Now let’s pass to adversarial situations where the AI has to take intelligent choices to win against an opponent. For example, in the game tic tac toe, we can make the AI play as X player and we are the O player. The AI does not have to find a solution to exit a maze, solve a puzzle or something similar. It has to make smart decision depending on the situation to win against the player.

-Minimax: this is the algorithm we are going to review, an algorithm used where there is an opponent of the AI. Since PCs can understands the meaning of Win or Lose (True or False) we can assign values to different end game situations. 0 wins = -1, No One wins = 0, X wins = 1.   
Let’s call the 2 players in a different way: The X player is called Max Player and the O one is called Min Player. This is because in the Minimax algorithm the Max player aims to maximize the score and the Min player aims to minimize the score (Max player, which is the X player wants to reach the case = 1 (the higher one) where X wins. Min player, O player wants to reach the case = -1 (the lowest one) where O wins.   
If Max player cannot get the 1 case, it will prefer to get the 0 instead of the -1).

What does the algorithm need to encode?

*S*0: the initial state

Player(*s*): a function that return whose turn is, so the AI knows what to do

Actions(*s*): return legal moves in *state s*

Result(s, a): where *s* stands for *state* and *a* for *action*. Returns the result of an action applied on current state.

Terminal(*s*): checks if *state* s is a “goal state”, checks if the game’s ended

Utility(*s*): returns a numerical value for *terminal state* to determine who won.

S0 (an empty tic tac toe field).

Player(s): if the number of Xs and Os are the same it’s Max player’s turn, else Min player’s turn. In the initial case, since X and O = 0 it’s Max player’s turn.

Action(s): in the initial state, since it’s X turn, the result of this function will be an array with an X in all the position

Result(s, a): a is the result of Action(s) and the AI decides to use the action that puts the X in the middle. The result is a “sum” of current state and action and the result is the field with an X in the middle.

Terminal(s): if AI runs this function with the current state (only an X in the middle), the result will be a False because the game’s not finished. If we take a board where there are 3 Xs/Os on the same line or the board is totally full, the result is a True..

Utility(s): works with the result of Terminal(s) function. If the Max player won it gives to the state a value of 1, if the board is totally filled and no one won it assigns to this state the value of 0. If Min player won it assigns to the state the value of -1.

So, how does the AI choose the move?

Immagine che contiene tavolo

Descrizione generata automaticamente

This is the current state  
and the Player(s) function  
says that is Min player’s  
turn. Action(s) says that   
the MinP can put an O in  
the top left corner  
and in the middle   
bottom square.

Now the AI asks itself: what would my opponent do if a put the O in the top left corner? The value will be 1. What if a follow the other action? The value will be 0, so it follows the second action. So, if the MinP puts the O in the top left corner, the maximum value that can be reached with that state is 1, the worst. If we put in the bottom middle square the maximum value will be 0, better than 1. The AI can choose between an option that will result 1 or 0, and since MinP wants the lowest value, it will choose the state that will be 0.