Technical University of Crete

COMP412

Copenhagen Hnefatafl using Q-learning

Nikolaos Pnevmatikatos - 2017030002

Introduction

Copenhagen Hnefatalf is a scandinavian board game belonging under the wider umpbrella of tafl games. It is a perfect information, asymmetric, zero-sum game where the defender must get their king piece to one corner of the board, and the attacker must prevent them from doing so.

Rules

The game rules are somewhat simple, though this doesn't make playing it easy. Each player can move any of their pieces along the vertical & horizontal axes (like a rook in chess). A piece can be captured (whereupon it is removed from play) by placing two oppising pieces adjecent to it, and opposite each other, additionally there are 5 "hostile tiles" (the four corners and center tile (if unoccupied), which count as hostile to any piece & may take part in a capture. The center tile may only be occupied by the king but other pieces may pass through it. The board edge doesn't count as hostile.

The win conditions for each player are as follows:
The defener must get their king (center piece) to one of the four corners of the board, alternatively, they may create an "exit fort". An exit fort is an arrangement of pieces that fulfils the following conditions:

1. The king must be on the edge of the board.
2. The king must be surrounded by defenders.
3. The surrounding defenders must be impossible to capture.

Conversly, the attacker must capture the king by placing four pieces adjecent to him. The king may not be captured on the edge of the board. If the king reaches the edge of the board, the attacker must capture every other defender piece in order to win.

This is not an exhaustive list of the rules of Copenhagen Hnefatafl, but it will suffice for the purposes of this report. A full list of rules may be found here.

The model

Since the game is played in a 11x11 board, using tabular qlearning with a state-board representation is impractical (if not impossible). Instead, we used a linear approximation function with the following features:

1. Bias
2. Own piece count
3. Oppponent piece count
4. Own mobility (number of possible legal moves)
5. Opponent mobility
6. Opponent piece loss
7. Move distance
8. King progress (Manhatan distance to the nearest corner)
9. King threat (attacker pieces adjecent to the king)
10. Exit fort[1]
11. Winner

Training

Because of the inherent assymetry of the game two models are required, one for the attacker & one for the defender. The agents were trained against a random agent (an agent that plays random legal moves) for 40K games (defender) and 12K games (attacker)[2].

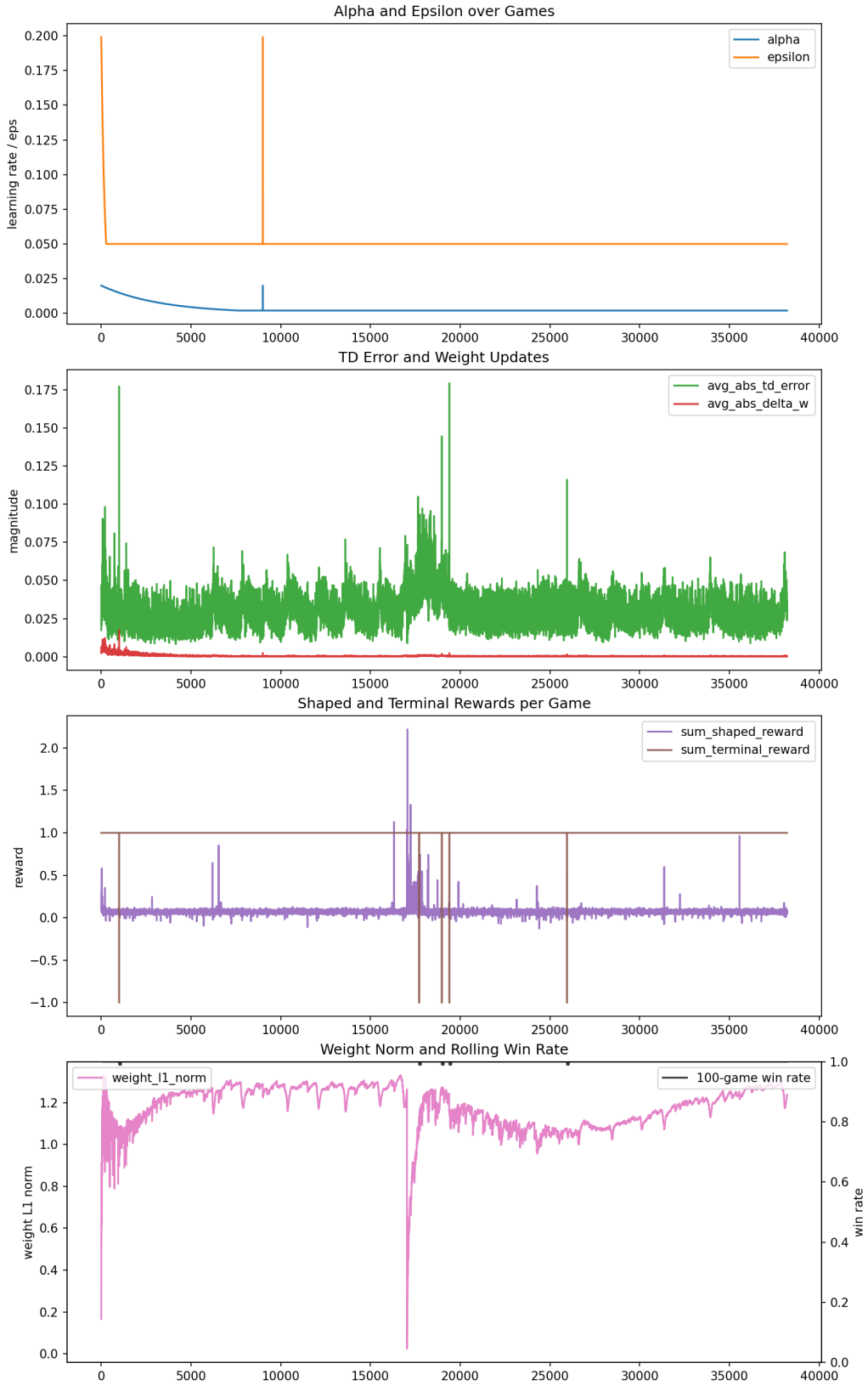
After the original training phase, the two models were pitted against each-other for another 10K games using a “frozen” opponent. This means that the opponent model didn’t train during this process, instead, a random checkpoint from the previous training session was used for every game.

Below are the learning metrcis for the first learning phase.

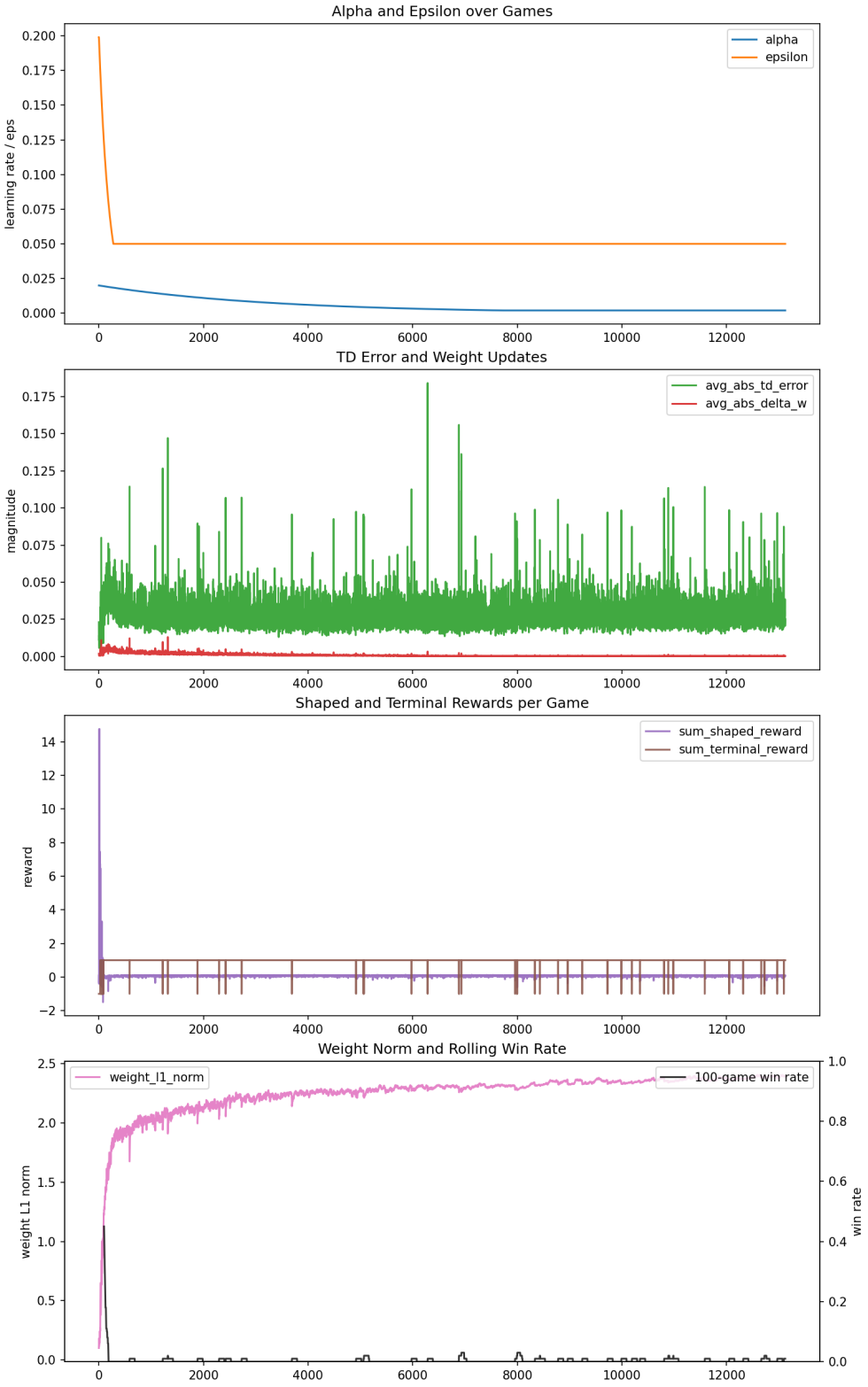
1 Nonzero if the king is on the edge, and increases as more friendly pieces exist in a 2-tile radius of the king. Further increases if those pieces cannot be captured.

2 Because the attacker win condition is more complicated to achieve than that of the defender, training was substantially slower, leading to time constraints.

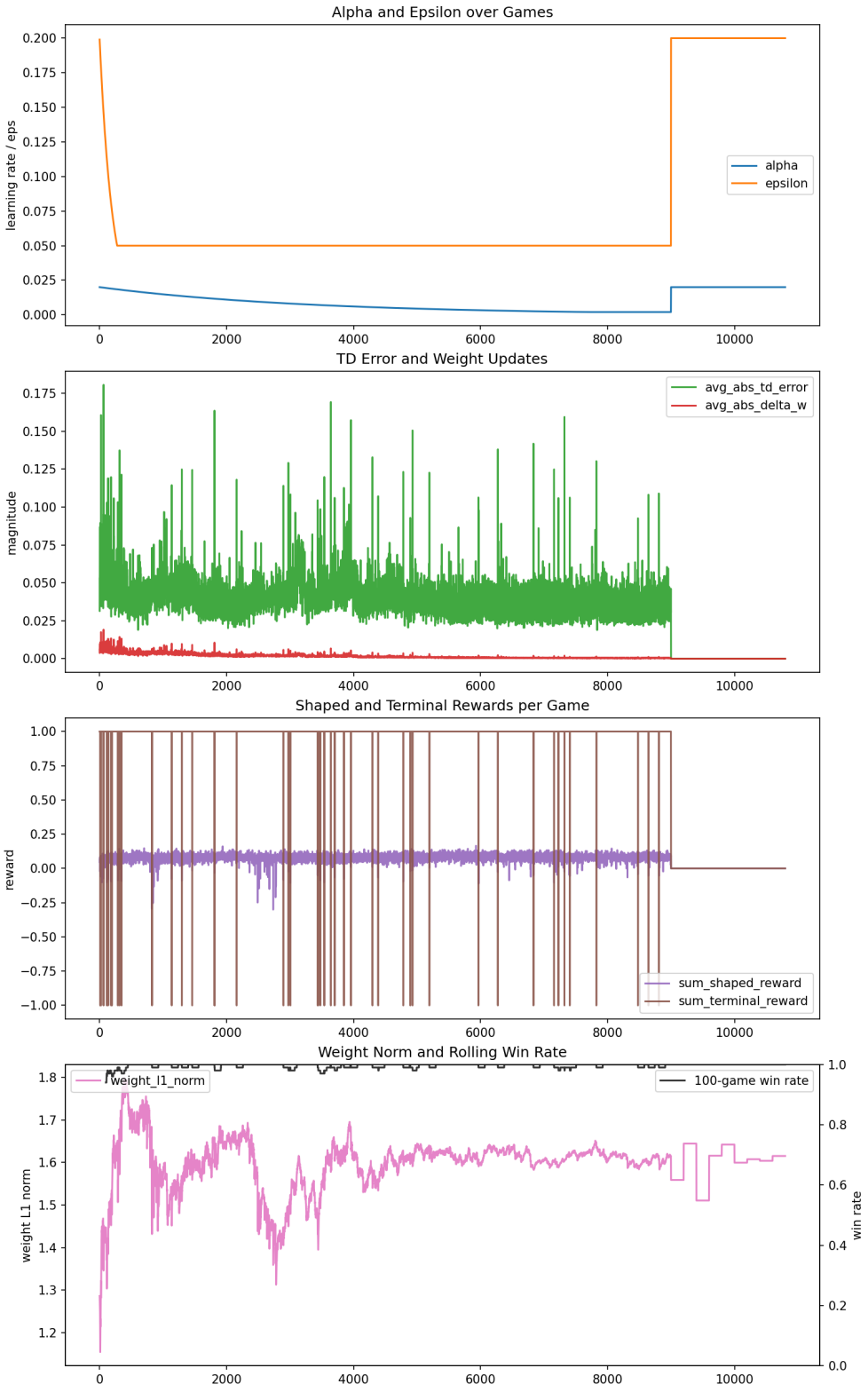
Defender 1st Phase:



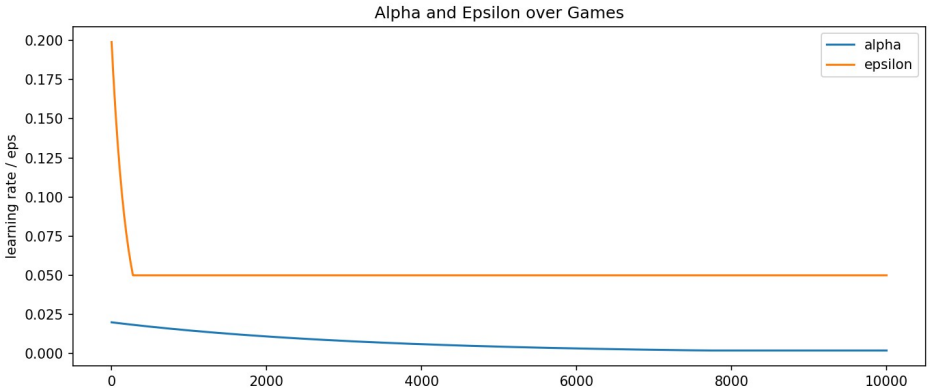
Attacker 1st phase:

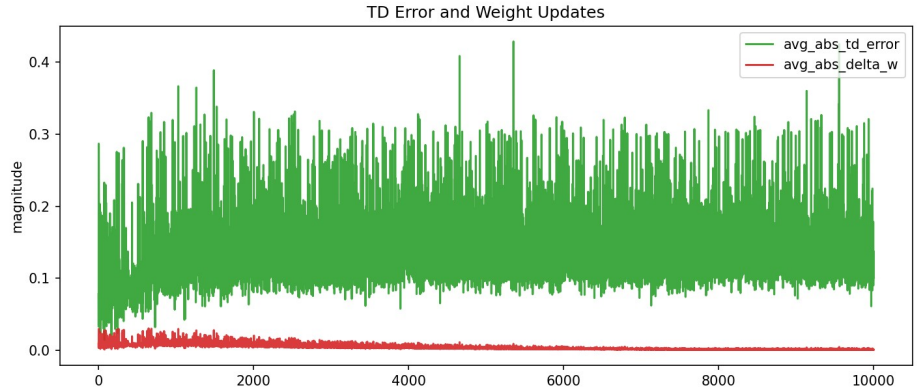


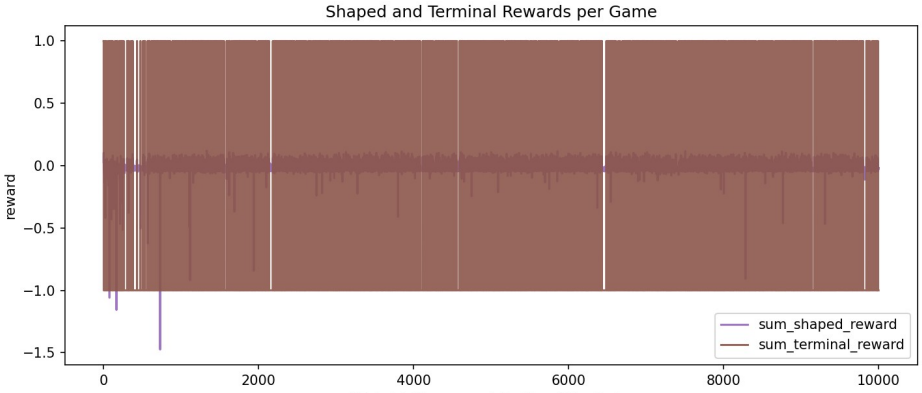
Defender 2nd Phase:

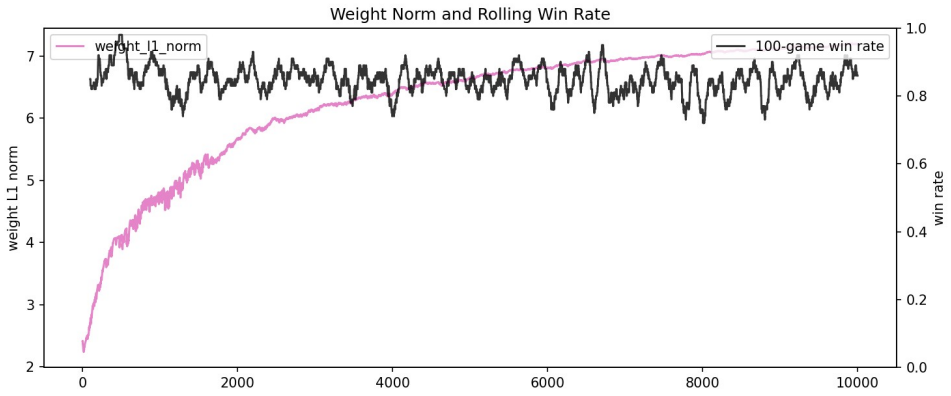


Attacker 2nd Phase:





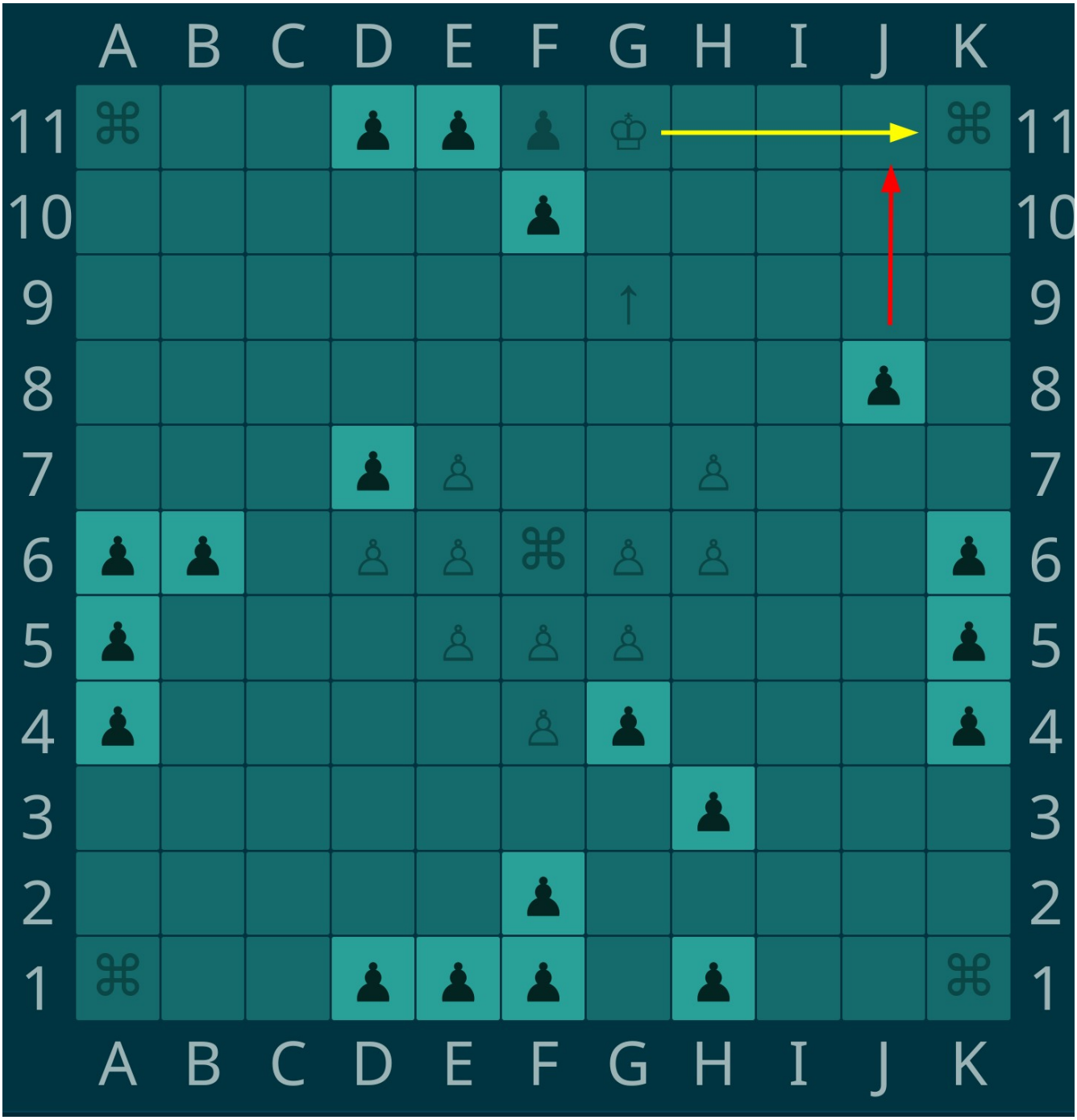




Lastly, the two models were evaluated by playing against them, both manually and with a minimax search model with depth 2.

Limitations & Future considerations

Upon evaluating the models it because aparent that the feature vector we chose was woefully inadequate for the task at hand. Consider the following position:



For a human (or a minimax search agent), the best move is trivially easy to determine. The attacker MUST block the kings escape, otherwise the game will be lost on the very next move. However, since the “king threat” feature wouldn’t be increased, the attacker will likely choose to capture another piece instead, and subsequently lose. Similar issues exist from the perspective of the defender. These issues may be alleviated by using a richer feature vector, for example we could add a feature representing possible escape paths for the king. Of course this would require re-training from scratch, so we leave it as an exercise for the reader.

Another limitation, though not as significant, was time. Because of the linear nature of the training process and the inherent delay in the communication between the two players and the engine, the training process was rather slow (30K games took ~2days, not counting runtimes errors). Ideally we would have trained the model on a higher number of games (100K games initially, 50K games with frozen opponents), unfrotunately this would have taken a prohibitively long amount of time.