19CST302-

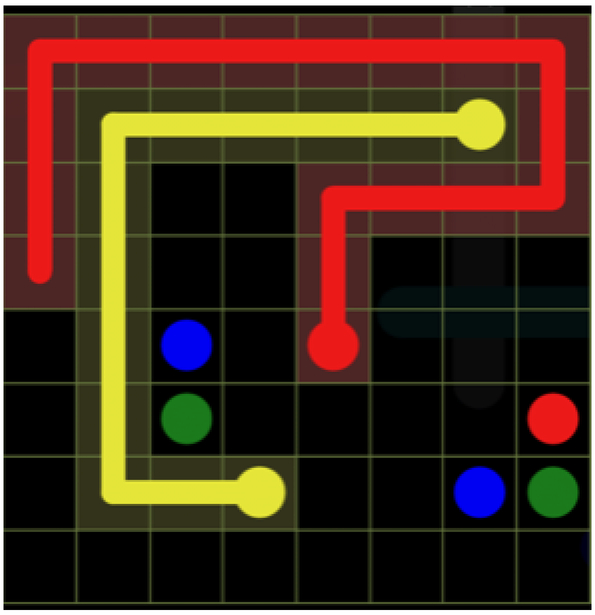
Neural Networks and Deep Learning

Approximate Q-learning

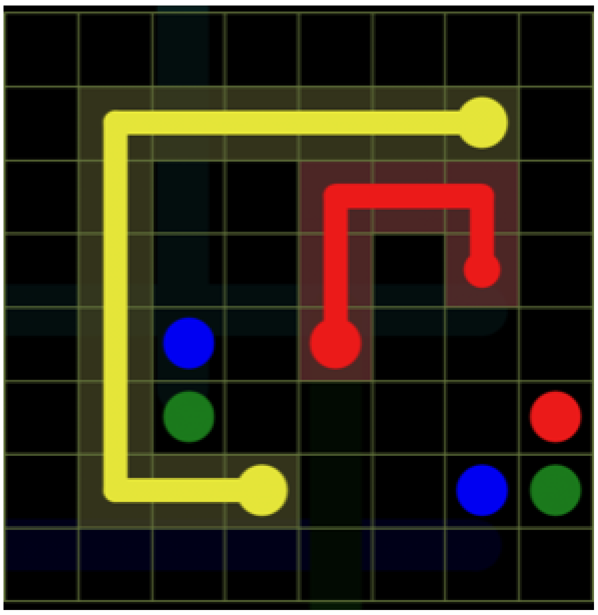
The applications of approximate Q-learning are fascinating, particularly in Games. Rather than the agent deciding a best action in a given state, the idea is to give it some intuitive features that it can quickly calculate at each step, independently of the exact state (configuration of the board), and let it decide which ones are the most important. This can be a game-changer in games like Pacman (e.g. decide your next move based on the distance to the nearest pellet and to the nearest ghost, rather than an action for every single possible state), or of course Flow Free, where the number of possible states is simply too large for exact Q-learning to be effective.

The trade-off is that it’s now on the developer to pick the "right" features. I narrowed the list down to features I knew were important in many cases (e.g. total remaining Manhattan distance to close each path), and some I suspected were important (but had no way to prove), which I would just let the algorithm figure out. These include:

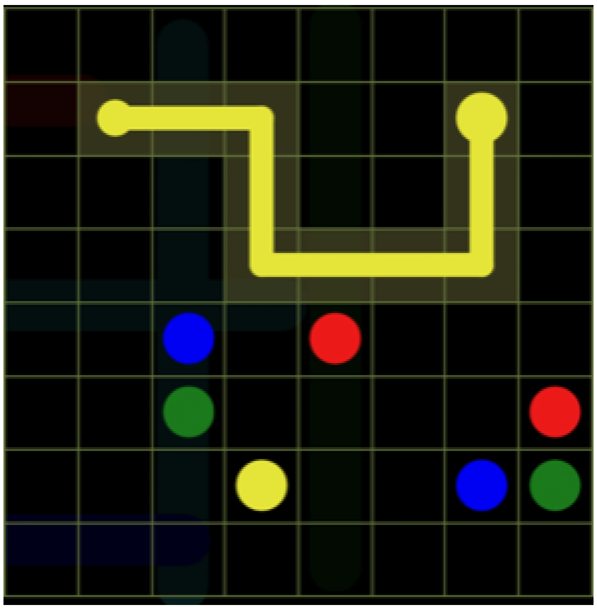
- Total remaining Manhattan distance to close each path
- Number of "turns"
- Number of "boxes"
- Number of valves in boxes
- Number of boxes with no valves (one can prove logically that this should never happen – I wanted to see whether the algorithm could figure it out)
- Whether a path is "hugging a wall" (i.e. if a path can stick to another adjacent path)



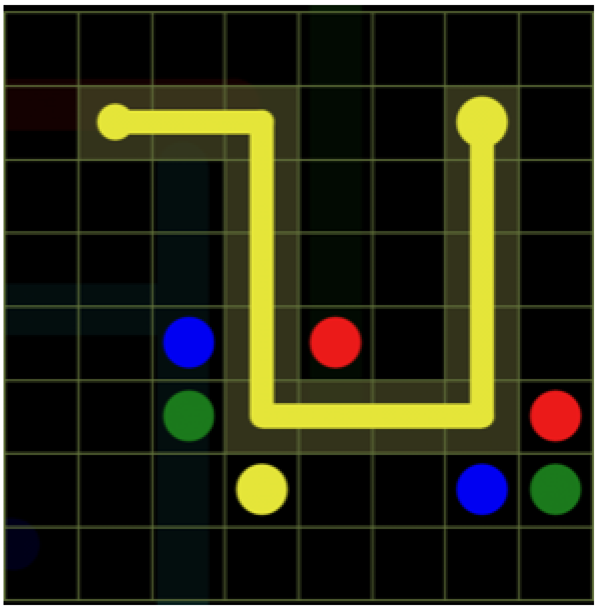
Red is “hugging” yellow and the wall



Red is not hugging yellow or the wall



A “box” with no valves



A “box” with 1 valve