MABSearch in Supervised Learning: Adaptive Learning Rate Optimization

Internship report submitted **under**

IIITDM-SIES Programme

by

M. K. Guruprasad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KANCHEEPURAM

August 2025

Certificate

I, **M. K. Guruprasad** from Amrita Vishwa Vidyapeetham, Amaravati hereby declare that the material presented in the Internship Report titled **MAB Search** represents original work carried out by me in the **Department of Computer Science and Engineering** at the **Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram** during the year **2025**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student's Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this Report is carried out under my supervision, and is worthy of consideration for the requirements of internship work during the period May 2025 to July 2025.

Advisor's Name: Advisor's Signature

Abstract

Gradient Descent (GD) is a fundamental optimization algorithm that is frequently used in machine learning tasks because of its efficiency and ease of computation for convex and differentiable objective functions. The learning rate, a crucial hyperparameter in GD, regulates the size of updates during optimization. The choice of a suitable learning rate is crucial for convergence and generalization, but it is still difficult because this parameter is highly problem-dependent.

We extend the recent work that introduced MABSearch, a reinforcement learning strategy based on multi-armed bandits (MABs) for autonomously choosing the best learning rate in vanilla GD, to supervised machine learning. In particular, using the California Housing dataset, this study examines the suitability of MABSearch for learning rate selection in the training of an MLP Regressor.The proposed method uses performance-based rewards to explore and select a specific set of candidate learning rates during training. It treats the selection of learning rates as a bandit problem.

Experimental results show that the proposed method successfully identifies the optimal learning rate from a predefined set using the epsilon-greedy MAB strategy, leading to performance that closely matches the best manually tuned configurations. By improving training reliability and reducing the need for manual hyperparameter tuning, the study demonstrates that ideas from reinforcement learning can be applied to traditional machine learning optimization. This approach enables the use of smart, self-adjusting optimization methods in real-world machine learning processes.

Acknowledgements

I take this opportunity to express my sincere gratitude to everyone who supported and guided me during the course of this internship and the successful completion of this project.

I am deeply thankful to Dr. A. S. Syed Shahul Hameed, Assistant Professor, Department of Computer Science and Engineering, for his constant guidance and valuable mentorship. His insights and support throughout the project were instrumental in shaping my work.

I would also like to thank Dr. N. Rino Nelson, Internship Coordinator, Department of Computer Science and Engineering, for his coordination and timely support during the internship.

I am grateful to the Department of Computer Science and Engineering, IIITDM Kancheepuram, for providing the resources and environment necessary for carrying out this work.

Finally, I thank my family and friends for their encouragement and support throughout this journey.

Contents

List of Figures

List of Tables

Abbreviations

GD	**G**radient **D**escent
LR	**L**earning **R**ate
RL	**R**einforcement **L**earning
MAB	**M**ulti-**A**rmed **B**andit
MDP	**M**arkov **D**ecision **P**rocess
OP	**O**ptimization
SR	**S**uccess **R**ate
MLP	**M**ulti-**L**ayer **P**erceptron
EMA	**E**xponential**M**oving **A**verage
MSE	**M**ean **S**quared **E**rror
RMSE	**R**oot **M**ean **S**quared **E**rror
ML	**M**achine **L**earning

Symbols

α	Learning rate
β	Smoothing factor for reward updates
λ	Decay constant for ϵ
ϵ	Exploration probability in ϵ-greedy policy
ϵ_t	Exploration probability at time step t
X_t	Model parameters at time step t
X_{t+1}	Updated model parameters after one iteration
$\nabla f(X)$	Gradient of the objective function at X
$\nabla f(X_t)$	Gradient of the loss function at X_t
$f(X)$	Objective function
$\mathcal{A}$	Set of available actions (learning rates)
R_a	Reward for selecting action a
R_i	Estimated reward (inverse of loss) for learning rate α_i
r_t	Observed reward (loss) at time step t
D	Dimensionality of the input space
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

Chapter 1

Introduction

1.1 Background

Gradient Descent (GD) is a widely used optimization technique in both numerical methods and machine learning. Its popularity comes from its simplicity and efficiency in minimizing loss functions that are convex and differentiable. The key idea behind GD is to iteratively adjust model parameters by moving in the direction opposite to the gradient, gradually approaching the function's minimum. Central to this process is the learning rate, a hyperparameter that controls how large each step is during the update.

Choosing the right learning rate is crucial for gradient descent to work effectively. If the learning rate is too low, the algorithm converges very slowly or might get stuck in suboptimal areas. On the other hand, if it's too high, the algorithm may overshoot the minimum or even diverge completely. Despite how important this parameter is, it's often chosen using manual tuning or heuristics—approaches that don't scale well with complex models or large datasets [1, 2, 3].

To overcome this limitation, researchers have started exploring reinforcement learning (RL) as a way to make adaptive, data-driven optimization decisions [4]. A simplified but powerful variant of RL is the Multi-Armed Bandit (MAB) model, which offers a lightweight and effective way to make sequential decisions under uncertainty. When applied to gradient descent, each possible learning rate can be treated like an "arm" of the bandit [5]. By assigning a reward based on the model's performance with a given learning rate, the MAB algorithm learns over time which rate works best.

1.2 Motivation

This approach replaces the traditional trial-and-error method of learning rate tuning with an adaptive, feedback-driven system. Using an epsilon-greedy policy, which is common in MAB algorithms, the method strikes a balance between trying new learning rates (exploration) and sticking with the best-performing one (exploitation). This turns the learning rate selection process into a problem of maximizing rewards [6].

While the original MABSearch algorithm performed well on synthetic benchmarks [7, 8], its real-world effectiveness in machine learning had not been studied. Deep learning models like Multi-Layer Perceptrons (MLPs), which often have complex and non-convex loss landscapes, are ideal candidates for such adaptive strategies. This project aims to test the practical applicability of MABSearch in real-world supervised learning settings.

Additionally, this work is motivated by recent trends in multi-objective and hybrid metaheuristic approaches such as MOPSO [9], which highlight the growing interest in adaptive optimization methods that balance performance and robustness.

1.3 Objectives of the Work

The main goals of this project are:

- To implement and embed the MABSearch algorithm into the training pipeline of a supervised learning model.
- To apply the approach to a regression task using the California Housing dataset and evaluate its effectiveness.
- To compare MABSearch-driven learning rate selection with fixed-rate strategies in terms of model performance and consistency.

- To assess how practical and beneficial reinforcement learning—particularly the multi-armed bandit model—is for adaptive hyperparameter tuning in real-world machine learning.

Chapter 2

Methodology

This chapter presents the methodology followed in this work, starting with the foundations of gradient descent and reinforcement learning. It introduces the Multi-Armed Bandit framework, explains the proposed MABSearch algorithm, its adaptation to a machine learning context, and finally describes the evaluation metrics used for performance assessment.

2.1 Overview of Gradient Descent

Gradient Descent (GD) is a widely used optimization algorithm in both machine learning and numerical computing. It is a first-order iterative method designed to minimize an objective function by moving in the direction of its negative gradient. The fundamental idea is that the gradient of a function points in the direction of steepest ascent, so moving in the opposite direction leads to a reduction in the function's value.

Mathematically, for a differentiable objective function $f(X)$, the parameter vector X is updated at each iteration t as follows:

$$X_{t+1} = X_t - \alpha \nabla f(X_t)$$

Here, α is the **learning rate**, a positive scalar that determines the step size in the direction of the negative gradient. A well-chosen learning rate leads to fast and stable convergence, while a poorly chosen one may result in slow training, suboptimal solutions, or even divergence.

- If the learning rate α is too small, the convergence becomes slow, and the algorithm may get stuck in local minima.
- If α is too large, the updates may overshoot the minimum, causing oscillations or divergence.

Despite its simplicity, selecting an appropriate learning rate remains a challenge. Traditional approaches rely on trial-and-error or fixed schedules (e.g., exponential decay), which often require domain-specific knowledge and extensive tuning. These limitations motivate the need for adaptive learning rate strategies that can adjust dynamically during training based on model performance — an idea explored further in this work through the Multi-Armed Bandit (MAB) based MABSearch algorithm.

2.2 Multi-Armed Bandit (MAB) Framework

The Multi-Armed Bandit (MAB) problem is a classical formulation in reinforcement learning that models sequential decision-making under uncertainty. It involves an agent repeatedly choosing from a set of possible actions (referred to as "arms"), each associated with an unknown and potentially stochastic reward. The objective is to maximize cumulative reward over time by strategically balancing the trade-off between exploring new actions and exploiting known high-reward ones.

In this framework:

- **Arms:** Each action the agent can take — in this work, each arm corresponds to a candidate learning rate.
- **Reward:** The numerical feedback received after choosing an arm — here, derived from the model's loss after training with a selected learning rate.

- **Policy:** The strategy used to select which arm to pull at each step.

A popular policy for balancing exploration and exploitation is the **ϵ-greedy strategy**. Under this approach:

- With probability ϵ, the agent explores by randomly selecting an arm.
- With probability $1 - \epsilon$, the agent exploits by selecting the arm with the current best estimated reward.

Over time, ϵ is typically decayed to encourage more exploitation once sufficient information has been gathered through exploration. Epsilon decay has been explored in different forms to optimize exploration strategies, including recent variants like reward-based decay [10].

2.3 MABSearch Algorithm

MABSearch is a reinforcement learning-inspired optimization strategy that integrates the Multi-Armed Bandit (MAB) framework with the gradient descent (GD) algorithm. It addresses the challenge of selecting an optimal learning rate by dynamically choosing from a predefined set of candidates based on observed model performance [5].

The motivation behind MABSearch aligns with the broader trend of using adaptive decision-making algorithms to overcome hyperparameter tuning challenges in machine learning [11, 12]. Bandit-based approaches offer a lightweight alternative to full reinforcement learning, making them suitable for integration into gradient-based optimization workflows [13, 6].

Problem Formulation

Let $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ be a finite set of candidate learning rates. Each learning rate α_i is treated as an arm in the MAB setup. In this work, we use:

$$\mathcal{A} = \{0.1, 0.0001\}$$

The model is trained iteratively using gradient descent, where the learning rate at iteration t is selected from $\mathcal{A}$ using a decision policy. The parameter update rule is:

$$X_{t+1} = X_t - \alpha_t \nabla f(X_t)$$

Here, $\alpha_t \in \mathcal{A}$ is the learning rate selected at time t by the MAB agent, based on past reward estimates.

Application to Machine Learning

To evaluate the effectiveness of MABSearch in a real-world task, it was applied to a supervised regression problem using the California Housing dataset [14]. The model used was a simplified configuration of `MLPRegressor` with no hidden layers and an identity activation function, effectively making it a linear model. The task is to predict median house value based on numerical features.

Arm Selection: Epsilon-Greedy Strategy

The learning rate is selected at each step using an ϵ-greedy strategy:

$$\alpha_t = \begin{cases} \text{randomly select from } \mathcal{A}, & \text{with probability } \epsilon_t \\ \arg\min_{\alpha_i \in \mathcal{A}} R_i, & \text{with probability } 1 - \epsilon_t \end{cases}$$

The exploration probability ϵ_t is decayed over time to encourage more exploitation:

$$\epsilon_t = \epsilon_{\min} + (\epsilon_{\max} - \epsilon_{\min}) \cdot \exp(-\lambda t)$$

This adaptive strategy ensures a balance between exploration and exploitation and follows the approach proposed in [4]. Recent enhancements such as reward-sensitive decay mechanisms further improve convergence in dynamic environments [10].

Reward Estimation via Training Error

After each training step, the model's performance is evaluated on the training data using Mean Squared Error (MSE). The cost estimate for the selected learning rate α_i is updated using an exponential moving average:

$$R_i \leftarrow (1-\beta)R_i + \beta \cdot \mathrm{MSE}_t$$

where:

- R_i is the current cost estimate for learning rate α_i,
- MSE_t is the observed error at step t,
- β is a smoothing factor, typically set to 0.9.

This strategy helps mitigate noise during training and has been previously explored in both optimization and behavioral modeling contexts [6].

Incremental Training Strategy

Training is performed incrementally using `fit()` with `max_iter=1` and `warm_start=True`. This allows the model to retain learned weights across updates while enabling dynamic learning rate changes. The training loop proceeds as follows:

1. Select a learning rate α_t using the ϵ-greedy strategy.
2. Perform one step of gradient descent: $X_{t+1} = X_t - \alpha_t \nabla f(X_t)$.
3. Compute the training MSE and update the reward estimate R_i.
4. Update the exploration rate ϵ_t.

By combining reinforcement learning concepts with classical optimization, MABSearch provides a lightweight and adaptive strategy for hyperparameter tuning. Its integration with gradient descent allows real-time adjustment of the learning rate, improving convergence behavior without the need for manual tuning or grid search. This aligns with the broader goal of building self-adaptive machine learning systems as discussed in [1, 2].

2.4 Evaluation Metrics

Model performance was assessed using the following metrics:

- **Mean Squared Error (MSE)** — measures the average of the squared differences between the predicted and actual values. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  where y_i is the true target value, $\hat{y}_i$ is the predicted value, and n is the number of samples.

- **Root Mean Squared Error (RMSE)** — the square root of MSE, providing a metric in the same unit as the target variable:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

These metrics were computed on both the training and test sets to evaluate the model's convergence behavior and generalization performance.

Chapter 3

Work Done

This chapter outlines the detailed steps taken throughout the project, from dataset preparation and baseline model evaluation to the integration and testing of the proposed MABSearch-based learning rate selection approach.

3.1 Dataset Preparation and Preprocessing

The experiments in this project were conducted on the California Housing dataset, a widely used benchmark dataset for regression tasks. It contains 20,640 instances, each with 8 numerical features such as median income, housing age, number of rooms, population, and geographic coordinates. The target variable is the median house value in a block group.

The dataset was accessed using `scikit-learn`'s built-in function `fetch_california_housing()` [14]. Basic preprocessing steps were performed as follows:

- **Imputation:** Missing values in features (if any) were replaced using the mean strategy with `SimpleImputer`.
- **Feature Scaling:** All features were standardized using `StandardScaler` to have zero mean and unit variance.

- **Train-Test Split:** The dataset was split into training (80%) and test (20%) sets using `train_test_split` with a fixed random seed for reproducibility.

3.2 Baseline Model with Fixed Learning Rates

To understand the impact of the learning rate on performance, the `MLPRegressor` was trained using fixed learning rates selected from the set:

$$\{0.1, 0.01, 0.001, 0.0001\}$$

The model was configured with:

- No hidden layers (`hidden_layer_sizes=()`)
- Identity activation function
- Stochastic Gradient Descent (SGD) as the optimizer
- Full batch training using the entire dataset
- `max_iter=100`, `shuffle=False`, and `warm_start=False`

Each model was trained independently with a different learning rate, and its Mean Squared Error (MSE) was calculated on the test set. These results serve as a benchmark to evaluate the performance of the dynamic learning rate strategy later.

3.3 Adaptive Learning Rate using MABSearch

The core objective of this project was to adapt the MABSearch algorithm, originally proposed for optimizing synthetic benchmark functions, to a real-world machine learning setting.

Action Space

The learning rate selection problem was framed as a Multi-Armed Bandit (MAB) task. The available learning rates were treated as arms:

$$\mathcal{A} = \{0.1, 0.0001\}$$

These values represent two extremes — a high learning rate prone to instability but fast convergence, and a low learning rate offering more stable yet slower updates.

Training Configuration

The same model architecture was used as in the baseline: a shallow linear MLPRegressor. However, instead of fixing the learning rate, it was dynamically selected at each iteration based on the ϵ-greedy bandit strategy.

Training was performed incrementally:

- The model was trained for 1 epoch at a time using `fit()` with `max_iter=1` and `warm_start=True`.
- The learning rate was adjusted at each iteration.
- The process was repeated for 99 steps.

Bandit Strategy and Reward Update

At each step, the algorithm chose between exploration and exploitation:

- With probability ϵ, a random learning rate was chosen.
- With probability $1-\epsilon$, the learning rate with the lowest estimated MSE was selected.

The exploration rate ϵ was decayed over time using an exponential decay schedule:

$$\epsilon_t = \epsilon_{\text{min}} + (\epsilon_{\text{max}} - \epsilon_{\text{min}}) \cdot \exp(-\lambda t)$$

Observed training loss was used to update the reward estimate for the selected learning rate using an exponential moving average:

$$R_i \leftarrow (1 - \beta) R_i + \beta \cdot \text{MSE}_t$$

where $\beta = 0.9$ is the smoothing factor.
This learning strategy builds on the work of prior researchers applying MAB to dynamic systems and optimization tasks [6, 9, 3].

Final Evaluation

After 99 steps, the final trained model was evaluated on the test set to assess generalization performance. The resulting MSE was then compared against the baseline results from fixed learning rates to evaluate the effectiveness of the MABSearch-driven adaptive learning rate mechanism.

All code was implemented using Python and `scikit-learn`, and executed within Jupyter notebooks for transparency and reproducibility.

Chapter 4

Results and Discussions

This chapter presents and analyzes the results obtained from evaluating both fixed and adaptive learning rate strategies applied to a regression task using the California Housing dataset. The goal of this evaluation is to compare the performance of traditional fixed learning rate selection with the proposed MABSearch approach that dynamically adapts the learning rate during training using a reinforcement learning-based strategy.

4.1 Performance with Fixed Learning Rates

To establish a performance baseline, the model was trained with four different fixed learning rates: $\{0.1, 0.01, 0.001, 0.0001\}$. Each configuration used a shallow `MLPRegressor` with no hidden layers, trained for 100 iterations in full-batch mode using stochastic gradient descent (SGD).

4.2 Adaptive Learning Rate Selection with MABSearch

To overcome the limitations of manual learning rate tuning, the MABSearch algorithm was applied to dynamically select the learning rate during training. MABSearch models the learning rate selection problem as a Multi-Armed Bandit (MAB) scenario, where each candidate learning rate is treated as an arm.

In this experiment, the action space consisted of two learning rates: $\mathcal{A} = \{0.1, 0.0001\}$. At each training iteration, one of the two was selected using an ϵ-greedy policy, and a single step of training was performed with that learning rate. The selection policy was influenced by the performance (MSE) of previously used learning rates, which was tracked using an exponential moving average reward update.

The training process continued for 99 iterations with the model state preserved across steps using `warm_start=True`. After training, the final model was evaluated on the test set.

4.3 Comparison of Learning Rate Strategies

The test MSEs obtained using both fixed learning rates and the adaptive MABSearch strategy are presented in Table 4.1. This unified view allows for easy comparison of all configurations.

The adaptive strategy not only matched but slightly outperformed the best manually selected fixed learning rate. This outcome highlights the potential of reinforcement learning-based optimization to automatically tune hyperparameters in a data-driven way without the need for manual experimentation.

4.4 Visualization and Analysis

To further understand the training dynamics and behavior of the MABSearch algorithm, several plots were generated.

Reward Tracking

Figure 4.1 shows how the estimated reward values (inversely related to MSE) for each learning rate arm evolved over time. It illustrates how MABSearch quickly learned to favor the better-performing learning rate.

Training MSE Trend

Figure 4.2 compares the training MSE trends between fixed learning rates and the MABSearch strategy. It highlights how MABSearch initially explores but then converges toward a performance comparable to or better than fixed learning rates.

Learning Rate Selection Pattern

Figure 4.3 visualizes the learning rate choices made by the MABSearch algorithm at each step. The decay of ϵ is also plotted, showing the shift from exploration to exploitation over time.

4.5 Discussion

The results clearly demonstrate that MABSearch is an effective approach for adaptive learning rate selection in gradient descent. Despite using only two candidate learning rates, the algorithm successfully identified the optimal one through reward-based learning and decision-making.

This dynamic tuning approach offers several advantages:

- It reduces reliance on manual hyperparameter search.
- It can respond to non-stationary optimization landscapes by adapting the learning rate over time.
- It integrates seamlessly into the training loop with minimal overhead.

Furthermore, the use of training loss as a feedback signal, coupled with an exponential moving average reward update, ensured stability despite the stochastic nature of training.

Although the current study used a shallow model to clearly observe learning rate effects, the same framework can be extended to deep learning scenarios and larger action spaces. Future work can explore the inclusion of more candidate learning rates, integration with other optimizers, and evaluation across diverse datasets and tasks.

Table 4.1: Comparison of Fixed and Adaptive Learning Rates

Strategy	**Test MSE**
Fixed Learning Rate (0.1)	0.5630
Fixed Learning Rate (0.01)	0.7413
Fixed Learning Rate (0.001)	1.3350
Fixed Learning Rate (0.0001)	5.8914
MABSearch (Adaptive LR)	0.5518

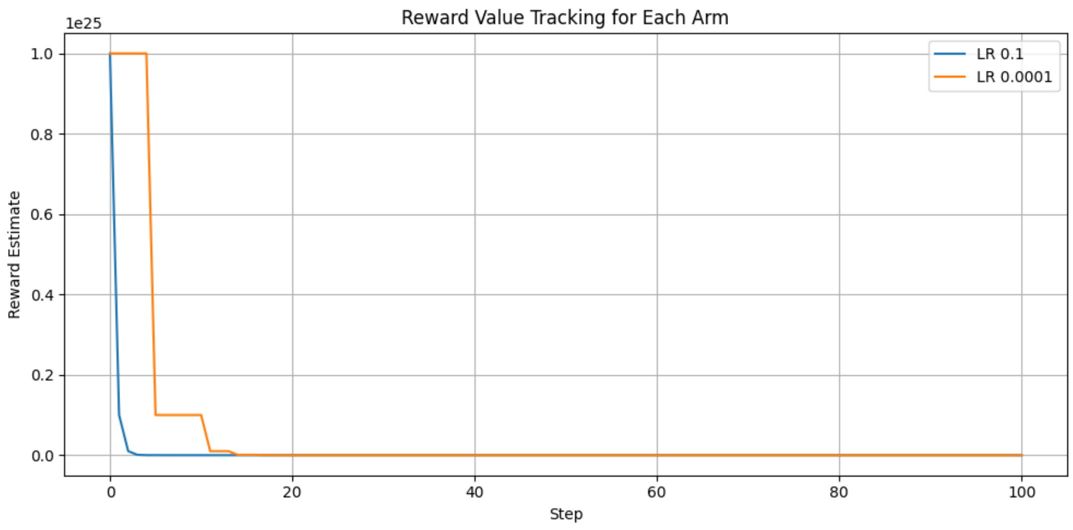

Figure 4.1: Reward value tracking for each arm over training steps.

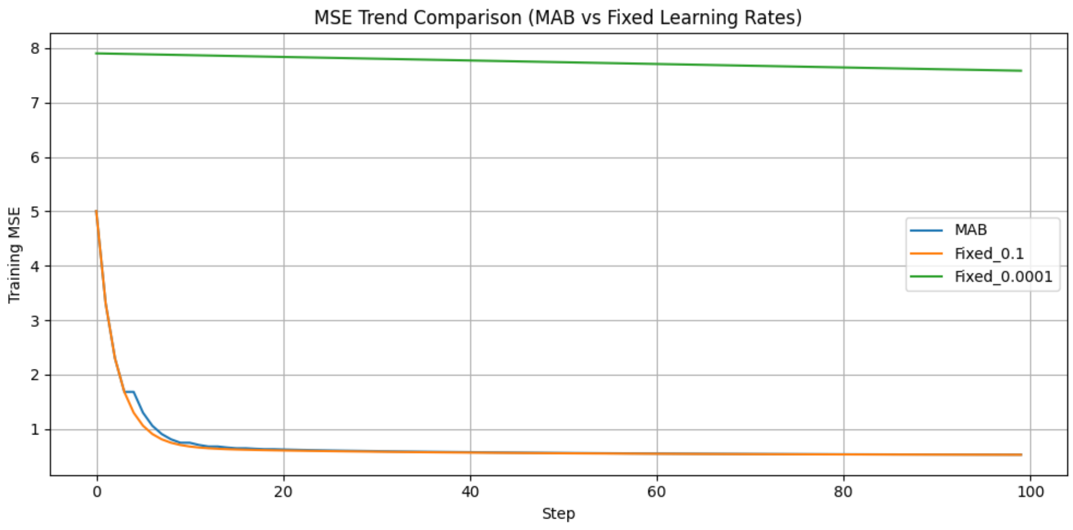

Figure 4.2: Training MSE trend comparison: MABSearch vs fixed learning rates.

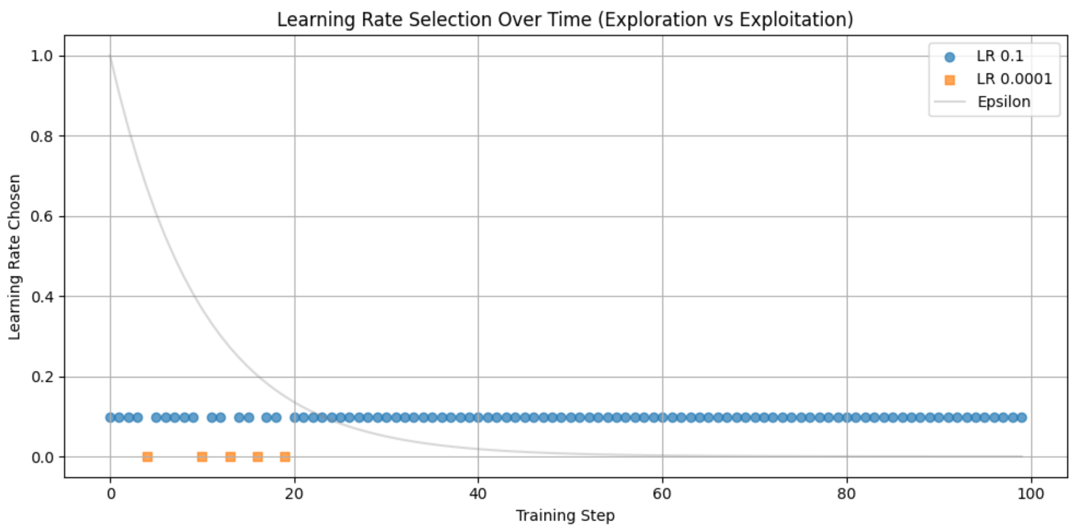

FIGURE 4.3: Learning rate selection over time (exploration vs exploitation).

Chapter 5

Conclusions and Extensions

This work explored the integration of reinforcement learning principles into the gradient descent process for learning rate selection. Specifically, we extended the previously proposed MABSearch algorithm—based on the Multi-Armed Bandit (MAB) framework—to a practical supervised learning task using the California Housing dataset.

The study demonstrated that MABSearch can successfully adapt the learning rate over time, using an ϵ-greedy policy and reward-based feedback derived from training loss. Compared to traditional fixed learning rate strategies, MABSearch not only avoided manual hyperparameter tuning but also achieved comparable or better performance. The adaptive nature of the algorithm allowed it to converge toward the most effective learning rate without prior knowledge or human intervention.

By evaluating both fixed and dynamic learning rate approaches, we showed that reinforcement learning techniques like MABs can be effectively embedded into standard machine learning training loops. This work serves as a concrete step toward more autonomous and self-tuning optimization methods.

Future Extensions

While the current study focused on a simplified regression model, the following directions can be considered for extending this work:

- **Deep Neural Networks:** Applying MABSearch to deep architectures (e.g., CNNs, RNNs) to investigate its scalability and stability in more complex settings.
- **Expanded Action Space:** Increasing the number of candidate learning rates to allow finer granularity in decision-making and richer exploration.
- **Alternative Reward Signals:** Experimenting with other reward metrics such as validation loss, training speed, or convergence rate instead of MSE alone.
- **Other Hyperparameters:** Extending the MAB-based search to include other hyperparameters like batch size, momentum, or weight decay in a multi-dimensional optimization setting.
- **Comparison with Other RL Methods:** Evaluating MABSearch against other adaptive strategies like Bayesian optimization or full reinforcement learning (e.g., Q-learning).

Overall, this work illustrates how reinforcement learning, even in its simplest form, can provide elegant and efficient solutions to common optimization challenges in machine learning.

Bibliography

[1] K. Sörensen, "Metaheuristics: The metaphor exposed," *International Transactions in Operational Research*, vol. 22, 2015. [Online]. Available: https://doi.org/10.1111/itor.12001

[2] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications, Chapter 3.* Cambridge University Press, 2020. [Online]. Available: https://doi.org/10.1017/9781108690935

[3] M. Locatelli and F. Schoen, "(global) optimization: Historical notes and recent developments," *EURO Journal on Computational Optimization*, vol. 9, 2021. [Online]. Available: https://doi.org/10.1016/j.ejco.2021.100012

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 2018.

[5] A. Syed Shahul Hameed and N. Rajagopalan, "Mabsearch: The bandit way of learning the learning rate—a harmony between reinforcement learning and gradient descent," *National Academy Science Letters*, vol. 47, pp. 29–34, 2024. [Online]. Available: https://doi.org/10.1007/s40009-023-01292-1

[6] V. Agrawal and P. Shenoy, "Tracking what matters: A decision-variable account of human behavior in bandit tasks," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 43, 2021.

[7] N. Rooy, "Landscapes python benchmark functions," https://github.com/nathanrooy/landscapes/blob/master/landscapes/single/_objective.py.Web, accessed: 30-Dec-2022.

[8] M. Jamil and X.-S. Yang, “A literature survey of benchmark functions for global optimisation problems,” *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, 2013.

[9] G. S. Mahapatra, B. Maneckshaw, and K. Barker, “Multi-objective reliability redundancy allocation using mopso under hesitant fuzziness,” *Expert Systems with Applications*, vol. 198, 2022. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.116696

[10] A. Maroti, “Rbed: Reward based epsilon decay,” *arXiv preprint*, 2019. [Online]. Available: http://arxiv.org/abs/1910.13701

[11] A. S. Syed Shahul Hameed and N. Rajagopalan, “Spgd: Search party gradient descent algorithm, a simple gradient-based parallel algorithm for bound-constrained optimization,” *Mathematics*, vol. 10, no. 5, 2022. [Online]. Available: https://doi.org/10.3390/math10050800

[12] R. Gupta, V. Mahendran, and V. Badarla, “Optimal searching of prefetched dash segments in fog nodes: A multi-armed bandit approach,” in *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2021.

[13] D. Glowacka, “Bandit algorithms in interactive information retrieval,” in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, 2017.

[14] R. K. Pace and R. Barry, “California housing prices dataset,” https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html, 1997, accessed via Scikit-learn.