University of San Francisco

Physics 302/CS 686-06 – Scientific Computation and Machine Learning
Spring, 2023

Final Project

Due: Before the start of the tournament (see the very end)

Artificial Neural Network Alak Player

The purpose of this project is to build an artificial neural network (ANN) algorithm to play the one-dimensional version of the ancient board game Go, or Alak.

We have gone over how to start this project in class. I would like to emphasize the following:

I. Simulating Alak

1. Randomly generate the next legal move
2. Add tests (see below) for checking capture
3. Decide if any pieces need to be removed from the board
4. The opposite side makes its move
5. Repeat until one side wins (the other side should have zero or one piece on the board)
6. Save all the rounds (each round consists of two moves, one by each side) of the simulated game in a pickle file.

II. Training

1. Generate lots of games and then throw away the obviously "dumb" ones (according to (i) and (ii) below) and train using the rest.

   (i) Throw away long games — they likely have lots of purposeless moves;
   (ii) Avoid games that have suicide moves — these are not games to learn from. In fact, once a game has a suicide move, stop playing and discard it.

2. Compute winning rate for the side you have chosen.

3. Feel free to come up with your own training scheme (or you can talk to me) — with training and validation — **no lengthy calculations or "brute-force" allowed: a better player has to be achieved through the ANN "learning" process and only a small amount of simple calculations (e.g., to avoid a move that will results in a kill by the other side).**

Important:

1. You have to have a method in your class definition to test capture. The following a list of six tests that your program has to pass:

   ```
   off_side = ['x', 'x', 'x', 'o', 'o', 'o']

   board_list = ['xoxoxx____', 'xooxooxx__', '__xoo__oxo',
   '_xoxxxoo____', '_x_oxxoo____',  '_xoxx___x_']
   ```

   Here:

   `off_side` — designates the offensive side, i.e., the side that has just made the move. Thus the `off_side` is `'x'` for the first three moves and `'o'` for the next three moves.

   `board_list` — each entry shows the board after the move by `off_side`.

   For every move, a test passes if the resulting broad (as determined by you coded) is the same as the corresponding entry in this list:

   ```
   board_expect = ['x_x_xx____', 'x__x__xx__', '__xoo__o_o',
   '_xo___oo____', '_x_o__oo____', '_x_xx___x_']
   ```

   These tests include:

   - simple kill
   - double kill
   - double kill that involves more than one piece
   - a suicide move

2. The tournament will be played interactively: you have to enter your opponent's move interactively and then your ANN-trained player will respond.

3. The tournament will be — tentatively — on **Tuesday, May 16, 2023, 4 - 6 PM.**