Modernized View of Learning Rates in Optimization Methods
Final Report

Theodore Gerasimov, Sathwik Yarlagadda
Rice University
COMP 514, 2024
fg50@rice.edu, sy81@rice.edu

Contents

1 Introduction

Learning rate schedules play a crucial role in the current era of large-scale machine learning models. The learning rate controls the step size at which the model parameters are updated during training using gradient descent. Traditionally, constant step-size learning rates or monotonically decreasing learning rate schedules like step decay or exponential decay have been used. However, recent research has explored novel learning rate techniques that can accelerate convergence.

In this report, we study three recent papers that propose innovative learning rate schedules:

- **"Acceleration via Fractal Learning Rate Schedules" [1]** derives what the authors call the Chebyshev fractal learning rate schedule, which is inspired by minimax polynomial optimization from numerical analysis.

- **"Super-Acceleration with Cyclical Step-sizes" Goujaud et al. [2]** shows that cyclical learning rate schedules offer a simple way to exploit spectral gaps in the Hessian matrix of the objective function. The authors develop a convergence analysis for quadratic objectives and propose a systematic approach to design optimal first-order methods.
- **"Provably Faster Gradient Descent via Long Steps" Grimmer [3]** establishes new convergence guarantees for gradient descent in smooth convex optimization by using a computer-assisted analysis technique. The authors show that allowing frequent long steps that potentially violate descent in the short term can lead to provably faster convergence in the long term.

We provide an overview of the key ideas and algorithms presented in the latter two papers. We then take a deeper dive into the Chebyshev learning rate schedule proposed in the first paper. Lastly, we provide an experiment, demonstrating that the Chebyshev fractal schedule can indeed accelerate convergence compared to traditional learning schedules and beyond quadratic objectives. In the appendix, we provide the plots produced in our experiment.

Our report aims to highlight these modernized views on learning rate schedules and their potential to push the boundaries of optimization for machine learning. Advanced learning rate schedules could help train complex models faster and more effectively.

2 Overview of Learning Rate Schedules

2.1 Traditional Learning Rate Schedules

In our course, we have studied and analyzed convergence of the following algorithms:

1. Constant step-size gradient descent
2. Heavy Ball gradient descent
3. Decaying step-size gradient descent

Each of these algorithms has a very clear idea behind it, and under certain assumptions such as Lipschitz-continuous gradients or strong convexity, these algorithms happen to have good convergence guarantees.

However, in real-life optimization problems, the actual objective functions rarely satisfy these "good" assumptions needed to guarantee fast convergence. Therefore, there is a lot of room for experimentation for any given model and there is still no consensus on the "best" learning rate schedule.

2.2 Modern Learning Rate Schedules

2.2.1 Gradient Descent with Long Steps

In the smooth, convex optimization setting, the paper [3] considers nonconstant step-size learning policies that periodically take large steps. These steps may violate the monotone decrease in objective value typically required for thorough convergence analysis. However, contrary to the common intuition, these periodic long steps provably speed up convergence in the long term, with increasingly large gains as longer and longer steps are periodically included. The paper draws a link with accelerated momentum methods, which also depart from ensuring a monotone objective decrease at every iteration.

Given an L-smooth, convex function f and a sequence of step sizes $h = (h_0, h_1, \dots h_{t-1})$ the paper considers the following iteration:

$$x_{k+1} = x_k - \frac{h_{(K \bmod t)}}{L} \nabla f(x_k) \tag{1}$$

The authors do worst-case analysis of convergence of the algorithm. Denote

$$p_{L,D}(\delta) := \begin{cases} \max_{x_0, x_\star, f} & f(x_t) - f(x_\star) \\ \text{s.t.} & f \text{ is convex, } L\text{-smooth} \\ & \|x_0 - x_\star\|_2 \leq D \\ & f(x_0) - f(x_\star) \leq \delta \\ & \nabla f(x_\star) = 0 \\ & x_{k+1} = x_k - \frac{h_k}{L} \nabla f(x_k) \quad \forall k = 0, \ldots, t-1. \end{cases}$$

and call the step-size pattern h *straightforward* if for some $\delta \in (0, \frac{1}{2}]$ we have

$$p_{L,D}(\delta) \leq \delta - \frac{\sum_{i=0}^{t-1} h_i}{LD^2} \delta^2 \quad \forall \delta \in \left[0, LD^2 \Delta\right]$$

for any $L, D > 0$.

Now, if we consider traditional constant step-size gradient descent, then we have the following convergence result:

Lemma 2.1. *Denote* $D = \sup \left\{ \|x - x_0\| \;\middle|\; f(x) \leq f(x_0) \right\}$ *and take* $h = (1)$. *Then for the algorithm 1 with pattern* h *we have the following rate of convergence:*

$$f(x_T) - f(x_*) \leq \frac{LD^2}{4T + 2}$$

If we take our constant step size between 1 and 2, we can further improve convergence rate by a factor of 2.

In the spirit of lemma 2.1, the authors set out to obtain convergence guarantees of the form

$$f(x_T) - f(x_*) \leq \frac{LD^2}{c_h \cdot T} + O\left(\frac{1}{T^2}\right),$$

i.e. their goal is to find learning rate patterns h that give the best constant c_h.

One of the main results of the paper [3] is the following:

Lemma 2.2. *Denote* $D = \sup \left\{ \|x - x_0\| \;\middle|\; f(x) \leq f(x_0) \right\}$ *and let* h *be a straightforward step-size pattern. Then for the algorithm 1 with pattern* h*, we have the following rate of convergence:*

$$f(x_T) - f(x_*) \leq \frac{LD^2}{\text{avg}(h)T},$$

where $\text{avg}(h) = \frac{1}{T} \sum_{i=0}^{T-1} h_i$.

The authors experimentally produce several stepsize patterns h and show convergence guarantees. For example, one of the first patterns the authors consider is $h = (1.5, 2.9)$. The reasoning behind choosing this pattern is the following. We know that constant gradient step-size between 1 and 2 has good convergence guarantees and a larger stepsize 2.9 is chosen experimentally. Similar stepsize patterns with longer (relative to the average step size) are introduced, and the convergence of gradient descent with these patterns can be seen in the following table:

Table 1: Stepsize patterns and their convergence

Pattern Length	**"Straightforward" Stepsize Pattern h (longest stepsize marked in bold)**	**Convergence Rate $\left(O(1/T^2)\right)$ omitted)**
$t = 2$	$(\mathbf{3} - \eta, 1.5)$ for any $\eta \in (0, 3)$	$\frac{LD^2}{(2.25 - \eta/2) \times T}$
$t = 3$	$(1.5, \mathbf{4.9}, 1.5)$	$\frac{LD^2}{2.63333\ldots \times T}$
$t = 7$	$(1.5, 2.2, 1.5, \mathbf{12.0}, 1.5, 2.2, 1.5)$	$\frac{LD^2}{3.1999999 \times T}$
$t = 15$	$(1.4, 2.0, 1.4, 4.5, 1.4, 2.0, 1.4, \mathbf{29.7},$ $1.4, 2.0, 1.4, 4.5, 1.4, 2.0, 1.4)$	
$t = 31$	$(1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 8.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, \mathbf{72.3},$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 8.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4)$	$\frac{LD^2}{3.8599999 \times T}$
$t = 63$	$(1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 14.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, \mathbf{164.0},$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 14.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4)$	$\frac{LD^2}{4.6032258 \times T}$
$t = 127$	$(1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 12.6,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 23.5,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 12.6,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, \mathbf{370.0},$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 7.2,$ $1.4, 2.0, 1.4, 3.9, 1.4, 2.0, 1.4, 12.6,$ $\ldots$ $)$	$\frac{LD^2}{4.6032258 \times T}$

At the end of the paper, the authors discuss the complexity of finding good-performing patterns h and discuss potential future extensions to other gradient descent-based algorithms. One of the approaches for looking for optimal patterns h is using branch-and-bound algorithms, developed by Gupta et al [4].[1] The authors conclude by mentioning the computational aspects of searching for optimal certificates h.

Cyclic, periodically long step sizes have proven to be useful in convex and deterministic settings. However, new techniques handling nonconvexity and stochasticity would need to be developed to describe the effect of long steps in machine learning.

2.3 Gradient Descent with Cyclical Step-sizes

In the paper [2], the authors develop a convergence-rate analysis for the Polyak Heavy-Ball method with step-sizes chosen in cyclical order. The update in the Cyclical Heavy-Ball algorithm with a fixed

[1] Dr. Shuvomoy Das Gupta is joining the Rice CMOR department in 2025!

pattern $h = (h_0, \ldots, h_{K-1})$ of size K looks as follows:

$$x_1 = x_0 - \frac{h_0}{1+m}\nabla f(x_0)$$
$$x_{k+1} = x_k + h_{(k \bmod K)}\nabla f(x_k) + m(x_k - x_{k-1}), \quad k = 1, \ldots$$

The authors develop a worst-case analysis of modifications of this algorithm for quadratic objectives. Consider the problem

$$\min_{x \in R^n} x^T H x - b^T x$$

and suppose that $Sp(H) \subset \Lambda \subset [\mu, L] \subset R_{++}$. Moreover, suppose that the eigenvalues are contained on two intervals of equal length: $\Lambda \subset [\mu_1, L_1] \cup [\mu_2, L_2]$. For this case, the authors present and analyze the Cyclical Heavy-Ball algorithm with cycle of length 2. This algorithm and its analysis is the main contribution of this paper. First, define

$$\rho = \frac{L_2}{L_2 - \mu_1}$$
$$R = \frac{L_2 - \mu_2}{L_2 - \mu_1}$$
$$m = \left(\sqrt{\frac{\rho^2 - R^2 - \sqrt{\rho^2 - 1}}{1 - R^2}}\right)^2$$

and consider the following simple algorithm:

Input : Initial iterate x_0, $\mu_1 < L_1 \leq \mu_2 < L_2$ (where $L_1 - \mu_1 = L_2 - \mu_2, \mu_1 < L_2$)
Output : Sequence $\{x_t\}$ that converges to the solution of $\min_x f(x)$

$x_1 \leftarrow x_0 - \frac{1}{L_1}\nabla f(x_0)$;
for $t = 1, 2, \ldots$ **do**
  if t *is even* **then**
    $h_t \leftarrow \frac{1+m}{L_1}$;
  else
    $h_t \leftarrow \frac{1+m}{\mu_2}$;
  end
  $x_{t+1} \leftarrow x_t - h_t \nabla f(x_t) + m(x_t - x_{t-1})$;
end

Algorithm 1: Cyclical (K = 2) Heavy Ball with optimal parameters

This algorithm gives an improvement in the convergence guarantee for the worst-case convergence analysis in the class $\mathcal{C}_\Lambda$ of quadratic functions:

Theorem 2.3. *The worst-case rate r_t in Algorithm 1 over $\mathcal{C}_\Lambda$ for even iteration number t is*

$$r_t^{Alg.2} = \left(\frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\right)^t \left(1 + t\sqrt{\frac{\rho^2 - 1}{\rho^2 - R^2}}\right)$$

Interestingly, Cyclical Heavy-Ball performs well in the case of non-quadratic objectives, and authors present local convergence results for this case in the appendix.

At the end of the paper, the authors discuss worst-case analysis approaches in the spirit of [6] and [5], drawing a connection between optimal first-order methods and minimax polynomials. The authors present a general framework for designing optimal momentum and step size cycles for a given spectral structure and cycle length.

Lastly, the authors present experiments, applying the Cyclic Heavy Ball to quadratic and logistic regression, each applied on two datasets, and the MNIST handwritten digits, as well as a synthetic dataset. These experiments demonstrate the improved convergence of the cyclical approach compared to the standard heavy ball method.

2.4 Chebyshev Fractal Learning Rate Schedules

The paper [1] introduces a novel approach to optimizing gradient descent by utilizing a fractal permutation of learning rates derived from Chebyshev polynomials. This method leverages the properties of these polynomials to strategically manipulate learning rates, aiming to enhance both the stability and convergence speed of the optimization process.

Chebyshev Polynomials and Learning Rates: Chebyshev polynomials, $T_n(x)$, are extensively used because of their extremal properties, particularly their ability to minimize the maximum error in polynomial approximations over a specified interval. The roots of these polynomials are utilized to derive the step sizes for gradient descent. The roots $\gamma_t, t = 1, \dots T$ of a Chebyshev polynomial of degree T are given as

$$\gamma_t = \frac{1}{2}\left(\cos\left(\frac{(2t-1)\pi}{2T}\right) + 1\right), \quad t = 1, \dots T$$

The learning rates η_t are inversely proportional to these nodes:

$$\eta_t = \frac{1}{\gamma_t}$$

This direct correlation between the Chebyshev nodes and the learning rates is crucial for controlling the convergence behavior of the algorithm.

Fractal Scheduling of Learning Rates: To enhance the algorithm's robustness to large step sizes that might otherwise lead to instability, a fractal permutation of the Chebyshev-derived learning rates is employed. This fractal ordering aims to intersperse large and small steps effectively, ensuring a balance that promotes convergence while mitigating potential divergence. Let $\sigma_1 = [1]$, and for each $T \geq 1$ a power of 2, define

$$\sigma(2T) = \text{interlace}(\sigma(T), 2T + 1 - \sigma(T)),$$

where $\text{interlace}([a_1, \dots a_n], [b_1, \dots b_n]) = [a_1, b_1, a_2, b_2, \dots a_n, b_n]$.

This recursive definition ensures that the learning rates are applied in a non-monotonous yet structured fashion, reducing the risk of instabilities typical in methods that employ large learning rates sporadically.

Convergence Analysis: The convergence analysis focuses on the spectral properties of the gradient descent update matrix defined by the learning rate sequence. For the case of a quadratic objective, given a rate schedule $\{\gamma_t\}_{t=1}^T$, we have

$$x_{\text{out}} - x_* = \prod_{t=1}^{T}(I - \eta_t A)(x_0 - x_*)$$

Specifically, the residual polynomial, $P(A) = \prod_{t=1}^{T}(I - \eta_t A)$, representing the product of shifted identity matrices reduced by scaled Hessians, determines the convergence behavior. The spectral norm of this polynomial, particularly how tightly it can bound the spectrum of the Hessian matrix, A, is critical for understanding the rate at which the algorithm converges to the optimum. It follows from the eigenvalue decomposition that to achieve the best bound in the worst-case analysis, we need to minimize $\max_{x\in[\mu,L]} |P(x)|$ over the set of polynomials $\left\{ P(x) \middle| \deg P(x) = T, P(0) = 1 \right\}$. The Chebyshev fractal schedule chooses a pattern that achieves the desired minimum in the worst-case analysis. Furthermore, it employs an ordering that happens to accelerate convergence beyond traditional constant or adaptive learning rate schedules. It should be noted that changing the order of the learning rate schedule does not affect the worst-case analysis of the gradient descent algorithm. However, the ordering plays a crucial role in the actual convergence of the algorithm.

2.4.1 Experimental Results

Objective The objective of these experiments is to evaluate the effectiveness of different learning rate schedules on the training stability and performance of deep learning models across various datasets. This study extends the experiments described in paper [1] by applying these learning rate strategies to both classification tasks (CIFAR-10, MNIST) and a regression task (California Housing), thereby exploring their utility in diverse contexts.

Setup Three datasets were utilized: CIFAR-10 for image classification, MNIST for digit classification, and California Housing for regression analysis. The models used for the experiments are a modified ResNet-18 for CIFAR-10, a custom CNN for MNIST, and a custom dense network for the California Housing dataset. Each dataset utilized three types of learning rate schedules: a constant learning rate throughout training, a decreasing learning rate using exponential decay, and a dynamic learning rate using a Chebyshev polynomial-based schedule. All experiments were conducted on a personal computing setup equipped with an NVIDIA RTX 3060 GPU, using PyTorch as the machine learning framework.

CIFAR-10

- **Control Model:** Achieved high accuracy, demonstrating effective learning under stable conditions. The learning rate was constant at 0.2, and training stabilized with high accuracy early in the process. This is substantiated by the progression of training accuracy and loss, as detailed in Figure 1.
- **Decay Model:** Began with a learning rate of 0.3, decreasing by a factor of 0.90 per epoch (gamma=0.90). This model showed slower initial learning but increased stability over epochs. The graph in Figure 3 illustrates the gradual decrease in learning rate and its correlation with improved model performance over time.
- **Experimental Model:** Showed more variability in accuracy and loss across epochs. The learning rate adjustments led to fluctuations in performance but ultimately achieved high accuracy. Parameters for Chebyshev were m=0.45, M=1, T=128. The fractal scheduling and its effects on model stability and convergence are vividly captured in Figure 5.

MNIST

- **Control Model:** Demonstrated very fast convergence to high accuracy, achieving above 98% accuracy within just a few epochs. This model efficiently handled the simpler MNIST dataset with a constant lower learning rate of 0.1, as shown in Figure 6.
- **Decay Model:** Adapted the learning rate from a higher starting point of 0.2, reducing it exponentially (gamma=0.95). It matched the control model's performance closely but took slightly longer to stabilize, showing minor fluctuations in accuracy. The graphical representation in Figure 8 depicts the trajectory of the learning rate and its direct impact on training metrics.
- **Experimental Model:** Occasionally outperformed the decay model in terms of convergence speed and final accuracy, highlighting the potential benefits of adaptive learning rate schedules in classification tasks. Parameters for Chebyshev were m=4.08, M=7.85, T=8. Figure 10 showcases the fractal Chebyshev schedule, illustrating how it dynamically adapts to training demands to optimize performance.

California Housing

- **Control Model:** Used a constant learning rate of 0.008. It demonstrated steady improvement in MSE and R^2 Score, indicating that the model was learning effectively. The performance stability is evident in Figure 11.
- **Decay Model:** Improved MSE and R^2 scores faster than the control model, particularly in the early epochs, thanks to a starting learning rate of 0.01 that reduced over time (gamma=0.9). This approach likely helped in managing the larger parameter updates initially, with the decay ensuring that the updates did not destabilize the model as it approached optimal

values. This relationship between learning rate adjustment and performance improvement is graphically detailed in Figure 13.

- **Experimental Model:** Showed the most significant fluctuations in MSE and R^2 scores, reflecting the nature of the fractal Chebyshev schedule. Despite these fluctuations, it achieved the highest R^2 scores among the models. Parameters for Chebyshev were m=7, M=30, T=8. The impact of the fractal schedule on regression performance is detailed in Figure 15.

Conclusion: The results indicate that adaptive learning rate schedules can significantly impact the training dynamics and performance of deep learning models, both in classification and regression tasks. The Chebyshev-based dynamic schedule, in particular, often led to faster convergence and higher or comparable final performance metrics than the constant and decay-based schedules. This suggests that such adaptive strategies might be more widely applicable, potentially offering improvements in training deep learning models across various types of tasks and datasets.

References

[1] N. Agarwal, S. Goel, and C. Zhang. Acceleration via fractal learning rate schedules. *CoRR*, abs/2103.01338, 2021. URL https://arxiv.org/abs/2103.01338.

[2] B. Goujaud, D. Scieur, A. Dieuleveut, A. Taylor, and F. Pedregosa. Super-acceleration with cyclical step-sizes, 2022.

[3] B. Grimmer. Provably faster gradient descent via long steps, 2024.

[4] S. D. Gupta, B. P. G. V. Parys, and E. K. Ryu. Branch-and-bound performance estimation programming: A unified methodology for constructing optimal optimization methods, 2023.

[5] F. Pedregosa. Residual polynomials and the chebyshev method. http://fa.bianp.net/blog/2020/polyopt/, 2020.

[6] F. Pedregosa. Acceleration without momentum. http://fa.bianp.net/blog/2021/no-momentum/, 2021.

A Appendix

A.1 Figures and Graphs

A.1.1 CIFAR-10 Results

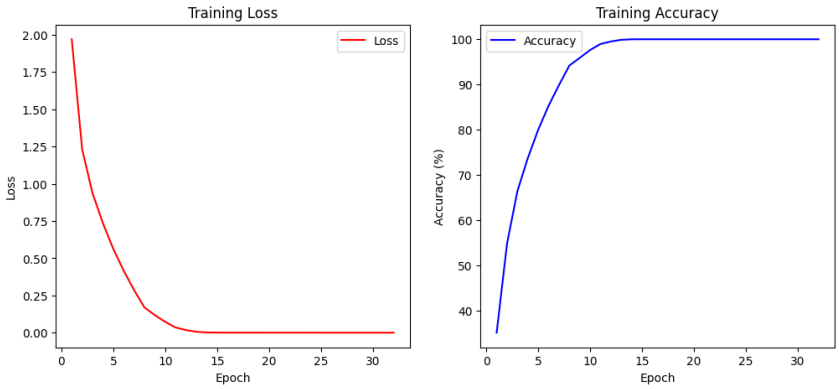

Figure 1: CIFAR-10 Control Model - Training accuracy and loss.

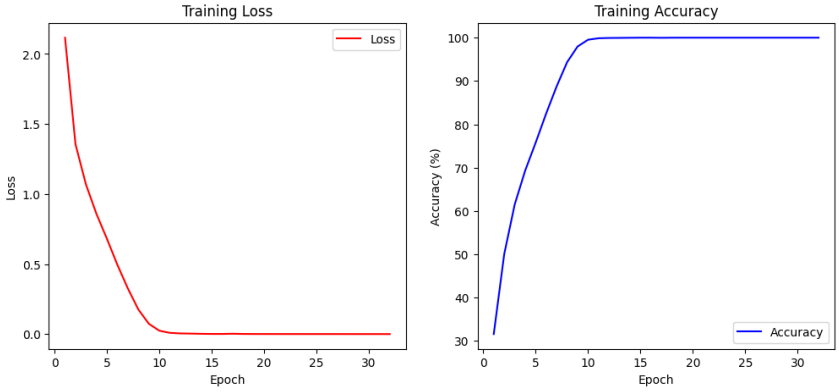

Figure 2: CIFAR-10 Decay Model - Training accuracy and loss.

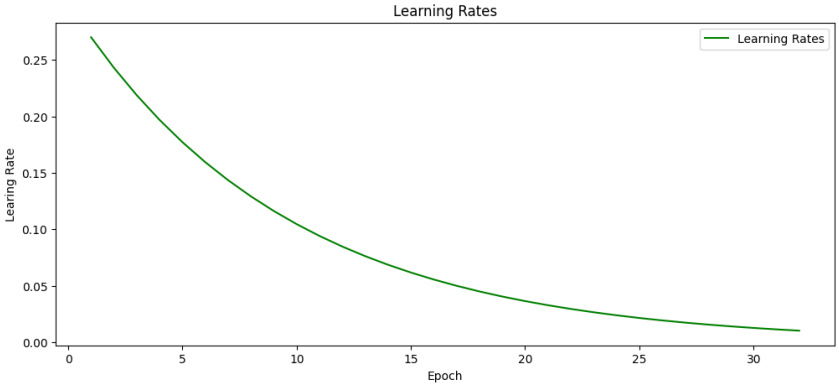

Figure 3: CIFAR-10 Decay Model - Learning rate over epochs.

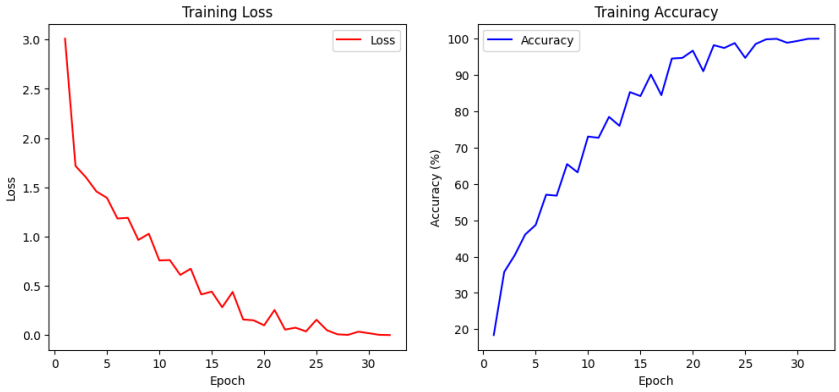

Figure 4: CIFAR-10 Experimental Model - Training accuracy and loss.

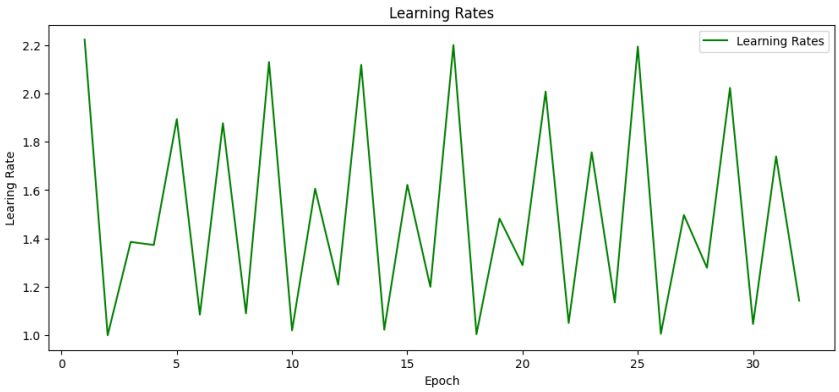

Figure 5: CIFAR-10 Experimental Model - Fractal Chebyshev learning rate schedule.

A.1.2 MNIST Results

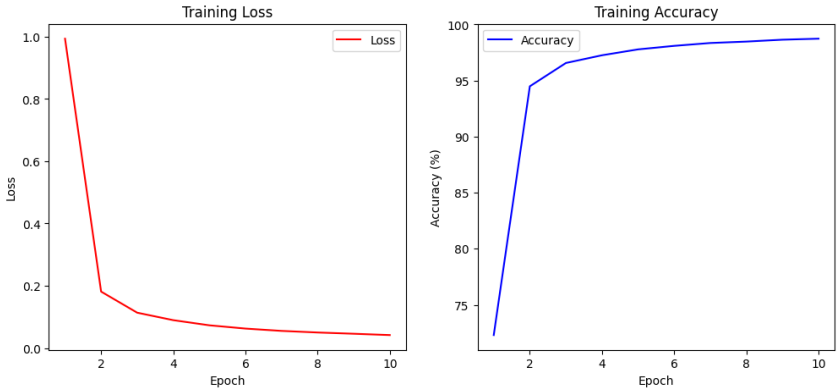

Figure 6: MNIST Control Model - Training accuracy and loss.

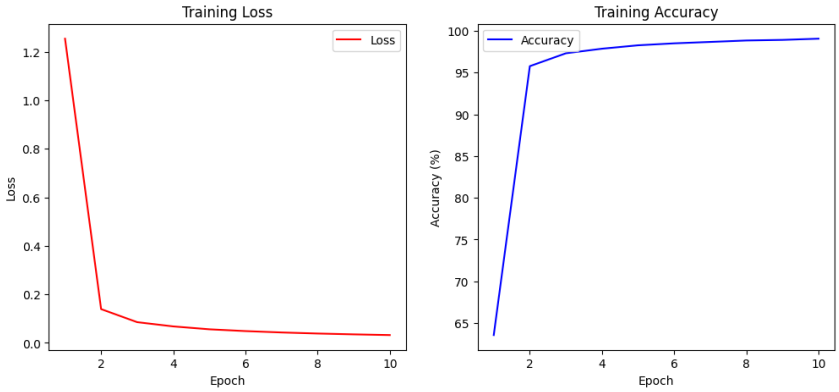

Figure 7: MNIST Decay Model - Training accuracy and loss.

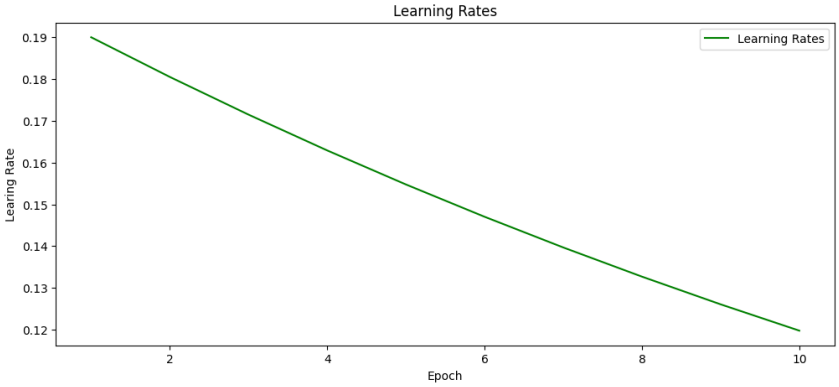

Figure 8: MNIST Decay Model - Learning rate over epochs.

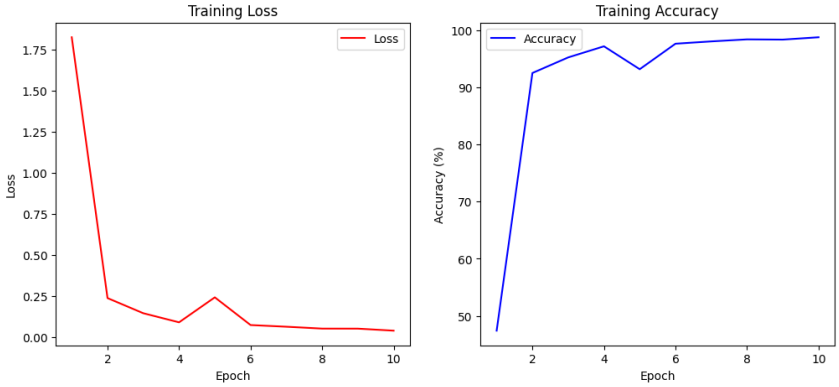

Figure 9: MNIST Experimental Model - Training accuracy and loss.

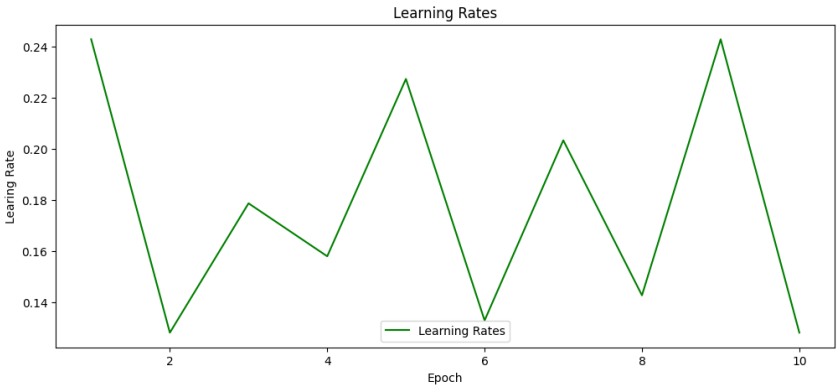

Figure 10: MNIST Experimental Model - Fractal Chebyshev learning rate schedule.

A.1.3 California Housing Results

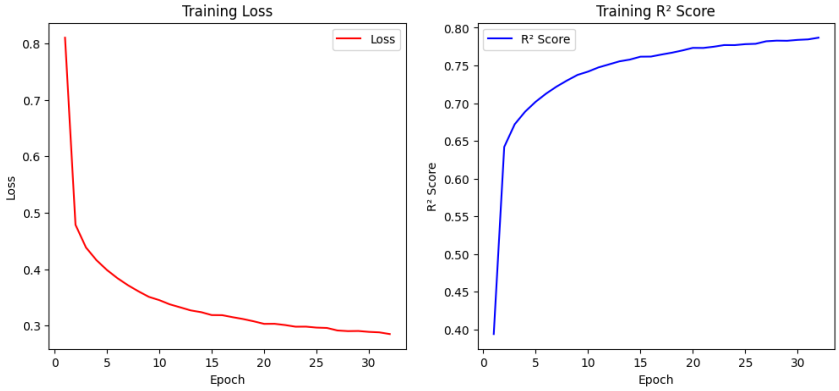

Figure 11: California Housing Control Model - MSE and R² Score.

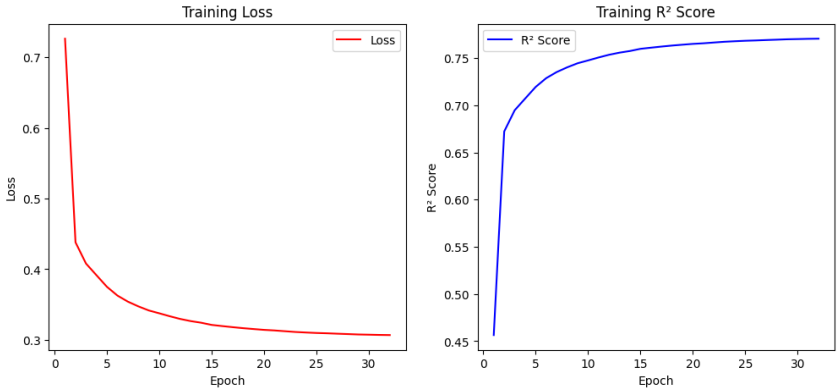

Figure 12: California Housing Decay Model - MSE and R² Score.

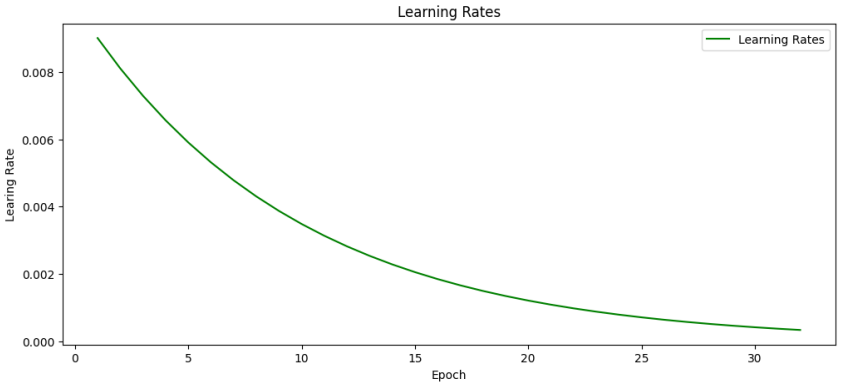

Figure 13: California Housing Decay Model - Learning rate over epochs.

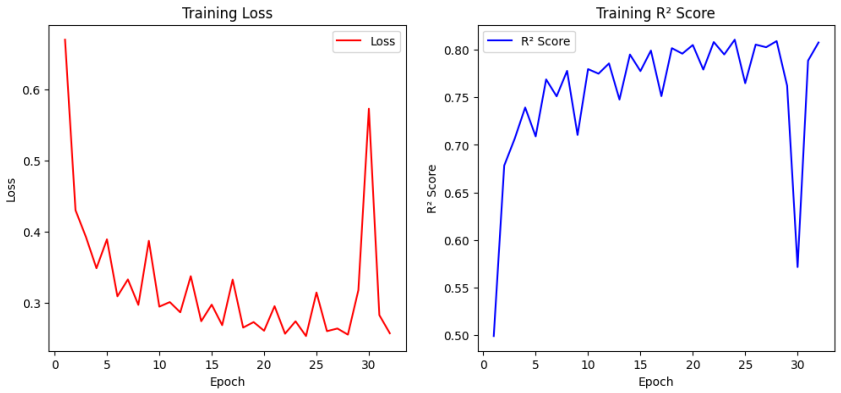

Figure 14: California Housing Experimental Model - MSE and R^2 Score.

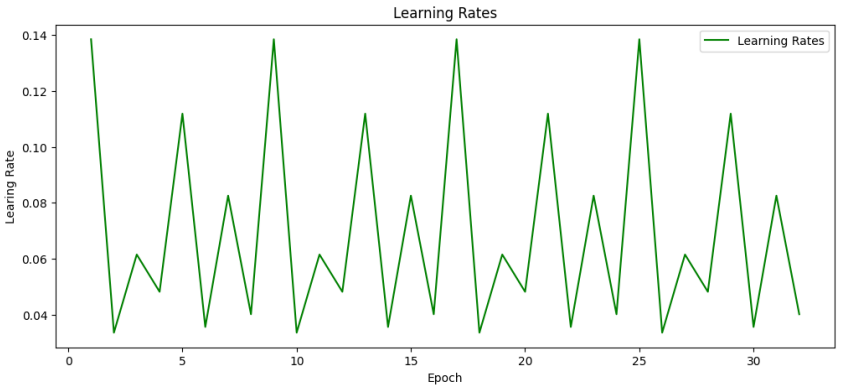

Figure 15: California Housing Experimental Model - Fractal Chebyshev learning rate schedule.