochastic optimization problem, it is assumed that random variable ξ is independent and identically distributed. Variance of objective function is bounded as

$$E_{\xi}\left[(f(\theta,\xi) - f(\theta))^2\right] < \sigma^2 < \infty \tag{12.71}$$

If derivative of function exists, under certain regularity condition it follows,

$$\nabla_{\theta} f(\theta) = E_{\xi}[\nabla_{\theta} f(\theta,\xi)] \tag{12.72}$$

Using [31], derivative of objective function is approximated by using central differences. Derivative is approximated as

$$g(\theta_k, \mu_k, \xi_k) = \left[\frac{f(\theta_k + \mu_k e_n, \xi_n^+) - f(\theta_k - \mu_k e_n, \xi_n^-)}{2\mu_k}\right] \tag{12.73}$$

To achieve convergence, bounds are applied on objective function, step size. There are always limitations on series of step sizes. Convergence in distribution [32] is given as

$$\frac{1}{k^{\gamma}}(\theta_k - \theta_*) \rightarrow N(0, H) \tag{12.74}$$

where H is a covariance matrix in its elements consists of algorithm parameters and derivative of objective functions. It has been shown that $\gamma = 1/3$ [33]. Gradient estimation in common random number regimes is

$$g(\theta_k, \mu_k, \xi_k) = [\delta_c(f(.,\xi_k); \theta_k; e_n; \mu_k)] \tag{12.75}$$

To compute finite difference approximation, single realization of ξ_k is employed. By considering different realizations of ξ, $f(\theta_k)$ can be computed more accurately. These belongs to sample average approximation methods. Performance of these methods depends on sample size and accuracies used per iteration. These methods shown to achieve convergence for smooth objective for sequence of sample sizes [34]. Sample size can also be dynamically selected from iteration to balance the deterministic and stochastic errors. Stochastic error is given by,

$$\left| f(\theta_k) - \frac{1}{p_k}\sum_{i=1}^{p_k} f(\theta_k, \xi_{k,i}) \right| \tag{12.76}$$

Deterministic error using first-order Taylor approximation is given as,

$$\left| f(\theta_k - \eta\nabla_{\theta} f(\theta_k)) - (f(\theta_k) - \eta\nabla_{\theta} f(\theta_k)^2) \right| \tag{12.77}$$

12.5 Optimization methods challenges and issues in machine learning

While applying optimization methods to optimize the models in machine learning, some challenges are there. Issues such as insufficient training data, complexity of machine-learning models, nonconvex objective function are very common to occur. Deep neural networks (DNNs) may have sometimes insufficient data to train the model. In the absence of sufficient training data, it may cause high variances and overfitting [35]. Inappropriate selection of learning rate and overfitting may affect the accuracy of model. Number of iteration in SGD method also affect the performance of model which makes it unable to converge.

For the nonconvex optimization problem, it may be possible to attain a local optimal position instead of global optimal position. In case of nonconvex optimization problem, several local optimal solutions are present and choosing global solution out of this local solution is difficult task. There are two approaches to solve the nonconvex problems. First approach transforms the nonconvex problem into convex problem and after that solve it using convex optimization methods. Other approach is to apply specific optimization methods to solve it directly. In sequential models, if the sequence is very large, samples gets truncated and this truncation may lead to deviation. If length of data is not an integer times of sample size, then some previous sampled data are added to sample size while training the model. Complexity of machine-learning models increases. Some of particular optimization problems are developed for specific machine-learning application which cannot be applied to other applications.

12.6 Conclusion

Theoretical basis of fundamental optimization methods is described here. Based on gradient information, Optimization method could be categorized into first-order techniques, high-order techniques and derivative-free optimization techniques. First-order techniques including GD method and its types are presented. Popular algorithms of gradients descent methods are also briefly discussed. SGD method is thoroughly explained along with its advantages and issues. SGD method has low convergence rate and low computational complexity. To improve the convergence rate and to use curvature information, high-order optimization methods were employed in literature. High-order techniques, namely, Newton's method, GNM, QNM, natural gradients methods along with some derivative-free techniques are also explained here. Challenges and issues occurs in optimization problems is also presented here.

References

[1] S. Ruder, An Overview of Gradient Descent Optimization Algorithms, September 15, 2016 arXiv preprint arXiv:1609.04747.

[2] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization, Math. Comput. 24 (111) (1970) 647–656.

[3] A.S. Berahas, R.H. Byrd, J. Nocedal, Derivative-free optimization of noisy functions via quasi-Newton methods, SIAM J. Optim. 29 (2) (2019) 965–993.

[4] L.M. Rios, N.V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global Optim. 56 (3) (July 2013) 1247−1293.

[5] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. (September 1, 1951) 400−407.

[6] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, Adv. Neural Inf. Process. Syst. 26 (2013) 315−323.

[7] C. Darken, J. Chang, J. Moody, Learning rate schedules for faster stochastic gradient search, in: In Neural Networks for Signal Processing 2, August 31, 1992.

[8] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (7) (July 1, 2011).

[9] I. Sutskever, Training Recurrent Neural Networks, University of Toronto, Toronto, Canada, January 1, 2013.

[10] R. Ge, F. Huang, C. Jin, Y. Yuan, Escaping from saddle points—online stochastic gradient for tensor decomposition, in: In Conference on Learning Theory, PMLR, June 26, 2015, pp. 797−842.

[11] N. Qian, On the momentum term in gradient descent learning algorithms, Neural Network. 12 (1) (January 1, 1999) 145−151.

[12] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: In International Conference on Machine Learning, PMLR, May 26, 2013, pp. 1139−1147.

[13] Y. Nesterov, A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ2), in: In Doklady an USSR 269, 1983, pp. 543−547.

[14] L. Baird, A.W. Moore, Gradient descent for general reinforcement learning, Adv. Neural Inf. Process. Syst. (July 20, 1999) 968−974.

[15] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980. 2014 Dec 22.

[16] M.D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701. 2012 Dec 22.

[17] T. Dozat, Incorporating Nesterov Momentum into Adam, 2016.

[18] N.L. Roux, M. Schmidt, F. Bach, A stochastic gradient method with an exponential convergence rate for finite training sets, arXiv preprint arXiv:1202.6258. 2012 Feb 28.

[19] R.H. Byrd, G.M. Chin, W. Neveitt, J. Nocedal, On the use of stochastic hessian information in optimization methods for machine learning, SIAM J. Optim. 21 (3) (July 1, 2011) 977−995.

[20] J.E. Dennis, J.J. Moré, A characterization of superlinear convergence and its application to quasi-Newton methods, Math. Comput. 28 (126) (1974) 549−560.

[21] J. Nocedal, Updating quasi-Newton matrices with limited storage, Math. Comput. 35 (151) (1980) 773−782.

[22] A. Agarwal, P.L. Bartlett, P. Ravikumar, M.J. Wainwright, Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization, IEEE Trans. Inf. Theor. 58 (5) (January 30, 2012) 3235−3249.

[23] A. Bordes, L. Bottou, P. Gallinari, SGD-QN: careful quasi-Newton stochastic gradient descent, J. Mach. Learn. Res. 10 (July 2009) 1737−1754.

[24] N.N. Schraudolph, Fast curvature matrix-vector products, in: In International Conference on Artificial Neural Networks, Springer, Berlin, Heidelberg, August 21, 2001, pp. 19−26.

[25] S.I. Amari, Natural gradient works efficiently in learning, Neural Comput. 10 (2) (February 15, 1998) 251−276.

[26] S.I. Amari, H. Nagaoka, Methods of Information Geometry, American Mathematical Soc., 2000.

[27] H. Park, S.I. Amari, K. Fukumizu, Adaptive natural gradient learning algorithms for various stochastic models, Neural Network 13 (7) (September 1, 2000) 755—764.

[28] M. Dodangeh, L.N. Vicente, Worst case complexity of direct search under convexity, Math. Program. 155 (1—2) (January 1, 2016) 307—332.

[29] H. Robbins, Some aspects of the sequential design of experiments, Bull. Am. Math. Soc. 58 (5) (September 1952) 527—535.

[30] P. Auer, Using confidence bounds for exploitation-exploration trade-offs, J. Mach. Learn. Res. 3 (Nov) (2002) 397—422.

[31] J. Kiefer, J. Wolfowitz, Stochastic estimation of the maximum of a regression function, Ann. Math. Stat. (September 1, 1952) 462—466.

[32] R. Durrett, Probability: Theory and Examples, Cambridge university press, April 18, 2019.

[33] P. L'Ecuyer, G. Yin, Budget-dependent convergence rate of stochastic approximation, SIAM J. Optim. 8 (1) (February 1998) 217—247.

[34] R. Pasupathy, On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization, Oper. Res. 58 (4-part-1) (August 2010) 889—901.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (January 1, 2014) 1929—1958.

View publication stats