Abstract:

provide a short description of your work and try to convince the reader that your paper is worth reading!

A stand alone condensed version of the article •No more than 250 words; written in the past tense •Uses keywords and index terms

2. Introduction:

<https://davidbarber.github.io/blog/2017/11/07/Learning-From-Scratch-by-Thinking-Fast-and-Slow-with-Deep-Learning-and-Tree-Search/>

Exit is a general strategy for learning and the apprentice and expert can be specified in a variety of ways. In board games Monte Carlo Tree Search (MCTS) is a strong playing strategy[6](https://davidbarber.github.io/blog/2017/11/07/Learning-From-Scratch-by-Thinking-Fast-and-Slow-with-Deep-Learning-and-Tree-Search/#fn:MCTS) and is a natural candidate to play the role of the expert. Deep Learning has been shown to be a successful method to imitate the play of strong players[4](https://davidbarber.github.io/blog/2017/11/07/Learning-From-Scratch-by-Thinking-Fast-and-Slow-with-Deep-Learning-and-Tree-Search/#fn:AlphaGo) which we therefore use as the apprentice.

explain the purpose of your work and motivates it - why is what you are doing important?

Obtain good data for training

What atres decisión trees

Why so important laveled data// games big source of laveled data

Generate a lookbook

- Generally known information about the topic

- Your hypothesis and an overview of the results

- How the article is organized

3. Background:

description of similar efforts done in the past (i.e., literature survey). Discuss any previous work on the topics and go beyond the provided references.

Not the first time done this, go, 1991 already done, a lot of potential

Another sampling data for any problema

Big data

Go game

4. Methodology:

Our search tree will be based on Reinforcement Learning, where the algorithm learns by performing actions and receiving an outcome from a specific environment. Each interaction will give a result that can be seen as a reward, and thus (the action and the outcome) can be used as input data to train the algorithm.

In our case, instead of learning at the same time that we are interacting with the environment, we will first generate all the data needed for the algorithm to train itself. We generate all the possible situations and its outcome before the training process. In order to collect all this information we have to take into account that it has to be fully levelled and has to contain almost, if not all, possible scenarios that we can face.

<https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419>

A good source from where we can obtain this complete information is games, where the information is not hidden and where the rules do not allow a random component. This allows that the methods described below will work with any perfect information game, such as XOX, in our case.

<https://medium.com/swlh/tic-tac-toe-at-the-monte-carlo-a5e0394c7bc2>

The game chosen is XOX (aka Tic-Tac-Toe) since it is easy to model and all the states can be easily explored, so we can provide our algorithm a complete dataset. This dataset, as mentioned before, has to contain a game state and a reward. At first we can pose de data as the final state and if the first player has won or not, thus the data can be collected as easy as generating games and saving them. But this does not give enough information to our tree to be able to predict which move should be done in any intermediate state of the game.

A dataset with all the intermediate states and the best move to be done in every case have to be created. To generate all this we will draw on Monte Carlo Tree Search (MCTS) algorithms which allow finding the best move given a state by expanding all the possible plays and return the one with highest chance to lead towards the victory.

The provided code \*\*\* is a script that performs different games (XOX, Othello…), being the computer both players and using MCTS to do each move. For each turn, given the present state, explores all the game possibilities up to a certain depth and performs a Upper Confidence Bound (UCB) to the present state to determine which movement is better and can lead to winning the game, maximizing your possibilities and minimizing the ones from the opponent.

Since the thinking from both sides is the same and always maximizes its chances, it is normal that most of the games, in the case of XOX, end up in a tie. Yet, that’s not a trouble, because for the data the player who won that particular game is not crucial, only the state and the chosen move is necessary, the algorithm already aimed for the win.

Each state (node) and the decision that the MCTS chooses is saved in an array. The first 9 numbers will represent the XOX board, with value 0 if empty, 1 if there’s an X and -1 if there is an O. The tenth digit represents the position where the algorithm chooses to play. Keeping it all in numbers will help later on for the learning process.

Describe the dataset/s you are going to use, including how the data was collected (or generated).

5. Experiments:

outline any experiments/analyses you will perform and explain the rationale behind them/it. If there are explicit results from other studies, list them here.

The results of the data, compare that all the intermediate states are similar.

Next step with training trees

At the Expert Improvement phase we use the apprentice to direct the MCTS algorithm toward promising moves, effectively reducing the game tree search breadth and depth. In this way, we bootstrap the knowledge acquired by Imitation Learning back into the planning algorithm.

6. Discussion:

explain how you will evaluate the results and how you will gain insights from your experiments.

A well trained tree how should perform

7. Conclusion:

any concluding remarks you might have.

Good data, but random to explore space

- Author name, article title, publication name, publisher, year published, volume, chapter and page number

References

<https://arxiv.org/pdf/1805.05935.pdf>

8. Plan: Provide a breakdown of the work needed to complete the project and how long it will take. Use dates or a Gantt chart. Be realistic about what you can achieve