PLAYING MASTERMIND WITH REINFORCEMENT LEARNING

Nithish Bolleddula

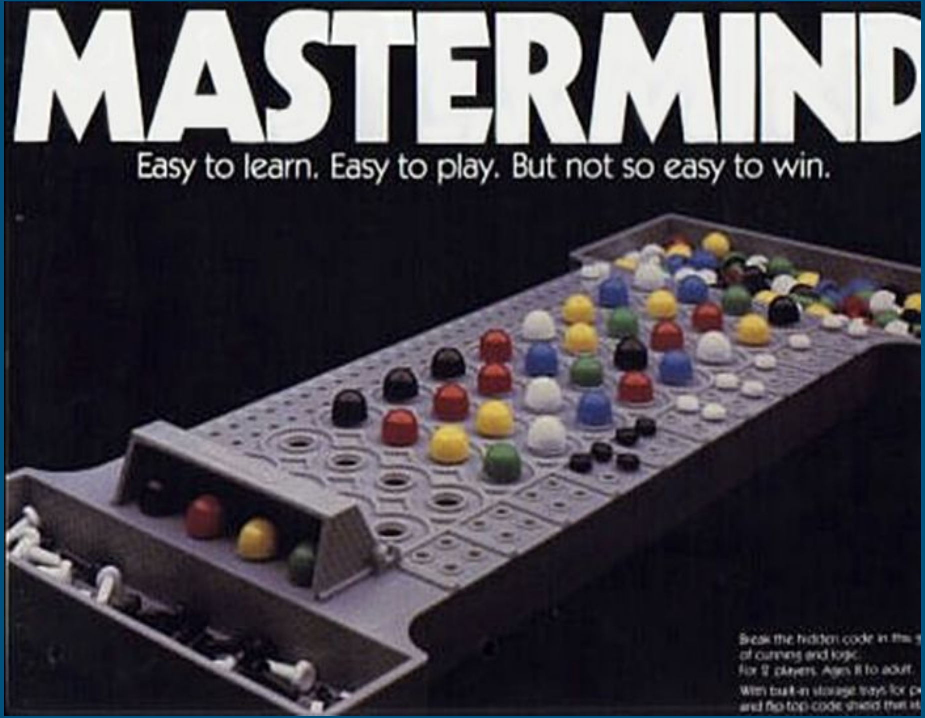
Shreejaya Bharathan

Outline

- Problem introduction
- Reinforcement Learning framing
- Modelling approach
- Interactive Demo
- Results
- What didn’t work
- Future improvements
- Conclusion

Introduction

- Mastermind is a two player code breaking game
- The aim of the game is to guess the (4 digit) code in n - turns


MASTERMIND
Easy to learn. Easy to play. But not so easy to win.
Break the hidden code in this g
of cunning and logic.
For 2 players. Ages 8 to adult.
With built-in storage trays for p
and flip-top code shield that st

Mastermind as an RL problem

- Environment : Mastermind game board
- Agent : Plays a move guessing the pattern
- State : Any combination of 4 colours as a guess
- Feedback : (Number of colors guessed correctly in right position, Number of colors guessed but in wrong position)
- Reward : +1, -1

Modeling Approach

- Existing solution - 5 guess algorithm

- Baseline - random agent

  We implemented two RL frameworks:

- Q learning - Final model 👑
- Policy gradient - Another RL approach

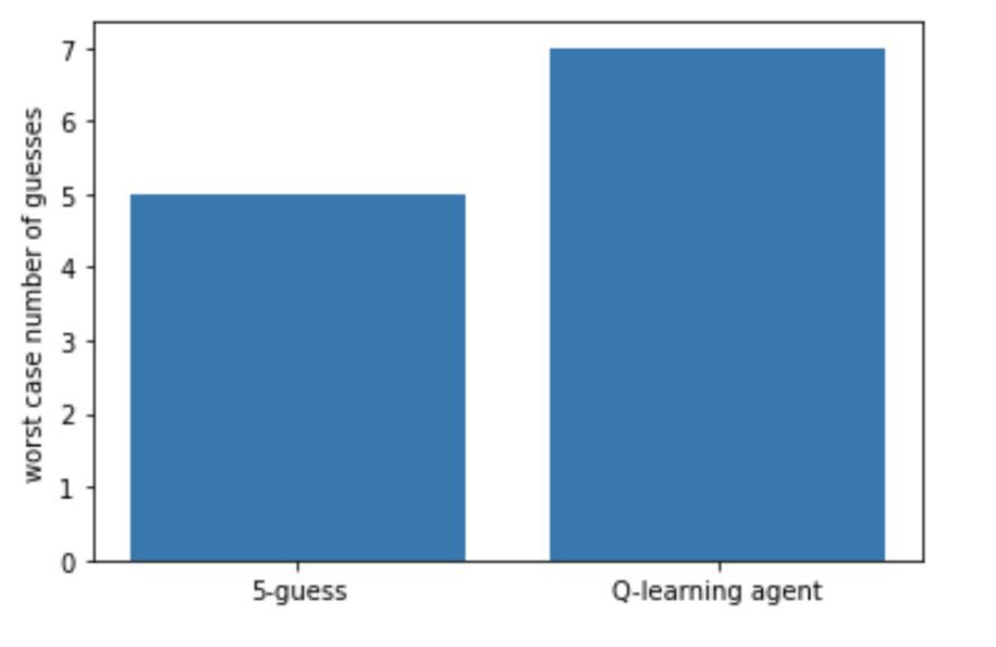
Demo of our final model!

Interactive demo - https://github.com/ShreejayaB/mastermind/blob/master/Interactive_play.ipynb

Q learning code - https://github.com/ShreejayaB/mastermind/blob/master/Q-learning.ipynb

Results

Q learning agent is able to perform close to the popular 5 guess algorithm by Donald Knuth.

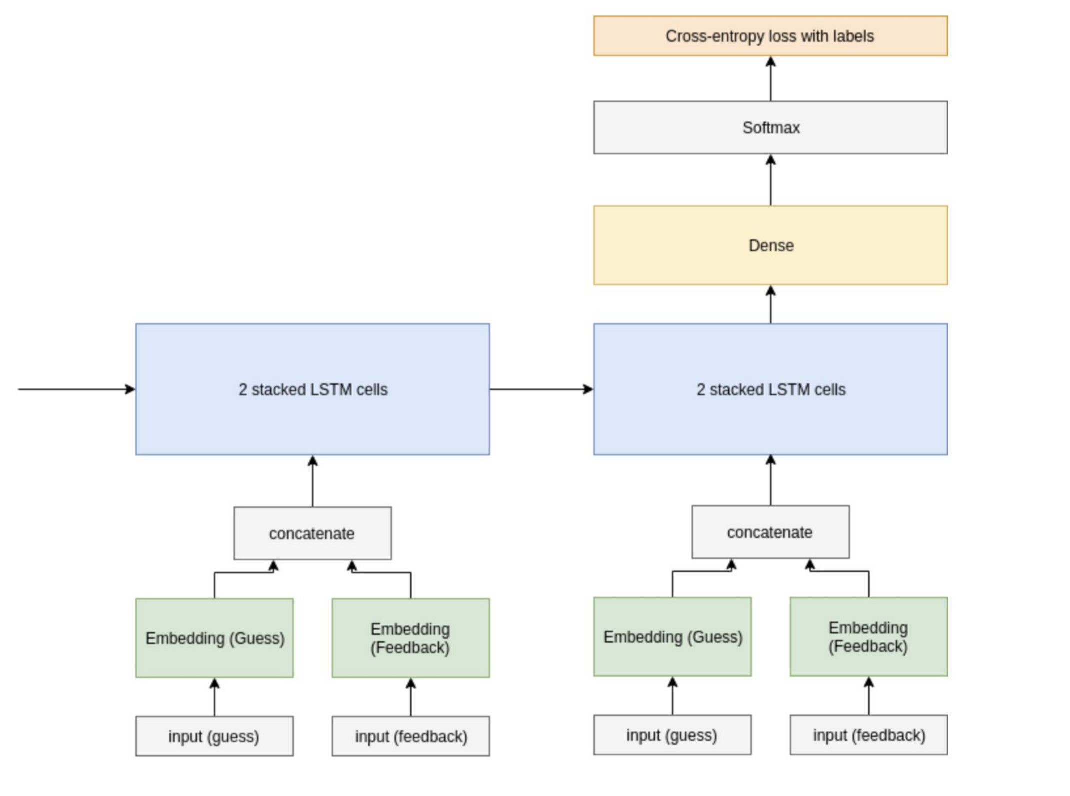


How to interpret our results?

- Initially Q learning was unstable
- After restricting the state space, we obtained a stable algorithm
- The reason for instability is the complex nature of the game - environment, feedback and reward change with each episode based on secret code
- Given the changing nature of the environment, implementing an RL algorithm to outperform the deterministic 5 guess is challenging

Policy Gradient approach & what didn't work

- To learn different game patterns, we used an LSTM network
- The LSTM learns embeddings of the guesses
- This meant more data simulation, making the process VERY slow



Reference - https://github.com/egeromin/mastermind/

Future improvements

- Parallelize the learning process
- Use GRU instead of LSTM
- Simulate the data efficiently
- Decrease the size of embedding of state and action vectors

Conclusions

- Mastermind is a fairly complex reinforcement learning problem
- Q Learning becomes unstable for dynamic environments
- Double Q learning may also help improve performance
- Policy gradients are computationally intensive. Use GPUs!!
- It’s not so easy to win mastermind afterall!

Thank you!