𝕄onte ℂarlo 𝕋ree 𝕊earch

MCTS builds a statistics tree (detailing value of nodes) that *partially* maps onto the entire game tree.

Statistics tree guides the AI to “*look only / mostly at* ***the most interesting nodes*** *in the game tree*”.

Value of nodes determined by simulations.

→ Definition

* N: the number of times an action “*a*” has been explored from the state “*s”*.
* W: the total value of the next state. Every time we hit a leaf node, we query the neural network and look up the value of that state on backup. We then add the leaf nodes value to the W of each node on the path back to route.
* Q: the mean value of the next state (W/N).
* P: the prior probability of selecting an action “*a*”, this is cord from the neural network policy head every time we hit a leaf node.
* Stochastically: sample randomly from the probability distribution (training).
* Deterministically: choose the action with the greatest N (competition).
* Brute Force: FILL.

**Steps**

1. Choose an action that maximises Q + U (**SELECTION**).

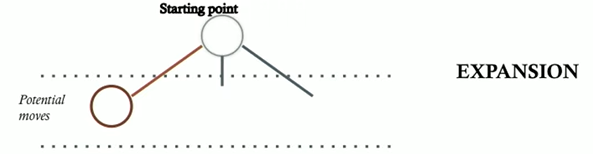
⇒ U is a function of P and N that increases if an action has not been explored much, relative to the other actions or if the prior probability of the action is high.

1. Continue until a leaf node is reached (**EXPANSION**).

⇒ The game state of the leaf node is passed into the neural network which output predictions about two things:

P: move probabilities.

V: value of the state (for the current player).

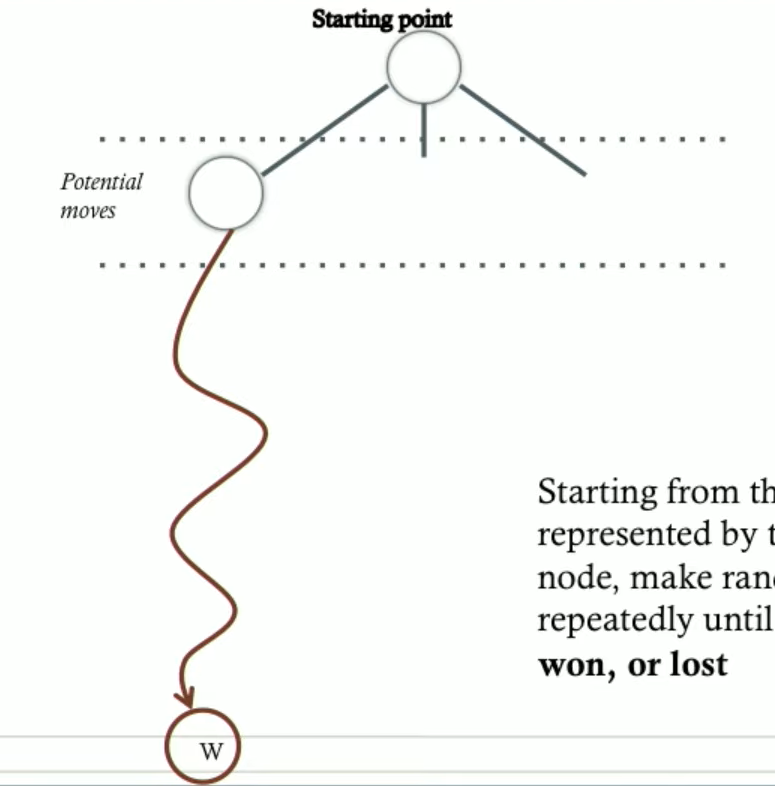


**Add a new node** to the stats tree, representing a position in the game that the AI will « investigate » (how good the move is) next.

1. Backup previous edges (**UPDATE**).

⇒ Each edge that was traversed to get to the leaf is updated as follow.

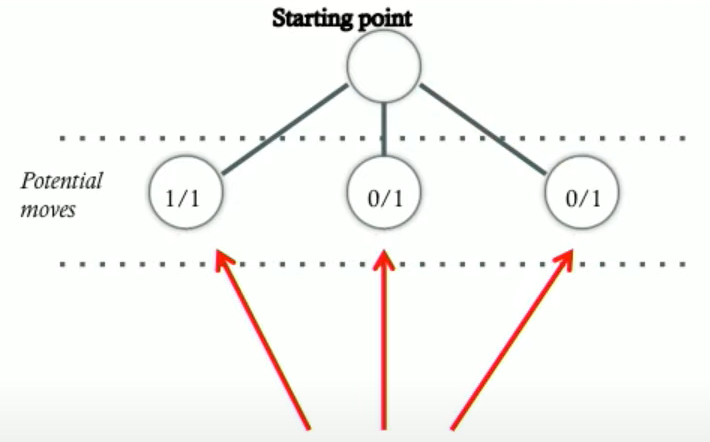
N → N + 1



W → W + V

Q = W / N

\* Repeat these steps a lot of times (1600 times?) to increase the accuracy of probabilities.

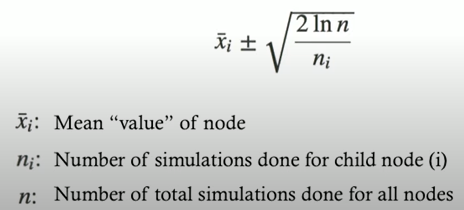


#’s say **left node** is the *better move*, but chances are, these numbers (produced by a single random simulation) are probably not a good indicator of how good any of the moves are!

⇒ More simulations will make them more accurate!

After that, all child nodes have now been visited at least once. Now AI can **select** which child node to be **investigated further**. (*the higher the value, the “better” the move is*).

The next selection gets based on two things:



* How good are the stats?
* How much has child node been ignored?

⇒ These simulations can go on for as long as you want. The more simulations there are, the bigger the stats tree, and the more accurate the node values are

♠ Select a move.

→ Choose the action with the greatest N (deterministically).

→ Sample randomly from the probability distribution (stochastically).

⇒ The selected state becomes the new root node. Retain all leaves in the search tree stemming from the chosen move and discard the rest. Continue repeating all the above until the game is over.

