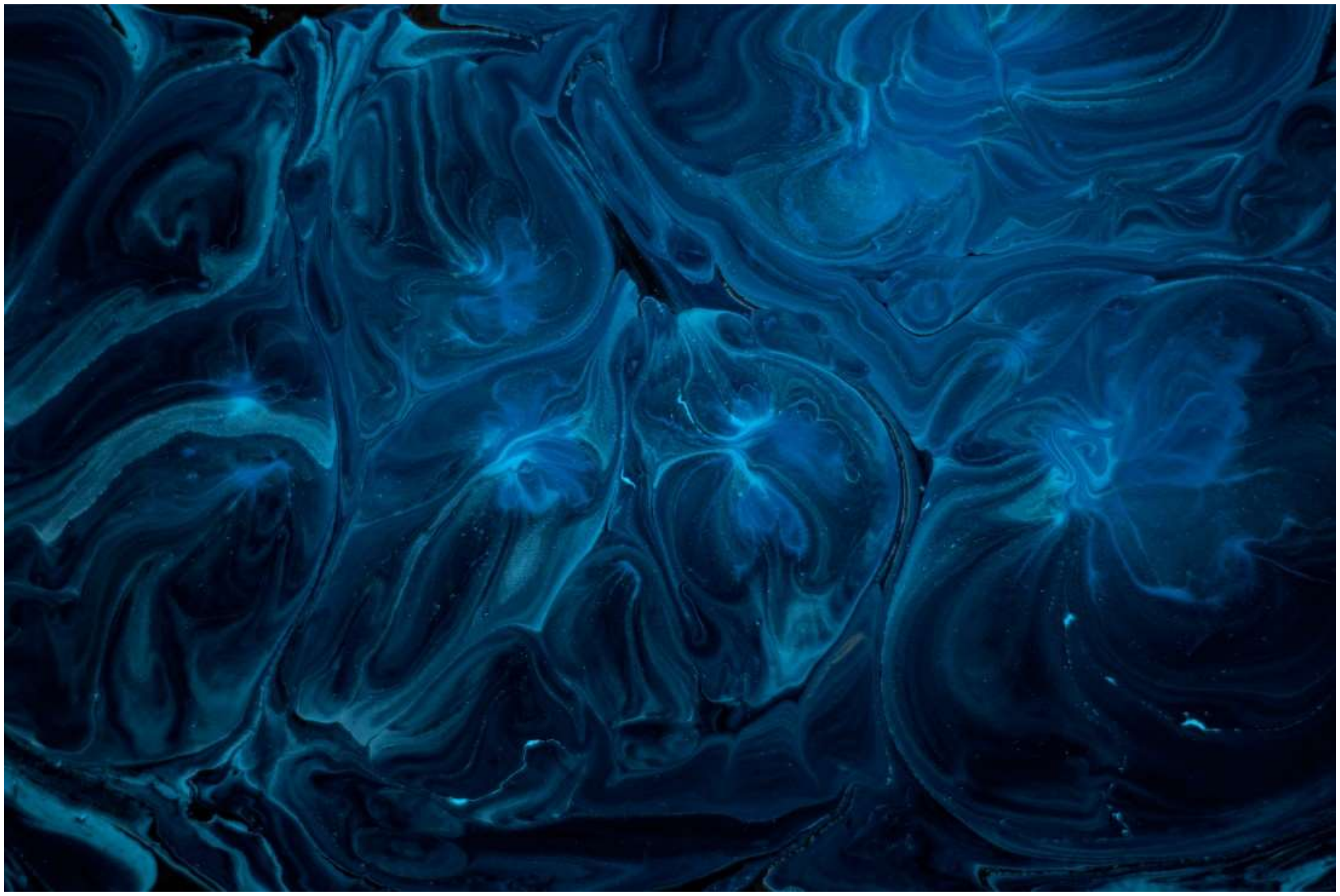
Ziad SALLOUM Follow

Dec 30, 2019 · 5 min read · · Listen

Neural Fictitious Self-Play

Deep Reinforcement Learning in Imperfect-Information Games



Introduction

This article is based on a scientific paper by Heinrich & Silver that introduces the first scalable end-to-end approach to lear 9 ate Nash equilibria **without prior domain knowledge**.

Important Reminders

- Fictitious Play: is an iterative method that finds Nash Equilibrium in Two Player Zero Sum game. Its problem is that it applies to Normal Form Games which are tabular and do not capture time or sequence. Details can be found in the article "Introduction to Fictitious Play".
- Fictitious Self Play: is a method that fixes the problem of the Fictitious Play by integrating time/sequence by using Extensive Form Game. It also uses Reinforcement Learning to find an approximation of the best response and Supervised Learning to update the average strategy. It is proven that it can converge to a Nash Equilibrium. More details in the article "Fictitious Self Play"

Imperfect Information Games

In imperfect information, players are simply unaware of the actions chosen by other players. However they know who the other players are, what their possible strategies/actions are, and the preferences/payoffs of these other players. Hence, information about the other players in **imperfect information is complete**.

On the other hand incomplete information games, players may or may not know some information about the other players, e.g. their "type", their strategies, payoffs and their preferences.

Chess is an example of a game with **perfect information** as each player can see all the pieces on the board at all times. Other examples of games with perfect information include tic-tac-toe, checkers, infinite chess, and Go.

Card games where each player's cards are hidden from other players such as poker and

the probability of choosing a data item is 1/N, even though N is unknown and the memory size is limited and less than N, so not all the stream data could be buffered prior to sampling.
In summary, the algorithm works such as for the ***ith*** item, we spin a wheel with ***i*** slots containing only one winning slot. If the winning slot is selected then we take the item in the sample. This method can be proven to have 1/N probability for each item to be selected.

Anticipatory Dynamics

Anticipatory Dynamics is a way to enable qualitative changes in the stability of learning game dynamics. In the case of general-sum multiplayer games, this means convergence to mixed strategy Nash equilibrium. More details can be found in "Anticipatory Learning in General Evolutionary Games" paper.

The idea behind this technique is to introduce an **anticipated correction** to the action. The easiest way to understand it, is to imagine shooting at a moving target. You **correct** your shot by aiming at some distance in front of the target.

So you introduce an anticipatory correction parameter. In the case of Neural Fictitious Self Play, this parameter is a probability called "η" .

Neural Fictitious Self-Play

Neural Fictitious Self Play (NFSP) introduces Neural Network to Fictitious Self Play.
It uses two independent networks Q(s, a | θ(Q)), and Π(s, a | θ(Π)) and two memory buffers Mrl and Msl assigned to each of them respectively:

- Mrl is a circular buffer that stores agent experience in the form of

$$(s_t, a_t, r_{t+1}, s_{t+1})$$

- Msl is a reservoir that stores the best behaviour in the form of: [s(t), a(t)] state and action at time step t.

strategy, β = ε-greedy(Q), which selects a random action with probability ε , and the action that maximizes Q-value with probability (1-ε).

- Π(s, a | θ(Π)), is a neural network that maps states to action probabilities and defines the agent's average strategy, π = Π. In other words, it tries to reproduce the best response behaviour using supervised learning from its history of previous best response behaviour in Msl.

So now we have two Neural Networks which predict actions.
Q(s, a | θ(Q)) by using the past experience, the other Π(s, a | θ(Π)) by pulling from a history of best responses. But the agent has to use one action at each turn, and has to choose which of these networks to use an action from.
In principle, the agent should play using it is average strategy Π, and uses Q to generate best response that will be fed into Msl to train its average policy network, Π(s, a | θ(Π)).
However there is a problem! How can Q generate experience from its own action? Remember that Q generates β = ε-greedy(Q) and feeds it to Π(s, a | θ(Π)) which will train its weights θ(Π) using supervised learning. So the action sampled from Π is not β, which is problematic for the experience of Q because it does not observe the result of its immediate action.

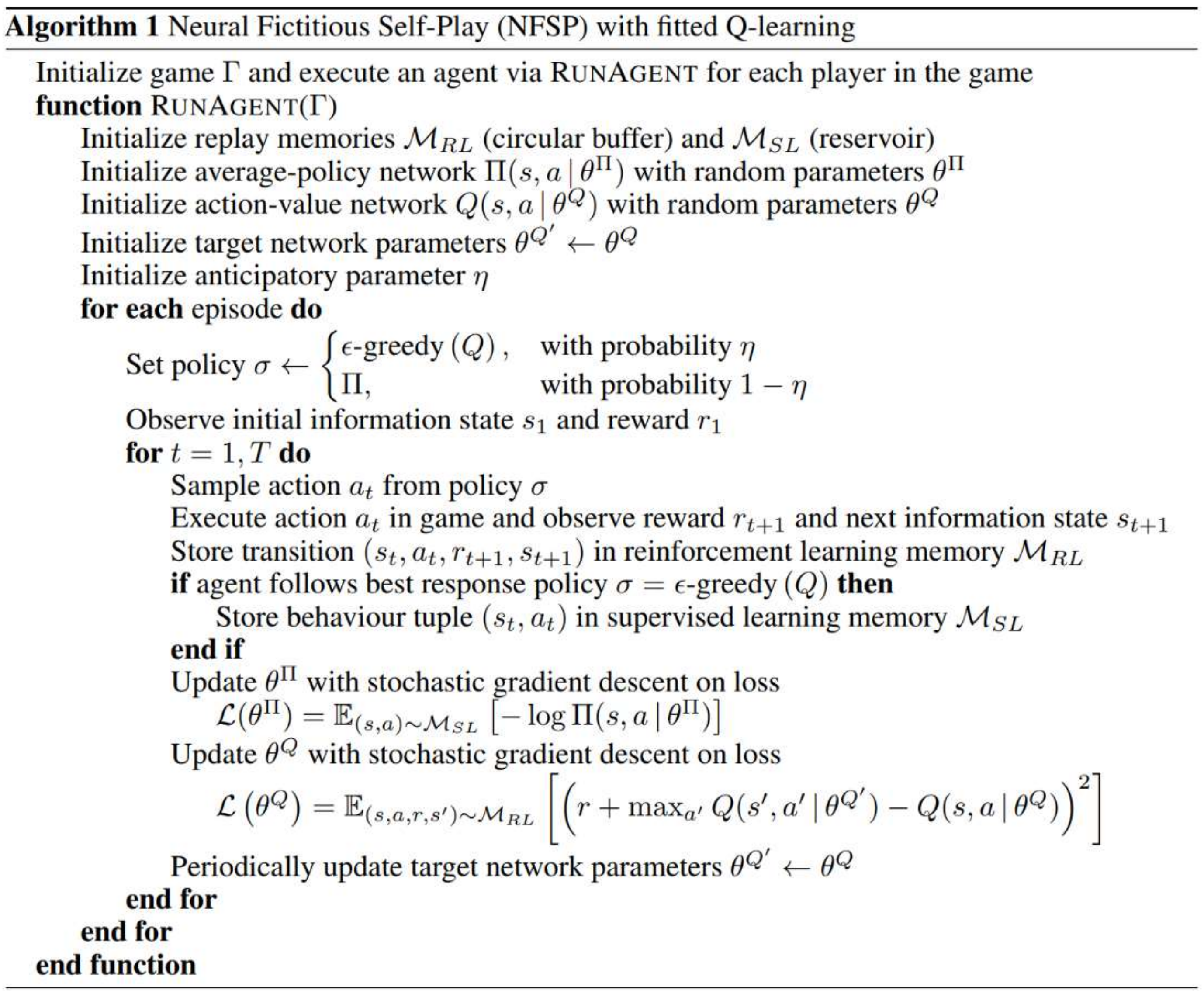
It appears that the best way to solve this issue is by using anticipatory parameter η such that it uses ε-greedy(Q) with probability η and the Π with probability (1-η).

The final algorithm main steps are like the following:

- define policy σ such that it is ε-greedy(Q) with probability η, and the Π with probability (1-η).
- get action from σ, execute it in the game, observe the the reward r(t+1) and next state s(t+1) and the transition [s(t), a(t), r(t+1), s(t+1)] in Mrl.
- if the σ is ε-greedy(Q), the store [s(t), a(t)] in Msl.

- periodically update the target network weights θ(Q') ← θ(Q)

The detailed algorithm is as follows:



Algorithm 1 Neural Fictitious Self-Play (NFSP) with fitted Q-learning

```
Initialize game $\Gamma$ and execute an agent via RUNAGENT for each player in the game
function RUNAGENT($\Gamma$)
    Initialize replay memories $\mathcal{M}_{RL}$ (circular buffer) and $\mathcal{M}_{SL}$ (reservoir)
    Initialize average-policy network $\Pi(s, a \mid \theta^{\Pi})$ with random parameters $\theta^{\Pi}$
    Initialize action-value network $Q(s, a \mid \theta^{Q})$ with random parameters $\theta^{Q}$
    Initialize target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
    Initialize anticipatory parameter $\eta$
    for each episode do
        Set policy $\sigma \leftarrow \begin{cases} \epsilon\text{-greedy}(Q), & \text{with probability } \eta \\ \Pi, & \text{with probability } 1-\eta \end{cases}$
        Observe initial information state $s_1$ and reward $r_1$
        for $t = 1, T$ do
            Sample action $a_t$ from policy $\sigma$
            Execute action $a_t$ in game and observe reward $r_{t+1}$ and next information state $s_{t+1}$
            Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in reinforcement learning memory $\mathcal{M}_{RL}$
            if agent follows best response policy $\sigma = \epsilon\text{-greedy}(Q)$ then
                Store behaviour tuple $(s_t, a_t)$ in supervised learning memory $\mathcal{M}_{SL}$
            end if
            Update $\theta^{\Pi}$ with stochastic gradient descent on loss
                $\mathcal{L}(\theta^{\Pi}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\left[-\log \Pi(s, a \mid \theta^{\Pi})\right]$
            Update $\theta^{Q}$ with stochastic gradient descent on loss
                $\mathcal{L}(\theta^{Q}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}_{RL}}\left[\left(r + \max_{a'} Q(s', a' \mid \theta^{Q'}) - Q(s, a \mid \theta^{Q})\right)^2\right]$
            Periodically update target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
        end for
    end for
end function
```

Conclusion

Heinrich & Silver argue that NFSP, is the first end-to-end deep reinforcement learning approach that approximates Nash equilibria of imperfect-information games from self-play. NFSP is scalable without prior domain knowledge. They showed that NFSP converges reliably to approximate Nash equilibria in a small poker game, whereas DQN's greedy and average strategies did not. NFSP learned a strategy that is

Heinrich & Silver.

"Anticipatory Learning in General Evolutionary Games" by Gurdal Arslan and Jeff S. Shamma

"Perfect information" - Wikipedia

"Reservoir sampling"- Wikipedia