Two-Timescale Algorithms for Learning Nash Equilibria in General-Sum Stochastic Games

H.L. Prasad†, Prashanth L.A.♯ and Shalabh Bhatnagar♯

†Streamoid Technologies, Inc  ♯Indian Institute of Science
Multi-agent RL setting

\[ \text{Reward } r = \langle r^1, r^2, \ldots, r^N \rangle, \]
\[ \text{next state } y \]

\[ \text{Action } a = \langle a^1, a^2, \ldots, a^N \rangle \]
Problem area

Normal-form Games

\((N, \mathcal{A}, r),\)
\(N\)-agents

Markov Decision Processes

\((\mathcal{S}, \mathcal{A}, p, r, \beta),\)
\(N\)-agents

Markov Chains

\((\mathcal{S}, p),\)

single agent

Stochastic Games

\((N, \mathcal{S}, \mathcal{A}, p, r, \beta),\)
\(N\)-agents
Problem area (revisited)

Design Objective:
- Online algorithm,
- Convergence to Nash equilibrium

1 If NE is a useful objective for learning in games, then we have a strong contribution!
A General Optimization Problem
Value function

\[ v^\pi(s) = E \left[ \sum_t \beta^t \sum_{a \in A(x)} r(s_t, a) \, \pi(s_t, a) \mid s_0 = s \right] \]

A stationary Markov strategy \( \pi^* = \langle \pi^1, \pi^2, \ldots, \pi^N \rangle \) is said to be Nash if

\[ v^i_{\pi^*}(s) \geq v^i_{\langle \pi^i, \pi^{-i}^* \rangle}(s), \forall \pi^i, \forall i, \forall s \in S \]
A stationary Markov strategy \( \pi^* = \langle \pi_1^*, \pi_2^*, \ldots, \pi_N^* \rangle \) is said to be Nash if

\[
v^i_{\pi^*}(s) \geq v^i_{\langle \pi^i, \pi^{-i*} \rangle}(s), \forall \pi^i, \forall i, \forall s \in S
\]
A stationary Markov strategy $\pi^* = \langle \pi^*_1, \pi^*_2, \ldots, \pi^*_N \rangle$ is said to be Nash if

$$v^{\pi^*_i}(s) \geq v^{\langle \pi^*_i, \pi^{\neg i} \rangle_i}(s), \forall \pi^*_i, \forall i, \forall s \in S$$
A stationary Markov strategy $\pi^* = \langle \pi_1^*, \pi_2^*, \ldots, \pi_N^* \rangle$ is said to be Nash if

$$v_{\pi^*}^i(s) \geq v_{\langle \pi^i, \pi^{-i}^* \rangle}^i(s), \forall \pi^i, \forall i, \forall s \in S$$
A stationary Markov strategy \( \pi^* = \langle \pi^1, \pi^2, \ldots, \pi^N \rangle \) is said to be Nash if

\[
v^i_{\pi^*}(s) \geq v^i_{\langle \pi^i, \pi^{-i*} \rangle}(s), \forall \pi^i, \forall i, \forall s \in S
\]
Dynamic Programming Idea

\[ v_{\pi^*}^i(x) = \max_{\pi^i(x) \in \Delta(A^i(x))} \left\{ E_{\pi^i(x)} Q_{\pi-i^*}^i(x, a^i) \right\}, \]

Optimal (Nash) Value

Marginal Value after fixing \( a^i \sim \pi^i \)

where Q-value is given by

\[ Q_{\pi-i^*}^i(x, a^i) = E_{\pi-i}(x) \left[ r^i(x, a) + \beta \sum_{y \in U(x)} p(y|x, a) v^i(y) \right] \]
Dynamic Programming Idea

\[ v^i_{\pi^*}(x) = \max_{\pi^i(x) \in \Delta(A^i(x))} \left\{ E_{\pi^i(x)} Q^i_{\pi - i^*}(x, a^i) \right\}, \]

Optimal (Nash) Value

Marginal Value after fixing \( a^i \sim \pi^i \)

where Q-value is given by

\[ Q^i_{\pi - i}(x, a^i) = E_{\pi - i(x)} \left[ r^i(x, a) + \beta \sum_{y \in U(x)} p(y|x, a)v^i(y) \right] \]
Dynamic Programming Idea

\[ v_{\pi^*}(x) = \max_{\pi^i(x) \in \Delta(A^i(x))} \left\{ E_{\pi^i(x)} Q_{\pi^i-\pi^*}^i(x, a^i) \right\}, \]

Optimal (Nash) Value

Marginal Value after fixing \( a^i \sim \pi^i \)

where Q-value is given by

\[ Q_{\pi^i-\pi^*}^i(x, a^i) = E_{\pi^i-\pi^*}(x) \left[ r^i(x, a) + \beta \sum_{y \in U(x)} p(y|x, a) v^i(y) \right] \]
Dynamic Programming Idea

\[ v^i_{\pi^*}(x) = \max_{\pi^i(x) \in \Delta(A^i(x))} \left\{ E_{\pi^i(x)} Q^{i}_{\pi - i^*}(x, a^i) \right\}, \]

Optimal (Nash) Value

Marginal Value after fixing \( a^i \sim \pi^i \)

where Q-value is given by

\[ Q^i_{\pi - i}(x, a^i) = E_{\pi - i}(x) \left[ r^i(x, a) + \beta \sum_{y \in U(x)} p(y|x, a) v^i(y) \right] \]
Optimization problem - informal terms

Need to solve:

\[ v^i_{\pi^*}(x) = \max_{\pi^i(x) \in \Delta(A^i(x))} \left\{ E_{\pi^i}(x)Q^i_{\pi-i^*}(x, a^i) \right\} \]  \hspace{1cm} (1)

Formulation:

Objective. minimize the Bellman error \( v^i(x) - E_{\pi^i}Q^i_{\pi-i}(x, a^i) \) in every state, for every agent

Constraint 1. ensure policy \( \pi \) is a distribution

Constraint 2. \( Q^i_{\pi-i}(x, a^i) \leq v^i_{\pi}(x) \leftarrow \) a proxy for the max in (1)
Optimization problem in formal terms

\[
\min_{v, \pi} f(v, \pi) = \sum_{i=1}^{N} \sum_{x \in S} (v^i(x) - E\pi Q^i_{\pi-i}(x, a^i))
\]

subject to

\[
\pi^i(x, a^i) \geq 0, \forall a^i \in A^i(x), x \in S, i = 1, 2, \ldots, N,
\]

\[
\sum_{i=1}^{N} \pi^i(x, a^i) = 1, \forall x \in S, i = 1, 2, \ldots, N.
\]

\[
Q^i_{\pi-i}(x, a^i) \leq v^i(x), \forall a^i \in A^i(x), x \in S, i = 1, 2, \ldots, N.
\]
Solution approach

**Usual approach:** Apply KKT conditions to solve the general optimization problem

**Caveat:** Imposes a tricky linear independence requirement

**Alternative:** Use a simpler set of SG-SP conditions
A sufficient condition

**SG-SP Point** A point \((v^*, \pi^*)\) is said to be an SG-SP point if it is feasible and for all \(x \in \mathcal{X}\) and \(i \in \{1, 2, \ldots, N\}\)

\[
\pi^i(x, a^i) g^i_{x,a^i}(v^*, \pi^{-i}^*(x)) = 0, \quad \forall a^i \in A^i(x)
\]

where \(g^i_{x,a^i}(v^i, \pi^{-i}(x)) := Q^i_{\pi^{-i}}(x, a^i) - v^i(x)\).

**Nash ⇔ SG-SP:**

A strategy \(\pi^*\) is Nash if and only if \((v^*, \pi^*)\) is an SG-SP point.
An Online Algorithm: ON-SGSP
ON-SGSP’s decentralized online learning model
ON-SGSP - operational flow

Policy evaluation: estimate the value function using temporal difference (TD) learning

Policy improvement: perform gradient descent for the policy using a descent direction

Descent direction ensures convergence to a global minimum of the optimization problem
Policy evaluation: estimate the value function using temporal difference (TD) learning

Policy improvement: perform gradient descent for the policy using a descent direction

Descent direction ensures convergence to a global minimum of the optimization problem
More on the descent direction

Descend along
\[- \sqrt{\pi^i(x, a^i)} \left| g_{x,a}^i(v^i, \pi^{-i}) \right| \times \text{sgn} \left( \frac{\partial f(v, \pi)}{\partial \pi^i} \right)\]

From
Lagrange multiplier and slack variable theory

Solution tracks an ODE with limit as an SG-SP point

1 \text{sgn} is a continuous version of \text{sgn}
Experiments
A single state non-generic 2-player game

**Payoff Matrix**

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Player 2 →</th>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>( a_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>1, 0</td>
<td>0, 1</td>
<td>1, 0</td>
<td></td>
</tr>
<tr>
<td>( a_2 )</td>
<td>0, 1</td>
<td>1, 0</td>
<td>1, 0</td>
<td></td>
</tr>
<tr>
<td>( a_3 )</td>
<td>0, 1</td>
<td>0, 1</td>
<td>1, 1</td>
<td></td>
</tr>
</tbody>
</table>
## Results from 100 simulation runs

<table>
<thead>
<tr>
<th></th>
<th>NashQ</th>
<th>FFQ (Friend Q)</th>
<th>ON-SGSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oscillate or converge to non-Nash strategy</td>
<td>95%</td>
<td>40%</td>
<td>0%</td>
</tr>
<tr>
<td>Converge to (0.5, 0.5, 0)</td>
<td>2%</td>
<td>0%</td>
<td>99%</td>
</tr>
<tr>
<td>Converge to (0, 0, 1)</td>
<td>3%</td>
<td>60%</td>
<td>1%</td>
</tr>
</tbody>
</table>
Stick-Together Game

Figure: Stick Together Game for $M = 3$

For $M = 30$, STG has 810000 states!
Results for STG with $M = 30$

ON-SGSP takes agents to within a $4 \times 4$-grid, while NashQ/FFQ to a $8 \times 8$-grid.

Foe Q-learning/NashQ have higher per-iteration complexity than ON-SGSP.
Thank You!