JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Evolutionary Computational Methods and Techniques for Data Science Optimization

Mohamed Abdeldaiem Mahboub[1*], T.Gopi Krishna[2], Pyla Srinivasa Rao[3]

[1]Department of Information Systems, Faculty of Information Technology University of Tripoli, Libya

[2]CSE Department, Adama Science and Technology University, Ethiopia

[3]Senior Manager, Cyber Security, Capgemini, India

Abstract : **The emerging field of data science is dedicated to the analysis of vast datasets through diverse methodologies, aiming to enhance data comprehensibility. A solid grasp of statistics, linear algebra, and optimization is essential for a profound understanding of data science. The pursuit of model performance improvement involves the optimization of parameters and hyperparameters, seeking the most effective values for achieving desired outcomes. This study endeavors to introduce accessible algorithms, both new and established, emphasizing ease of implementation and comprehension. The focus extends to appreciating theoretical analyses and discerning which algorithm suits specific data science challenges. The objectives encompass the identification of various optimization problems, linking them with applicable solution methods, and the manual application of optimization techniques to solve modest problems. Additionally, the study delves into discussions on interpreting the sensitivity of optimization solutions to changes in parameter values.**

IndexTerms – Optimization, ML, DL, Data Science.

I. INTRODUCTION

Optimization refers to a method or process employed to identify the most efficient solution based on specified criteria. The nature of the requirement determines whether the objective is to minimize or maximize a particular value within a given category. For example, if a company aims to maximize returns on its products, the condition is set for maximum value; conversely, if the goal is to minimize product costs in production, the requirement is for a minimum value. In the realm of data science, optimization emerges as a potent tool for addressing intricate problems. It proves instrumental in tasks such as model selection, hyperparameter tuning, and feature prioritization. Moreover, optimization contributes to enhancing the accuracy and performance of machine learning algorithms, mitigating issues like overfitting and underfitting, and accelerating training speeds.

Recent advancements in data science optimization, encompassing automated machine learning, deep reinforcement learning, and neural architecture search, have revolutionized the speed and efficiency with which optimal solutions are identified by data scientists. Mathematically, an optimization problem involves identifying the optimal solution from a set of candidate or feasible options. In the context of data science, the quality of a data model is continually assessed through a cost function. Minimizing the cost function equates to finding an optimal parameter set that yields the minimum possible error in the system.

Given the intricacies and interconnectedness of advanced engineering systems, an analyst with comprehensive domain knowledge is essential for effective system optimization. The application of optimization theory and techniques to various scientific and research domains represents a crucial facet of Applied Mathematics. This study focuses on attaining the best solution, whether it be a minimum or maximum, and emphasizes that the optimization process is an art form dedicated to finding the optimal response to prevailing situations [1,2,3]

The subsequent step in the optimization process involves determining the problem's type or classification, a presentation of which is included in this study. Following this classification, the study deems it necessary, based on the problem's scope and requirements, to select the most compatible optimization tool from the available commercial and academic options. This chosen tool is then applied in the designated domain to optimize functions and derive optimal or approximate solutions aligned with the problem's needs [1,2,3]

II TYPES OF OPTIMIZATION TECHNIQUES USED IN DATA SCIENCE

There are several types of optimization techniques used in data science. These include gradient descent, simulated annealing, evolutionary algorithms, Bayesian optimization, and particle swarm optimization. Each technique has its own advantages and disadvantages. The choice of which technique to use depends on the problem at hand.

2.1 Gradient Descent

Gradient descent is a popular optimization technique used in data science. It is a first-order iterative optimization algorithm that uses the gradient of the cost function to update the parameters in order to minimize the cost function. Gradient descent is used in many machine learning algorithms, such as linear regression and logistic regression. The gradient descent approach is currently the method of choice for optimization. The objective of this tactic is to update the variables iteratively in a way that is opposed to how the gradients of the objective function move. This strategy directs the model to discover the target with each update, eventually converging to the optimal value for the objective function. So now that we know what a gradient descent is and how it works, let's start implementing the same in python. But, before we get to the code logic of the same, let's first take a look at the data we are going to be using and the libraries that we will be importing for our purpose.

For this implementation, we are going to use the promotion dataset. This is a dataset that gives us the total sales for different products, after marketing them on Laptop, GPU and TV. Using our algorithm, we can find out which medium performs the best for our sales and assign weights to all the mediums accordingly. Moving on to the imports, we will be using the libraries pandas and numpy, which will be used to read the data as well as perform mathematical functions on it. We will also be using matplotlib and seaborn to plot our findings and explain the results graphically. Once we have imported the libraries we will use pandas to read our dataset, and print the top 5 columns to reassure that the data has been read correctly. Once we have read the data, we will set the X & Y variables for our gradient descent curve. Gradient descent is an optimization algorithm which can be used for estimating the values of regression parameters, given a dataset with inputs and outputs [4,5].

Python Code sample:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as sb
# Read the CSV file 'Promotion.csv' into a DataFrame
df = p.read_csv('Promotion.csv')
# Display the first few rows of the DataFrame
df.head()
```

Here X would represent Laptop, GPU and TV's while Y would represent our sales. As all these purchase might be on different scales, we then normalize our X & Y variables. After Normalize

```
X=df[['Laptop','GPU','TV']]
Y=df['Purchase']
Y=np.array((Y-Y.mean())/Y.std())
X=X.apply(lambda rec:(rec-rec.mean())/rec.std(),axis=0)
```

Once we have a normalized dataset, we can start defining our algorithm. To implement a gradient descent algorithm we need to follow 4 steps:

Step1:Start by randomly initializing the bias and weight (θ).
Step2:Compute the predicted values of Y using the initialized bias and weight.
Step3:Evaluate the cost function based on the predicted values and the actual values of Y.
Step4:Determine the gradients and update the weights accordingly.
gradient = (dy/dx)=(d/dx)(4*x^2) = 8*x (Calculating the gradient function) Fig.1 shows Gradient descent cost and weight

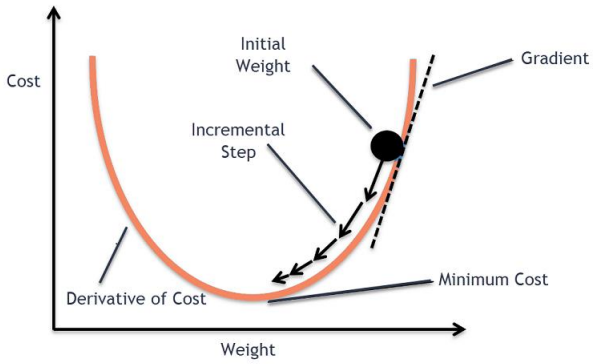

Fig 1 Gradient descent cost and weight

2.2 Simulated Annealing

Simulated annealing serves as a probabilistic optimization method employed to discover the global optimum of a specified problem. The approach involves stochastic exploration of the search space to identify a set of parameters associated with the lowest cost. Subsequently, the algorithm adjusts these parameters, leveraging insights from the current best solution. Simulated annealing finds application in diverse domains, including neural networks and clustering. A visual representation of simulated annealing is depicted in Fig.2 [6,8,9].

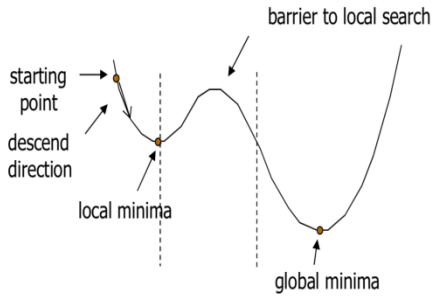

Fig 2 Simulated Annealing

2.3 Evolutionary Optimization Algorithms

Inspired by biological evolution, evolutionary algorithms are metaheuristic optimization techniques. These algorithms generate a population of potential solutions and subsequently identify the most promising ones according to their fitness. Within the realm of data science, evolutionary algorithms find diverse applications such as feature selection, model selection, and hyperparameter tuning [7,9].

2.3.1 Bayesian Optimization

Bayesian optimization is a probabilistic optimization technique that utilizes Bayesian inference to optimize parameters. It works by constructing a prior distribution over the parameter space and then updating it based on the data. Bayesian optimization is used in hyperparameter tuning and model selection. Fig 3 shows three iterations of Bayesian optimization [7,9].

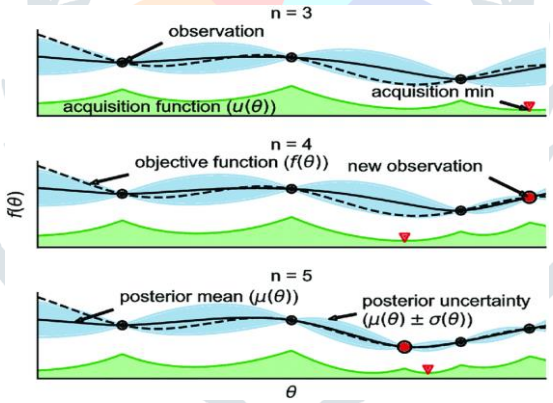

Fig 3. Three iteration of Bayesian Optimization

2.3.2 Particle Swarm Optimization

Derived from the collective behavior of birds flocking, particle swarm optimization is a population-based optimization technique. This method involves forming a swarm of particles, with each particle representing a potential solution. The positions of these particles are then adjusted based on the best solutions identified. Particle swarm optimization finds application in various domains, including feature selection, model selection, and hyperparameter tuning.

Optimization can be used to solve a variety of complex problems in data science. It can be used to select the most suitable model for a given problem, tune hyperparameters, and select the most relevant features. Optimization can also be used to improve the accuracy and performance of machine learning algorithms, reduce overfitting and underfitting, and increase training speed.

Fig 4 shows PSO [7,9].

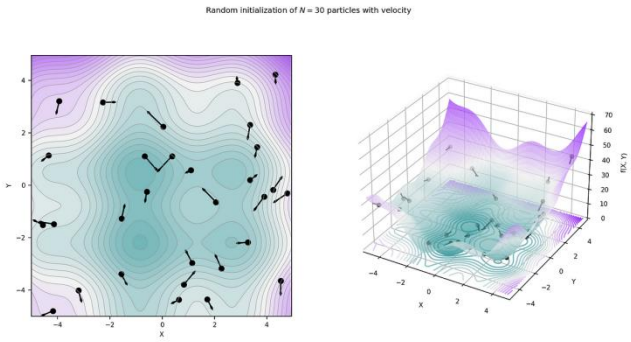

Fig 4. PSO

2.4 Model Selection

Optimization can be used to select the most suitable model for a given problem. This is done by testing different models and evaluating their performance using a given metric. The model with the highest score is then chosen as the best model. Fig 5. Shows Data driven model selection for optimization [7,9].

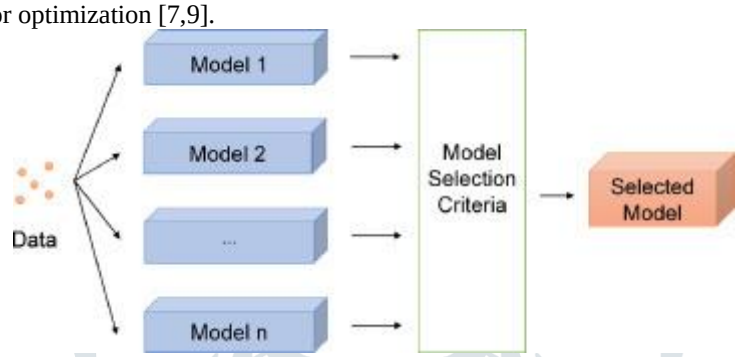

Fig.5 Data driven evolutionary optimization based on model

2.5 Hyperparameter Tuning

The refinement of model performance by adjusting the values of model parameters is known as hyperparameter tuning. Techniques like gradient descent, simulated annealing, and evolutionary algorithms are employed in this process to identify the optimal parameter set for a given model. Figure 6 provides a visual representation of hyperparameter tuning [7,9].

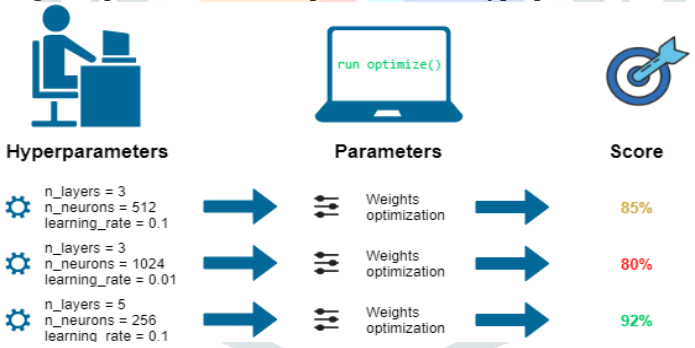

Fig 6. Hyperparameter Tuning Process for optimization

2.6 Feature Selection and Engineering

The process of choosing the most pertinent features for a given problem is referred to as feature selection. Optimization techniques play a crucial role in identifying the optimal set of features for a given model. Concurrently, feature engineering involves transforming existing features into more advantageous forms, and optimization can be applied to determine the most effective ways to carry out these transformations. Fig. 7 illustrates the feature selection process [7,9].

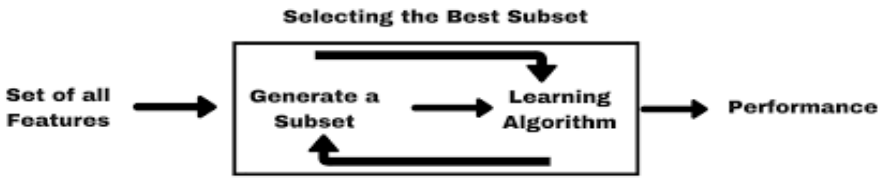

Fig 7. Feature Selection Process

2.7 Optimization Techniques in Data Science, Machine Learning, and Deep Learning

There are two types of supervised learning tasks: Regression & Classification [7,9].

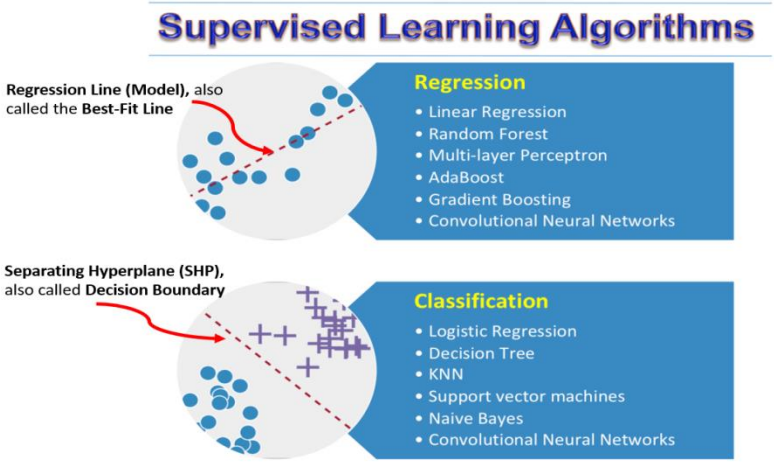

Fig.8 Supervised learning tasks-Regression and Classification

Table 1 presents a list of error functions minimized in widely utilized regression algorithms.

Table 1 Cost function or Loss function in Regression

Name of Regression Algorithm	Loss (Cost) Function minimized
Linear Regression	Mean Square Error (MSE)
Polynomial Regression	Mean Square Error (MSE)
Regularized Regression (Ridge/LASSO)	SSE + Penalty Term
Decision Tree Regressor	Sum of Squares of Errors (SSE)
Support Vector Regressor (SVR)	L1-loss (same as Sum of absolute values of Errors) or L2-Loss (similar to SSE)

In addition to the information provided in Table 1, it is worth noting that certain machine learning (ML) and deep learning (DL) algorithms also incorporate the use of Huber loss.

In the context of classification problems, the primary objective is to determine the optimal separating hyperplane, typically a line or curve that passes between the data points. This hyperplane should effectively segregate the majority of data points into two or more classes. The goal is to minimize the classification error, which represents inaccurately labeled data points. The specific form of the classification error varies based on the type of classifier algorithm in use. The diverse expressions of this error for different ML algorithms are summarized in the following Table.2.

Table 2 Cost function in Classification

Name of Classifier Algorithm	Loss (Cost) Function minimized
Logistic Regression	Binary Cross Entropy (BCE), commonly called as negative-log-loss
Support Vector Classifier (SVC)	Hinge Loss
Decision Tree Classifier	Gini or Entropy or "log_loss" (you can choose)

III IMPACT OF OPTIMIZATION ON MACHINE LEARNING ALGORITHMS

Enhancing the performance of machine learning algorithms is greatly influenced by optimization. This process contributes to refining the accuracy and overall performance of the model, mitigating issues like overfitting and underfitting, and accelerating training speeds. Additionally, optimization plays a pivotal role in assisting data scientists in identifying the most fitting model for their specific problem, fine-tuning hyperparameters, and selecting the most pertinent features.

3.1 Improving Accuracy and Performance

Leveraging optimization techniques can enhance the accuracy and overall performance of machine learning algorithms. Data scientists can fine-tune parameters or hyperparameters to ensure the model operates at its peak, resulting in heightened accuracy and improved performance..

3.2 Reducing Overfitting and Underfitting

Optimization can help reduce overfitting and underfitting by finding the optimal set of parameters or hyperparameters. By doing so, data scientists can ensure that their model is not too complex or too simple. This can lead to improved accuracy and performance.

3.3 Increasing Training Speed

Optimization can also be used to increase the training speed of machine learning algorithms. By optimizing parameters or hyperparameters, data scientists can ensure that the model is running efficiently and quickly. This can lead to improved training speed. Implementing optimization strategies in data science projects can be a challenging task. To ensure success, data scientists should follow the following steps:

3.4 Establishing Objectives and Constraints

The first step in implementing optimization strategies is to establish the objectives and constraints of the project. This includes defining the goals and constraints of the project, such as the desired accuracy and performance, the computational budget, and the time limit.

3.5 Choosing the Right Optimization Technique

Once the objectives and constraints have been established, data scientists should choose the right optimization technique for their project. This can involve comparing different optimization techniques and selecting the one that best fits the project's needs.

3.6 Preparing for Optimization

Prior to initiating the optimization process, data scientists should ready their data and code. This involves tasks such as data cleaning, normalization, segmentation into training and test sets, and the preparation of essential code for executing the optimization process.

3.7 Executing the Optimization Process

Once the data and code are ready, data scientists can begin the optimization process. This involves running the optimization algorithm on the data and adjusting the parameters or hyperparameters until the desired result is achieved.

IV RECENT ADVANCES IN DATA SCIENCE OPTIMIZATION

In recent years, there have been several advances in data science optimization. These include automated machine learning, deep reinforcement learning, and neural architecture search. These methods can automate the optimization process and enable data scientists to find optimal solutions faster and more efficiently.

4.1 AutoML: Streamlining Machine Learning Processes

Automated machine learning (AutoML) is a technique that automates the optimization process. AutoML algorithms are able to automatically select the best model and tune its parameters or hyperparameters to achieve the desired performance [8,9].

4.2 Deep Reinforcement Learning

Deep reinforcement learning, a machine learning algorithm, integrates both deep learning and reinforcement learning. Its application extends to optimizing parameters or hyperparameters to attain the desired performance level [8,9].

4.3 Neural Architecture Search

Neural architecture search is an automated method for designing neural networks, employing optimization algorithms to identify the most effective architecture tailored to a specific task.

V ADVANCED OPTIMIZATION METHODS AND TECHNIQUES

5.1 Stochastic Gradient Descent

The stochastic gradient descent (SGD) algorithm was developed to solve the high computing cost involved in each iteration of the process when dealing with enormous amounts of data. The equation can be written as follows:

Back-propagation refers to taking the values and iteratively modifying them depending on various parameters to lower the loss function.

Instead of directly calculating the exact value of the gradient, this approach updates the gradient (theta) by randomly using one sample at each iteration. This is done in place of directly calculating the gradient's value. The stochastic gradient approximates the true gradient that does not consider any outside factors. This optimization approach cuts down on the amount of time needed to do an update when working with a large number of samples, and it also eliminates some redundant computational work. Fig 9. Shows the absolute minimum of stochastic gradient descent.

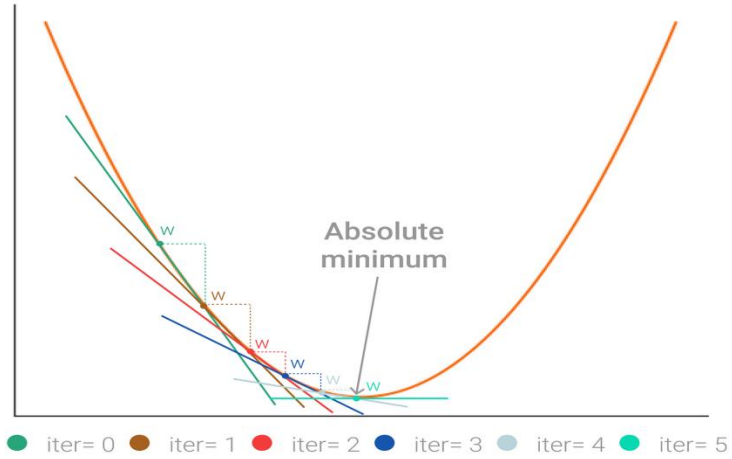

Fig 9. Stochastic Gradient Descent

5.2 The Technique of Adaptive Learning Rate

Learning rate is one of the primary hyperparameters optimized during the process. The model's behavior depends on the learning rate, which determines whether it will pass over particular data sections. When the learning rate is high, there is a possibility that the model will overlook more nuanced parts of the data. If it is low, then it is preferable for applications that take place in the real world. The rate of learning has a significant impact on the Stochastic Gradient Descent (SGD). It can be difficult to determine what the optimal value of the learning rate should be. It was suggested that adaptive approaches could automatically do this tweaking.

In Deep Neural Networks (DNNs), the adaptive forms of SGD have seen the widespread application. Methods such as AdaDelta, RMSProp, and Adam all use exponential averaging to offer accurate updates while keeping the process as straightforward as possible. Weights with a steep gradient will have a slow learning rate and vice versa. This is known as the degrade formula. RMSprop modifies the Adagrad method so that it slows the rate at which it monotonically decreases the amount learned. Adam is virtually identical to RMSProp, except that he possesses momentum.

The Alternating Direction Method of Multipliers, sometimes known as ADMM, is an additional alternative to the Stochastic Gradient Descent method (SGD).

The learning rate is not predetermined in either the gradient descent or AdaGrad approaches, which is the key distinction between the two. The computation for it makes use of all of the historical gradients that have been accumulated up to the most recent iteration. Fig 10. Shows impact of learning rate of Neural Network.

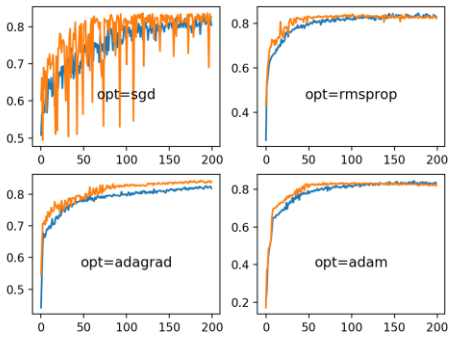

Fig 10. Impact of Learning Rate on Neural Network

5.3 Method of the Conjugate Gradient

The conjugate gradient (CG) method is applied in the process of solving nonlinear optimization issues in addition to large-scale linear systems of equations. A slow convergence speed is characteristic of first-order approaches. On the other hand, the approaches of the second order require a lot of resources. An intermediate algorithm known as conjugate gradient optimization combines the benefits of first-order information with the high-order methods' ability to ensure rapid convergence.

5.4 Optimization without the use of derivatives

Certain optimization problems are almost always solvable by employing a gradient, notwithstanding the possibility that the derivative of the objective function does not exist or is difficult to calculate. This is due to the fundamental characteristics of the issue. Derivative-free optimization enters the picture at this point to help solve the problem. Instead of deriving solutions methodically, it uses a heuristic algorithm that picks approaches that have previously been successful. There are numerous examples of this: genetic algorithms, particle swarm optimization, and classical simulated annealing arithmetic. Fig.11 shows the

zeroth order optimization.

5.5 Zeroth order optimization

Recent years have seen the development of derivative-free optimization's successor, zeroth order optimization, which aims to remedy the faults of its predecessor. Methods of optimization that don't use derivatives have trouble scaling up to huge problems and don't provide a way to analyze how quickly they converge on a solution.

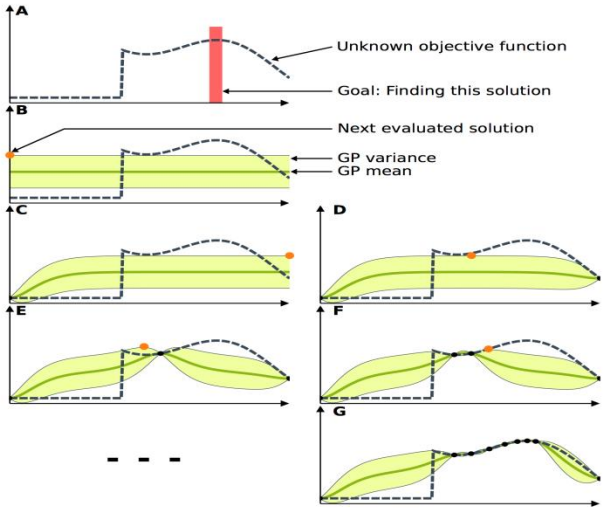

Fig 11. Zeroth Order Method

Advantages of the Zeroth Order include the following:

Ease of implementation that requires only a minor adjustment to the gradient-based methods that are most frequently employed.

Approximations of derivatives that are computationally efficient in situations when the derivatives themselves are difficult to compute convergent rates that are comparable to those of first-order algorithms.

VI Elements of an Optimization Problem

Typically, an optimization problem consists of three main elements.
minimize f(x), w.r.t x, subject to a $\leqslant$ x $\leqslant$ b

a. **The objective function (f(x))**: The Objective Function (f(x)): The initial component is the objective function, denoted as f(x), which aims to be either maximized or minimized. Typically, discussions focus on minimization problems, as a maximization problem with f(x) can be transformed into a minimization problem by considering -f(x). Hence, without sacrificing generality, the focus is primarily on minimization problems [10].
b. **Decision Variables (x):** The second component involves decision variables, denoted as x, which are chosen to minimize the function. This is expressed as min f(x) [10].
c. **Constraints (a ≤ x ≤ b):** The third component comprises constraints that limit the range of x to a specific set, often expressed as a $\leqslant$ x $\leqslant$ b. When examining an optimization problem, it is crucial to identify these three components.

VI Varieties of Optimization Problems

Exclusively based on the nature of constraints:

a. **Constrained Optimization Problems:** When a set of constraints is provided, and the solution must adhere to these constraints, we refer to them as constrained optimization problems.
b. **Unconstrained optimization problems:** In cases where the constraint is missing we call them unconstrained optimization problems.

Varied by the nature of objective functions, decision variables, and constraints:

a. **In the scenario where the decision variable (x) is continuous:** A variable x is considered continuous if it can assume an infinite number of values. In such cases, x has the potential to take on an unending range of values within the interval from -2 to 2 [10].
min f(x), x $\in$ (-2, 2)
b. **Linear Programming Problem:** If the decision variable (x) is continuous and both the objective function (f) and all constraints are linear, the problem falls under the category of a linear programming problem. Therefore, in this context, the decision variables are continuous, the objective function is linear, and all constraints are linear as well [10].
c. **Nonlinear Programming Problem:** When the decision variable (x) remains continuous, but either the objective function (f) or the constraints are non-linear, the problem is termed a non-linear programming problem. Therefore, a

programming problem becomes non-linear if either the objective or the constraints are non-linear [10].

If the decision variable (x) is an integer variable: Integers are defined as numbers whose fractional part is 0 (zero), such as -3, -2, 1, 0, 10, and 100 [10].

min f(x), x $\in$ [0, 1, 2, 3]

d. **Linear Integer Programming Problem:** When the decision variable (x) is an integer variable, and both the objective function (f) and all constraints are linear, the problem is termed a linear integer programming problem. Thus, in this scenario, the decision variables are integers, the objective function is linear, and the constraints are also linear [10].

e. **Nonlinear Integer Programming Problem:** When the decision variable (x) is maintained as an integer, but either the objective function (f) or the constraints are non-linear, the problem is referred to as a nonlinear integer programming problem. Hence, a programming problem becomes nonlinear if either the objective or the constraints assume a non-linear form [10].

f. **Binary Integer Programming Problem**: When the decision variable (x) is restricted to only binary values, such as 0 and 1,the problem is categorized as a binary integer programming problem [10].

min f(x), x $\in$ [0, 1]

In the case where the decision variable (x) is a mixed variable: If a combination of continuous and integer variables is employed, this type of decision variable is referred to as a mixed variable [10].

min f(x1, x2), x1 $\in$ [0, 1, 2, 3] and x2 $\in$ (-2, 2)

g. Mixed-Integer Linear Programming Problem: When the decision variable (x) is a mixed variable, and both the objective function (f) and all constraints are linear, the problem is categorized as a mixed-integer linear programming problem. Consequently, in this scenario, the decision variables are mixed, the objective function is linear, and the constraints are also Linear [10].

h. Mixed-Integer Nonlinear Programming Problem: When the decision variable (x) is maintained as mixed, but either the objective function (f) or the constraints are non-linear, the problem is identified as a mixed-integer nonlinear programming problem. Thus, a programming problem transitions into a nonlinear state if either the objective or the constraints assume non-linear characteristics [10].

VII APPROACHES TO ADDRESSING OPTIMIZATION PROBLEMS

Usually, there are two approaches to addressing the majority of intricate numerical or scientific problems:

- **Analytical Method:** This illustrates the conventional theoretical method, often referred to as the "pen-and-paper" approach, typically taught in traditional school or university curricula. These methods provide a precise solution for well-defined mathematical problems that are solvable. For instance, when tasked with identifying the values of x for which the function f(x) = 2x^2 – 3x + 1 equals zero (commonly known as the "roots" of the polynomial), we are conventionally instructed to solve for the roots of a generic second-degree polynomial ax^2 + bx + c = 0, as demonstrated below:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = 0.5 \text{ and } 1.0$$

- **Numerical Methods Approach:** In this approach, the customary procedure involves initializing the process with an initial guess value as a potential solution. Subsequent computations are then executed, entailing the assessment of the function f(x), and the solution is iteratively refined until reaching the desired outcome, a state denoted as convergence. To illustrate, if a tolerance level of 0.0001 is set, the iterations will halt when the value of f(x) equals or falls below 0.0001, indicating the attainment of convergence in the entire optimization process. Well-known numerical methods for zero-finding encompass Newton's method, the Secant method, Brent's method, and the Bisection method [12].

7.1 Optimization problems and solution methods

To address optimization problems, we employ two distinct approaches: Fig 11. Shows the solution methods

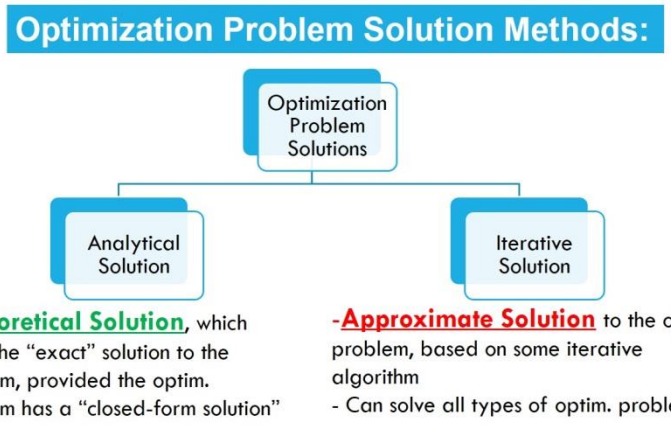

Fig 11. Methods of optimization problem

- **Analytical method:** An example of this is the Ordinary Least Squares Method (OLS) utilized by the Linear Regression class in the scikit-learn package in Python and the numpy polyfit function. As previously mentioned, the OLS technique furnishes accurate coefficients (slopes and intercept terms) for the best-fit line [12].
- **Iterative Solution:** The commonly embraced iterative strategy for addressing optimization problems in machine learning involves the utilization of the Gradient Descent Algorithm and its variations, encompassing Stochastic Gradient Descent and MiniBatch Gradient Descent. Furthermore, advanced variants like RMSprop, Adam, Adadelta, Adagrad, Adamax, and Nadam find application, especially in the domain of deep learning [12].

7.2 Determining the Appropriate Approach for Resolving Specific Optimization Problems

For unconstrained minimization, several methods are at one's disposal, including the Conjugate Gradient (CG), Newton's Conjugate Gradient, or the quasi-Newton Broyden, Fletcher, Goldfarb, and Shanno algorithm (BFGS). Additional options include the Dog-leg Trust-region algorithm, Newton Conjugate Gradient Trust-region algorithm, and the Newton GLTR trust-region algorithm. In a similar vein, when dealing with constrained optimization problems, both with and without bounds, a variety of algorithms can be employed, such as Nelder-Mead, Limited-memory BFGS (L-BFGS), Powell's method, Truncated Newton's Conjugate Gradient, Sequential Least Squares Programming (SLSQP), and others[12].

7.3 Implementation of Optimization

Certainly, optimization holds a pivotal role in every machine learning, including deep learning, algorithm, and data science pursuit. Delve into the examination of the three data points presented in Table 3.

Table 3. Sample dataset

X1	Y
1	4.8
3	12.4
5	15.5

Individuals with a background in machine learning will quickly recognize X1 as the independent variable, often referred to as "features" or "attributes," while Y represents the dependent variable, also termed the "target" or "outcome." The primary goal for any machine is to discern the relationship between X1 and Y, a process essentially "learned" by the machine through data, giving rise to the concept of "machine learning." Similar to human learning from experiences, machines assimilate comparable experiences in the form of data. Figure 12 illustrates optimization errors. Now, let's consider a scenario where the objective is to determine the best-fit line for a set of three given data points. The plot below showcases these points represented by blue circles. Additionally, there is a red line (with squares) identified as the "best-fit line" through these three data points. For comparison, a "poor-fitting" line (depicted in yellow) is also presented.

Fig 12. Error comparison of 3 data points

The ultimate goal is to derive the equation for the best-fitting straight line that encompasses the three data points mentioned in the above Table.3 [12].

$$\hat{\mathbf{Y}} = \mathbf{w}_0 + \mathbf{w}_1\mathbf{X}_1 \qquad \text{---------- (1)}$$

Represents the equation of the best-fit line (depicted by the red line in the preceding plot), with w1 as the slope of the line and w0 as the intercept of the line. In the realm of machine learning, this optimal fit is commonly referred to as the Linear Regression (LR) model, and w0 and w1 are termed as model weights or coefficients [12].

The anticipated values derived from the Linear Regression model (denoted as Y^) are illustrated by red squares in the preceding plot. It is important to note that these predicted values may not precisely match the actual values of Y (depicted by blue circles). The vertical disparity between them signifies the error in the prediction, as indicated by:[12].

$$\mathbf{Error_i} = \hat{\mathbf{Y}}_\mathbf{i} - \mathbf{Y_i} \qquad \text{---------- (2)}$$

For any i^{th} data point, I assert that this best-fit line will exhibit the minimum prediction error compared to all possible infinite random "poor-fit" lines. The cumulative error across all data points is quantified by the Mean Squared Error (MSE) Function, which is minimized for the best-fit line [12].

$$\mathbf{MSE} = \frac{1}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{N}}(\mathbf{Error_i})^2 = \frac{1}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{N}}(\hat{\mathbf{Y}}_\mathbf{i} - \mathbf{Y_i})^2 \qquad \text{---------- (3)}$$

Where N represents the total number of data points in the dataset (in this instance, N equals 3). The mathematical pursuit of minimizing or maximizing any quantity is designated as an optimization problem, and the point where the minimum/maximum is attained is termed the optimal values of the variables [12].

(a) OLS Method:

$$L(w_j) = \left(\frac{1}{N}\right)\sum_{i=1}^{N}(w_0 + w_1 X_{1i} - Y_i)^2 \qquad \text{..... (4)}$$

Evidently, L is a function contingent on model weights (w0 and w1), and the task at hand is to identify their optimal values through the minimization of L. The optimal values are denoted by (*) in the Figure 13. [12].

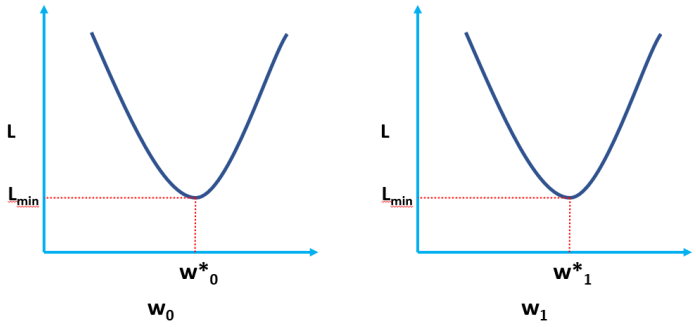

Fig 13. Optimal values find upon minimizing L

Without delving into the derivations in this article (which are planned for a follow-up piece), the ultimate optimal values of w0 and w1, where the MSE Loss function attains its minimum, are provided as follows: [12].

$$w_1^* = \frac{\left[\overline{X_1 Y} - \overline{X_1}\,\overline{Y}\right]}{\left[\left(\overline{X_1^2}\right) - \left(\overline{X_1}\right)^2\right]} \quad \text{and} \quad w_0^* = \overline{Y} - \left(w_1^* \cdot \overline{X_1}\right) \qquad \text{..... (5)}$$

Where $\overline{X_1}$ represents the mean of the values of X_1 feature. Similarly, $\overline{Y}$ is the mean of the Y values, $\overline{X_1 Y}$ is the mean of the products X1 times Y, $\overline{X_1}\,\overline{Y}$ represents the product of the means of X1 & Y, and $\overline{X_1^2}$ represents the mean of the squares of the values of X_1. The calculations are illustrated in the table below.

Table 4. Calculation of Mean of the square

	X_1	Y	X_1^2	X_1*Y
	1	4.8	1	1x4.8
	3	12.4	9	3x12.4
	5	15.5	25	5x15.5
Mean	9/3 = 3	32.7/3 =10.9	35/3 = 11.67	(119.5)/3 = 39.83

$$w_1^* = \frac{[39.83-(3\times10.9)]}{[(11.67)-(3)^2]} = \frac{7.13}{2.67} = 2.67$$

$$w_0^* = 10.9-(2.6704*3) = 2.88$$

7.4 Exploring Convexity and Concavity

In the realm of data science and optimization, a foundational understanding of convex and concave functions is crucial. These mathematical concepts have significant implications for uncovering optimal solutions and gaining insights into the behavior of various algorithms.

7.4.1 Convex Function

A function is considered convex if, for any two points within its domain, the line segment connecting those two points lies either above or coincides with the graph of the function. Mathematically, a function f(x) is convex when the following inequality holds for any points x and y within its domain, and for any value of t between 0 and 1: [12].

$f((1-t)*x+t*y) \leq (1-t)*f(x)+t*f(y)$

A function is deemed convex when, for any two points within its domain, the line segment connecting those two points either lies above or coincides with the graph of the function. In mathematical terms, a function f(x) is convex if:

7.4.1.1 Optimization with Convex Functions in Data Science

- Linear regression's optimization problem centers on minimizing the convex function, the mean squared error. This error metric is convex concerning model parameters (coefficients) due to its composition as the sum of squared differences, with the inherent convexity of the square function.

- Support Vector Machines (SVM) formulate their optimization problem as the search for a hyperplane that maximizes data separation, translating into a convex quadratic programming problem. The objective of optimizing the margin is inherently a convex function.

- Logistic regression employs the logistic loss function, or cross-entropy loss, which is inherently convex. This function quantifies the difference between predicted probabilities and true binary class labels.

- Ridge regression addresses its optimization problem by minimizing a convex function, a combination of the mean squared error and the L2 regularization term. The convexity is maintained through the square function applied to the L2 regularization term, which is the squared sum of model coefficients.

- Support Vector Regression (SVR) minimizes a convex function representing the loss due to deviations from true target values. It allows specific deviations within a tolerance margin (epsilon).

- Elastic Net's optimization problem revolves around minimizing a convex function, incorporating the mean squared error and a combination of L1 and L2 regularization terms. The convex nature of both L1 and L2 regularization terms ensures the overall convexity of the objective function [12].

7.4.2 Utilizing Convex Functions for Optimization in Data Science

$f((1 - t) * x + t * y) \geq (1 - t) * f(x) + t * f(y)$ for $0 \leq t \leq 1$

A concave function exhibits a shape reminiscent of an inverted bowl or a cave, featuring a distinctive global maximum. Consequently, any local maximum within the function is also its global maximum. Figure 14 illustrates both convex and concave functions [12].

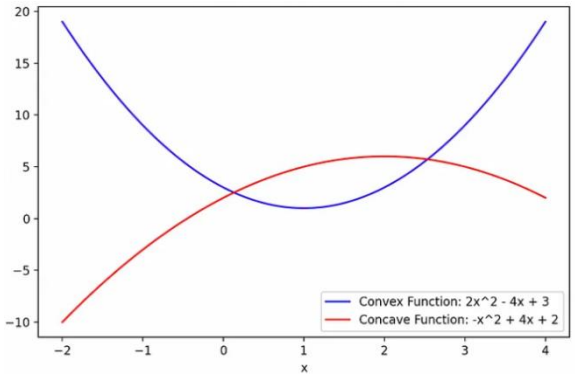

Fig 14. Convex and Concave Functions

7.4.2.1 Optimization with Concave Functions in Data Science

o Within decision trees, the entropy function serves as a metric for impurity or uncertainty within a node. The primary goal is to diminish this uncertainty by choosing a split that maximizes information gain, effectively minimizing the concave entropy function.

o The likelihood function evaluates how effectively a model's parameters explain observed data. The transformation of the optimization problem into a concave function, achieved through the negative logarithm of the likelihood, streamlines the minimization process to uncover maximum likelihood estimates of model parameters.

o Numerous data science tasks, encompassing probabilistic modeling, clustering, and information retrieval, often revolve around determining the optimal probability distribution aligned with the target distribution. This alignment is attained through the minimization of the concave Kullback-Leibler (KL) divergence between the two distributions.

o Various machine learning endeavors, including clustering and feature selection, employ conditional entropy to optimize the assignment of data points to clusters or select the most informative features.

o In Bayesian statistics, the negative log-likelihood of the Gaussian distribution functions as a concave loss function for specific regression tasks, such as maximum a posteriori (MAP) estimation.

o Reinforcement learning agents aspire to acquire policies maximizing cumulative rewards. The utilization of the negative exponential function of the expected reward motivates agents to take actions leading to higher rewards while avoiding those with lower rewards. Figure 15 provides an illustration of commonly used methods for optimizing data science tasks [12].

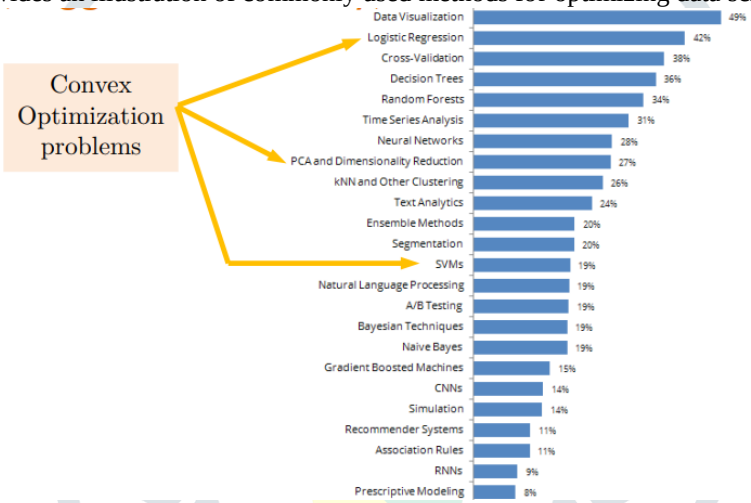

Fig 15. Data Science Methods Most used

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Analytics Vidya, for their invaluable guidance and mentorship throughout the course of this research. Their expertise and unwavering support significantly contributed to the success of this project. I am thankful to University of Tripoli, dept of IT for their collaborative efforts, insightful discussions, and technical assistance. Their contributions greatly enriched the quality of this research. I am also grateful to my colleagues and friends who provided encouragement and support during the various stages of this research.Lastly, I would like to acknowledge the anonymous reviewers and editors whose constructive feedback greatly improved the final version of this manuscript.

CONCLUSION

In conclusion, the burgeoning field of data science is dedicated to unraveling the intricacies of extensive datasets through diverse methodologies, with the overarching goal of augmenting data comprehension. A foundational understanding of statistics, linear algebra, and optimization proves indispensable in navigating the nuances of data science. The relentless pursuit of refining model performance involves the intricate process of optimizing parameters and hyperparameters, meticulously seeking optimal values to achieve desired outcomes. This study serves as a comprehensive exploration, introducing a spectrum of algorithms, both contemporary and established, with a keen emphasis on accessibility and ease of implementation. The narrative extends beyond algorithmic intricacies to encompass an appreciation of theoretical analyses, aiding in the discernment of which algorithm is most suited to particular data science challenges. The study's objectives encompass the identification of diverse optimization problems, establishing connections with relevant solution methods, and the hands-on application of optimization techniques for tackling modest problems. Furthermore, the study engages in insightful discussions, shedding light on the nuanced interpretation of optimization solutions and their sensitivity to variations in parameter values. By providing a holistic perspective, this research contributes to the foundational knowledge base essential for practitioners and researchers in the dynamic realm of data science.

REFERENCES

[1] Du, D. Z.; Pardalos, P. M.; Wu, W. (2008). "History of Optimization". In Floudas, C.; Pardalos, P. (eds.). Encyclopedia of Optimization. Boston: Springer. pp. 1538–1542.
[2] A.G. Malliaris (2008). "stochastic optimal control," The New Palgrave Dictionary of Economics, 2nd Edition.
[3] Siadati, Saman. (2013). Evolutionary Computation.10.13140/RG.2.2.36143.15526.
[4] C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, (September):1–11, 1992.
[5] Herty, M.; Klar, A. (2003–01–01). "Modeling, Simulation, and Optimization of Traffic Flow Networks". SIAM Journal on Scientific Computing. 25 (3): 1066–1087.
[6] Robbins, H. and Monro, S. (1951). A stochastic approximation method. Annals of Mathematical Statistics, 22(3).
[7] A. Beck. First Order Methods in Optimization. MOS-SIAM Series on Optimization, 2017.
[8] Sixin Zhang, Anna Choromanska, and Yann LeCun. Deep learning with Elastic Averaging SGD. Neural Information Processing Systems Conference (NIPS 2015), pages 1–24, 2015.
[9] Heena Rijhwani, Analytics Vidhya,web blog,2020.
[10] **https://www.geeksforgeeks.org/optimization-for-data-science/ ,** 16 Jul, 2020.
[11]. https://en.wikipedia.org/wiki/Ordinary_least_squares.
[12] https://medium.com/tag/data-science