

Reinforcement Learning Algorithms used in Tic Tac Toe

November 17, 2019

1 TD(0) Learning

For episode = 1, M **do**

Initialize a fresh game

For t = 1, T **do**

Play move $a_t = \begin{cases} \text{random move} & \text{with probability } \epsilon \\ \operatorname{argmax}_a \tilde{V}(\operatorname{succ}(s_t, a), \theta) & \text{for white} \\ \operatorname{argmin}_a \tilde{V}(\operatorname{succ}(s_t, a), \theta) & \text{for black} \end{cases}$

Receive reward r_t

Store the transition (s_t, s_{t+1}, r_t, a_t) in the replay buffer D

Sample a random minibatch transition from D

Set the TD-target: $y_t = \begin{cases} r_t & \text{if game terminates at step } t + 1 \\ r_t + \gamma \tilde{V}(s_{t+1}, \theta) & \text{otherwise} \end{cases}$

Perform a stochastic gradient decent step on $[y_t - V(s_t, \theta)]^2$ with respect to θ

Every c steps set $\tilde{V} = V$

End For

End For

γ is the discount factor and θ the neural network parameters.

2 TD(λ)

procedure REFRESH(l)

For transition $(s_t, s_{t+1}, r_t, R_t^\lambda, a_t) \in l$ processing back-to-front **Do**

If terminal(s_{t+1}) **Then**

 Update $R_t^\lambda \leftarrow r_t$

Else

 Get adjacent transition $(s_{t+1}, s_{t+2}, r_{t+1}, R_{t+1}^\lambda, a_{t+1})$ from l

 Update $R_t^\lambda \leftarrow r_t + \gamma[\gamma R_{t+1}^\lambda + (1 - \lambda)V(s_{t+1}, \theta)]$

End If

End For

End procedure

For episode = 1, M **do**

 Initialize a fresh game

For t = 1, T **do**

 Play move $a_t = \begin{cases} \text{random move} & \text{with probability } \epsilon \\ \operatorname{argmax}_a V(\operatorname{succ}(s_t, a), \theta) & \text{for white} \\ \operatorname{argmin}_a V(\operatorname{succ}(s_t, a), \theta) & \text{for black} \end{cases}$

 Receive reward r_t

 Append the transition $(s_t, s_{t+1}, r_t, R_t^\lambda, a_t)$ to L , where R_t^λ is arbitrary

If terminal(s_{t+1}) **Then**

 REFRESH(L)

 Store L in D

End If

 Sample a random minibatch transition from D

 Perform a stochastic gradient decent step on $[R_t^\lambda - V(s_t, \theta)]^2$ with respect to θ

 Every c steps REFRESH(D)

End For

End For

γ is the discount factor and θ the neural network parameters.

3 Q-Learning

```

For episode = 1, M do
  Initialize a fresh game
  For t = 1, T do
    Play move  $a_t = \begin{cases} \text{random move} & \text{with probability } \epsilon \\ \operatorname{argmax}_a \tilde{Q}(s_t, a, \theta) & \text{for white} \\ \operatorname{argmin}_a \tilde{Q}(s_t, a, \theta) & \text{for black} \end{cases}$ 
    Receive reward  $r_t$ 
    Store the transition  $(s_t, s_{t+1}, r_t, a_t)$  in the replay buffer  $D$ 
    Sample a random minibatch transition from  $D$ 
    Set the target:  $y_t = \begin{cases} r_t & \text{if game terminates at step } t+1 \\ r_t + \gamma \max_{a'} \tilde{Q}(s_{t+1}, a', \theta) & \text{otherwise} \end{cases}$ 
    Perform a stochastic gradient decent step on  $[y_t - Q(s_t, a_t, \theta)]^2$  with respect to  $\theta$ 
    Every  $c$  steps set  $\tilde{Q} = Q$ 
  End For
End For

```

γ is the discount factor and θ the neural network parameters.

4 DQN(λ)

procedure REFRESH(l)

For transition $(s_t, s_{t+1}, r_t, R_t^\lambda, a_t) \in l$ processing back-to-front **Do**

If terminal(s_{t+1}) **Then**

 Update $R_t^\lambda \leftarrow r_t$

Else

 Get adjacent transition $(s_{t+1}, s_{t+2}, r_{t+1}, R_{t+1}^\lambda, a_{t+1})$ from l

 Update $R_t^\lambda \leftarrow r_t + \gamma[\gamma R_{t+1}^\lambda + (1 - \lambda)\max_{a'} Q(s_{t+1}, a', \theta)]$

End If

End For

End procedure

For episode = 1, M **do**

 Initialize a fresh game

For t = 1, T **do**

 Play move $a_t = \begin{cases} \text{random move} & \text{with probability } \epsilon \\ \operatorname{argmax}_a Q(s_t, a, \theta) & \text{for white} \\ \operatorname{argmin}_a Q(s_t, a, \theta) & \text{for black} \end{cases}$

 Receive reward r_t

 Append the transition $(s_t, s_{t+1}, r_t, R_t^\lambda, a_t)$ to L , where R_t^λ is arbitrary

If terminal(s_{t+1}) **Then**

 REFRESH(L)

 Store L in D

End If

 Sample a random minibatch transition from D

 Perform a stochastic gradient decent step on $[R_t^\lambda - Q(s_t, a_t, \theta)]^2$ with respect to θ

 Every c steps REFRESH(D)

End For

End For

γ is the discount factor and θ the neural network parameters.

5 AlphaZero

5.1 Monte-Carlo Tree Search (MCTS)

5.1.1 Upper Confidence Bound

$$U(s, a) = Q(s, a) + \sqrt{\frac{2 \ln \sum_b N(s, b)}{1 + N(s, a)}}$$

$U(s, a)$ is the upper confidence bound for the current state s and action a

$Q(s, a)$ is the expected reward by taking action a in state s

$N(s, a)$ is the number of times we took action a from state s

$\sum_b N(s, b)$ is the total number of plays from state s

5.1.2 Upper Confidence Bound Alpha Zero

$$U(s, a) = Q(s, a) + c_{puct} P(s, a) \sqrt{\frac{\sum_b N(s, b)}{1 + N(s, a)}}$$

$U(s, a)$ is the upper confidence bound for the current state s and action a .

$Q(s, a)$ is the expected reward by taking action a in state s .

c_{puct} is a constant that controls the amount exploration

$P(s, a)$ probability to take action a in state s as predicted by the neural network

$N(s, a)$ is the number of times we took action a from state s

$\sum_b N(s, b)$ is the total number of plays from state s

5.1.3 Alpha Zero Tree Search

```

procedure SEARCH( $s$ )
  If terminal( $s_t$ ) Then
    Return  $r_t$ 
  End If

  If not exists( $P(s, \cdot)$ ) Then
    predict  $P(s, \cdot)$  and  $v(s)$  with the neural network
     $N_s(s) = 0$ 
     $Q(s, a) = 0$  for all  $a$ 
     $N(s, a) = 0$  for all  $a$ 
    If player == BLACK Then
      Return  $-v(s_{t+1})$ 
    Else
      Return  $v(s_{t+1})$ 
    End If
  End If

   $U(s, a) = Q(s, a) + c_{puct}P(s, a)\frac{\sqrt{N_s(s)}}{1+N(s, a)}$  for all  $a$ 
   $a_t = \operatorname{argmax}_a U(s, a)$ 
  Execute  $a_t$  to get next state  $s_{t+1}$ 
   $v(s_{t+1}) = \text{SEARCH}(s_{t+1})$ 

  If player == BLACK Then
     $v = -v(s_{t+1})$ 
  Else
     $v = v(s_{t+1})$ 
  End If

   $Q(s, a) = \frac{N(s, a)Q(s, a) + v}{N(s, a) + 1}$ 
   $N(s, a) = N(s, a) + 1$ 
   $N_s(s) = N_s(s) + 1$ 
  Return  $v$ 
End procedure

procedure MCTSAZ( $s$ )
  For simulation = 1, M Do
    SEARCH( $s_t$ )
  End
  Return  $N(s, a)$ 
End procedure

```

5.1.4 Training Algorithm

```

For episode = 1, M Do
  For t = 1, T Do
    Initialize  $N_s$ ,  $N$ ,  $Q$ ,  $U$  and  $P$ 
    Initialize a fresh game
     $N(s_t, a) = \text{MCTSAZ}(s)$ 
    If  $temp == 0$  Then
       $a_t = \text{argmax}_a N(s, a)$ 
       $P(s_t, a) = \begin{cases} 1 & \text{for } a_t \\ 0 & \text{otherwise} \end{cases}$ 
    Else
       $P(s_t, a) = N(s, a)^{\frac{1}{temp}}$ 
       $P(s_t, a) = \frac{P(s, a)}{\sum_b P(s_t, b)}$ 
    End If

    Append the training example  $(s_t, P(s_t, a), v_t)$  to  $L$ , where  $v_t$  is arbitrary
    Pick action  $a_t$  by sampling from  $P(s_t, a)$ 
    Play move  $a_t$ 
  End

  Observe the final reward  $r_T$  of the game
  For training example  $(s_t, P(s_t, a), v_t) \in L$  Do
    If player == WHITE Then
      Update  $v_t \leftarrow r_T$ 
    Else
      Update  $v_t \leftarrow -r_T$ 
    End If
  End
End

```