INDEX

Kanpur Philosophers
ISSN 2348-8301
International Journal of humanities, Law and Social Sciences
Published biannually by New Archaeological & Genological Society
Kanpur India

Special Issue IV, Issue I: 2023

ANALYSIS OF OPTIMIZATION TECHNIQUES IN MACHINE LEARNING ALGORITHMS

Stanley John Varghese[1] & Siji Jose Pulluparambil[2]

[1]Student, Naipunnya Institute of Management and Information Technology, Pongam, Kerala, India.
[2]Assistant Professor, Naipunnya Institute of Management and Information Technology, Pongam, Kerala, India.

ABSTRACT

PURPOSE

The primary intent of machine learning is to employ algorithms and data to mimic human learning potential, hence steadily reducing its learning curve and adopting accuracy. Moreover, it is no longer an abstract form or methodology - but a driving technology that powers applications and industries in the real world. Examples include image and speech recognition, traffic prediction, virtual personal assistance, customer service, self-driving cars, stock market trading, and the pharmaceuticals industry. Under machine learning, a multiplicity of algorithms are embedded to solve problems. However, these algorithms are designed to operate manually, hence opening up the thought of whether such algorithms can be learned. The highlights will affirm a brief background of various optimization techniques utilized along withtheir applications and challenges.

APPROACH

The research was conducted based on secondary data available for evaluating optimization techniques and their founding principles.

Keywords

Optimization techniques, Machine Learning, Machine Learning algorithms, Classification of differentiable and non-differentiable objective functions

1. INTRODUCTION

Over a few decades, there has been an exponential rise in processing power and storage capacities leading to the revolution in Big Data to cultivate and analyze real-time information and make qualitative decisions. With such technological advances, revolutionary products adopted machine learning as a means to predict the accurate possibility of a future action based on past event data [1]. Currently, optimization is embedded onto ML and leverages it to result close to, if not, exact predictions using computing platforms. Companies such as Salesforce, IBM, Google, Intel, and even Twitter have benefitted from integrating machine learning techniques into their business models to grow their user base and potentially understand what the consumers want and predict their next queries or actions. These techniques have paved the way to solve complex business and even societal problems to a certainextent which are seen predominantly over the past decade - effectively implemented in both first-world and third-world countries. However, to achieve explicit accuracy and avoid small errors in dynamic real-time situations, optimization is key. The chief focus is to highlight the optimization sphere and discuss its further potential to predict foreseeable and unforeseeable (important) events.

2. OBJECTIVES

This research paper analyzes copious data available on optimization techniques built on top of machine learning algorithms. The objectives are:

1. To discuss optimization and its pivotal role in ML algorithms.
2. To state merits and challenges faced when using optimization in ML algorithms.
3. To elucidate optimization classification.
4. To discuss optimization techniques used in ML algorithms, their mathematical foundations, and challenges.

3. DEFINITION

A simple yet common engineering design optimization process is shown in the figure below. The designer's task is to state a problem specification that provides data such as the: constraints, constants, objectives, and parameters that are to be accomplished. The designer is responsible for constructing the problem and quantifying the advantages of prospective designs. Once the designer creates an initial design, it should point towards the optimization algorithm [2].

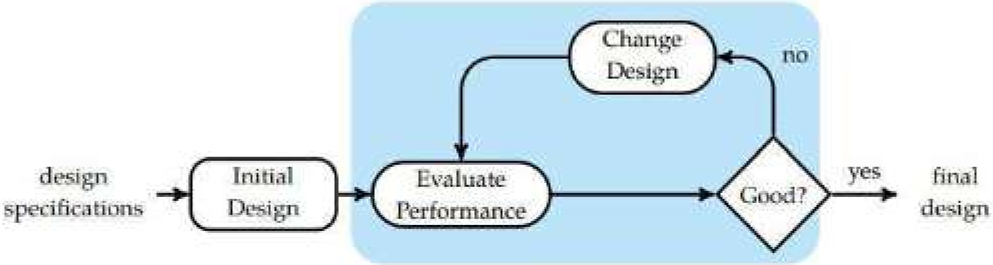

Fig. 1: The design improvement process. Area in blue denotes potential automation in the optimization process.

Furthermore, it can be defined as the process of calibrating hyperparameters in order to reduce the cost function by using any of the optimization techniques available. It is essential to scale down the cost function since it expounds the deviation between the estimated parameters, the true value and what the model has predicted. Ultimately, these 'optimizers' change neural network attributes to minimize potential losses. The goal is to find the global minima of a function, but if minima occur, then the gradient of the function is zero. However, a zero-gradient does not imply optimality.

To move further, it is vital to grasp the subtle differences between parameters and hyperparameters of a model:

- Hyperparameters are set up before a model gets trained. They vary from learning rate, number ofclusters, weights, and more. Overall, they describe the model's framework.
- Parameters, on the other hand, are produced during training. They are not obtainable beforehand. Examples can include biases and weights for neural networks. The data is restricted internally to the ML models and is altered based on the inputs provided.

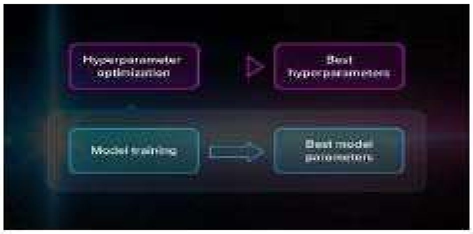

Fig. 2: Hyperparameters in training model

To adjust the model, hyperparameter optimization is utilized. Reduction of errors with a highly accurate model is present if the nonpareil combination of values is found. After each

3.5.1 Stochastic methods

Background

Randomization is key to strategically aid in exploring the search space for an optimum when using stochastic methods. Using randomness assists in escaping local optima and enhances the probability of finding a global optima. To ensure repeatability, these methods use pseudo-random number generators. However, do note that such generators produce random numbers based on a deterministic process.

Facing a substantial amount of randomness is usually ineffective since it prevents prior evaluation points to assist in leading the search efficiently. The methods under this category are capable of controlling thedegree of randomness in search [29].

Techniques used Simulated Annealing

Metallurgy is the cornerstone inspiration for this method. Annealing can be described as a process when

a material is heated and then cooled hence making it easy to work with. When turned hot, the atoms in the material are flexible to move around with random motion leading to better position settlement. An unhurried cooling action transforms the material into an ordered, crystalline state. On the other hand, a quick instant quenching results in defects due to the material being forced to adjust in its contemporary condition.

Temperature is the quantity that controls the stochasticity degree during a randomized search. To elucidate, the temperature begins high, and allows the process to move around freely the search space

with the aim that in this phase, it will identify an optimal region with the ideal local minima [30].

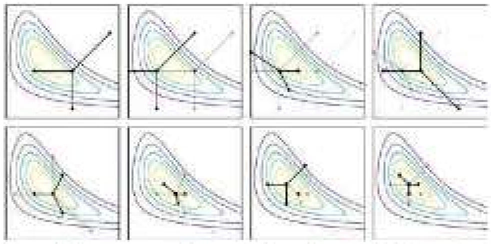

Fig. 25: Mesh adaptive direct search starting from left to right, top to bottom.

The method terminates when the temperature declines low enough such that the improvement is out of the picture. Termination happens when the latest function value is not far from ϱ from prior nϱ iterations and the optimal function value is gained over the execution journey.

Cross-entropy method

The method, known as the proposal distribution, stands separately from other methods discussed so far since it regulates an explicit probability distribution over the search space. The method is based on the process of minimizing cross-entropy by fitting the distribution and is also known as the Kullback–Leibler divergence. Using certain conditions, reducing the cross-entropy leads to searching for the maximum chance of the distribution parameters. In addition, it offers new samples for the following iteration(s). With each iteration, the sample is provided from each proposal distribution which then is updated to fit a collection of ideal samples. The objective of convergence is for proposal distribution to focus on global optimum value [31].

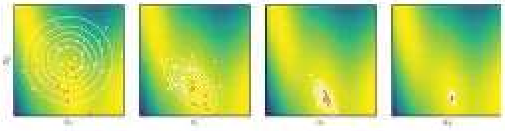

Fig. 26: The cross-entropy method with m = 40 experminted to the Branin function (2-dimensional function) implementing a bivariate Gaussian proposal distribution. The 10 supreme samples for each iteration are highlighted in red.

Natural Evolution Strategies

Similar to the cross-entropy method, these techniques adjust a θ-parameter proposal distribution. It is important to detail the proposal distribution and the sample quantity. Rather than fitting optimized samples, these strategies integrate gradient descent [32]. The gradient can be approximated from the samples through the equation:

$$\begin{aligned}
\nabla_{\theta} \mathbb{E}_{\mathbf{x}\sim p(\cdot\mid\theta)}[f(\mathbf{x})] &= \int \nabla_{\theta} p(\mathbf{x}\mid\theta) f(\mathbf{x})\,d\mathbf{x} \\
&= \int \frac{p(\mathbf{x}\mid\theta)}{p(\mathbf{x}\mid\theta)} \nabla_{\theta} p(\mathbf{x}\mid\theta) f(\mathbf{x})\,d\mathbf{x} \\
&= \int p(\mathbf{x}\mid\theta) \nabla_{\theta} \log p(\mathbf{x}\mid\theta) f(\mathbf{x})\,d\mathbf{x} \\
&= \mathbb{E}_{\mathbf{x}\sim p(\cdot\mid\theta)}[f(\mathbf{x}) \nabla_{\theta} \log p(\mathbf{x}\mid\theta)] \\
&\approx \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}^{(i)}) \nabla_{\theta} \log p(\mathbf{x}^{(i)}\mid\theta)
\end{aligned}$$

Fig. 27: The gradient approximation formulae

3.5.2 Population Algorithms

Background

Previous sections discussed so far have a single design point that is shifted incrementally en route for a minimum. Population methods consist of optimization via 'individuals' - group of design points. When a substantial number of individuals are distributed throughout the search space, it allows the algorithm to prevent being stuck at a local minimum.

Data at spread out points in the search space can be allocated between individuals to optimize the objective function globally. A majority of these methods are stochastic in nature, and it is usually simple toembed parallel computations [33]. It starts off with initialization wherein the methods are introduced with an initial population - similar to how descent methods expect an initial design point. The initial population must propagate to maximize the changes for samples to be near the best regions.

Techniques used Genetic algorithms

The methods obtain inspiration from biological evolution, where deemed-fit individuals are probable to spread their genes to the next generation. In fact, an individual's fitness level for reproduction is inversely proportional to the value of the objective function at a specified point.

The design point connected to an individual is addressed as a chromosome. Every single generation of fitter individuals' chromosomes are passed to the subsequent generation after undertaking the crossover, genetic operations, and mutations [34].

Chromosomes

There are multiple ways to represent chromosomes. The easiest method is to integrate a binary string chromosome - a representation that is alike to how DNA is encoded.

Fig. 28: A chromosome representation using a binary string.

Binary strings are commonly used since they allow expressing crossover at ease along with mutation. Still, the decoding process of a binary string and generating a design point is not guaranteed to be smooth. Oftentimes, the binary string may not associate a valid point in the search space. It is natural to showcase a chromosome using a list of real values. These chromosome values are vectors R^d that are directly proportional to points in the search space.

Initialization

The algorithms begin with a randomized initial population. Binary string chromosomes are usually triggered using random bit strings.

Selection

Selection is the process of picking chromosomes to identify as parents for the upcoming generation. A population with m chromosomes will have a selection method that generates an m-parental pairs list for m children of the following generation. Do note that the selected pairs may contain duplicates.

Crossover

Crossover joins the parents' chromosomes to produce children. There are a few crossover schemes:

- **Single-point crossover**

The initial portion of parent A's chromosomes generates the initial portion of the child chromosome, and the latter part of parent B's chromosomes creates the latter part of the child chromosome. The intersected point where the transition happens is fixed uniformly at random.

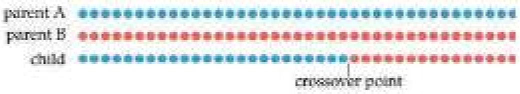

Fig. 29: Single-point crossover

- **Two-point crossover**

Two crossover points are used

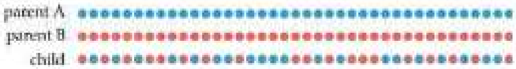

Fig.30 : Two-point crossover

The aforementioned crossover methods are applicable for real-valued chromosomes. There are definition schemes to allow additional crossover regimes that interpolate between real values.

The real values are interpolated in a linear fashion between the matured individual values xa and xb:

$$\mathbf{x} \leftarrow (1-\lambda)\mathbf{x}_a + \lambda\mathbf{x}_b$$

Fig. 31: Real value interpolation between distinct values

where λ is a scalar condition usually assigned to one-half.

Mutation

If the latest chromosomes were created only via crossover, many characteristics not available in the random initial population would never happen, and the elite-fit genes could overly spread the population. Mutation, however, enables new traits to voluntarily form thus allowing the genetic method to discover extra state space. Additionally, child chromosomes go through mutation after crossover. Each binary-valued chromosome bit comprises a minor probability of being reversed. For example, a chromosome with m-bits will have a mutation rate of 1/m hence causing an average of a single mutation for each child chromosome. Real-valued chromosomes' mutations can be executed using bitwise flips; nevertheless, it is usual to include zero-mean Gaussian noise too.

Differential Evolution

Differential evolution method tries to upgrade each individual in the population by readjusting and combining other individuals within the sample based on a straightforward formula:

It requires a parameter of a differential weight w and a crossover probability [35]. Usually, w is between

0.4 and 1. For each x:

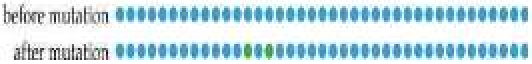

Fig. 32: Representation of binary string before and after mutation application

- Select three random unique individuals a, b, and c
- Create an interim design $z = a + w \cdot (b - c)$
- Enable optimization by opting for a random dimension $j \in [1, \ldots, n]$ in n dimensions
- Design the distinct candidate individual x via binary crossover

$$x'_i = \begin{cases} z_i & \text{if } i = j \text{ or with probability } p \\ x_i & \text{otherwise} \end{cases}$$

- Place the optimized design between x and x' into the upcoming generation

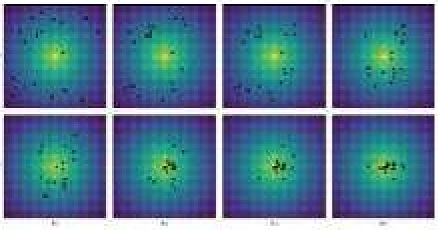

Fig. 33: Demonstration of differential evolution with p=0.5 and w=0.2 applied to Ackley's function - *a function that is used to test a method's probability of being stuck in local minima. It consists of two maincomponents - an exponential bell curve fixed at the center of the origin and a sinusoidal component that creates a collection of local minima which institutes the global minimum of a function.*

Particle Swarm Optimization

Particle Swarm Optimization initiates momentum to obtain minima by accelerating convergence. Each particle (individual) in the population sample maintains a track of its present velocity, position, and the ideal position it's viewed from so far. Momentum also enables an individual to gather speed in a beneficial direction that is distinct from local deviations.

Within each iteration, every individual moves towards both the ideal position it has seen and the optimum position found compared to any other individual thus far [36]. The accelerated movement is weighted by a random parameter with distinct random numbers being yielded for each acceleration. The updated equations are shown as: where x_{best} is the greatest location identified so far compared to other particles; r1 and r2 are random numbers drawn from U(0,

1) which is a popular strategy that accepts inertia w to fester over time; w, c1, and c2 are parameters [37].

1. CONCLUSION

As machine learning is integrated into a variety of applications and industries, its learning ability takes a steep curve; however, with introduction to optimization of such algorithms, the methods themselves begin to learn and embed the best parameters to achieve the optimum value to make decisions and solve problems.

Each of the methods share a common trait that helps segregate and identify the best algorithm needed for a given problem. Moreover, they do not define a one-size-fits-all solution. With this idea, the paper analyzes:

- Optimization algorithms that can be identified into those that require derivatives and those that execute without.
- Classical methods implement the first and oftentimes the second derivative of the objective function.
- Stochastic and direct search algorithms are constructed as objective functions where function derivatives are absent.
- Overall review of algorithms utilized to showcase utility presence in a practical environment

REFERENCES:

1. IBM Cloud Education. (2020). https://www.ibm.com/cloud/learn/machine-learning. [Accessed:05-August-2022].
2. Mykel Kochenderfer and Tim Wheeler (2019). Algorithms for optimization, (Pg4). [Accessed: 07-August-2022].
3. Yulia Gavrilova (2020). An Overview of Machine Learning Optimization Techniques. https://serokell.io/blog/ml-optimization
4. Mykel Kochenderfer and Tim Wheeler. (2019). Algorithms for optimization, (Pg5). [Accessed: 10-August-2022].
5. Jason Brownlee (2021). How to Choose an Optimization Algorithm. https://machinelearningmastery.com/tour-of-optimization-algorithms/. [Accessed: 10-August-2022].

6. Luke, S. (2013). *Essentials of metaheuristics* (Vol. 2). Raleigh: Lulu. [Accessed: 10-August-2022].

7. Hassin, R. (1981). On maximizing functions by Fibonacci search. Fibonacci Quarterly, 19, 347-351. [Accessed: 12-August-2022]

8. Hanson, F. Mcs 471 class optimization notes: Method of golden section search. [Accessed: 12-August-2022]

9. Wood, G.R. (1992).The bisection method in higher dimensions. Mathematical Programming 55, 319–337. https://doi.org/10.1007/BF01581205. [Accessed: 14-August-2022]

10. Elias D. Nino-Ruiz. (2021). A line-search optimization method for non-Gaussian data assimilation via random quasi-orthogonal sub-spaces. Journal of Computational Science. Volume 53, 101373, ISSN 1877-7503, https://doi.org/10.1016/j.jocs.2021.101373. (https://www.sciencedirect.com/science/article/pii/S1877750321000624) [Accessed:

11. 16-August-2022]

12. Do, S.T., Reynolds, A.C. (2013). Theoretical connections between optimization algorithms based on an approximate gradient. Comput Geosci 17, 959–973.

13. https://doi.org/10.1007/s10596-013-9368-9 [Accessed: 16-August-2022]

14. Beck, A. (2017). First-order methods in optimization. Society for Industrial and AppliedMathematics. [Accessed: 16-August-2022]

15. Lan, G. (2020). First-order and stochastic optimization methods for machine learning (p. 123). New York: Springer. [Accessed: 17-August-2022]

16. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprintarXiv:1609.04747. [Accessed: 17-August-2022]

17. Dehghani, M., & Samet, H. (2020). Momentum search algorithm: A new meta-heuristic optimization algorithm inspired by momentum conservation law. SN Applied Sciences, 2(10),1-15. [Accessed: 17-August-2022]

18. Battiti, R. (1992). First-and second-order methods for learning: between steepest descent and Newton's method. Neural computation, 4(2), 141-166.[Accessed: 18-August-2022]

19. Stoer, J. (2002). R. bulirsch introduction to numerical analysis springer-verlag. Texts

in AppliedMathematics, 12, 30.

20. Pilanci, M., & Wainwright, M. J. (2017). Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. SIAM Journal on Optimization, 27(1), 205-245. [Accessed: 18-August-2022]

21. Dennis, Jr, J. E., & Wolkowicz, H. (1993). Sizing and least-change secant methods. SIAM Journal on Numerical Analysis, 30(5), 1291-1314. [Accessed: 19-August-2022]

22. Byrd, R. H., & Nocedal, J. (1989). A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. SIAM Journal on Numerical Analysis, 26(3), 727-739.[Accessed: 19-August-2022]

23. Finkel, D. (2003). DIRECT optimization algorithm user guide. North Carolina State University. Center for Research in Scientific Computation.

24. Velásquez, J. D. (2010). An enhanced hybrid chaotic algorithm using cyclic coordinate search and gradient techniques. Revista de Ingeniería, (32), 45-53. [Accessed: 20-August-2022]

25. Powell, M. J. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. The computer journal, 7(2), 155-162. [Accessed: 21-August-2022]

26. Kramer, O. (2010). Iterated local search with Powell's method: a memetic algorithm for continuous global optimization. Memetic Computing, 2(1), 69-83. [Accessed: 22-August-2022]

27. Hooke, R., & Jeeves, T. A. (1961). ``Direct Search"Solution of Numerical and Statistical Problems. Journal of the ACM (JACM), 8(2), 212-229. [Accessed: 23-August-2022]

28. Aghamohammadi, M. R., & Pourgholi, M. A. H. D. I. (2008). Experience with SSSFR test for synchronous generator model identification using Hook–Jeeves optimization method. International Journal of System Applications, Engineering and Development, 2(3), 122-127. [Accessed: 23-August-2022]

29. Dolan, E. D., Lewis, R. M., & Torczon, V. (2003). On the local convergence of pattern search. SIAM Journal on Optimization, 14(2), 567-583.[Accessed: 25-August-2022]

30. Dennis, J. E., & Woods, D. J. (1987). Optimization on microcomputers: The Nelder-Mead simplex algorithm. New computing environments: microcomputers in large-scale

computing, 11,

31. 6-122.[Accessed: 26-August-2022]
32. Boender, C. G. E., & Romeijn, H. E. (1995). Stochastic methods. Handbook of global optimization.[Accessed: 27-August-2022]
33. Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.[Accessed: 27-August-2022]
34. Rubinstein, R. Y., & Kroese, D. P. (2004). The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning (Vol. 133). New York: Springer.[Accessed: 27-August-2022]
35. Rechenberg, I. (1978). Evolutionary strategies. In Simulation Methods in Medicine and Biology (pp. 83-114). Springer, Berlin, Heidelberg.[Accessed: 28-August-2022]
36. Ali, M. M., & Törn, A. (2004). Population set-based global optimization algorithms: some modifications and numerical studies. Computers & Operations Research, 31(10), 1703-1725. [Accessed: 28-August-2022]
37. Goldberg, DE (1992). Genetic Algorithms in Search, Optimization & Machine Learning, 401pp., Addison-Wesley (1989). 7(1), 168-168. [Accessed: 30-August-2022]
38. Das, S., & Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. IEEE transactions on evolutionary computation, 15(1). 4-31.[Accessed: 31-August-2022]
39. Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). Swarm Intelligence Morgan Kaufmann Publishers. San Francisco. [Accessed: 02-September-2022]
40. Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In MHS'95. Proceedings of the sixth international symposium on micro machine and human science (pp. 39-43). IEEE. [Accessed: 03-September-2022]