Comprehensive Overview of Optimization Techniques in Machine Learning Training

K. Karthick
Department of Electrical and Electronics Engineering, GMR Institute of Technology, Rajam - 532127, Andhra Pradesh, India
Email: kkarthiks@gmail.com
*Corresponding Author

***Abstract*—This article offers a comprehensive overview of optimization techniques employed in training machine learning (ML) models. Machine learning, a subset of artificial intelligence, employs statistical methods to enable systems to learn and improve from experience without explicit programming. The paper delineates the significance of optimization in ML, emphasizing its role in adjusting model parameters to minimize loss functions, thereby ensuring efficient model training and improved generalization. The discussion encompasses various optimization methods, including Gradient Descent Variants, Adaptive Learning Rate Methods, Second-Order Optimization Methods, Regularization Methods, Constraint-based Methods, and Bayesian Optimization. Each section elucidates the principles, applications, and benefits of these techniques, highlighting their relevance in addressing challenges such as overfitting, scalability, and computational efficiency. The article aims to guide researchers, practitioners, and enthusiasts in navigating the complex landscape of optimization techniques tailored for diverse machine learning algorithms and applications.**

Keywords*—*Optimization Techniques, Machine Learning, Gradient Descent Variants, Adaptive Learning Rate Methods, Bayesian Optimization

I. INTRODUCTION

Optimization methods play a crucial role in training machine learning models efficiently. Machine learning (ML) is a subset of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. It uses statistical techniques to enable machines to improve at tasks by being exposed to more data over time. There are three primary types of machine learning namely Supervised, Unsupervised and Reinforcement learning methods [1], [2].

Supervised Learning Algorithms learn from labeled data, and the goal is to learn a mapping from inputs to outputs based on input-output pairs. A classic example is image classification, where the algorithm learns to recognize objects based on labeled images. Unsupervised Learning Algorithms learn from unlabeled data, identifying patterns and structures in the data. An example is clustering, where the algorithm groups similar data points together without predefined labels. Reinforcement Learning Algorithms learn by interacting with an environment to achieve a goal or maximize some notion of cumulative reward [3]. An illustrative example is training a computer program to play chess, where the algorithm learns optimal moves through trial and error, receiving rewards for successful strategies.

Optimization in the context of machine learning refers to the process of adjusting model parameters to minimize (or maximize) a loss function. The primary objective is to find the best possible model parameters that result in the lowest value of the loss function. A loss function quantifies how well the model performs by measuring the difference between its predicted outputs and the actual outcomes. The goal during training is to minimize this 'loss' – essentially fine-tuning the model to make accurate predictions. It ensures that the model is continually improving its predictive capabilities. By iteratively adjusting the model's parameters to reduce the difference between predictions and actual results, its ability can be enhanced to generalize well to unseen data. In simpler terms, optimization helps in fine-tuning a model so that it performs better on unseen or test data.

Machine learning models are trained using algorithms that adjust model parameters to minimize a loss function. Optimization methods facilitate this adjustment process, ensuring that models learn from data effectively [4]-[6].

Many machine learning problems involve complex models with millions or even billions of parameters. Efficient optimization methods enable the training of these models within reasonable time and computational resources.

Effective optimization techniques help in building models that generalize well to unseen data. In other words, optimized models are less likely to overfit (perform well on training data but poorly on test data) [7], [8].

As datasets grow larger and models become more complex, the need for scalable optimization methods becomes crucial. Optimization techniques ensure that models can handle vast amounts of data and remain computationally feasible [9].

The field of machine learning encompasses various algorithms and models, each requiring specific optimization techniques tailored to its characteristics. Optimization methods provide the flexibility to adapt and fine-tune these algorithms according to the specific problem at hand.

II. GRADIENT DESCENT VARIANTS

Gradient Descent updates parameters in the opposite direction of the gradient of the loss function. Stochastic Gradient Descent (SGD) uses a subset of data to compute the gradient, making it faster but more noisy than standard gradient descent. Mini-Batch Gradient Descent is a compromise between batch (full dataset) and SGD where updates are made using small random batches of data. Momentum adds a fraction of the update vector from the past step to the current update vector, helping to navigate through areas with high curvature. Gradient Descent Variants are versatile optimization techniques that find applications across

a wide range of problem statements in machine learning, deep learning, optimization theory, and other related domains. Whether it's minimizing a loss function, optimizing model parameters, or finding the best configuration for a given objective, gradient-based methods offer powerful tools to navigate complex optimization landscapes.

Training deep learning models, such as multi-layer perceptrons (MLPs), involves minimizing a loss function using gradient-based optimization techniques.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) architectures also utilize variants of gradient descent to update weights and biases during the training phase.

Fig. 1 [10] illustrates a contrast between noiseless and noisy SGD training. Using a single example leads to fluctuations, resulting in winding and slower convergence paths during iterations. In Fig. 2 [11], the gradient exhibits vertical oscillations in a specific direction. One straightforward approach to address this issue is to stabilize the gradient horizontally while minimizing its vertical fluctuations.

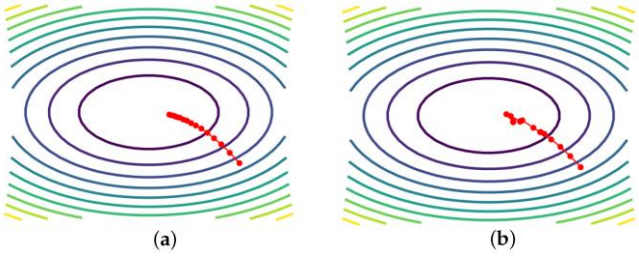

Fig. 1. Comparison of SGD and GD. (a) GD, (b) SGD [10]

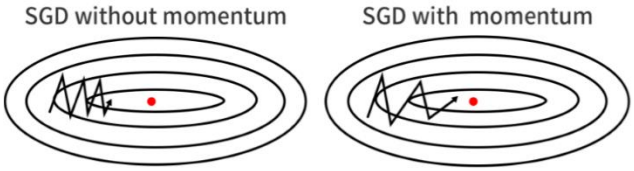

Fig. 2. Comparison of SGD algorithms with and without momentum [11]

III. Adaptive Learning Rate Methods

Adaptive learning rate methods, such as RMSprop and Adam, play a crucial role in optimizing machine learning models. These techniques dynamically adjust the learning rates of all model parameters based on historical gradients.

RMSprop is an extension of Adagrad that divides the learning rate by an exponentially decaying average of squared gradients. Adam (Adaptive Moment Estimation) combines ideas from Momentum and RMSprop. It uses both first and second moments of the gradients to adapt the learning rates.

Adaptive learning rate methods can accelerate the convergence of optimization algorithms by adjusting the learning rate dynamically. This adaptability ensures that the optimization process progresses efficiently, especially in scenarios where the landscape of the loss function varies across dimensions or iterations. In standard optimization algorithms, a fixed learning rate can lead to oscillations or divergent behaviors, especially in regions of the parameter space with steep gradients. Adaptive methods help mitigate these issues by scaling the learning rate based on the nature and history of gradients. The generic framework of adaptive optimization methods have been shown in Fig. 3. It outlines how different algorithms align with the general framework. The primary distinctions among these algorithms revolve around the choice of "averaging" functions and the manner in which parameters Φ and ψt are adjusted. This abstraction helps to highlight the discrepancies among the different optimizers. Adaptive learning rate methods offer robustness to various challenges encountered during optimization, such as saddle points, vanishing or exploding gradients, and non-stationary objectives. By adjusting the learning rate adaptively, these methods can navigate complex optimization landscapes more effectively. Traditional optimization algorithms often require manual tuning of hyperparameters like the learning rate, which can be challenging and time-consuming. Adaptive learning rate methods alleviate this burden by automatically adjusting hyperparameters based on the characteristics of the optimization problem, reducing the need for extensive hyperparameter tuning.

Algorithm 1 Generic framework of adaptive optimization methods

Input: $w_t \in \mathcal{F}$, initial step α_0, functions $\{\phi_t, \psi_t\}_{t=1}^{T}$

1: **for** t=1 **to** T **do**
2: $g_t = \nabla f_t(w_t)$
3: $m_t = \phi_t(g_1, ..., g_t)$ and $V_t = \psi_t(g_1, ..., g_t)$
4: $\alpha_t = \alpha_0/\sqrt{t}$
5: $\hat{w}_{t+1} = w_t - \alpha_t m_t/\sqrt{V_t}$
6: $w_{t+1} = \Pi_{\mathcal{F},\sqrt{V_t}}(\hat{w}_{t+1})$
7: **end for**

Fig. 3. Generic framework of adaptive optimization methods [12]

IV. Second-Order Optimization Methods

Newton's Method uses the second derivative (Hessian matrix) to guide the optimization process, which can be computationally intensive for large datasets [13].

Quasi-Newton Methods [14] approximates the Hessian matrix using gradient information to speed up convergence without explicitly computing the second derivative. Second-order optimization methods, as the name suggests, involve using second-order derivatives, specifically the Hessian matrix, in the optimization process. While first-order methods (like gradient descent) utilize only the first derivative (gradient) of the objective function, second-order methods incorporate information from the second derivative to guide the optimization process.

Second-order methods can converge to the optimal solution in fewer iterations compared to first-order methods in certain scenarios, especially when dealing with functions that are not well-conditioned. For functions that are highly non-linear or ill-conditioned (i.e., where the curvature varies significantly), second-order methods can provide more accurate and efficient convergence compared to first-order methods [15]. Second-order methods can offer insights into the local curvature of the optimization landscape, helping algorithms navigate more efficiently towards local minima or maxima. This can be particularly beneficial in complex optimization problems where the first-order gradient might provide limited information. In some cases, first-order methods may oscillate or exhibit erratic behavior due to the lack of curvature information. Second-order methods, by considering curvature, can offer more stable convergence behavior. For optimization problems where the Hessian matrix [16] provides valuable information about the curvature and condition of the objective function, second-

order methods can be particularly beneficial in adjusting step sizes adaptively.

V. Regularization Methods

L1 (Lasso) Regularization adds a penalty proportional to the absolute value of the coefficients. L2 (Ridge) Regularization adds a penalty proportional to the squared magnitude of the coefficients [17].

Elastic Net combines L1 and L2 regularization penalties. Regularization is a technique used to prevent overfitting in machine learning models, especially when dealing with a large number of features or parameters [18]. Overfitting occurs when a model learns the training data too closely, including its noise and outliers, which can lead to poor generalization on unseen data. The primary reason for using regularization is to prevent overfitting. For binary outcomes, LASSO adds an $L1$-norm penalty term to the negative log-likelihood function for a logistic regression and estimates the regression parameters by minimizing.

$$-\ell_1(\beta_0, \beta) + \lambda\left|\left|\beta_0, \beta\right|\right|_1 \quad (1)$$

The term "-ℓ1(β0, β)" represents the negative log-likelihood function with respect to the parameters β0 and β. The term " $\lambda||\beta$0, $\beta||$1" represents the L1 regularization penalty, where λ is the regularization parameter. For logistic regression with binary outcomes, the regression parameters are estimated by minimizing.

$$-\ell_1(\beta_0, \beta) + \lambda_1||\beta_0, \beta||1 + \lambda_2\left|\left|\beta_0, \beta\right|\right|_{22} \quad (2)$$

The term " $\lambda_2||\beta_0$, $\beta||_{22}$" represents the L2 regularization penalty, where λ_2 is another regularization parameter. By adding a regularization term to the loss function during training, models are discouraged from fitting the noise in the training data, resulting in improved performance on unseen data. In linear models, where predictors are highly correlated, regularization methods can help handle multicollinearity, ensuring stable and reliable estimates of model parameters. Some regularization techniques, like L1 regularization (Lasso), can drive certain feature weights to zero, effectively performing feature selection and simplifying the model.

VI. Constraint-Based Methods

Projected Gradient Descent updates parameters using gradient descent but projects the updated parameters back to a feasible region if they violate certain constraints. Evolutionary Algorithms: Genetic Algorithms Inspired by the process of natural selection, these algorithms use mechanisms like mutation, crossover, and selection to evolve solutions to optimization problems. Particle Swarm Optimization (PSO) uses a population of candidate solutions (particles) that move through the search space based on their own best position and the best position found by the entire swarm. Fig. 4 shows the classification of metaheuristic algorithms.

Constraint-based methods refer to optimization techniques where solutions are sought subject to specific constraints. In the context of machine learning and optimization, these methods are used to find optimal solutions within predefined boundaries or conditions. The primary objective is to ensure that the solutions generated adhere to certain criteria or limitations set by the problem's nature or domain requirements. Many real-world problems come with inherent constraints that solutions must satisfy. For instance, in resource allocation or scheduling problems, there might be constraints related to time, capacity, availability, or other resource limitations. Constraint-based methods ensure that solutions are both optimal and feasible within these constraints. Certain optimization problems have unique characteristics or domain-specific requirements that necessitate the incorporation of constraints. For example, in engineering design problems, solutions must often satisfy multiple design constraints related to materials, safety, cost, etc. Constraint-based methods allow for the determination of optimal solutions while ensuring that these solutions meet all specified constraints. This ensures a balance between optimality and feasibility, ensuring that solutions are not only optimal in terms of objective function values but also practical and implementable in real-world scenarios. Incorporating constraints can help in mitigating risks associated with certain decisions or solutions. By ensuring that solutions adhere to safety, operational, or regulatory constraints, constraint-based methods help in reducing potential risks or adverse outcomes. In complex optimization problems involving multiple objectives or criteria, constraints help in structuring the decision-making process by providing clear boundaries or guidelines. This structured approach ensures that solutions are coherent, logical, and aligned with the problem's broader objectives.

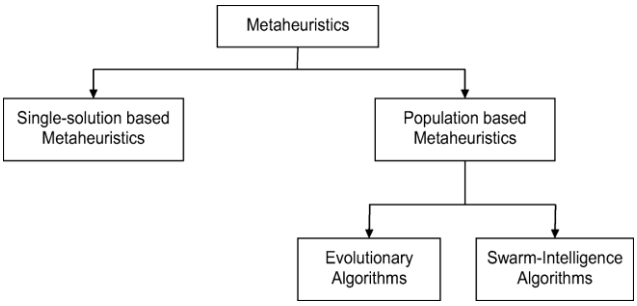

Fig. 4. Classification of metaheuristic algorithms

VII. Bayesian Optimization

Bayesian Optimization is a probabilistic model-based optimization technique primarily used for optimizing expensive and black-box functions. Unlike traditional optimization methods that require derivatives or gradient information, Bayesian Optimization focuses on building a probabilistic model of the objective function and then using this model to determine the next best point to evaluate [19]. At its core, Bayesian Optimization employs probabilistic models (typically Gaussian Processes) to capture and model the objective function's behavior. This model provides insights into the function's behavior, such as estimating the function's value at unexplored points and the uncertainty associated with those estimates. Bayesian Optimization is sequential in nature. It iteratively selects a new point to evaluate based on the current probabilistic model, evaluates the objective function at that point, updates the model with the new observation, and repeats this process until convergence or a predefined stopping criterion is met. A key strength of Bayesian Optimization lies in its ability to balance exploration (searching regions of the parameter space where

the objective function is uncertain) and exploitation (focusing on regions that are predicted to yield optimal or near-optimal values). This balance ensures efficient optimization without getting trapped in local optima.

In real-world scenarios, evaluating the objective function (e.g., performance of a machine learning model, physical experiments) can be time-consuming, computationally expensive, or costly. Bayesian Optimization aims to find the optimal solution with as few evaluations as possible, making it particularly suitable for scenarios where function evaluations are expensive.

When the mathematical form of the objective function is unknown or too complex to derive gradients or other analytical properties, Bayesian Optimization provides a robust framework to optimize such black-box functions solely based on function evaluations. Traditional optimization methods like gradient descent might get stuck in local optima, especially for non-convex and multimodal functions. Bayesian Optimization's exploration-exploitation strategy allows it to navigate complex landscapes and search for the global optimum efficiently. In machine learning and deep learning, models often have hyperparameters (e.g., learning rate, regularization parameters) that significantly impact performance. Bayesian Optimization offers an efficient way to tune these hyperparameters, leading to better model performance without exhaustive grid or random search.

VIII. Strengths and Limitations of Optimization Methods

Gradient descent variants are simple to implement, applicable to a wide range of problems, and computationally efficient, making them popular optimization methods. However, they may converge slowly, particularly in high-dimensional spaces, and are prone to getting trapped in local minima. Adaptive learning rate methods, such as Adam, dynamically adjust learning rates for each parameter, resulting in quicker convergence and enhanced performance in deep learning tasks. Nonetheless, they may demonstrate instability or sensitivity to hyperparameters, necessitating careful tuning.

Second-order optimization methods like Newton's Method leverage curvature information for faster convergence, especially in regions with high curvature. Yet, they are computationally intensive due to the necessity of computing and inverting the Hessian matrix, and they may suffer from numerical instability. Regularization methods, such as L1/L2 regularization, mitigate overfitting by penalizing large parameter values, leading to more resilient models. Nevertheless, they can introduce bias and require additional hyperparameter tuning.

Constraint-based methods enforce constraints on parameter values to ensure model stability and interpretability, making them valuable. However, they may be computationally demanding, particularly for intricate constraints, and may necessitate specialized optimization techniques. Bayesian optimization models the objective function as a probabilistic surrogate, efficiently exploring the parameter space and enhancing sample efficiency. Nonetheless, it relies on prior knowledge or assumptions about the objective function and may be less effective in high-dimensional spaces.

IX. Conclusion

In conclusion, optimization techniques serve as the cornerstone for training efficient and robust machine learning models, addressing inherent challenges like overfitting, scalability, and computational complexity. This comprehensive overview elucidated various optimization methodologies, including Gradient Descent Variants, Adaptive Learning Rate Methods, Second-Order Optimization Methods, Regularization Methods, Constraint-based Methods, and Bayesian Optimization. Each method offers unique advantages and applications, catering to diverse machine learning algorithms and scenarios. As machine learning continues to evolve and find applications across various domains, the understanding and effective implementation of optimization techniques become paramount. Future research endeavors should focus on integrating these techniques, exploring hybrid approaches, and addressing emerging challenges to propel the advancements in machine learning and artificial intelligence domains. The insights provided in this article aim to equip researchers, practitioners, and stakeholders with knowledge and tools to navigate the intricacies of optimization in machine learning, fostering innovation, and driving transformative solutions in the era of artificial intelligence.

References

[1] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," *Computers*, vol. 12, no. 5, p. 91, 2023, https://doi.org/10.3390/computers12050091.

[2] A. Haleem, M. Javaid, M. A. Qadri, R. P. Singh, R. Suman, "Artificial intelligence (AI) applications for marketing: A literature-based study," *International Journal of Intelligent Networks*, vol. 3, pp. 119-132, 2022, https://doi.org/10.1016/j.ijin.2022.08.005.

[3] R. Pugliese, S. Regondi, R. Marini, "Machine learning-based approach: global trends, research directions, and regulatory standpoints," *Data Science and Management*, vol. 4, pp. 19-29, 2021, https://doi.org/10.1016/j.dsm.2021.12.002.

[4] M. Gupta, K. Rajnish, V. Bhattacharjee, "Impact of Parameter Tuning for Optimizing Deep Neural Network Models for Predicting Software Faults," *Scientific Programming*, vol. 2021, pp. 1-17, 2021, https://doi.org/10.1155/2021/6662932.

[5] M. J. Bianco *et al*., "Machine learning in acoustics: Theory and applications," *The Journal of the Acoustical Society of America*, vol. 146, no. 5, pp. 3590-3628, 2019, https://doi.org/10.1121/1.5133944.

[6] T. Q. Bao and C. Gutiérrez, "Ekeland variational principles for vector equilibrium problems," *Optimization*, vol. 73, no. 1, pp. 29-62, 2024, https://doi.org/10.1080/02331934.2022.2094264.

[7] S. Ali, "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence," *Information Fusion*, vol. 99, p. 101805, 2023, https://doi.org/10.1016/j.inffus.2023.101805.

[8] S. Raschka, J. Patterson, C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence," *Information*, vol. 11, no. 4, p. 193, 2020, https://doi.org/10.3390/info11040193.

[9] U. Sivarajah, M. M. Kamal, Z. Irani, V. Weerakkody, "Critical analysis of Big Data challenges and analytical methods," *Journal of Business Research*, vol. 70, pp. 263-286, 2017, https://doi.org/10.1016/j.jbusres.2016.08.001.

[10] Y. Tian, Y. Zhang, H. Zhang, "Recent Advances in Stochastic Gradient Descent in Deep Learning," *Mathematics*, vol. 11, no. 3, p. 682, 2023, https://doi.org/10.3390/math11030682.

[11] L. Bottou, O. Bousquet, "The tradeoffs of large scale learning," *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pp. 161-168, 2008, https://proceedings.neurips.cc/paper_files/paper/2007/hash/0d3180d672e08b4c5312dcdafdf6ef36-Abstract.html.

[12] G. Ioannou, T. Tagaris, A. Stafylopatis, "AdaLip: An Adaptive Learning Rate Method per Layer for Stochastic Optimization," *Neural Processing Letters*, vol. 55, pp. 6311–6338, 2023, https://doi.org/10.1007/s11063-022-11140-w.

[13] I. K. Dassios, "Analytic Loss Minimization: Theoretical Framework of a Second Order Optimization Method," *Symmetry*, vol. 11, no. 2, p. 136, 2019, https://doi.org/10.3390/sym11020136.

[14] J. Z. Zhang, N.Y. Deng, L. H. Chen, "New Quasi-Newton Equation and Related Methods for Unconstrained Optimization," *Journal of Optimization Theory and Applications*, vol. 102, pp. 147-167, 1999, https://doi.org/10.1023/A:1021898630001.

[15] T. Guo, Y. Liu, C. Han, "An Overview of Stochastic Quasi-Newton Methods for Large-Scale Machine Learning," *Journal of the Operations Research Society of China*, vol. 11, pp. 245-275, 2023, https://doi.org/10.1007/s40305-023-00453-9.

[16] J. M. Bofill, "Updated Hessian matrix and the restricted step method for locating transition structures," *Journal of Computational Chemistry*, vol. 15, no. 1, pp. 1-11, 1994, https://doi.org/10.1002/jcc.540150102.

[17] O. Demir-Kavuk, M. Kamada, T. Akutsu, E. Knapp, "Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features," *BMC Bioinformatics*, vol. 12, no. 412, 2011, https://doi.org/10.1186/1471-2105-12-412.

[18] L. Liu, J. Gao, G. Beasley, S. Jung, "LASSO and Elastic Net Tend to Over-Select Features," *Mathematics*, vol. 11, no. 17, p. 3738, 2023, https://doi.org/10.3390/math11173738.

[19] X. Wang, Y. Jin, S. Schmitt, M. Olhofer, "Recent Advances in Bayesian Optimization," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1-36, 2023, https://doi.org/10.1145/3582078.