Reinforcement Learning for Computing Numerical Eigenvalues and Eigenvectors

Project Proposal

March 2025

Abstract

Using RL to find numerical values of eigenvalues and eigenvectors. Does this make sense? What should be reward formulations?

1 Timelines

1. Make sure that you selected this project and mentioned your name in sheet. Link: Google Sheet Numerical Algorithms
2. ChatGPT or any other coding assistant is allowed as much as you want!
3. Read the material of the project selected, and run the basic starter code, if it exists.
   - did you understand the code? if not, then read it again!
4. Phase-1 evaluation on 25/26 March. Did you mention this in sheet? If not, then do it!
   - Done with evaluation? Was something suggested during evaluation? Did you clearly figured out? If not, then contact instructor for clarification!
5. Project objectives are well defined! Continue working.
6. Phase-2 evaluation 10 April. Are you doing OK?
   - You are at a stage where you are midway, setup baselines, and entered research phase.
7. Final evaluation 7th or 8th May
   - Introduce your work, objectives, and your outcomes. Time 15 minutes. Keep the slides clear, one line per item. Use overleaf and beamer! Select presentation format: Overleaf presentation template. Then click on open as template. This creats your own copy.
8. Most projects are doable on google colab. Link: Google Colab. If your code requires longer runs, then do a checkpoint to save the intermediate data, then run it again.
9. Submission: Report 5 pages, codes.
10. Do you need more GPU resources? Let us know ASAP. Dom kae sure that you have a base code running on colab for small dataset.

2 Introduction

Eigenvalue problems are fundamental in many areas of science and engineering, including quantum mechanics, stability analysis, and numerical simulations. Classical numerical methods such as power iteration, QR decomposition, and Krylov subspace methods are widely used for eigenvalue computation Golub and Van Loan [1996]. However, these methods can be computationally expensive for large-scale problems, particularly in high-dimensional applications.

Recent advances in reinforcement learning (RL) have shown promise in solving complex numerical problems by learning adaptive strategies. This project aims to investigate the use of RL-based approaches for eigenvalue and eigenvector computation, leveraging deep RL to develop efficient iterative solvers. The goal is to explore whether RL agents can learn adaptive search strategies that outperform traditional numerical methods in specific cases.

3 Starter Code

Starter Code

4 Problem Formulation

Given a square matrix $A \in \mathbb{R}^{n\times n}$, the goal is to compute its dominant eigenvalues λ and associated eigenvectors v satisfying:

$$Av = \lambda v. \tag{1}$$

The RL agent is tasked with learning an iterative method for estimating eigenvalues by interacting with the system, receiving feedback, and optimizing a policy that minimizes the residual error:

$$R = -\|Av - \lambda v\|^2. \tag{2}$$

5 Challenges and Research Questions

- How can reinforcement learning policies be designed to approximate eigenvalues more efficiently than classical methods?
- Can RL learn a generalizable eigenvalue solver that works across different matrix classes?
- How does the RL-based method compare in accuracy, convergence speed, and computational cost to power iteration and Lanczos methods Saad [2011]?
- Can we use RL to accelerate convergence in iterative eigenvalue solvers, particularly in large sparse systems?

6 Proposed Approach

We propose the following reinforcement learning framework:

- State Representation:
  - The state consists of the current vector approximation v_t and the residual error $r_t = Av_t - \lambda_t v_t$.
- Action Space:
  - The RL agent selects an update direction for v_t, similar to power iteration or Krylov subspace selection.

- Reward Function:
  - The agent receives a reward proportional to the reduction in residual error: $R_t = -\|Av_t - \lambda_t v_t\|^2$.
- Learning Algorithm:
  - Proximal Policy Optimization (PPO) Schulman et al. [2017] is used to train the agent to optimize eigenvalue convergence.

7 Experimental Setup

We will test the RL-based eigenvalue solver on:

- Dense Matrices: Randomly generated symmetric and non-symmetric matrices.
- Sparse Matrices: Large sparse systems from real-world applications.
- Quantum Systems: Hamiltonian matrices from physics simulations.
- Graph-Based Eigenvalue Problems: Spectral clustering in machine learning.

8 Evaluation Metrics

- Eigenvalue Approximation Error: $\|Av - \lambda v\|^2$.
- Convergence Speed: Number of iterations required for a given tolerance.
- Computational Efficiency: Time complexity comparison with classical methods.
- Generalization Performance: Performance on unseen matrix distributions.

9 References and GitHub Code

- Papers:
  - Matrix Computations Golub and Van Loan [1996].
  - Numerical Methods for Eigenvalue Problems Saad [2011].
  - Proximal Policy Optimization Algorithms Schulman et al. [2017].
  - Reinforcement Learning for Numerical Methods Arjona-Medina et al. [2021].
- GitHub Repositories:
  - OpenAI Baselines for RL Experiments.
  - SciPy: Eigenvalue Solvers in Python.
  - Google Research: RL for Numerical Methods.

10 Expected Outcomes

- Development of an RL-based eigenvalue solver that adapts to different matrix types.
- Comparative study on accuracy and efficiency with classical solvers.
- Investigation of the generalization capability of RL for eigenvalue problems.

References

Jose Arjona-Medina, Jonas Schneider, Laura von Rueden, Jochen Garcke, and Christof Sch"utte. Learning to solve pdes with physics-informed reinforcement learning. *arXiv preprint arXiv:2103.10297*, 2021.

Gene H Golub and Charles F Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

Yousef Saad. *Numerical Methods for Large Eigenvalue Problems.* Society for Industrial and Applied Mathematics (SIAM), 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.