Tiny Project Report

Instructor: Huynh Trung Hieu

Member: Nguyen Trong Huy - 10423046

Project Report: Linear System Solver and Evaluator

Introduction

The objective of this project is to design and implement an object-oriented C++ system that can solve linear systems using techniques from linear algebra. This includes handling regular square systems, symmetric positive definite systems, and non-square systems such as those encountered in data modeling.

The project is divided into two parts:

- **Part A:** Create reusable C++ classes to represent vectors, matrices, and various types of linear systems.
- **Part B:** Apply these classes to solve a real-world linear regression problem using the UCI CPU Performance dataset.

Through these parts, the project demonstrates how object-oriented programming (OOP) principles and numerical linear algebra can be combined to build modular, testable, and extensible computational tools.

GitHub Repository: ://github.com/TrogHuy/tinyProjectFinal

Overview of Implemented Classes

1. **Vector**
   Represents a dynamic array of doubles using 0-based indexing. Supports element access, dot product, scalar operations, and printing.
2. **Matrix**
   Implements a 2D array using 1-based indexing. Features include:
   - Matrix operations (addition, multiplication, transpose)
   - Determinant and inverse for square matrices
   - Pseudo-inverse for non-square matrices
3. **LinearSystem**
   A base class for solving square systems using Gaussian elimination. Accepts a square matrix A and a vector b and solves $Ax = b$.

4. **PosSymLinSystem**
   Inherits from `LinearSystem`, using the Conjugate Gradient method for symmetric positive definite matrices.

5. **NonSquareSystem**
   Solves $Ax = b$ where A is not square, using the Moore–Penrose pseudo-inverse computed as $(A^T A)^{-1} A^T$.

6. **DataLoader**
   Loads and parses the `machine.data` file from the UCI dataset, extracting the 6 input features as matrix A and the target PRP as vector b.

7. **Evaluator**
   Provides functions to split the dataset into training/testing subsets and compute the Root Mean Square Error (RMSE) between predicted and actual target values.

Applying the Classes to Part B

In Part B, the classes developed in Part A are used to build a linear regression model that predicts CPU performance (PRP) from six features: MYCT, MMIN, MMAX, CACH, CHMIN, and CHMAX.

Workflow:

1. Load the dataset using `LoadDataFromFile("machine.data", A, b)`.

2. Split the data using `SplitTrainTest(A, b, trainA, trainb, testA, testb)`.

3. Solve for the regression coefficients using:

```
NonSquareSystem model(&trainA, &trainb);
Vector x = model.Solve();
```

4. Make predictions: `Vector predicted = testA * x;`

5. Compute RMSE: `double rmse = ComputeRMSE(predicted, testb);`

Result:

The system was tested with an 80/20 train-test split. A sample result is:

```
RMSE on test set: 36.4278
```

This shows the system's capability to model real data using linear algebra techniques implemented from scratch.

Conclusion

This project demonstrates the power of combining object-oriented design with mathematical modeling. The classes developed for matrices, vectors, and solvers provide a reusable foundation for linear algebra computations. In Part B, these tools are applied to a regression task, yielding reasonable predictive performance and validating the correctness of the implementation.

References

- UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets/Computer+Hardware`
- Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations*. JHU Press.
- Trefethen, L. N., & Bau, D. (1997). *Numerical Linear Algebra*. SIAM.