A state-of-the-art survey of advanced optimization methods in machine learning

Muhamet Kastrati[a] and Marenglen Biba[a]

[a] *University of New York in Tirana, Tirana, Albania*

Abstract

The main objective of this paper is to provide a state-of-the-art survey of advanced optimization methods used in machine learning. It starts with a short introduction to machine learning followed by the formulation of optimization problems in three main approaches to machine learning. Then optimization is presented along with a review of the most recent state-of-the-art methods and algorithms that are being extensively used in machine learning in general and deep neural networks in particular. The paper concludes with some general recommendations for future work in the area.

Keywords 1

Optimization methods, machine learning, deep neural networks, gradient descent, stochastic gradient descent

1. Introduction

In recent years, machine learning has made significant progress and received enormous attention in the research community and industry. Machine learning is applied successfully to a wide range of problems ranging from image recognition, speech recognition, text classification, online advertising, web search, recommendation systems, etc. However, there also exist many challenging problems in machine learning including minimization of loss (error) function, hyperparameters tuning, feature selection, dimensionality reduction, finding the optimum combination from a pool of base classifiers, etc. Optimization is an efficient and robust tool to tackle some of these challenges, and since its beginning has played a vital role in statistical and machine learning. In today's data-intensive technology era, machine learning models in general, and deep neural networks in particular, rely more and more on optimization methods [1].

The relationship between optimization methods and machine learning is one of the most relevant topics in modern computational science.

More recently a lot of work has been done by both the optimization and machine learning community by achieving state-of-the-art results on this matter. Generally, it is considered a relationship of great intimacy, optimization has proved to be a core tool in machine learning and at the same time, machine learning can be considered as an important source of inspiration for new optimization ideas [2].

From point of view of gradient information in optimization, popular optimization can be grouped into three major classes: the first-order optimization methods, which are mainly based on stochastic gradient methods, the second class comprises of high-order optimization methods, here Newton's method represents a typical example of this class and the third class comprises of heuristic derivative-free optimization methods, in which the coordinate descent method is a representative [3].

Proccedings of RTA-CSIT 2021, May 2021, Tirana, Albania
EMAIL: muhamet.kastrati@gmail.com; marenglenbiba@unyt.edu.al

The interplay of optimization and machine learning has also attracted the attention of various researchers and practitioners.

This paper serves as a complementary one to those previously published, at the same time provides a state-of-the-art review on advanced optimization methods used in machine learning. First, it gives a short introduction to machine learning followed by the formulation of optimization problems in three main types of machine learning approaches. Then, an overview of state-of-the-art advanced optimization methods used in machine learning in general and deep neural networks, in particular, has been presented. Finally, conclusions are drawn in the last section.

2. Background

The following is given a short introduction to machine learning models, followed by optimization methods and algorithms to facilitate understanding of these two fields and the interaction among them.

2.1. Machine learning

Machine learning is a subfield of artificial intelligence that deals with building a computer system that learns from data without being explicitly programmed [4, 5].

The authors in [6] start their definition for machine learning as data modeling and go further by treating machine learning as an optimization problem. In [7], author begins his formulation with "machine learning is programming computers to optimize a performance criterion using example data or past experience". These authors and others in their work treat machine learning as an optimization problem, which is our main goal of this section. Furthermore, almost all machine learning problems can be defined as an optimization problem [8]. This is done by having a solution (model) space defined, and an optimality criterion (objective function) determined. By taking into consideration that the solution (model) space can be very large or even infinite, usually is searched for the best suboptimal solutions [6]. By having the solution (model) space and objective function defined, then, any appropriate optimization method can be used to solve the machine learning problem as an optimization one.

As described in [6, 7] there are three main types of machine learning approaches: supervised learning, unsupervised learning, and reinforcement learning [9].

2.1.1. Optimization problems in supervised learning

Optimization problems in supervised learning aimed to find a function that best minimizes the loss function L on the training sample N, with respect to some parameters θ, when both input x^i and output y^i are given [3, 10, 11]

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta)) \quad (1)$$

Support Vector Machine - here loss function is a typical form of structured risk minimization. In this case, usually, the regularization items are added into the objective function to alleviate overfitting [3]

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta)) + \lambda |\theta\ \|_2^2 \quad (2)$$

here λ is a fixed parameter that usually is determined through cross-validation.

2.1.2. Optimization problems in unsupervised learning

The main task here is to find a function that best minimizes the loss function on training, when only input is given without output (target), for a set of defined parameters.

For the k-means clustering algorithm, the optimization problem can be formulated as minimizing the following loss function [3]

$$\min_{S} \sum_{k=1}^{K} \sum_{x \in \mathbf{S_k}} \| x - \mu_k \|_2^2 \quad (3)$$

with K being the number of clusters, x feature vector of samples, μ_k represents the center of cluster k, and S_k that represents the sample set of cluster k. The main goal of the objective function considers minimizing the variance through all clusters.

In the case of principal component analysis (PCA) the goal of the objective function is to minimize the reconstruction errors and has the form as given below:

$$\min \sum_{i=1}^{N} \| \overline{x^i} - x^i \|_2^2 \quad (4)$$

$$where \quad \overline{x^i} = \sum_{j=1}^{D'} z_j^i e_j \quad D >> D', \quad (4.1)$$

Here N denotes the number of samples, x^i denotes the D-dimensional vector, $\overline{x^i}$ denotes

the reconstruction of x^i. In the second part of the equation above, $z^i = z_1^i, \ldots, z_{D,}^i$ denotes the projection of x^i in D'-dimensional coordinates, and e_j denotes the standard orthogonal basis under D'.

In probabilistic models, the optimization goal is to find an optimal probability density function of $p(x)$, which maximizes the logarithmic likelihood function (MLE) [3] of the training samples,

$$\max \sum_{i=1}^{N} \ln p(x^i, \theta) \quad (5)$$

2.1.3. Optimization problems in RL

Here the main task is to maximize the expected reward after executing a series of actions which are defined by a policy mapping function [3, 9].

$$\max_{\pi} V_\pi(s) \quad (6)$$

$$V_\pi(s) = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | S_t = s] \quad (6.1)$$

2.2. Optimization

Optimization is a mathematical discipline that is used to solve any problem involving decision making, whether in engineering or economics. The basic idea used in optimization is choosing the best solution among various available alternatives, in the sense of the given objective function [12]. According to [13] an optimization problem can be defined in this form:

$$\begin{aligned} &\underset{x}{\text{minimize}} && f_0(x) \\ &\text{subj. to} && f_i(x) \leq b_i \;\; i = 1, \ldots, m, \end{aligned}$$

where vector $x = (x_1, \ldots, x_n)$ represents the optimization variable of the problem and $f_0: \mathbf{R}^n \to \mathbf{R}$ is the objective function, $f_i: \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, m$ are denoted the (inequality) constraint functions and with $b_1, \ldots, b_m$ are denoted limits for the constraints.

Depending on the particular forms of the objective and constraint functions there is a taxonomy of optimization problems including linear, quadratic, semi-definite, semi-infinite, integer, nonlinear, goal, geometric, fractional, etc. [13, 14].

In this paper, we have tried to reflect mainly on the advanced optimization methods that are used in machine learning in general and deep neural networks in particular.

3. Optimization methods in machine learning

The following is given an overview of the state-of-the-art optimization methods applied in machine learning models.

3.1. First-order methods

First-order optimization methods are also known as gradient-based optimization and as the name implies, these methods are mainly based on first-order derivatives or gradient descent. This section dealt with state-of-the-art first-order optimization methods in machine learning that are currently in use apart.

a. Gradient Descent - is one of the oldest and most popular algorithms used to perform optimization. Because of its relevant role in solving optimization problems in machine learning and deep learning, it is part of every state-of-the-art deep learning library (e.g. lasagne's, caffe's, and keras') [15].

The key idea behind gradient descent is to minimize an objective function $L(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by performing the iterative update of the parameters in the opposite direction of the gradient of the objective function $\Delta_\theta L(\theta)$ with regard to the parameters. During the gradient procedure, a learning rate η is used, that determines step-size or how much to move in each iteration until an optimal value is reached. The iteration process is repeated until the derivative becomes zero, which means we have reached the minimum value [15]. As described by [15] there are three distinct variants of gradient descent known as batch gradient descent, mini-batch gradient descent, and stochastic gradient descent, which differ in how much data is used during the gradient procedure of the objective function.

Batch gradient descent - also called villa gradient descent, is a simple variant of gradient descent that performs gradient procedure of the cost function w.r.t. to the parameters η for the entire training dataset at once [15]

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \quad (7)$$

One of the key advantages of batch gradient descent is its ability to guarantee to converge to the global optimum for the convex function and to a local optimum for the non-convex one.

The main drawback of batch gradient descent is the run-time, it can be very slow because the entire dataset used to perform just one update, at the same time, is inconvenient for datasets that are too large to fit in memory.

Stochastic gradient descent - in contrast to batch gradient descent, it does not use the entire dataset but rather performs parameter update only for one training example at a time.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \quad (8)$$

Because the stochastic gradient descent uses only one training example before updating the gradients, especially in the case of a large training set, it can be much faster compared to batch gradient descent. Due to being much faster, it can also be used to learn online. However, there is a drawback in the context of the parameters, they will "oscillate" toward the minimum rather than converge smoothly. Further details on the stochastic gradient descent can be found in the following section.

Mini-batch gradient descent - this approach benefits from combining the best of both worlds and performs an update for every mini-batch of *n* training examples:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \quad (9)$$

There are two main advantages to this: first, it further reduces the variance of the parameter updates, which can lead to more stable convergence. Second, it can make use of highly optimized matrix optimizations typical to state-of-the-art deep learning libraries. In general, mini-batch has sizes ranging from 50 up to 256 but this can be larger depending on the applications. Because of its advantages, mini-batch gradient descent is a method of choice for training a neural network and the term SGD usually is employed also when mini-batches are used [8, 15].

b. Stochastic Gradient Descent (SGD) [16] - was proposed to allow online updates for large-scale datasets, which was considered as the main disadvantage faced with batch gradient descent, caused by high computational complexity in each iteration. SGD is based on the idea of using only one example randomly selected to update the gradient per iteration [16]. Therefore, the cost of the SGD algorithm is independent of the number of examples used and can achieve sublinear convergence speed [17]. In this way, the SGD reduces the update time especially when deals with large data sets, and this makes it possible to significantly accelerates the calculation. In strong convex problem, SGD can achieve the optimal convergence speed [18, 19].

c. Momentum [20] - is a method that helps SGD to increase the rate of convergence in the relevant direction and reduce the oscillations. All this was made possible by adding the momentum term of the update vector of the previous time step to the current update vector [15]. On the other hand, ravines are areas in a slope, where the surface curvature is steeper in one dimension than another and these are common around local optima. When oscillating in these ravines, SGD progress becomes very slow to obtain the local optimum position. To handle this slow progress, SGD adds up a parameter called Momentum. Usually, the good momentum term is considered $\gamma = 0.9$, which accelerates the convergence in the relevant direction.

$$\begin{aligned} \nu_t &= \gamma \nu_{t-1} - \eta \nabla_\theta J(\theta) \\ \theta &= \theta - \nu_t \end{aligned} \quad (10)$$

with γ is denoted momentum term, ν_{t-1} is update vector in the previous step, $\eta \nabla_\theta J(\theta)$ is update vector in the current step and with $\theta = \theta - \nu_t$ is denoted update of the parameter.

In other words, the idea of using momentum is similar to the ball effect down a hill. It starts with an initial momentum value and accumulates momentum as it goes down a hill, it speeds up on the way (until finally terminal velocity is reached, i.e. $\gamma < 1$). In the same way, the momentum term increases when moving in the direction of the gradient and decreases updates for dimensions whose gradient is in different directions. As a result, the convergence speeds up and oscillation has been reduced [15].

d. Nesterov Accelerated Gradient (NAG) [21] - is intended to further improve the traditional momentum method [22]. It is based on the idea of using the momentum term $\gamma \nu_{t-1}$ to move the parameters θ. By using the term $\theta - \gamma \nu_{t-1}$ makes enable estimation of the next position of the parameters (the gradient is missing for the full update), in other words, a rough idea of where parameters are going to be. As result, is possible to effectively look ahead by calculating the gradient with regard to the approximate future position of the parameters instead of the current position.

e. AdaGrad [23] - is another gradient-based optimization algorithm. It is based on the idea of adapting the learning rate dynamically based on the historical gradient in some previous iteration. In other words, adopting smaller learning rates for features occurring often, and higher learning rate for features not occurring often. By doing this, it is well-suited for dealing with sparse data. Here are the updating formulae as presented by [15]

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

Then, the SGD update performed for every parameter θ_i at each step t takes the form:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

As given in the equation above, Adagrad adopts the general learning rate dynamically η for every parameter θ_i at each step t based on the previous gradients computed for θ_i.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii}+\epsilon}} \cdot g_{t,i} \quad (12)$$

where $G_t \in \mathbb{R}^{dxd}$ here is a diagonal matrix where each diagonal element i,i is the sum of the squares of the gradients w.r.t. θ_i up to time step t, ϵ is a smoothing term that avoids division by zero, $g_{t,i}$ is the gradient of the loss function with respect to the parameter θ_i at time step t [15].

As mentioned above, the Adagrad is characterized by the ability to adapt the learning rate dynamically based on historical data, which makes it convenient to handle sparse data [24].

f. AdaDelta [25] - is an extension of Adagrad that is aimed to reduce its weaknesses with regard to the learning rate. In contrast to the previous method, which accumulates all past gradients, the new method restricts the accumulating window of the past gradients to some fixed size w.

The running average $E[g^2]_t$ at time t then depends only on the previous average and the current gradient [15]:

$$E[g^2]_t = \gamma E[g^2]_{t-1} - (1-\gamma) g^2{}_t \quad (13)$$

where $E[g^2]_t$ is the running average at time step t that depends only on the previous average and the current gradient. The parameter is updated as:

$$\Delta\theta_t = -\eta \cdot g_{t,i}$$
$$\theta_{t+1} = \theta_t - \Delta\theta_t \quad (14)$$

In Adadelta there is no need to set the learning rate because it is defined by default and has been eliminated from the update rule [15].

g. RMSProp [26] - is an adaptive learning rate method proposed independently at about the same time as AdaDelta. Both RMSProp and AdaDelta aimed to tackle the radically diminishing learning rates issue of AdaGrad.

h. Adaptive moment estimation (Adam) [8] - is one of the most-popular first-order gradient-based optimization algorithms, which combines the advantages of both AdaGrad and RMSProp. Adam is based on adaptive estimates of lower-order moments. Some of the main benefits of this method are as follows: easy to implement, computationally efficient, requires less memory space, invariant to diagonal rescale of the gradients, and appropriate for problems with large data and/or a large number of parameters

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (15)$$

Here m_t and v_t are denoted the estimates of the first and the second moment of the gradients respectively.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii}+\epsilon}} \cdot g_{t,i} \quad (16)$$

As proposed by authors in [15], good default values for the tested machine learning problem are 0.001 for stepsize, 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ.

i. Nesterov-accelerated Adaptive Moment Estimation (Nadam) [27] - was proposed to further improve Adam, by combining the benefits of both Adam and NAG. Adam is characterized by two main components - a momentum and an adaptive learning rate. Although, regular momentum can be shown conceptually and empirically to be inferior to NAG algorithm.

(j) Other recent adaptive optimizers - several new variants of adaptive optimizers have been proposed more recently. These include AdamW [28], SC-Adagrad and SC-RMSProp [29], Yogi [30], and Adafactor [31].

k. Variance Reduction Methods - over the years several variance reduction techniques have been proposed. Some of the most well-known including stochastic average gradient (SAG) [32], stochastic dual coordinate descent (SDCA) [33], stochastic variance reduction gradient (SVRG) [17], incremental surrogate optimization (MISO) [34] and SAGA [35].

SAG - is a variance reduction method that has been proposed with the aim to improve the convergence speed. SAG iterations have a linear convergence rate that makes them faster than SGD, and has favorable advantages compared to other stochastic gradient

algorithms, especially when the loss function is smooth and convex [32, 36].

SVRG - SGD is characterized by slow convergence due to the inherent variance. The SVRG method was proposed with the idea to overcome this drawback. It has the same fast convergence rate with SAG and SDCA for smooth and strongly convex functions, but unlike SAG and SDCA it does not require the storage of gradients. As result, SVRG has very good performance and easily applicable to complex models such as non-convex neural networks [17, 37, 38].

l. Alternating Direction Method of Multipliers (ADMM) [39, 40] - is a very simple and well-known optimization method that is suitable for distributed convex optimization and in particular to problems that are present in applied statistics and machine learning [41, 42]. It has the root in the augmented Lagrangian method (aka. the method of multipliers).

As presented by authors in [42], thanks to its ability to deal with objective functions separately and synchronously, ADMM turned out to be a method of choice in the field of large-scale data-distributed machine learning and big-data related optimization. It is characterized by its capability to efficiently analyze, parallelize, and solve large optimization problems in a wide range of applications [43].

Many different variants of ADMM have been proposed including stochastic alternating direction method of multipliers (S-ADMM) algorithm [41, 44], which incrementally approximates the full gradient in the linearized ADMM formulation, zeroth-order online alternating direction method of multipliers (ZOO-ADMM), which enjoys advantages of both worlds, gradient-free operation and ADMM to accommodate complex structured regularizers [45].

m. Frank-Wolfe Method - also known as conditional gradient algorithm for smooth optimization, was originally proposed by Wolfe in 1956. It is one of the simplest and earliest known iterative algorithms used for solving quadratic programming problems with linear constraints [46]. The idea underlying the Frank-Wolfe method is to approximate the convex objective function with a linear function. For further details and extensions, we refer to [46, 47, 48].

3.2. High-order methods

The use of curvature information in addition to the gradient offers a convenient approach where an objective function exhibits a highly nonlinear and is ill-conditioned. For example, second-order optimization methods have been proposed with this aim in mind. When these methods are used in the context of machine learning, they work effectively by providing faster convergence rates, stability to parameter tuning, and robustness to problem conditioning. However, the benefits derived from the use of curvature information in the form of a dense Hessian matrix come at the cost of increased per-iteration complexity [3, 49].

In the last decades, with the continuous improvement of high-order optimization methods, there has been increased interest in the exploration of large-scale data by using stochastic techniques [50, 51, 52]. Following, we provide a brief introduction on some of the most prominent second-based methods including the conjugate gradient method, classical quasi-Newton method, the Hessian-Free method, and natural gradient.

a. Conjugate gradient (CG) - can be seen as being between the method of steepest descent and Newton's method. It is one of the most effective iterative methods used for solving sparse systems of linear equations [53], but can be easily applied to solve nonlinear optimization problems [54]. In general, the first-order methods are simple but characterized by a slow convergence rate, and the second-order methods are fast but characterized by high computational cost. The idea underlying CG is to combine both methods and develop something in between, whose goal is to accelerate the convergence rate like high-order and at the same time using the advantages of the first-order information [3].

b. Quasi-Newton Methods - methods of Newton or quasi-Newton's type are typically well-known to be the most convenient choice for solving nonlinear minimization problems when the objective function is twice continuously differentiable [55]. As we mentioned previously, gradient descent employs first-order information, but it suffers from a slow convergence rate. To overcome this limitation, several more sophisticated second-order methods have been introduced, e.g., Newton's method [56]. Newton's method is

based on the idea of using both the first-order derivative (gradient) and the second-order derivative (Hessian matrix) to approximate the objective function with a quadratic function. This process repeats until the updated variable converges.

During the last decades, several methods based on Newton's has been developed and some of the most well-known include BFGS [57] and L-BFGS [58].

c. Stochastic Quasi-Newton Method - both stochastic Newton (SN) and stochastic quasi-Newton (SQN) methods have been proposed as a solution to overcome some issues present in first-order optimization methods. The central idea of these methods is to combine the speed of Newton's method and the scalability of first-order methods by integrating curvature information. The resulting class of these combined methods is enjoying increasing popularity as a robust method for several large-scale machine learning tasks [59]. Many different variants and extensions have been proposed to further improve the efficiency of SN and SQN methods including online-LBFGS, which is variants of BFGS proposed by [50], block BFGS [60] and its stochastic variant [61], two sampled QN methods such as sampled LBFGS and sampled LSR1, which are proposed by [59].

d. Hessian-Free Method - the working principle of the Hessian-free (HF) method is quite similar to Newton's method, which incorporates second-order gradient information. In contrast to Newton's method discussed above, for HF method as the name implies does not make any approximation to the Hessian *H*, but rather the product *Hd* is accurately computed using the finite differences method [62].

Sub-sampled Hessian-Free Method - HF optimization is also known as truncated-Newton, is a popular method, and has also been subject to a number of studies in the optimization research community for decades, but never seriously applied in machine learning because of its limitations. In order to overpass these limitations, a sub-sampled technique has been introduced in HF, resulting in an improved HF method [62, 63].

e. Natural Gradient - the natural gradient descent approach [64] is an optimization method typically inspired from the perspective of information geometry, and is often used as an alternative to conventional stochastic gradient descent. Over the last decades, has received increased interest, motivating many new and related approaches.

In some applications, a smaller number of iteration is required in the natural gradient compared to gradient descent, which makes it an attractive alternative method. However, computing the natural gradient has been shown to be impractical for models with very many parameters such as large neural networks, due to the extreme size of the Fisher matrix [65].

3.3. Derivative-free methods

For some optimization problems arising from practical applications, the derivative of objective and constraint functions may not exist or is difficult to calculate. This is where the use of derivative-free, or zeroth-order, optimization comes into play. Derivative-free optimization (DFO) is a discipline in mathematical optimization that aimed to find the optimal solution without requiring the availability of derivatives [66, 67].

In general, there are two major types of ideas with regard to derivative-free optimization. To the first type belong heuristic algorithms and the second type is made up of the coordinate descent method.

a. Heuristic algorithms - heuristic and metaheuristic techniques have been successfully used to solve several optimization problems in the different research areas of science and engineering. Some of the most well-known examples of these techniques include optimization methods include genetic algorithms, simulated annealing, differential evolution, harmony search, ant colony algorithms, and particle swarm optimization. Although, the implementation strategy of heuristic or metaheuristic algorithm in large-scale machine learning problems is still rarely investigated [67]. Therefore, these techniques are not thoroughly discussed in this paper but for further details we refer to [68].

b. Coordinate descent method (CD) - is a well-known typical derivative-free [69] optimization technique, which is based on the idea of updating one variable at a time by minimizing a single-variable subproblem. It is very simple and easy to be implemented and can be very efficient and scalable when applied to large-scale machine learning problems.

4. Conclusion

Over the last decades, there has been a huge interest in optimization across multiple fields in science and engineering including machine learning. However, there also exist many challenging problems in machine learning because of the increasing scale and complexity of machine learning and computational platforms.

Optimization is an efficient and robust tool to tackle some of these challenges, and since its beginning has played a vital role in statistical and machine learning. In today's data-intensive technology era, machine learning models in general, and deep neural networks in particular rely more and more on optimization methods.

More recently a lot of work has been done by both the optimization and machine learning community by achieving state-of-the-art results on this matter. Generally, optimization has proved to be a core tool in machine learning and at the same time, machine learning can be considered as an important source of inspiration for new optimization ideas.

We hope that the issues discussed in this paper will push forward the discussion in the area of optimization and machine learning, on the same time it may serve as complementary material for other researchers interested in both these topics.

5. References

[1] G. Lan, "Lectures on optimization methods for machine learning," *H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA*, 2019.

[2] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*. Mit Press, 2012.

[3] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2020.

[4] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[5] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[6] I. Kononenko and M. Kukar, *Machine learning and data mining*. Horwood Publishing, 2007.

[7] E. Alpaydin, *Introduction to machine learning*. MIT press, 2010.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] A. Haj-Ali, N. K. Ahmed, T. Willke, J. Gonzalez, K. Asanovic, and I. Stoica, "A view on deep reinforcement learning in system optimization," *arXiv preprint arXiv:1908.01275*, 2019.

[10] C. Gambella, B. Ghaddar, and J. Naoum-Sawaya, "Optimization problems for machine learning: a survey," *European Journal of Operational Research*, 2020.

[11] S. Bubeck, "Theory of convex optimization for machine learning," *arXiv preprint arXiv:1405.4980*, vol. 15, 2014.

[12] E. K. Chong and S. H. Zak, *An introduction to optimization*. John Wiley & Sons, 2004.

[13] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[14] K. P. Bennett and E. Parrado-Hernandez, "The interplay of optimization and machine learning research," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1265–1281, 2006.

[15] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[16] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp.400–407, 1951.

[17] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[18] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[19] A. Agarwal, M. J. Wainwright, P. L. Bartlett, and P. K. Ravikumar, "Information-theoretic lower bounds on the oracle complexity of convex optimization," in *Advances in Neural*

Information Processing Systems, 2009, pp. 1–9.
[20] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.
[21] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence...)," 1983.
[22] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.
[23] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.
[24] R. Ward, X. Wu, and L. Bottou, "Adagrad stepsizes: Sharp convergence over nonconvex landscapes," in *International Conference on Machine Learning*, 2019, pp. 6677–6686.
[25] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv*:1212.5701, 2012.
[26] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Technical Report*, 2012.
[27] T. Dozat, "Incorporating nesterov momentum into adam," 2016.
[28] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
[29] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," *arXiv preprint arXiv:1706.05507*, 2017.
[30] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Advances in neural information processing systems*, 2018, pp. 9793–9803.
[31] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," *arXiv preprint arXiv:1804.04235*, 2018.
[32] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Advances in neural information processing systems*, 2012, pp. 2663–2671.
[33] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 567–599, 2013.
[34] J. Mairal, "Optimization with first-order surrogate functions," in *International Conference on Machine Learning*, 2013, pp. 783–791.
[35] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in neural information processing systems*, 2014, pp. 1646–1654.
[36] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.
[37] Z. Allen-Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," in *International conference on machine learning*, 2016, pp. 699–707.
[38] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *International conference on machine learning*, 2016, pp. 314–323.
[39] R. Glowinski and A. Marroco, "Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite d'une classe de problemes de dirichlet non lineaires," ESAIM: *Mathematical Modelling and Numerical Analysis - Modelisation Mathematique et Analyse Numerique*, vol. 9, no. R2, pp. 41–76, 1975.
[40] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & mathematics with applications*, vol. 2, no. 1, pp. 17–40, 1976.
[41] H. Ouyang, N. He, L. Tran, and A. Gray, "Stochastic alternating direction method of multipliers," in *International Conference on Machine Learning*, 2013.
[42] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, 2011.
[43] D. Hallac, C. Wong, S. Diamond, A. Sharang, R. Sosic, S. Boyd, and J.

Leskovec, “Snapvx: A networkbased convex optimization solver,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 110–114, 2017.

[44] W. Zhong and J. Kwok, “Fast stochastic alternating direction method of multipliers,” in *International Conference on Machine Learning*, 2014, pp. 46–54.

[45] S. Liu, J. Chen, P.-Y. Chen, and A. Hero, “Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 288–297.

[46] M. Frank, P. Wolfe et al., “An algorithm for quadratic programming,” *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[47] K. L. Clarkson, “Coresets, sparse greedy approximation, and the frank-wolfe algorithm,” *ACM Transactions on Algorithms (TALG)*, vol. 6, no. 4, pp. 1–30, 2010.

[48] M. Patriksson, *The traffic assignment problem: models and methods*. Courier Dover Publications, 2015.

[49] S. B. Kylasa, “Higher order optimization techniques for machine learning,” Ph.D. dissertation, Purdue University Graduate School, 2019.

[50] N. N. Schraudolph, J. Yu, and S. Gunter, “A stochastic quasi-newton method for online convex optimization,” in *Artificial intelligence and statistics*, 2007, 436–443.

[51] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-newton method for large-scale optimization,” *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[52] P. Moritz, R. Nishihara, and M. Jordan, “A linearly-convergent stochastic l-bfgs algorithm,” in *Artificial Intelligence and Statistics*, 2016, pp. 249–258.

[53] J. R. Shewchuk et al., “An introduction to the conjugate gradient method without the agonizing pain,” 1994.

[54] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[55] D. Steck and C. Kanzow, “Regularization of limited memory quasi-newton methods for large-scale nonconvex minimization,” *arXiv preprint arXiv:1911.04584*, 2019.

[56] M. Avriel, *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

[57] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[58] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization,” *Mathematical programing*, vol. 45, no. 1-3, pp. 503-528, 1989.

[59] A. S. Berahas, M. Jahani, and M. Takac, “Quasi-newton methods for deep learning: Forget the past, just sample,” *arXiv preprint arXiv:1901.09997*, 2019.

[60] W. Gao and D. Goldfarb, “Block bfgs methods,” *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1205–1231, 2018.

[61] R. Gower, D. Goldfarb, and P. Richt´arik, “Stochastic block bfgs: Squeezing more curvature out of data,” in *International Conference on Machine Learning*, 2016, pp. 1869–1878.

[62] J. Martens, “Deep learning via hessian-free optimization.” in *ICML*, vol. 27, 2010, pp. 735–742.

[63] R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal, “On the use of stochastic hessian information in optimization methods for machine learning,” *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 977–995, 2011.

[64] S.-I. Amari, “Natural gradient works efficiently in learning,” *Neural computation*, vol. 10, no. 2, 1998.

[65] J. Martens, “New insights and perspectives on the natural gradient method,” *arXiv preprint arXiv:1412.1193*, 2014.

[66] J. Larson, M. Menickelly, and S. M. Wild, “Derivative-free optimization methods,” *arXiv preprint arXiv:1904.11585*, 2019.

[67] A. S. Berahas, R. H. Byrd, and J. Nocedal, “Derivative-free optimization of noisy functions via quasi-newton methods,” *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 965–993, 2019.

[68] M. Kastrati, and M. Biba, “Stochastic local search: a state-of-the-art review”. *International Journal of Electrical and Computer Engineering*, 11(1), 716, 2021.

[69] D. P. Bertsekas, “Nonlinear programming,” *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.