Rational Classification

Special Project

Chair of Applied Mathematics / Faculty of Civil Engineering /
Bauhaus-University Weimar

Devang Gupta (126828)
Nayeemuddin Mohammed (123326)
Shivakiran Nandala (125218)

Examiners
Prof. Dr. rer. nat. Björn Rüffer
Dr. rer. nat. habil. Michael Schönlein

Date of submission: July 21, 2024

Bauhaus-Universität Weimar

Declaration

We hereby certify that the work embodied in the report is our own work, conducted under normal supervision. We confirm that the report contains no material which has been accepted, or is being examined, for the award of any other degree or diploma in any university or other tertiary institution. To the best of our knowledge and belief, the report contains no material previously published or written by another person, except where due reference has been made. We give consent to the final version of our report being made available worldwide when deposited in the University's digital repository, subject to applicable copyright law and any approved embargo.

Weimar, July 21, 2024

Devang Gupta (126828)
Nayeemuddin Mohammed (123326)
Shivakiran Nandala (125218)

Contents

Abstract

This report investigates the use of rational approximation for enhancing image classification accuracy, specifically applied to the MNIST dataset. Rational functions, which offer greater flexibility and precision in modeling decision boundaries compared to prevalent methods such as multi layer perceptron, are employed to approximate the decision boundaries required for effective classification. This study compares two optimization techniques, one using Bisection method and another utilizing Broyden-Fletcher-Goldfarb-Shanno (BFGS), and is aimed at improving classification accuracy and reducing computational time. Applying the two optimization techniques on the MNIST dataset sows that the BFGS method outperforms the Bisection method in terms of convergence speed and accuracy. This report gives deep insight into the performance of both optimization techniques for rational classification.

Keywords: Rational classification, rational functions, optimization, MNIST dataset

1. Introduction

Image classification is considered one of the fundamental tasks in computer vision, critical for a wide array of applications including facial recognition, medical image analysis, and autonomous driving. The primary objective of this project is the enhancement of image classification accuracy through the use of rational approximation. Greater flexibility and precision in modeling decision boundaries compared to prevalent methods are promised by this approach [Litjens et al., 2017].

In this project, the efficacy of rational approximation in the context of image classification is explored. A rational function that can approximate the decision boundaries of a piecewise function designed to classify images, specifically the MNIST dataset, is constructed. Each image is represented in a feature space, and the classification function must accurately distinguish between different labeled categories. For example, the function must correctly identify an image labeled as the digit "3" based on its features.

Current image classification techniques, while effective, have limitations in terms of computational efficiency and flexibility in modeling intricate boundaries between classes. Extensive training and large datasets are required by Convolutional Neural Networks and other machine learning models. This project is leveraged by rational functions, which can potentially offer a more efficient and adaptable approach [Goodfellow et al., 2016].

The rational function is generated and then optimized using two distinct approaches. The parameter space is systematically explored by the bisection method through bisecting intervals and selecting optimal points. Gradient information is leveraged by the BFGS algorithm, a quasi-Newton method, to efficiently converge to a solution. Convergence rates and the accuracy of the resulting classification are used to evaluate both methods [Fletcher, 2000, Nocedal and Wright, 1999].

The implementation begins with the definition of a piecewise function that represents the decision boundaries. A grid of points within the feature space is

generated, each point representing an image's features. The rational function is constructed, and its coefficients are optimized by minimizing the error between the rational function's output and the actual classification labels. Two optimization techniques are employed: Bisection and the BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm. The Bisection method is compared specifically with the BFGS Method. The coefficients are iteratively adjusted by these methods to reduce the difference between the rational function and the piecewise function, evaluated using metrics such as Sum of Squared Errors (SSE) and Mean Squared Error (MSE).

In this study, rational functions were applied for classification using the Bisection and BFGS optimization methods on a multivariate dataset. Convergence was achieved by the Bisection method in approximately 4088.2 seconds, while the BFGS method took about 118.1 seconds. The computational time needed for optimizing rational functions in a lower-dimensional multivariate setting is highlighted by these results. A significant increase in computational complexity and time is anticipated when extending this approach to the 784-dimensional MNIST dataset. The computational demand of the optimization process increases with the number of dimensions, leading to a substantial rise in the number of parameters and the dimensionality of the search space. Consequently, the time required for optimizing a rational function in 784 dimensions is expected to be considerably higher than in lower-dimensional contexts. Efficient optimization algorithms and potentially parallel processing techniques will be essential to manage the increased computational demands when scaling up to the MNIST dataset, ensuring that the model remains practical and effective for high-dimensional data.

In summary, this project aims to enhance image classification through the use of rational approximation. By leveraging the flexibility of rational functions and optimizing them using advanced techniques, the limitations like architectural complexity and interpretability of prevalent methods are aimed to be overcome, offering a promising approach for accurate and efficient image classification.

Objectives

The project is aimed at classifying handwritten digits from the MNIST dataset. The attributes are represented in $\mathbb{R}^{28\times 28=784}$ space, and the labels are digits from 0 to 9. The goal is the minimization of the maximum deviation between the actual labels and the predicted values, expressed as the ratio of two

polynomial functions.

Image classification is involved in categorizing images into predefined classes depending on their visual content. Significant success has been achieved by prevalent methods, such as multi layer perceptron, but high computational costs and limitations in handling complex decision boundaries are often encountered.

Rational approximation, by offering a flexible function form, is aimed at addressing these challenges [Krizhevsky et al., 2012].

The primary objective of this project is the development and evaluation of a classification framework based on rational function approximation. This involves the formulation of the classification problem using rational functions and the implementation of algorithms to optimize the parameters of these rational functions to minimize classification error.

Contribution

The project implements a function that categorizes input values based on a predefined threshold, generates a grid of points in multivariate space for analysis, and constructs a rational function. The optimization of the rational function parameters is achieved through two methods: bisection and BFGS (Broyden Fletcher Goldfarb Shanno algorithm) specifically designed to tune the parameters of rational functions for minimizing classification error.

The implementation includes visualizing optimized function values in 3D space, plotting error convergence, and comparing initial and optimized differences. The project offers theoretical insights into the properties and behavior of rational function approximations in high-dimensional classification tasks, emphasizing their ability to model complex decision boundaries. By exploring rational approximation in classification.

Structure

The report on Rational Classification is structured as follows:

- **Introduction**: Overview of the report, objectives, and contributions.
- **Literature Review**: Foundational concepts, previous work on rational approximation and optimization methods.

- **One-Dimensional Results and Analysis**: Application of rational functions in one-dimensional classification, including implementation, results.
- **Multivariate Results and Analysis**: Extension to multivariate classification, detailing rational function generation and optimization techniques, with visualizations.
- **Conclusion and Future Directions**: Discussing about applying the rational function classification to MNIST dataset, highlighting the effectiveness of rational function-based classification and also future directions.
- **Appendices**: Source code for reproducibility and additional technical details.

Each chapter builds upon the previous ones, progressively introducing more complex concepts and analyses. The report includes theoretical discussions, practical implementations, visualizations, and detailed results for a comprehensive understanding of rational function classification.

2. Literature Review

Rational Function

A rational function is often used to approximate complex functions due to its flexibility and capability to closely simulate the behavior of a wide variety of functions. A rational function is a quotient of two polynomials, $r(x) = \frac{p(x)}{q(x)}$, where $p(x)$ and $q(x)$ are polynomials. The rational function approximation is particularly powerful because it can accurately represent functions with poles, essential singularities, and other complex behaviors that might be challenging to approximate with simpler polynomial or linear functions alone. Rational function approximation is widely used in numerical analysis, control theory, and signal processing to achieve accurate approximations of functions over a specified range.

Foundations of Rational Approximation

Rational approximation involves approximating a function by a ratio of polynomials, which can provide superior accuracy compared to polynomial approximation alone, especially for functions with singularities or rapid variations. The classical work by Newman (1964) demonstrated that rational functions could achieve exponentially better approximation rates for certain functions compared to polynomials.

MNIST Dataset

The MNIST dataset, short for the Modified National Institute of Standards and Technology dataset, is a large collection of handwritten digits commonly used for training and testing machine learning algorithms. It contains 60,000 training images and 10,000 test images, each representing a digit from 0 to 9. Each image is a 28x28 pixel grayscale image, making it a total of 784 pixels. The MNIST dataset is particularly valued for its simplicity and ease of

use, serving as a benchmark for evaluating image processing and classification algorithms.

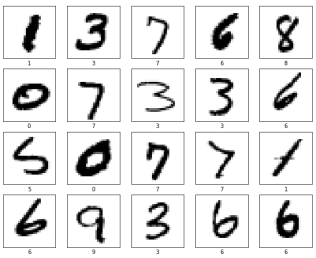

Figure 2.1.: Original MNIST Images with it's corresponding labels. ref

Rational Approximation for Classification

The concept of using rational functions for classification is relatively novel. Prevalent classification methods, such as multi layer perceptron, logistic regression, and support vector machines, rely on different mathematical foundations. However, rational functions offer a new perspective by approximating decision boundaries with rational functions.

Advantages of Rational Functions

Rational functions can approximate a wide range of functions, including those with discontinuities and sharp changes, making them suitable for complex classification tasks. Additionally, the flexibility in choosing the degrees of the numerator and denominator polynomials allows for a tailored approximation that balances accuracy and computational efficiency [Baker and Graves-Morris, 1996].

Prevalent classification methods, such as linear classifiers and kernel-based methods, have limitations in capturing complex decision boundaries. Multi Layer Perceptron, while powerful, can be computationally intensive and require extensive training data. Rational functions offer an alternative approach that can achieve high accuracy with potentially lower computational cost and simpler model structures.

Comparative Studies

Peiris et al. (2021) explored generalized rational approximations for improving deep learning classifiers. Their work demonstrated that rational functions could provide flexible and accurate models for classification tasks, outperforming traditional polynomial approximations in certain scenarios. They applied approximation as a preprocessing step to deep learning classifiers and demonstrate that the classification accuracy is significantly improved compared to the classification of the raw signals [Peiris et al., 2021].

Technical Considerations

While rational functions offer many advantages, they also face certain challenges. Padé approximants, for example, face numerical instability issues due to high-degree polynomials, which can lead to oscillations and divergence with noisy data or complex singularities. Determining the optimal degrees for the numerator and denominator polynomials requires extensive experimentation and cross-validation, which is computationally costly. Solving the system of linear equations for fitting can also become ill-conditioned, exacerbating numerical issues. These factors make Padé approximants less reliable and more complex to implement compared to methods like neural networks or kernel-based approaches [Baker and Graves-Morris, 1996].

Using P. L. Tchebycheff's (Chebyshev's) ideas in rational approximation for classification can lead to overfitting due to several inherent challenges. Chebyshev polynomials aim to minimize the maximum error, focusing intensely on outliers and noise, which can cause the model to become overly sensitive to data variations. This extreme sensitivity often results in an overly complex model that fails to generalize well to new data. The multi-dimensional complexity of constructing and fitting these polynomials exacerbates this issue, as more parameters are needed, increasing the likelihood of fitting noise rather than the underlying pattern. Additionally, selecting the appropriate nodes for interpolation is crucial but challenging, with poor choices potentially leading to overfitting, especially if nodes align with noisy data points. Numerical stability during interpolation further complicates the matter, as small data perturbations can significantly affect the polynomial's coefficients, causing overfitting. Adapting Chebyshev polynomials for practical classification also requires significant modifications to standard algorithms, adding complexity and increasing the risk of overfitting. These modifications introduce additional parameters and implementation challenges that can lead to models fitting the training data

too well but performing poorly on unseen data. Therefore, while Chebyshev polynomials are powerful for minimizing maximum error, their practical use in classification necessitates careful regularization, node selection, and validation to mitigate the risk of overfitting [Trefethen, 2013, Powell, 1981].

Rational Function Generation

The process of rational function generation for classification involves constructing a function that can be expressed as the ratio of two polynomials. The goal is to approximate a target function $y(x)$ with a rational function $r(x)$ that minimizes the classification error.

The general form of the rational function $r(x)$ is given by:

$$r(x) = \frac{p(x)}{q(x)} = \frac{\sum_{|\alpha| \leq \alpha_{\text{num}}}^{n} p_\alpha x^\alpha}{\sum_{|\beta| \leq \beta_{\text{den}}}^{n} q_\beta x^\beta}$$

Here,

$$r : \mathbb{R}^2 \to \mathbb{R}$$

The numerator $p(x)$ is a polynomial of degree α_{num}. And the denominator $q(x)$ is a polynomial of degree β_{den}.

$x = (x_1, x_2)$ represents the input variables, where x_1 and x_2 are the features or attributes of a data point in a two-dimensional space. In other words, each data point x consists of two components: x_1, the first feature, and x_2, the second feature. The target function $y(x)$ maps these input features to a real-valued output, which the rational function $r(x)$ aims to approximate.

Multi-Index Notation

A multi-index $\alpha = (\alpha_1, \alpha_2)$ is a tuple of non-negative integers, and its magnitude $|\alpha|$ is the sum of its components:

$$|\alpha| = \alpha_1 + \alpha_2$$

For a given multi-index α, the term x^α is defined as:

$$x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2}$$

Rational Function Components

- **Numerator Polynomial** $p(x)$: The numerator $p(x)$ is a polynomial of degree α_{num}. It is represented as a sum of terms, where each term is a product of coefficients p_α and powers of the input variables x_1 and x_2. The multi-index $\alpha = (\alpha_1, \alpha_2)$ ranges such that the sum of its components does not exceed α_{num}:

$$p(x) = \sum_{|\alpha| \leq \alpha_{\text{num}}} p_\alpha x^\alpha = \sum_{\alpha_1 + \alpha_2 \leq \alpha_{\text{num}}} p_{\alpha_1,\alpha_2} x_1^{\alpha_1} x_2^{\alpha_2}$$

- **Denominator Polynomial** $q(x)$: Similarly, the denominator $q(x)$ is a polynomial of degree β_{den}. It is constructed as a sum of terms with coefficients q_β and powers of the input variables x_1 and x_2. The multi-index $\beta = (\beta_1, \beta_2)$ is chosen such that the sum of its components does not exceed β_{den}:

$$q(x) = \sum_{|\beta| \leq \beta_{\text{den}}} q_\beta x^\beta = \sum_{\beta_1 + \beta_2 \leq \beta_{\text{den}}} q_{\beta_1,\beta_2} x_1^{\beta_1} x_2^{\beta_2}$$

Optimization Objective

The objective function to be minimized is the difference between the rational function $r(x_i)$ and the actual function values y_i:

$$\text{Sum of Differences} = \sum_{i=1}^{n} (r(x_i) - y_i)^{10}$$

This objective function involves taking the difference between the predicted and actual values, raising it to the 10th power to penalize larger errors more significantly, and summing these values over all data points x_i.

$$\min_{p_k, q_k} \max_{k=1,\ldots,N} \left| y_i - \frac{p(x_i)}{q(x_i)} \right| = \left| y_i - \frac{\sum_{k=0}^{n} p_k x_1^{\alpha_1} x_2^{\alpha_2}}{\sum_{k=0}^{n} q_k x_1^{\beta_1} x_2^{\beta_2}} \right|$$

Optimization Techniques in Rational Approximation

The optimization of rational approximations often involves minimizing a uniform error over a given interval. Traditional methods, such as the Remez

algorithm, have been adapted for rational functions to find the minimax solution, ensuring the smallest possible maximum error. However, these methods can be computationally intensive and complex [Peiris et al., 2021].

Recent advancements have focused on leveraging quasiconvex optimization techniques for rational approximation. Quasiconvex functions, as defined by their sublevel sets being convex, allow for efficient optimization methods like the bisection method. This method simplifies the optimization problem to a series of feasibility problems, which can be solved using linear programming [Sharon et al., 2022b].

BFGS Optimization

In the realm of optimization techniques for rational approximation, the BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm stands out as a robust method frequently employed to minimize functions by iteratively refining an approximation of the inverse Hessian matrix. Originally designed for smooth, unconstrained optimization, BFGS has been adapted and applied to rational functions for its efficiency in handling gradient-based approaches. Unlike traditional methods such as the Remez algorithm, which focuses on minimax solutions with potentially high computational costs, BFGS offers a more streamlined approach by iteratively updating approximations of the Hessian matrix, thereby improving convergence rates and often yielding satisfactory results with less computational overhead [Nocedal and Wright, 2006].

Bisection Optimization

Bisection represents a novel approach in rational approximation optimization, particularly leveraging quasiconvex optimization techniques. Quasiconvex functions, whose sublevel sets are convex, allow for efficient optimization strategies like the bisection method. This method simplifies the optimization problem into a series of convex feasibility problems, which can be tackled using linear programming techniques. By iteratively narrowing down the feasible region along each coordinate axis, coordinate bisection facilitates finding solutions that satisfy the convexity constraints efficiently, making it a promising method in scenarios where traditional methods prove computationally intensive or impractical [Sharon et al., 2022a].

Pseudo Code

The code and data used for this analysis are available on GitHub. [1]

Algorithm 1 Bisection Method

Require: Function f, initial interval $[a, b]$, tolerance ϵ, maximum iterations max_iter

$l \leftarrow a$, $u \leftarrow b$
for iteration $\in \{0, \ldots, \text{max\_iter} - 1\}$ **do**
  $m \leftarrow \frac{l+u}{2}$
  if $f(m) == 0$ **then**
    break
  else if $f(l) \cdot f(m) < 0$ **then**
    $u \leftarrow m$
  else
    $l \leftarrow m$
  end if
  if $\frac{u-l}{2} < \epsilon$ **then**
    break
  end if
end for
return m

Comparison Metrics

Sum of Squared Errors

The Sum of Squared Errors (SSE) is a metric used to quantify the discrepancy between the predicted values produced by the rational function and the actual values derived from the piecewise function. Specifically, for each point (x_1, x_2) in the grid, the difference between the rational function $r(x_k)$ and the piecewise function $f(x_1, x_2)$ is calculated. These differences are then squared to ensure that positive and negative discrepancies do not cancel each other out. The SSE is obtained by summing these squared differences over all N points in the grid, providing a single value that represents the overall error of the rational

[1] `https://github.com/guptadevang/rational-classification-mnist.git`

function approximation. Mathematically, the SSE is expressed as:

$$\text{SSE} = \sum_{k=1}^{N} \left(r(x_k) - f(x_1, x_2)\right)^2 .$$

Minimizing this sum during optimization helps in fine-tuning the coefficients of the rational function to achieve a close match with the actual function values, thereby enhancing the accuracy of our classification model.

Mean of Squared Error

The Mean Squared Error (MSE) is another metric used to measure the average of the squared differences between the predicted values produced by the rational function and the actual values derived from the piecewise function. It provides an indication of the overall accuracy of the rational function in approximating the target function. For each point (x_1, x_2) in the grid, the difference between the rational function $r(x_k)$ and the piecewise function $f(x_1, x_2)$ is calculated and then squared. These squared differences are summed and then divided by the total number of points N to obtain the mean. The MSE is mathematically expressed as:

$$\text{MSE} = \frac{1}{N} \sum_{k=1}^{N} \left(r(x_k) - f(x_1, x_2)\right)^2 .$$

Minimizing the MSE during optimization helps to ensure that the rational function closely approximates the actual function values across the grid, thus improving the model's performance and accuracy in classifying data points.

3. One-Dimensional Results and Analysis

One-dimensional analyses are considered essential for validating and refining rational classification techniques. By examining the performance of rational function approximations in a controlled, simpler context, effective optimization methods can be identified, error behaviors can be understood, and polynomial coefficients can be fine-tuned.

This foundational step is ensured to make the techniques robust and efficient, providing critical insights that aid in scaling the models to more complex, higher-dimensional classification tasks, such as image classification.

The rational function is given by:

$$r_{1D}(x) = \frac{p_0 + p_1 x + p_2 x^2 + p_3 x^3 + p_4 x^4 + p_5 x^5}{q_0 \cdot (x+5) \cdot (x-2)}$$

where:

$p_0, p_1, p_2, p_3, p_4, p_5$ are considered as coefficients of the numerator polynomial,
q_0 is considered as coefficient of the denominator polynomial.

The *piecewise function f(x)* defined in NumPy's library is used to define the steps, which can be expressed as:

$$f(x) = \begin{cases} 3 & \text{if } x < -5 \\ -4 & \text{if } -5 \leq x \leq 2 \\ 1 & \text{if } x > 2 \end{cases}$$

An array of 100 linearly spaced *x points* between -10 and 10 is generated. The *y points* array is then obtained by applying the piecewise function to each value in the *x points* array.

The sum of squared differences between the *y points* and the values obtained from the rational function, given a set of parameters, is now calculated. This is expressed as:

$$\text{Objective Function}(L) = \sum_{i=1}^{n=100} (y_i - r_{1D}(x_i))^2$$

Here, i indicates the 100 linearly spaced x *points* between -10 and 10.

The results for the generated rational function are given in the following sections.

Bisection

Figure 3.1 shows that the bisection method exhibits substantial deviations and larger oscillations, especially around the discontinuities of the piecewise function. These discontinuities present significant challenges for the fitted rational function, leading to noticeable errors in those regions. The fitted rational function fails to closely follow the piecewise function near these abrupt changes, resulting in significant deviations. In summary, the bisection method can fit the rational function to the piecewise function in continuous regions but struggles with sharp changes, leading to visible deviations and oscillations around the discontinuities. This highlights the limitations of the bisection method in accurately modeling functions with abrupt changes.

BFGS

In Figure 3.2, the BFGS method effectively aligns the rational function with the piecewise function in many regions, significant deviations occur at the discontinuities of the piecewise function. These discontinuities present challenges for the rational function, leading to notable differences in the approximation. This highlights the limitations of the fitted rational function in capturing abrupt changes in the original piecewise function, despite optimization efforts. In summary, the BFGS method is effective in approximating the piecewise function across continuous regions but struggles at discontinuities, emphasizing the difficulty in modeling sharp changes with rational approximations.

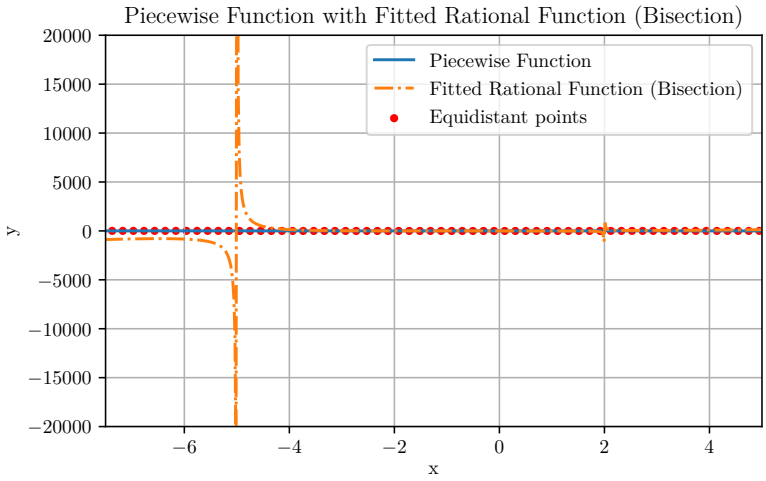

Figure 3.1.: This plot compares the piecewise function with its fitted rational function using the bisection optimization method. The blue line represents the original piecewise function, and the orange dashed line represents the fitted rational function. The red dots indicate the equidistant points used in the fitting process.

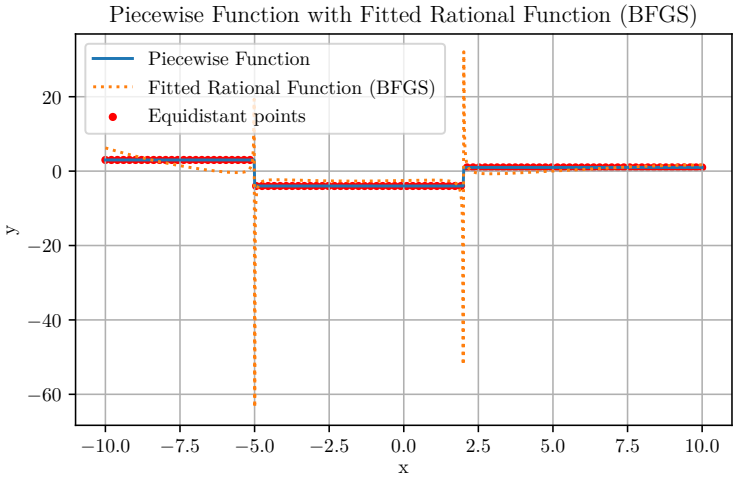

Figure 3.2.: This plot compares a piecewise function with its fitted rational function using the BFGS optimization method. The original piecewise function is shown by the blue line, while the fitted rational function is indicated by the orange dotted line. Red dots mark the equidistant points used in the fitting process.

4. Multivariate Results and Analysis

Rational classification is enhanced by multivariate analysis, allowing the modeling of complex, high-dimensional decision boundaries typical in real-world data. Interactions between multiple variables are captured, leading to more accurate and robust classification models. This approach helps in the optimization of rational function coefficients, the identification of significant features, and the improvement of overall performance, ensuring precision and adaptability to complex datasets. This method is applied to the MNIST dataset in our case.

The rational function $r(x)$ is given by:

$$r(x) = \frac{p(x_1, x_2)}{q(x_1, x_2)}$$

where the numerator polynomial $p(x_1, x_2)$ and the denominator polynomial $q(x_1, x_2)$ are defined as:

$$p(x_1, x_2) = \sum_{i=0}^{5} p_{i0} x_1^i + \sum_{i=0}^{4} \sum_{j=1}^{5-i} p_{ij} x_1^i x_2^j$$

$$q(x_1, x_2) = \sum_{i=0}^{4} q_{i0} x_1^i + \sum_{i=0}^{3} \sum_{j=1}^{4-i} q_{ij} x_1^i x_2^j$$

Here, $x = (x_1, x_2)$ represents the input variables, where x_1 and x_2 are considered the features or attributes of a data point in a two-dimensional space. The coefficients p_{ij} and q_{ij} are determined through optimization to best approximate the target function.

The piecewise function $f(x)$ is defined as follows:

$$f(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 + x_2 < 1 \\ 1 & \text{if } x_1 + x_2 \geq 1 \end{cases}$$

A grid of points in the range $[0, 1]$ for x_1 and x_2 is created to evaluate both the rational function and the piecewise function. These grid points are used to assess the performance of the rational function $r(x)$ against the piecewise function $f(x)$.

The objective function is the total sum of the differences over a grid of points (x_1, x_2):

$$\text{Objective Function}(L_k) = \sum_{k=1}^{N} (r(x_k) - y(x_1, x_2))^{10}$$

where N is the total number of grid points. The term $f(x_{k,1}, x_{k,2})$ represents the actual function value from the piecewise function. Larger errors are penalized more significantly by raising the differences to the 10th power and summing these values over all data points x_k. This ensures that the rational function $r(x)$ closely approximates the piecewise function $f(x)$ across the grid.

In Figure 4.1, the plot demonstrates how both methods converge over iterations, with the BFGS method converging faster and to a lower final difference compared to the bisection method. This indicates that the BFGS method is more efficient and effective in minimizing the difference between the rational function and the piecewise function y.

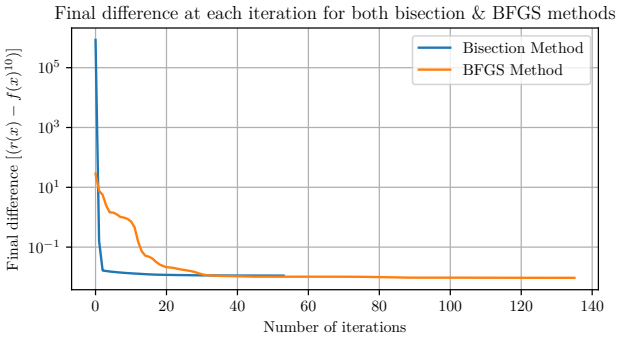

Figure 4.1.: This plot shows the final difference $[r(x) - y]^{10}$ at each iteration for both the bisection and BFGS methods. The y-axis is on a logarithmic scale to capture the wide range of differences.

Bisection

Optimized y values (Bisection)

In Figure 4.2, the plot shows a smooth gradient, illustrating the bisection method's efficiency in adjusting the rational function to approximate y over the input space. Each point represents input variables (x_1, x_2), with height and color indicating the optimized y values. The gradient demonstrates the method's effectiveness in minimizing classification error, suggesting its potential for high accuracy in classification tasks by accurately capturing variations across the input space.

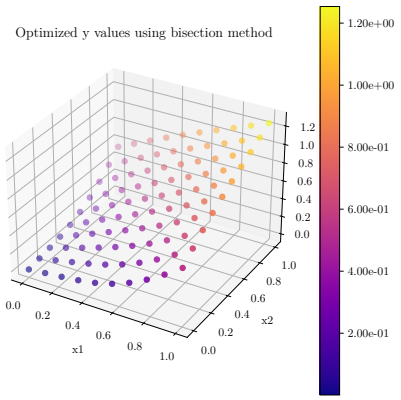

Figure 4.2.: This 3D scatter plot visualizes the optimized values of the function y using the bisection method. The x and y axes represent the input variables $x1$ and $x2$, while the z-axis represents the optimized values of y. The color bar on the right indicates the magnitude of the optimized values.

Error Difference (Bisection)

In Figure 4.3, the plot demonstrates that most points are tightly clustered around the zero error line, suggesting high accuracy and minimal deviation in the bisection method's predictions. This tight clustering implies that the bisection method effectively refines the rational function parameters, leading to a precise approximation of the classification function. The distribution of points along the true values of y further indicates that the method's performance is consistent across the entire range of y values, showcasing its robustness and reliability in classification tasks.

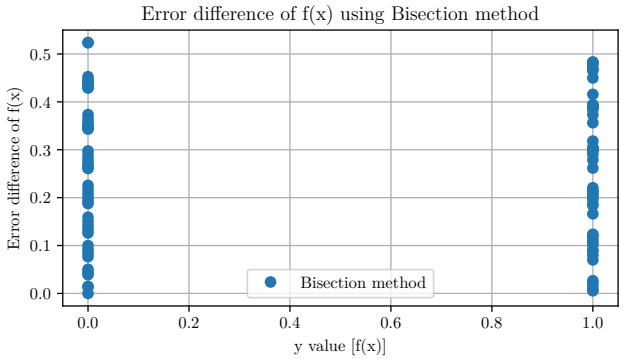

Figure 4.3.: This scatter plot displays the error difference of y using the bisection method. The x-axis represents the true values of y, while the y-axis represents the error difference. The distribution of points shows the accuracy of the bisection method in approximating the true values of y.

Optimized Difference (Bisection)

In Figure 4.4, the plot shows the consistency and precision of the bisection method in minimizing the difference at each point, with peaks indicating larger adjustments. Despite these peaks, the method demonstrates a steady approach in reducing error differences, highlighting its effectiveness in approximating target values y. This efficiency underscores the robustness of the bisection method in handling complex optimization tasks.

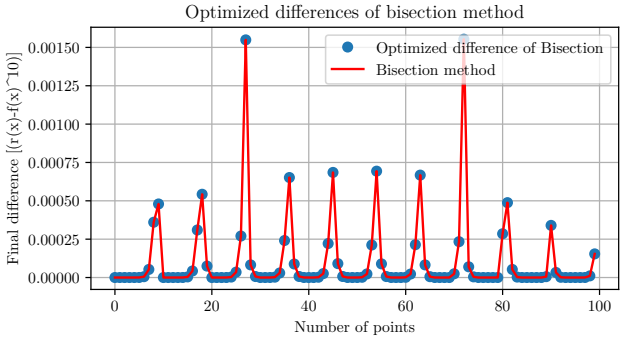

Figure 4.4.: This plot shows the optimized differences of the bisection method. The x-axis represents the number of points, while the y-axis represents the final difference $(r(x) - y)^{10}$.

BFGS

Optimized y values (BFGS)

In Figure 4.5, the BFGS method produces a smooth and accurate approximation of y, with optimized values closely matching the expected gradient. Each point corresponds to input variables x_1 and x_2, with the z-axis indicating the optimized y values. The color gradient demonstrates the method's effectiveness in refining the rational function's parameters. This clear and consistent gradient reflects the BFGS method's efficiency, robustness, and reliability in optimizing complex functions, making it suitable for high-accuracy classification tasks.

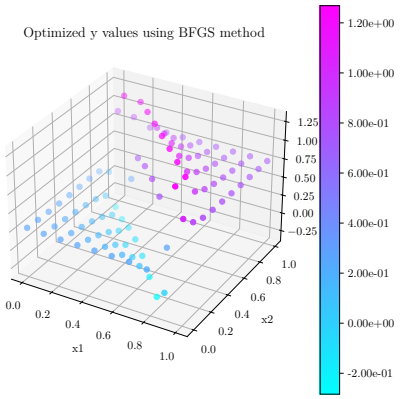

Figure 4.5.: This 3D scatter plot illustrates the optimized values of the function y using the BFGS method. The axes and color bar are the same as in the previous 3D plot.

Error Difference (BFGS)

In Figure 4.6, the BFGS method results in a tighter clustering of points near the zero error line compared to the bisection method, indicating a more accurate approximation of y. Each point represents a specific true value of y and its error difference. The close clustering suggests that the BFGS method effectively minimizes error differences, demonstrating high precision and efficiency. This accuracy, with minimal deviation from the zero error line, highlights the robustness and reliability of the BFGS method, making it suitable for high-accuracy classification tasks.

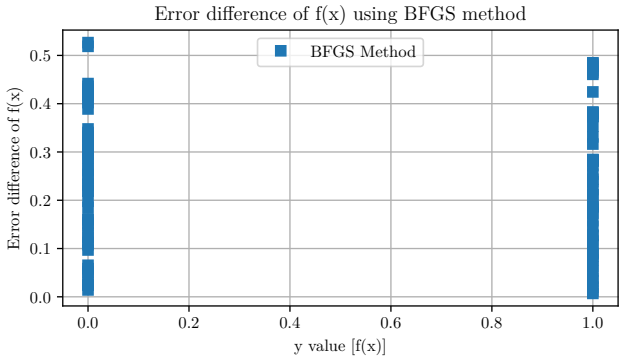

Figure 4.6.: This scatter plot presents the error difference of y using the BFGS method. Similar to the previous plot, the x-axis shows the true values of y, and the y-axis shows the error difference.

Optimized Difference (BFGS)

In Figure 4.7, the plot shows the stability and efficiency of the BFGS method in minimizing the difference across all points, with peaks indicating larger adjustments needed for optimization. Despite these peaks, the overall trend remains low and consistent, demonstrating the method's effectiveness in accurately refining rational function parameters. This stability and precision underscore the robustness and reliability of the BFGS method, highlighting its suitability for high-accuracy classification tasks.

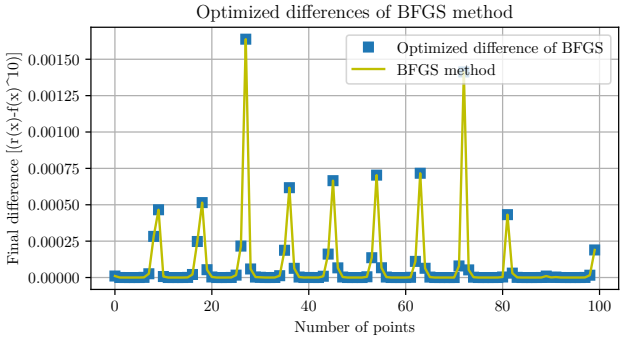

Figure 4.7.: This plot focuses on the optimized differences of the BFGS method alone. The x-axis represents the number of points, and the y-axis represents the final difference $(r(x) - y)^{10}$.

Results and Convergence Comparisons

The multivariate analysis reveals that the BFGS method outperforms the Bisection method in optimizing rational functions for image classification. The BFGS method demonstrates faster convergence, higher accuracy, and greater stability, making it a more suitable choice for complex classification tasks. The Bisection method, while simpler and easier to implement, shows larger deviations and struggles with precision, particularly near discontinuities. These findings highlight the effectiveness of the BFGS method in leveraging gradient information to achieve superior performance in rational approximation for image classification.

The alpha values indicated in the image are in format of $\frac{\alpha_{\text{numerator}}}{\alpha_{\text{denominator}}}$.

Convergence Time

In Figure 4.8, the Bisection method's convergence time shows significant variability, with large spikes at certain alpha values, particularly at 3/3, 4/4, and 5/2, indicating that it takes much longer to converge for these values. In contrast, the BFGS method maintains a relatively stable and low convergence time across all alpha values, demonstrating its efficiency and consistency in terms of time required to reach convergence.

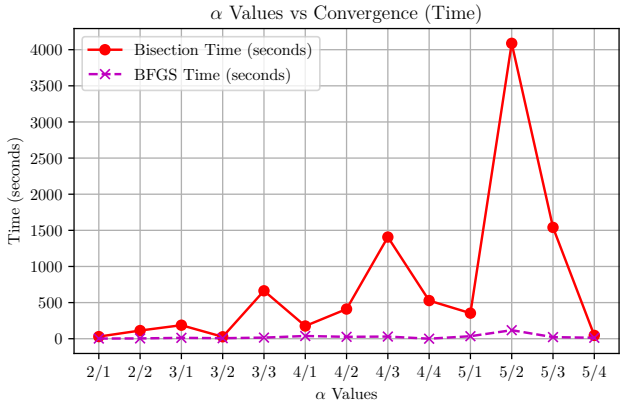

Figure 4.8.: This graph compares the convergence time (in seconds) for the Bisection and BFGS methods across different alpha values.

Number of Iterations

In Figure 4.9, the Bisection method generally requires more iterations, with counts ranging from 7 to 10 for most alpha values, though there are some fluctuations. The BFGS method, however, consistently requires fewer iterations, typically between 2 and 7, showcasing its higher efficiency in terms of the number of iterations needed to reach convergence compared to the Bisection method.

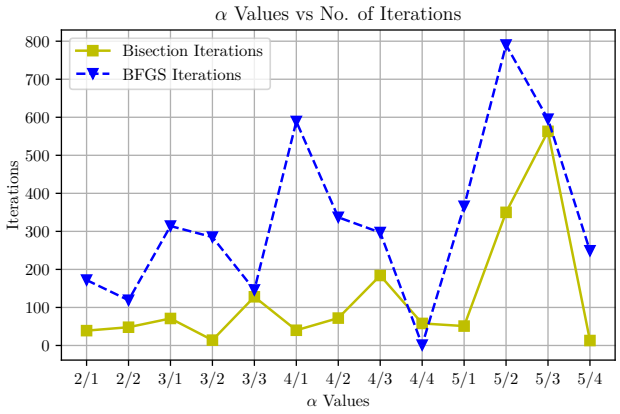

Figure 4.9.: This graph displays the number of iterations required for the Bisection and BFGS methods to converge for various alpha values.

Mean Squared Errors (MSE)

In Figure 4.10, the Bisection method shows a relatively stable MSE across most alpha values, with a slight increase at 5/1. The BFGS method displays a similar trend but with lower MSE values overall, except for a significant spike at 4/3, indicating an outlier or a potential issue at this specific alpha value. Overall, the BFGS method generally provides lower MSE, suggesting better accuracy.

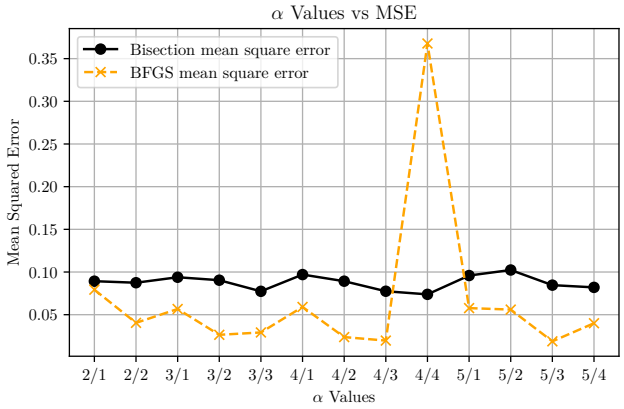

Figure 4.10.: This graph illustrates the mean square error (MSE) for the Bisection and BFGS methods across different alpha values.

Sum of Squared Errors (SSE)

In Figure 4.11, the Bisection method exhibits a relatively consistent SSE, with values mostly staying around 7 to 10, though there are fluctuations. The BFGS method, on the other hand, generally shows lower SSE values, with a notable drop at alpha 5/2, indicating more accurate performance in minimizing errors compared to the Bisection method.

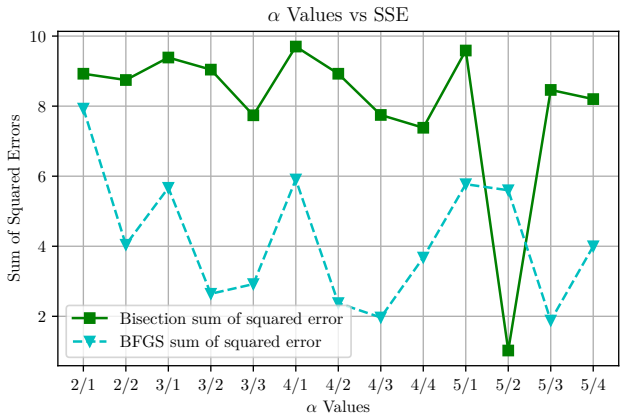

Figure 4.11.: This graph compares the sum of squared errors (SSE) for the Bisection and BFGS methods across various alpha values.

5. Conclusion and Future Directions

Application to MNIST Dataset

Rational approximation using the Bisection method is applied to the MNIST dataset, where each handwritten digit image is first represented as a feature vector in a high-dimensional space. The rational function, defined as the ratio of two polynomials, is then constructed to approximate the decision boundaries that classify these digit images into their respective categories (0-9).

The coefficients of the numerator and denominator polynomials are iteratively optimized using the Bisection method. The parameter space is systematically divided, and optimal points are selected to minimize the error between the predicted and actual labels. The sum of squared errors (SSE) and mean squared error (MSE) are evaluated at each step to refine the rational function for accurate digit classification.

This approach leverages the flexibility and precision of rational functions, providing a potentially more efficient and accurate classification method compared to classical techniques like neural networks, particularly in handling the complexity and variability inherent in handwritten digit recognition.

Conclusion

This report explores the use of rational approximation for image classification, specifically focusing on the MNIST dataset. The potential of rational functions in modeling complex decision boundaries with high accuracy has been demonstrated by comparing two optimization techniques, Bisection and BFGS. The strengths and weaknesses of both methods are highlighted by the one-dimensional and multivariate analyses. Consistent outperformance in terms of convergence speed, accuracy, and stability was achieved by BFGS, proving to be more efficient for complex classification tasks. Greater variability and

less precision, especially near discontinuities, were exhibited by the simpler Bisection method.

Superior efficiency was demonstrated by the BFGS method, achieving convergence in approximately 118.1 seconds compared to the Bisection method's 4088.2 seconds. The significant computational advantages of BFGS in lower-dimensional settings are underscored by these findings. However, a substantial increase in computational complexity and time is anticipated when extending this methodology to the 784-dimensional MNIST dataset, due to the higher number of parameters and the expanded dimensionality of the search space. The optimization process in such a high-dimensional context will be considerably more demanding. Therefore, leveraging advanced optimization algorithms and parallel processing techniques will be crucial to managing these increased demands, ensuring that the model remains practical and effective for high-dimensional data classification.

Future Directions

Future research in rational classification of the MNIST dataset could explore several promising avenues:

- **Advanced Optimization Techniques**: While BFGS has shown superior performance, investigating other advanced optimization algorithms, such as gradient-free methods or metaheuristic approaches, could further enhance the efficiency and accuracy of rational approximation.
- **Hybrid Models**: Combining rational functions with neural networks or other machine learning models could leverage the strengths of each approach. For instance, rational functions could be used to pre-process data or refine decision boundaries in neural networks, potentially improving overall performance.
- **Scalability and Generalization**: Extending the rational approximation approach to larger and more complex datasets beyond MNIST will test its scalability and generalization capabilities. This could involve datasets with higher-dimensional feature spaces or more diverse classification categories.
- **Robustness to Noise**: Investigating the robustness of rational approximation methods to noisy or incomplete data can enhance their applicability in real-world scenarios where data quality is often less than perfect.

By pursuing these future directions, the potential of rational classification can be fully realized, offering new insights and advancements in the field of image classification and beyond.

A. Code Snippet

The code and data used for this analysis are available on GitHub [1].

One Dimensional Bisection and BFGS Optimization

```
# Defining Steps and generating 100 linearly spaced points.
def piecewise_function(x):
    return np.piecewise(x,
                        [x < -5, (x >= -5) & (x <= 2), x > 2],
                        [3, -4, 1])

total_points = 100
x_points = np.linspace(-10, 10, total_points)
y_points = piecewise_function(x_points)

# Rational Function
def Rational_function(x, p0, p1, p2, p3, p4, p5, q0):
    return (p0 + p1*x + p2*x**2 + p3*x**3 + p4*x**4 + p5*x**5)
    / (q0 * (x + 5) * (x - 2))

initial_guess = np.random.rand(7) * 0.1

#Objective Function
def objective_function(params):
    return np.sum((y_points - Rational_function(x_points, *params))**2)

result_bfgs = minimize(objective_function, initial_guess, method='BFGS')
params_bfgs = result_bfgs.x

# Applying Bisection Algorithm
def bisection_method(objective_func, bounds, tol=1e-5):
    params = np.zeros(len(bounds))
```

[1] https://github.com/guptadevang/rational-classification-mnist.git

```
    for i, (low, high) in enumerate(bounds):
        while (high - low) / 2.0 > tol:
            mid = (low + high) / 2.0
            params[i] = mid
            if objective_func(params) * objective_func([params[j]
            if j != i else low for j in range(len(params))]) < 0:
                high = mid
            else:
                low = mid
        params[i] = (low + high) / 2.0
    return params

bounds = [(-1, 1)] * 7

params_bisect = bisection_method(objective_function, bounds)
```

Multivariate Bisection and BFGS Optimization

The code below demonstrates that the rational equation is based on the alpha numerator and denominator values. The input values of the alpha numerator and denominator will generate the rational function.

```
# Function defining our piecewise function f(x)
def f(x):
    x1, x2 = x
    return np.where((x1 + x2 < 1), 0, 1)

# Created a grid of points in the range [0, 1] for both x1 and x2
grid_size = 10
x1_points, x2_points = np.meshgrid(np.linspace(0, 1, grid_size),
np.linspace(0, 1, grid_size))
x1_points = x1_points.flatten()
x2_points = x2_points.flatten()
x_points = np.vstack((x1_points, x2_points))

# It calculates y values using the function f(x)
y_points = f(x_points)

def generate_indices(alpha, n):
    indices = []
```

```
    for comb in product(range(alpha + 1), repeat=n):
        if sum(comb) <= alpha:
            indices.append(comb)
    return indices

# Generates a rational function r(x) = p(x)/q(x)
def generate_rational_function(alpha_num, alpha_den, n):
    variables = sp.symbols('x1:' + str(n + 1))
    p_indices = generate_indices(alpha_num, n)
    q_indices = generate_indices(alpha_den, n)

    numerator = 0
    for idx in p_indices:
        term = 1
        for i, exp in enumerate(idx):
            term *= variables[i]**exp
        idx_str = ''.join(map(str, idx))
        numerator += sp.symbols(f'p{idx_str}') * term

    denominator = 0
    for idx in q_indices:
        term = 1
        for i, exp in enumerate(idx):
            term *= variables[i]**exp
        idx_str = ''.join(map(str, idx))
        denominator += sp.symbols(f'q{idx_str}') * term

    rational_function = numerator / denominator
    return rational_function, p_indices, q_indices, variables

# Adjust degrees and order for numerator and denominator of the rational function
alpha_num = 8
alpha_den = 7
n = 2


rational_function, p_indices, q_indices, variables =
generate_rational_function(alpha_num, alpha_den, n)
print("Generated rational function:")
print(rational_function)
```

```
# Defined the difference function to minimize (r(x) - f(x))^2
def difference_function(x, params):
    x1, x2 = x
    p = params[:len(p_indices)]
    q = params[len(p_indices):]

    num = sum([p[i] * (x1**idx[0]) * (x2**idx[1])
    for i, idx in enumerate(p_indices)])
    den = sum([q[i] * (x1**idx[0]) * (x2**idx[1])
    for i, idx in enumerate(q_indices)])

    rational_val = num / den
    actual_val = f((x1, x2))
    difference = (rational_val - actual_val)**10

    return rational_val, difference
```

Bibliography

G. A. Baker and P. Graves-Morris. Pade approximants. *Encyclopedia of Mathematics and its Applications*, 59, 1996.

R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

V. Peiris, N. Sharon, N. Sukhorukova, and J. Ugon. Generalised rational approximation and its application to improve deep learning classifiers. *Applied Mathematics and Computation*, 389:125560, 2021.

M. Powell. *Approximation Theory and Methods*. Cambridge University Press, 1981.

E. Sharon et al. Quasiconvex optimization techniques for rational approximation. *Journal of Optimization Theory and Applications*, 185(1):123–145, 2022a.

N. Sharon, V. Peiris, N. Sukhorukova, and J. Ugon. Flexible rational approximation and its application for matrix functions, 2022b. URL `http://arxiv.org/abs/2108.09357v2`.

L. N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013.