Int. J. Nonlinear Anal. Appl. 12 (2021) No. 2, 1453-1461
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.5261

Training analysis of optimization models in machine learning

Ahmed Alridha[a,*], Fadhil Abdalhasan Wahbi[a], Mazin Kareem Kadhim[a]

[a] *Department of Mathematics Science, Ministry of Education, Babylon, Iraq*

(Communicated by Madjid Eshaghi Gordji)

Abstract

Machine learning is fast evolving, with numerous theoretical advances and applications in a variety of domains. In reality, most machine learning algorithms are based on optimization issues. This interaction is also explored in the special topic on machine learning and large-scale optimization. Furthermore, machine learning optimization issues have several unique characteristics that are rarely seen in other optimization contexts. Aside from that, the notions of classical optimization vs machine learning will be discussed. Finally, this study will give an outline of these particular aspects of machine learning optimization.

Keywords: machine learning, mathematical programming, optimization techniques.

1. Introduction

One of the most important advances in current computer science is the interaction between optimization and machine learning. The use of optimization formulations and methods in the creation of algorithms to extract key knowledge from large amounts of data is proving to be crucial. Machine learning, on the other hand, isn't only a consumer of optimization technology; it's also a constantly expanding subject that generates new optimization concepts [7, 10, 11]. Our study represents the current state of the art in the intersection of optimization and machine learning in a way that academics in both domains can understand. Because of their broad application and appealing theoretical characteristics, optimization techniques have risen to prominence in machine learning. Because of the rising complexity, size, and variety of today's machine learning models, current assumptions must be reexamined. It explains the resurrection of well-known frameworks like first-order techniques,

*Corresponding author

Email addresses: amqa92@yahoo.com (Ahmed Alridha), Fadhilf40@gmail.com (Fadhil Abdalhasan Wahbi), mazen.marjan22@gmail.com (Mazin Kareem Kadhim)

Received: March 2021 *Accepted:* May 2021

stochastic approximations, convex relaxations, interior-point techniques, and proximal techniques in new situations. Regularized optimization, resilient optimization, gradient and subgradient methods, splitting approaches, and second-order methods are among the newest topics covered. Many of these approaches are based on other subjects, such as operations research, theoretical computer science, and optimization subfields. As it stands now, the rudder is in the direction of a closer relationship between machine learning and various other areas, as well as within the improvement community as a whole. Machine learning is the study of algorithms that learn from data rather than explicitly declaring the code to do a task [3, 6]. Algorithms can learn on their own without the need for human interaction thanks to machine learning. Machine learning algorithms are divided into various categories based on the type of problem to be solved. The following are the three primary types:

1. We use supervised machine learning techniques to construct a model that predicts labels based on characteristics given labeled data.
2. Unsupervised machine learning algorithms: Unsupervised machine learning algorithms do not have a goal value. Assume you need to organize the hypothetical spacecraft into groups based on their characteristics; you'll use a clustering technique to accomplish so. To find patterns in the dataset, unsupervised machine learning is performed. We don't know which cluster is whose, but we do know that all the spacecraft in one cluster look same; the right picture in Figure (1) illustrates this.

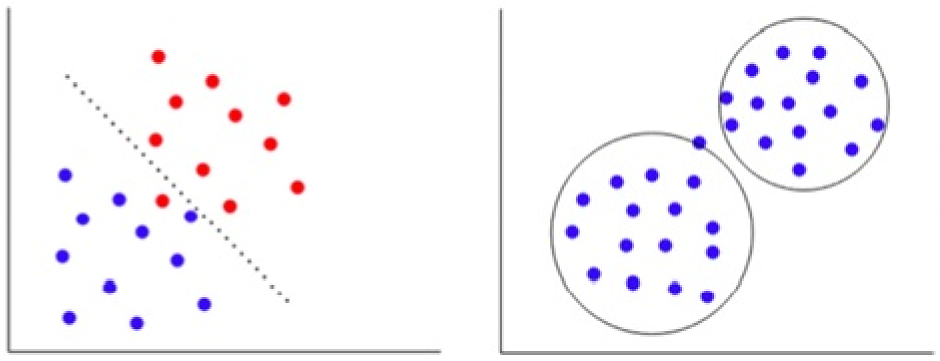

Figure 1: Examples of supervised machine learning (left) and unsupervised machine learning (right)

3. Reinforcement machine learning algorithms learn from their surroundings; if they perform well, they are rewarded.

2. Models and data

Data is used in machine learning models. They work with well-defined datasets, which are homogeneous collections of data points (for example, observations, photographs, or measurements) relevant to a certain situation, to make associations, uncover links, uncover trends, produce new samples, and more (for example, room temperature sampled every 5 minutes, or the weights of a group of individuals) Unfortunately, machine learning models' assumptions or conditions are not always obvious, and a protracted training procedure might end in a full validation failure. A model may be thought of as a gray box (the simplicity of many popular techniques ensures some transparency), where a vector input X retrieved from a dataset is turned into a vector output Y:

The model has been represented in the preceding diagram by a function that is dependent on a set of parameters provided by the vector. The dataset is made up of data taken from a real-world

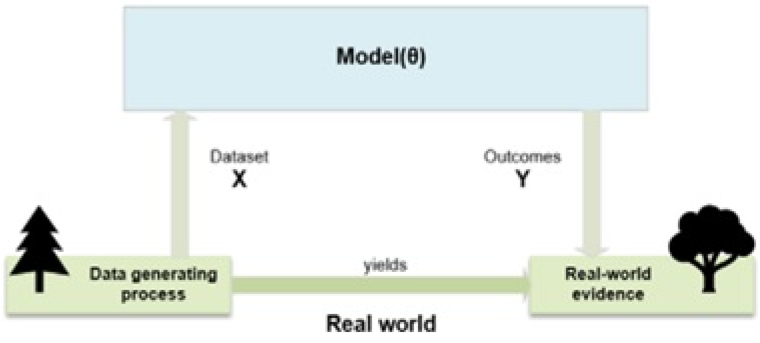

Figure 2: a generic model parameterized and its relationship with the real world

scenario, and the model's outputs must match the nature of the actual connections. In logic and probabilistic situations, where inferred circumstances must resemble natural ones, these requirements are quite strong [3, 10, 11].

3. Optimization in machine learning

The objective function of an optimization problem is described in terms of a collection of variables known as optimization variables. Actually, The goal of an optimization problem is to determine the variable values that maximize or minimize the objective function. In machine learning, it's customary to employ a minimization version of the objective function, and the accompanying objective function is known as a loss function. Now we'll look at several well-known optimization functions that are specified by the variable [5, 6, 8, 9].

3.1. Univariate optimization

Consider the following single-variable objective function $f(l)$:

$$f(l) = l^2 - 2l + 3$$

This objective function is an upright parabola with the formula $f(l) = (l-2)^2 + 2$. Figure (shows the objective function, which clearly reaches its minimum value at $l = 1$, when the nonnegative term $(1-2)^2$ drops to 0. The rate of change of $f(1)$ with respect to l is zero at the minimal value, because the tangent to the plot at that point is horizontal. The first derivative of the function $f(l)$ with respect to l may also be computed and set to 0 to determine the best value:
As a result, we arrive at $l = 1$ as the best value.

$$f^{'}(l) = \frac{df(l)}{dm} = 2l - 2 = 0$$

3.2. Bivariate optimization

The univariate optimization scenario is unrealistic since most optimization problems in real-world circumstances include many variables. We'll look at the circumstance of a two-variable optimization function to better understand the slight differences between single-variable and multivariable optimization. Bivariate optimization is a strategy that bridges the complexity gap between single-variable and multivariable optimization [5, 8]. For clarification, we'll look at bivariate extensions of

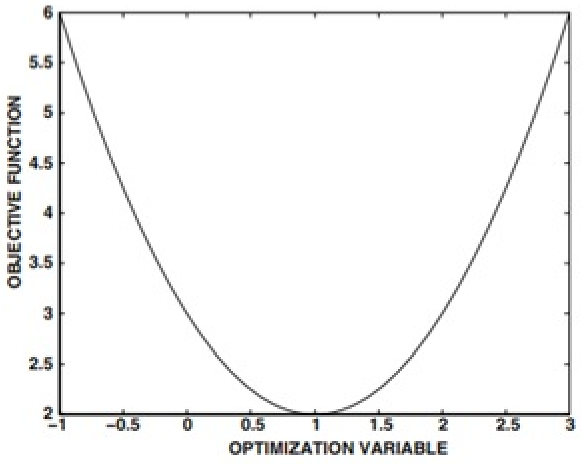

Figure 3: single global minimum $f(x) = x^2 - 2x + 3$

the univariate optimization functions in Figure (3). We may generate bivariate functions by merging two copies of the univariate function shown in Figure (4).

$$h(x, y) = f(l) + f(m) = l^2 + m^2 - 3l - 2m + 6$$
$$H(x, y) = h(l) + h(m) = \left[l^2 + m^2/3\right] - \left[l^3 + m^3/2\right] - l^2 - m^2 + 5$$

It's worth noting that these functions are simplified and have a unique structure; they're also additively separable. In reality, a "gradient" is a vector of partial derivatives by definition. The gradient of the function $h(l, m)$ in Figure (4)(a) may be calculated as follows:

$$\bigtriangledown h(l, m) = \begin{bmatrix} \frac{\partial g(l,m)}{\partial l} & \frac{\partial g(l,m)}{\partial m} \end{bmatrix}^T = \begin{bmatrix} 2l - 2 \\ 2m - 2 \end{bmatrix}$$

To indicate a function's gradient, the notation $\bigtriangledown$ is put in front of it. Because we have two optimization variables, l and m, the gradient is a column vector with two components in this case. The partial derivative of the objective function with respect to one of the two variables is represented by each component of the 2-dimensional vector. Setting the gradient $h(l, m)$ to zero is the easiest way to solve the optimization issue,

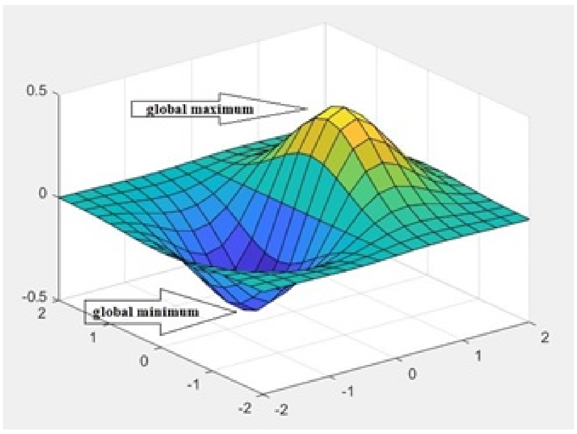

Figure 4: Global minimum and global maximum

resulting in the solution $[1, m] = [1, 1]$. Clear that global minimum of *h(l,m)* as follow

$$h(l, m) = f(l) + f(m) = l^2 + m^2 - 3l - 2m + 6$$

3.3. Multivariate optimization

A wide parameter space and numerous optimization variables characterize the majority of machine learning issues. “The variables in the optimization problem are variables that are utilized to build a prediction function for the visible and hidden characteristics of the machine learning issue. In a linear regression problem, the optimization variables $w_1, w_2, \dots, w_d$ are used to predict the dependent variable m from the independent variables $l1...ld$, for example:

$$m = \sum_{i=1}^{d} w_i \; l_i$$

Starting with this section, we’ll assume that only the notations w1...wd denote optimization variables, whereas the other “variables,” such as li and m, are actual observed values from the data set (which are constants from the optimization perspective). This is a standard notation for machine learning issues. Differences between observed and expected values of certain properties, such as the variable m given above, are frequently penalized by objective functions [1, 5, 8]. In machine learning, such objective functions are commonly referred to as loss functions. As a result, we’ll frequently use the phrase ”loss function” instead of ”objective function.” The loss function will be assumed to be a function of a vector of several optimization variables $w = \begin{bmatrix} w_1 & \dots & w_d \end{bmatrix}$w Setting the gradient vector to zero is the easiest way to solve the optimization issue directly (without gradient descent), resulting in the following set of d conditions:

$$\frac{\partial J(\bar{w})}{\partial w_i} = 0, \qquad \forall \; i \in \{1, 2, \dots, d\}$$

The parameters $w_1 \dots w_d$ may be determined by solving a system of d equations based on these circumstances. Similar to how one would characterize whether a critical point (i.e., zero-gradient point) is a maximum, minimum, or inflection point in univariate optimization, one would desire a way to characterize whether a critical point (i.e., zero-gradient point) is a maximum, minimum, or inflection point. The second-order condition kicks in at this point. Remember that the criteria for $f(w)$ to be a minimum in single-variable optimization is > 0. The Hessian matrix is used to expand this notion in multivariate optimization [2, 9].

4. Hyperparameters as a model ML optimization techniques.

Now we will present parameters and hyperparameters as model ML optimization strategies, but first we must grasp the distinction between parameters and hyperparameters of a model [4, 9, 10, 12]:

4.1. Model parameters

These are the entities that have been learnt from the training data during training. The designer does not establish them manually. The weights and biases are the model parameters for deep neural networks.

4.2. Model hyperparameters

These are the parameters that control how the model parameters are determined during training.- Heuristics are usually used to configure them manually. During the cross-validation step, they are fine-tuned (discussed later). Learning rate, number of layers, number of units in each layer, and many other factors are examples.

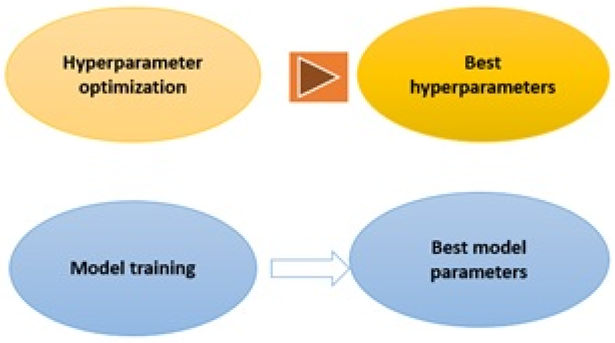

Figure 5: Parameters and hyperparameters of the model

Hyperparameter optimization is required to fine-tune the model. We can reduce error and develop the most accurate model by finding the best combination of their values.

4.3. Hyperparameter tuning technique

Multiple trials are done in a single training job to perform hyperparameter optimization. Each trial is a complete run of your training program, with values for your selected hyperparameters set within the constraints you define. The Cloud ML Engine training service maintains note of each trial's outcomes and adjusts for future trials. When the work is completed, you will receive a summary of all the trials as well as the most effective value configuration based on the criteria you select.

5. Top optimization techniques in machine learning

Let's take a look at some of the strategies for optimizing your model's hyperparameters.

5.1. Exhaustive search

It's the most basic of all the search strategies. The optimum of a function is bracketed by computing the values of the function at a number of evenly spaced locations, as shown in the diagram below (Figure 6). Typically, the search starts with a lower limit on the variable and compares three successive function values at a time, based on the premise that the function is unimodal. The search is either discontinued or continued based on the results of the comparison by changing one of the three points with a new point [2, 3].

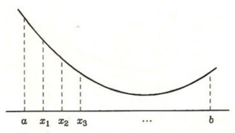

Figure 6: The exhaustive search method in simple case

Then, to minimize the error, we utilize gradient descent, which is the most frequent approach for model optimization. You must loop over the training dataset while re-adjusting the model to complete gradient descent. Our objective is to reduce the cost function since this ensures that the model has the minimum possible error and improves its accuracy.

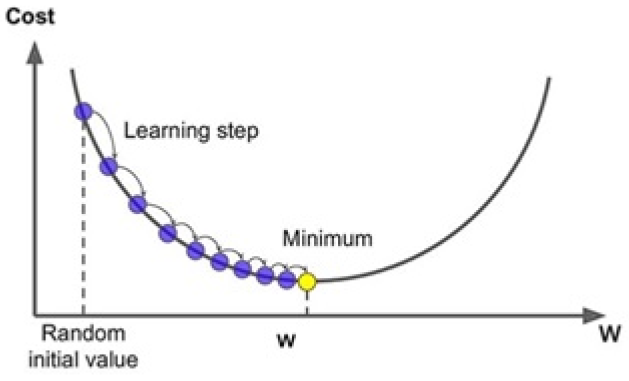

Figure 7: Gradient descent

You can see a graphical illustration of how the gradient descent process travels through variable space in the graph above. To get started, pick a random point on the graph and choose a direction at random. If you see that the mistake is growing, you've gone in the incorrect way. The optimization is complete when you can no longer improve (lower the error) and you have found a local minimum. A step-by-step description of how gradient descent works may be seen in the video below. In gradient descent, you take the same-sized steps ahead. If you set a learning rate that is too high, the algorithm will hop about in circles instead of coming closer to the correct answer. If it's too tiny, the computation will begin to resemble an exhaustive search, which is inefficient. As a result, you must pick your learning pace wisely. Gradient descent, when done correctly, becomes a computationally efficient and relatively rapid approach of model optimization [8, 9, 11].

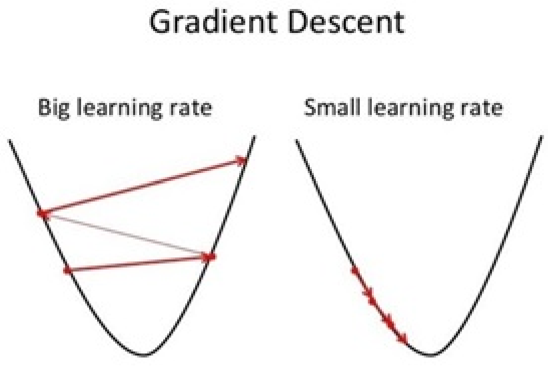

Figure 8: Small learning rate vs big learning rate

Another way to ML optimization is genetic algorithms. The underlying premise of these algorithms' reasoning is an effort to apply evolutionary theory to machine learning. Assume you have a collection of random algorithms at your disposal. This is the demographic you'll be dealing with. Some models with preset hyperparameters are better adjusted than others among several models. Let's go look for them! To begin, you must determine the correctness of each model. Then you just keep the ones that turned out the best. To produce a second generation of models, construct some offspring with comparable hyperparameters to the best models. We will continue this procedure many times until only the finest models remain at the end. Genetic algorithms aid in avoiding local minima and peaks. They are frequently used in the optimization of neural network models.

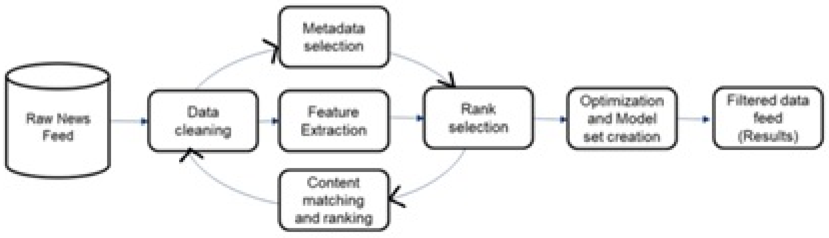
Raw News Feed
Data cleaning
Metadata selection
Feature Extraction
Content matching and ranking
Rank selection
Optimization and Model set creation
Filtered data feed (Results)

Figure 9: an example of the work of Genetic algorithms

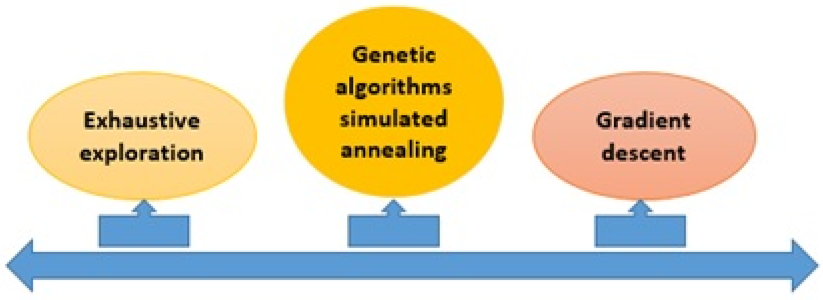

Figure 10: Optimization technique in our model

6. Conclusion

In this paper it is discussed several well-known optimization functions that are specified by the variable. Actually, ML algorithms must discover good solutions to small groups of problems with unique structures. Besides, models can be rewritten to allow for better algorithms as long as generalization is enhanced or not jeopardized. Because of the inherent flaws in machine learning models and the fact that erroneous answers are actively sought, high precision is not necessary. ML algorithms, on the other hand, must discover good solutions to small groups of problems with unique structures. Models can be rewritten to allow for better algorithms as long as generalization is enhanced or not jeopardized. Because of the inherent flaws in machine learning models and the fact that incorrect answers are actively sought as a type of regularization, high precision is not necessary. Finally, the focus is on hyperparameter optimization in automated machine learning.

References

[1] C. C. Aggarwal, *Linear algebra and optimization for machine learning*, Springer, 2020.
[2] G. Y. Ban, N. El Karoui and A. E. Lim, *Machine learning and portfolio optimization*, Manag. Sci. 3 (2018) 1136–1154.
[3] N. J. Browning, R. Ramakrishnan, O. A. Von Lilienfeld and U. Roethlisberger, *Genetic optimization of training sets for improved machine learning models of molecular properties*, J. Phys. Chem. Lett. 8 (2017) 1351–1359.

[4] C. Gambella, B. Ghaddar and J. Naoum-Sawaya, *Optimization models for machine learning*, A Survey. arXiv preprint arXiv:1901.05331, (2019).

[5] W. T. Garrison and M. D. Petty, *An analysis of machine learning online training approaches for simulation optimization*, IEEE Southeast Conf. (2019) 1–6.

[6] K. M. Hamdia, X. Zhuang and T. Rabczuk, *An efficient optimization approach for designing machine learning models based on genetic algorithm*, Neural Comput. Appl. 33 (2012) 1923–1933.

[7] Mirmozaffari, Mirpouya, M. Yazdani, A. Boskabadi, H. Ahady Dolatsara, K. Kabirifar and N. Amiri Golilarz, *A novel machine learning approach combined with optimization models for eco-efficiency evaluation*, Appl. Sci. 10(15) (2020) 5210.

[8] C. Song, T. Ristenpart and V. Shmatikov, *Machine learning models that remember too much.*, Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (2017) 587–601.

[9] S. Sun, Z. Cao, H. Zhu and J. Zhao, *A survey of optimization methods from a machine learning perspective*, IEEE Trans. Cyber. 50 (2019) 3668–3681.

[10] Z. Wang and M. O'Boyle, *Machine learning in compiler optimization*, Proc. IEEE, 11 (2018) 1879–1901.

[11] J. Wu X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei and S. H. Deng, *Hyperparameter optimization for machine learning models based on Bayesian optimization*, J. Elect. Sci. Tech. 1 (2019) 26–40.

[12] L. Yang and A. Shami, *On hyperparameter optimization of machine learning algorithms: Theory and practice*, Neurocomput. 415 (2020) 295–316.