Estimating Trigonometric Functions Using ML

Alexandra Walker and Mattie Fuller

University of Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80918
awalker17@uccs.edu and mfuller@uccs.edu

Abstract

This paper proposes training an RL agent to be able to estimate the solutions to triangles containing missing angles and sides. These estimates will be evaluated via mean squared error (MSE) and other similar metrics; should the margin of error be small, the hidden functions that the agent is approximating can be found. That is, to show that an RL agent can 'discover' trigonometric functions such as sin, cos or tan.

Introduction

Machine Learning (ML) is a method of computational learning utilizing large data sets to analyze algorithms for the purpose of finding trends in data. This proposal will outline a method of analyzing right triangles to estimate sine, cosine, tangent, and possibly more like Pythagorean theorem, Pythagorean identity and many more. Right triangles have many unique and intricate properties that can be expressed by elementary functions. To that end, this paper proposes an experiment to teach trigonometry to a ML algorithm, such as reinforcement learning.

Background

Machine Learning was imagined as soon as 1959 and since then has evolved to detect medical conditions (Verma and Peyada, 2020) such as breast cancer, or even predicting natural disasters (Gracia et al., 2021). In addition, papers around solving math problems, have been using RL to better break down algebraic equations for many purposes (Dabelow and Ueda, 2025). The process of finding functions to best fit the data even has its own field as summarized by Kumar and Bhatnagar (2022) for linear regression. For non-linear regression, Gauss-Newton, a popular method as proposed in Korbit and Zanon (2024) is a better regression technique for non-linear applications. Verma and Peyada mentioned earlier used Gauss-Newton to demonstrate a new method of breast cancer classification.

Literature Review

Reinforcement Learning (RL) has been used successfully for similar problems to this. Specifically Q-Learning has been used solve algebraic problems in an exact manner. (Dabelow and Ueda,2025). This was achieved by having the world model consist of 3 mutable stacks, one for the right hand side of the equation, one for the left hand side of the equation and the last effectively as scratch paper for the RL agent to solve the problem. The agent's actions were then defined as moving elements between the stacks. This approach yielded great results; a similar approach likely could be used for solving trigonometry problems.

Problem Statement

In its most simple terms: can a reinforcement learning algorithm learn to solve right triangles by estimating trigonometric functions.

Dataset

Triangles, especially right triangles, are particularly easy to generate with high accuracy. With this in mind, a custom dataset can be easily generated with thousands if not hundreds of thousands of samples in only a few minutes, so finding data to train on with solid data is easy to create. However, as discussed later, more information is necessary to allow reinforcement learning to go more smoothly. Including, missing data, the function method best suited to solve the triangle, and having the correct answer for the missing information is vital to train on. Our methodology section will go into how these will be used, but during training having the answer and the solve by method included into the dataset information simplifies the process. The best suited function is one of (potentially) many ways to solve for any given side or angle in a triangle. For example, a triangle with 2 angles included does not require any complex algorithm to solve for the third angle: $A3 = 180 - A1 - A2$, where as one angle, one side, and the right angle will require a sine, cosine, or tangent function to find the second side.

Methodology

Initially focusing on using Q-Learning, a Reinforcement Learning method, we seek to estimate sin, tan, and cos. As such, the key elements to consider are the following: States, Actions, Reward Policies, and the Environment model (Ghasemi and Ebrahimi, 2024).

Environment Model And State

Reinforcement Learning (RL) functions by allowing an autonomous agent to interact with a representation of the world via applying actions to a state; then rewarding the agent for 'good' choices and punishing for 'bad' choices. States are specific configurations of the environment, for instance, a chess board with black king h8, white pawn g7 and white king e7, is a state representing black in checkmate to white's pawn protected by white's queen.

The RL agent can only interface with and interpret the world via states. Given this, only information the RL agent needs to learn a task should be included. States can be expressed as N-dimensional vectors where N is the number of significant data types (Harmon and Harmon, 1996). For solving triangles with missing sides or angles, we propose 7 significant features.

| Side A-C | Angle a-c | Best Fit Function | Solve For |

Table 1: A table depicting our 7 key features for representing state.

Sides A-C represent the side lengths of their corresponding sides, likewise angles a-c represent the angle measurements of each angle. These 6 features are likely the minimum required to train the RL agent to solve triangles. We believe that including two more features: "Best Fit Function" and "Solve For" will aid in training the agent. The latter is simply a representation of which other feature the agent should try to solve for, expressed as 0-5 with each index corresponding to one of the 6 sides and angles.

The "Best Fit Function" feature represents what elementary mathematical function would typically be used to solve this type of equation; Sin could be the easiest way to solve a specific triangle, in such cases this feature would express that or any other function that would be typically used to solve the problem. By including this feature, the RL agent will effectively learn to distinguish the type of its problems. And in the best case be able to reverse engineer the original functions from this.

Actions

An RL agent views the world through states; it interacts with the world through actions. Any manipulation of a state must be represented as an action that the agent can preform. In this way, we must define all the possible ways that the agent is permitted to interact with the state.

We identified 23 possible actions the agent might need to perform to solve a triangle with missing sides/angles.

- Add or subtract K from the proposed answer, where k is an arbitrary small quantity. (2 actions)
- $K_1 - K_2$, where $K_1, K_2 \in [a, b, c]$ and $K_1 \neq K_2$ (6 actions)
- $K_1 * K_2$, where $K_1, K_2 \in [A, B, C]$ (9 actions)
- $K_1 \div K_2$, where $K_1, K_2 \in [A, B, C]$ and $K_1 \neq K_2$ (6 actions)

Though it does warrant stating this might be possible with just the 2 actions of adding or subtracting a small sum from the proposed answer. But this would likely result in much less accurate estimates.

Reward Policies

As shown in Jin et al. (2024) demonstrated a new RL reward algorithm designed for Q-Learning that showed "near-optimal policy with satisfactory convergence, and performs well across MDPs". Seeking to achieve similar results, their proposed algorithm will be the experimental reward policy of the RL agent.

Timeline

The project as it is completed will loosely follow the timeline depicted below, dates in bold are strict and cannot be changed:

2/19/25	Project Proposal Report
2/19/25-2/24/25	Generate Dataset
2/24/25-3/15/25	Build and debug RL 'model'
3/15/25-4/1/25	Train and debug RL 'model'
4/1/25	Midterm Report
4/1/25-5/8/25	Finalize Models and Report Paper
5/8/25	Final Demo

Evaluation

The success or failure of this project will be determined by two factors. Firstly, whether the RL agent is able to accurately estimate the correct answers to the trig problems. Secondly, if we can reverse engineer the trigonometric functions from the RL agent's learned estimations; that is can the RL agent 'discover' trigonometric functions such as sin(x), cos(x), and tan(x). Furthermore metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Squared Error (MSE) can be used to provide numerical representations of how well the RL agent estimates correct answers.

Conclusion

As such, we hope to show that an RL agent utilizing Q-Learning can estimate solutions to triangles with missing sides or angles. Then using that learned ability to solve triangles to hopefully derive/discover the actual function the agent is approximating.

References

Dabelow, L.; and Ueda, M. 2025. Symbolic equation solving via reinforcement learning. *Neurocomputing*, 613: 128732.

Ghasemi, M.; and Ebrahimi, D. 2024. Introduction to Reinforcement Learning. arXiv:2408.07712.

Gracia, S.; Olivito, J.; Resano, J.; del Brio, B. M.; de Alfonso, M.; and Álvarez, E. 2021. Improving accuracy on wave height estimation through machine learning techniques. *Ocean Engineering*, 236: 108699.

Harmon, M. E.; and Harmon, S. S. 1996. Reinforcement learning: A tutorial. *WL/AAFC, WPAFB Ohio*, 45433: 237–285.

Jin, Y.; Gummadi, R.; Zhou, Z.; and Blanchet, J. 2024. Feasible Q-Learning for Average Reward Reinforcement Learning. In Dasgupta, S.; Mandt, S.; and Li, Y., eds., *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, 1630–1638. PMLR.

Korbit, M.; and Zanon, M. 2024. Incremental Gauss-Newton Descent for Machine Learning. *arXiv preprint arXiv:2408.05560*. Accessed: 2025-02-18.

Kumar, S.; and Bhatnagar, V. 2022. A Review of Regression Models in Machine Learning. *JOURNAL OF INTELLIGENT SYSTEMS AND COMPUTING*, 3(1): 40–47.

Verma, H. O.; and Peyada, N. K. 2020. Parameter estimation of aircraft using extreme learning machine and Gauss-Newton algorithm. *The Aeronautical Journal*, 124(1272): 271–295.