Process:

Solution: to replace the Q table with a Network. train the Network with the states as input  and updated Q value as target.  The way we get Q[s.a] here is by input the states to a network and get the Q value as output.

* Alpha = 0.2, Gamma = 0.8, Rar = 0.999, Radr = 0.9999
* rar and rar\_decay factors. Doc says that for details refer to mc2p3 doc.

I am trying to use following approach (based on suggested paper in 2nd link provided on the wiki page for using Q(s, a, weights)- This was the breakthrough work done at Deepmind with Deep Q Learners trained to play Atari games back in 2013.

Pseudo code is on page 5.

1. Create one NN for each state with state coordinate as input and four actions as possible output. For back-propagation, use reward for each state to adjust the wight values
2. Initialize the weights of NN by random values [-1,1]. Keep one copy of NN weights for each state
3. For each iteration from Start to End position, updates the state's weight while moving from Goal to the start states.
4. Once trained, then each NN should process the given its own state and provide one of the posible action as output

Deep learning course: <http://course.fast.ai/index.html>

The optional textbook (Mitchell) Chapter 4 has a good overview on neural networks

Basic Video: <https://youtu.be/bxe2T-V8XRs>