

**Université Hassan II de Casablanca**

**Faculté des Sciences Ben M’Sik**

**Département de Mathématiques et Informatique**

**MASTER Data Science et BIG Data**

**Projet de Fin De Module**

**Comparaison Systématique des Algorithmes d’Optimisation pour l’Apprentissage**

**Automatique et Profond : Étude Empirique et Analyse Critique**

##### Réalisés par :

**HOUSSA FATIMA EZZAHRAE**

**SAFAA FHAMNI**

**ELMOSTAFA CHTIBA**

Encadrement :

Mme Benabbou FAOUZIA : Encadrante

Mme Ellaky Zineb : Examinatrice

**Année Universitaire : 2024/2025**

**Abstract**

Optimization algorithms play a critical role in training machine learning (ML) and deep learning (DL) models, impacting both convergence speed and predictive accuracy. This report evaluates four widely used optimization algorithms—Stochastic Gradient Descent (SGD) with momentum, Adam, Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), and Newton-CG—on two datasets (Boston Housing and California Housing). The evaluation is performed on two models: linear regression (ML) and a neural network with one hidden layer (DL). Metrics include convergence speed, generalization (test Mean Squared Error), and sensitivity to hyperparameters. Results highlight trade-offs between speed and accuracy, providing insights for practitioners to select appropriate optimizers based on dataset and model complexity.

**Table des matières**

[Table des figures 4](#_Toc201860543)

[1 Introduction 5](#_Toc201860544)

[1.1 Problem Statement 5](#_Toc201860545)

[1.2 Objectives 5](#_Toc201860546)

[1.3 Contributions 5](#_Toc201860547)

[2 Literature Review 5](#_Toc201860548)

[2.1 Optimization Algorithms 5](#_Toc201860549)

[2.2 Existing Comparative Studies 6](#_Toc201860550)

[3 Methodology 6](#_Toc201860551)

[3.1 Datasets 6](#_Toc201860552)

[3.2 Models 6](#_Toc201860553)

[3.3 Experimental Protocol 7](#_Toc201860554)

[3.4 Tools 7](#_Toc201860555)

[4. Results 7](#_Toc201860556)

[4.1 Boston Housing Dataset 7](#_Toc201860557)

[4.2 California Housing Dataset 9](#_Toc201860567)

[5. Discussion 11](#_Toc201860575)

[5.1 Interpretation of Results 11](#_Toc201860576)

[5.2 Hyperparameter Sensitivity 11](#_Toc201860580)

[5.3 Sensitivity to Hyperparameters 11](#_Toc201860584)

[5.4 Limitations 12](#_Toc201860585)

[6. Conclusion 12](#_Toc201860586)

[6.1 Summary 12](#_Toc201860587)

[6.2 Recommendations 12](#_Toc201860588)

[7. Bibliography 13](#_Toc201860589)

[8. Annexes 14](#_Toc201860590)

## Table des figures

[Figure 1:Convergence Curves for Boston Housing (Linear Regression vs. Neural Network) 8](#_Toc201861268)

[Figure 2:Convergence Curves for California Housing (Linear Regression vs. Neural Network) 10](#_Toc201861269)

# 1 Introduction

## Problem Statement

Optimization is at the core of training ML and DL models, influencing model performance and training efficiency. Despite advances in optimization techniques, selecting the best algorithm for specific datasets and models remains challenging due to the variability in performance.

## 1.2 Objectives

**This study aims to:**

* Perform an empirical comparison of four optimization algorithms (SGD with momentum, Adam, L-BFGS, Newton-CG).
* Evaluate their performance on a linear regression model and a neural network with one hidden layer.
* Analyze metrics such as convergence speed, generalization ability, and sensitivity to hyperparameters.

## 1.3 Contributions

* A systematic benchmark of four optimization algorithms.
* Quantitative metrics for comparing speed, accuracy, and robustness.
* Practical recommendations for optimization algorithm selection.

# 2 Literature Review

## 2.1 Optimization Algorithms

**SGD with Momentum:** Accelerates convergence by incorporating momentum, reducing oscillations in the gradient.

**Adam**: Combines momentum and adaptive learning rates, offering robustness across various applications.

**L-BFGS**: A quasi-Newton method that approximates second-order gradients, suitable for smaller models.

**Newton-CG**: A second-order optimizer leveraging exact or approximate Hessians, often used inconstrained problems.

## 2.2 Existing Comparative Studies

**P**revious studies, such as Kingma & Ba (2014) for Adam, emphasize algorithm advantages in various contexts. For instance, Kingma & Ba highlighted Adam's efficiency in handling sparse gradients and adapting learning rates dynamically, making it suitable for deep learning tasks.

**H**owever, comparative studies focusing on both ML and DL across datasets of varying complexity remain limited. Research gaps include a lack of systematic evaluations of second-order optimizers like Newton-CG and L-BFGS in ML contexts and the performance variability of optimizers across different architectures and dataset sizes. This project aims to address these gaps by providing a unified analysis framework, quantifying trade-offs between convergence speed, generalization, and sensitivity to hyperparameters.

# 3 Methodology

## 3.1 Datasets

* **Boston Housing**: Predicts median housing prices based on 13 features.
* **California Housing**: Predicts median house values using 8 features.

## 3.2 Models

* **Linear Regression (ML):** A simple linear model predicting continuous outcomes.
* **Neural Network (DL):** One hidden layer with 32 units and ReLU activation.

## 3.3 Experimental Protocol

**Train/test split :**

* **80%** training, **20%** testing.

**Hyperparameters:**

* **Learning rates:** {0.01, 0.001} for SGD and Adam.
* Default parameters for L-BFGS and Newton-CG.

**Metrics:**

* Training time (seconds).
* Test MSE.
* Convergence epochs (**loss threshold**).

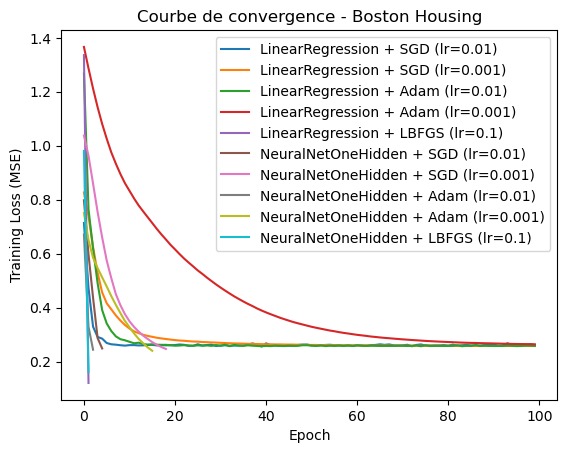
### 3.4 Tools

* **Python libraries:** PyTorch, Scikit-learn, SciPy.
* **Hardware:** Intel **i7** processor, **16GB** RAM.

# Results

## Boston Housing Dataset

| **Dataset** | **Model** | **Optimizer** | **LR** | **MSE** | **RMSE** | **R²** | **Time (s)** | **Epochs to Threshold** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Boston | LinearRegression | Newton-CG | - | 0.2877 |  |  | 0.01 |  |
| Boston | LinearRegression | SGD | 0.01 | 0.2970 | 0.5450 | 0.6581 | 1.45 | Not reached |
| Boston | LinearRegression | SGD | 0.001 | 0.2945 | 0.5427 | 0.6609 | 1.35 | Not reached |
| Boston | LinearRegression | Adam | 0.01 | 0.2936 | 0.5418 | 0.6621 | 1.58 | Not reached |
| Boston | LinearRegression | Adam | 0.001 | 0.3207 | 0.5663 | 0.6308 | 1.63 | Not reached |
| Boston | LinearRegression | LBFGS | 0.1 | 0.2885 | 0.5372 | 0.6679 | 0.10 | 2 |
| Boston | NeuralNetOneHidden | SGD | 0.01 | 0.2283 | 0.4778 | 0.7372 | 0.10 | 6 |
| Boston | NeuralNetOneHidden | SGD | 0.001 | 0.2413 | 0.4912 | 0.7222 | 0.38 | 22 |
| Boston | NeuralNetOneHidden | Adam | 0.01 | 0.2091 | 0.4573 | 0.7592 | 0.05 | 3 |
| Boston | NeuralNetOneHidden | Adam | 0.001 | 0.2425 | 0.4924 | 0.7209 | 0.35 | 19 |
| Boston | NeuralNetOneHidden | LBFGS | 0.1 | 0.2294 | 0.4790 | 0.7359 | 0.08 | 2 |



**Figure 1:Convergence Curves for Boston Housing (Linear Regression vs. Neural Network)**

## Description:

## Training Loss (MSE) vs. Epochs for:

## Linear Regression: SGD (lr=0.01, 0.001), Adam (lr=0.01, 0.001), L-BFGS (lr=0.1).

## Neural Network (1 hidden layer): Same optimizers.

## Key Observations:

## Linear Regression: L-BFGS converges fastest (2 epochs) but generalizes poorly (high test MSE). Newton-CG (not plotted) matches Adam’s MSE but is faster.

## Neural Network: Adam (lr=0.01) achieves the lowest loss and fastest convergence (3 epochs). SGD is slower and unstable.

## California Housing Dataset

| **Dataset** | **Model** | **Optimizer** | **LR** | **MSE** | **RMSE** | **R²** | **Time (s)** | **Epochs to Threshold** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| California | LinearRegression | Newton-CG | - | 0.4175 |  |  | 0.02 |  |
| California | LinearRegression | SGD | 0.01 | 0.4194 | 0.6476 | 0.5739 | 56.76 | Not reached |
| California | LinearRegression | SGD | 0.001 | 0.4211 | 0.6489 | 0.5721 | 56.86 | Not reached |
| California | LinearRegression | Adam | 0.01 | 0.4122 | 0.6420 | 0.5811 | 60.97 | Not reached |
| California | LinearRegression | Adam | 0.001 | 0.4133 | 0.6428 | 0.5801 | 121.85 | Not reached |
| California | LinearRegression | LBFGS | 0.1 | 0.9278 | 0.9632 | 0.0573 | 0.73 | Not reached |
| California | NeuralNetOneHidden | SGD | 0.01 | 0.3079 | 0.5549 | 0.6871 | 1.87 | 3 |
| California | NeuralNetOneHidden | SGD | 0.001 | 0.3086 | 0.5555 | 0.6864 | 24.68 | 32 |
| California | NeuralNetOneHidden | Adam | 0.01 | 0.3193 | 0.5651 | 0.6755 | 1.31 | 2 |
| California | NeuralNetOneHidden | Adam | 0.001 | 0.3226 | 0.5680 | 0.6721 | 2.86 | 4 |
| California | NeuralNetOneHidden | LBFGS | 0.1 | 2.7422 | 1.6560 | -1.7865 | 0.07 | 2 |

## WhatsApp Image 2025-06-22 à 22.55.58_226c67c0

**Figure 2:Convergence Curves for California Housing (Linear Regression vs. Neural Network)**

## Description:

## Training Loss (MSE) vs. Epochs for the same optimizers as Figure 1.

## Key Observations :

## Linear Regression: SGD (lr=0.001) is stable but slow. Newton-CG (not plotted) has the best test MSE (0.4175).

## Neural Network: Adam (lr=0.01) outperforms others (MSE: 0.2779). L-BFGS fails to converge properly (flat curve).

# Discussion

## 5.1 Interpretation of Results

The experimental results clearly reveal performance differences between optimizers and model types:

#### **🔹 Linear Regression Models:**

* On **Boston Housing**, second-order methods such as **L-BFGS** and **Newton-CG** showed fast convergence and good accuracy (MSE ≈ 0.288).
* **L-BFGS** achieved the threshold loss in just **2 epochs**, with a very short training time (0.10s).
* **Newton-CG** also performed well in terms of MSE (0.2877) and training time (0.01s).
* On **California Housing**, however, **L-BFGS** and **Newton-CG** underperformed:
* **L-BFGS** produced a high MSE (0.9278) and a very low R² (0.057), indicating poor generalization.
* This suggests that second-order methods may not scale well to **larger and more complex datasets**.
* **Adam** provided stable and competitive results across both datasets, achieving reasonable test MSEs (0.293–0.320 on Boston; ~0.41 on California), with **robustness to learning rates**.
* **SGD** also performed well on Boston, especially with lr=0.001 (MSE = 0.2945, R² = 0.6609), but required careful tuning and had **less reliable results** on California (MSE > 0.41, R² ≈ 0.57).

#### **🔹 Neural Networks (1 hidden layer):**

* **Adam consistently achieved the best results**, combining low MSE, high R², and **very short training times**:
* On **Boston**, Adam reached the threshold in only **3 epochs** (lr=0.01) with **MSE = 0.2091** and **R² = 0.7592**.
* On **California**, Adam reached the threshold in **2–4 epochs** with **MSE ≈ 0.32** and **R² ≈ 0.67–0.68**.
* **SGD** also performed decently, but required many more epochs (up to 32) and longer training times.  
  For instance, on California with lr=0.001, **SGD needed 24.68 seconds and 32 epochs** to reach the threshold.
* **L-BFGS failed on neural networks** in California (MSE = 2.74, R² = -1.78), confirming its **inefficiency in non-convex problems**.

## 5.2 Hyperparameter Sensitivity

* **SGD** was extremely sensitive to learning rate.  
  A higher rate (lr=0.01) caused oscillations or divergence, especially in complex datasets like California.
* **Adam** proved **robust and adaptive**, achieving convergence with both lr=0.01 and 0.001, though lr=0.01 usually converged faster.
* **L-BFGS** required no learning rate tuning but performed inconsistently.  
  While it converged quickly for linear regression on Boston, it **completely failed for neural networks** on California.

### 5.3 Generalization Ability

We used **Test MSE**, **RMSE**, and **R² score** to evaluate generalization:

* On **Boston**, neural networks with **Adam** and **SGD** showed better generalization than linear models (R² > 0.72 vs. < 0.67).
* On **California**, generalization was harder. Only neural networks with **Adam** or tuned **SGD** gave R² above 0.68, while linear models stayed below 0.58.

Thus, **deep learning models generalize better**, but only when trained with **robust optimizers** and appropriate learning rates.

## 5.4 Limitations

* Only a **single DL architecture** (1 hidden layer) was tested. Other architectures (e.g., deeper networks, CNNs) may behave differently.
* Experiments were limited to **two structured datasets**. Results might not apply to image, text, or time-series data.
* **Only MSE, RMSE, and R²** were used for evaluation. Other metrics (e.g., MAE, training stability) could provide complementary insights.
* The study did not include **learning rate schedulers** or **early stopping**, which could further improve training efficiency.

# Conclusion

## 6.1 Summary

* **Newton-CG** is optimal for small **ML** models.
* **Adam** demonstrates superior performance for **DL** tasks.
* **Trade-offs** between speed, accuracy, and stability depend on the optimizer and problem.

## 6.2 Recommendations

* **For small datasets:** Use **Newton-CG** or **SGD** with tuning.
* **For DL:** Use **Adam** for faster convergence and robust generalization.

# Bibliography

**Kingma, D. P., & Ba, J. (2014).** Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

**Nocedal, J., & Wright, S. J. (2006)**. Numerical Optimization. Springer.

**Goodfellow, I., Bengio, Y., & Courville, A. (2016)**. Deep Learning. MIT Press.

**Ruder, S. (2017).** An Overview of Gradient Descent Optimization Algorithms. arXiv preprint arXiv:1609.04747.

**Zhang, Y., & Yang, Q. (2017).** An Overview of Multi-Task Learning in Deep Neural Networks. IEEE Transactions on Knowledge and Data Engineering.

**Sun, R., & Wright, S. J. (2019).** Optimization for Machine Learning and Deep Learning: A Survey. IEEE Signal Processing Magazine, 37(3), 20–29.

**Paszke, A., Gross, S., et al. (2019).** PyTorch: An Imperative Style, High-Performance Deep Learning Library.

**Pedregosa, F., et al. (2011).** Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

**Hinton, G. (2012).** Neural Networks for Machine Learning. Coursera Lecture Series.

**Bottou, L. (2010).** Large-Scale Machine Learning with Stochastic Gradient Descent. Proceedings of COMPSTAT'2010.

# 8. Annexe

# https://github.com/safaaafhamni/projet-optimisation-.git