

Computational Intelligence in Games - Cheat Sheet -

Alexander Dockhorn

May 8, 2018

This short overview tries to provide you a guide for all equations and algorithms mentioned in the lecture Computational Intelligence in Games at the Otto von Guericke University Magdeburg. It can be quite hard to understand all the little details involved without being sure about the symbols used. We hope this guide helps you during your review of the course topics.

I currently do not recommend printing this guide, because it is the first version of this kind of overview. Even if I reviewed the contents multiple times, errors are still likely. Please always refer to the actual version in the Git-Repository.

In case you have any ideas how to improve this guide please let us know by writing an e-mail to: dockhorn@ovgu.de.

Contents

1	(Evolutionary) Game Theory	4
1.1	Basics in Game Theory	4
1.2	General 2-Player Games	4
1.3	Nash Equilibria	5
1.4	Evolutionary Game Theory	5
1.5	Additional Information on Evolutionary Game Theory	6
2	Reinforcement Learning	8
2.1	General Notation	8
2.2	Value and Action-Value Functions	10
2.3	Dynamic Programming	11
2.4	Monte Carlo Method	12
2.5	Temporal Difference Learning	14

List of Equations

1	Expected payoff of pure strategy s_i , $E(s_i, y)$	5
2	Expected payoff of mixed strategy x , $E(x, y)$	5
3	Fitness of strategy s_i , $f_i(x)$	6
4	Average population fitness, $\Phi(x)$	6
5	Replicator Equation for infinite populations, \dot{x}	6
6	Weak Selection, \dot{x}	7
7	Probability for replacing a strategy using weak selection ρ	7
8	Comparison of using T,R,S,P or b,c for describing a payoff matrix	7
9	Simple (undiscounted) Return, G_t	9
10	Discounted Return, G_t	9
11	Expected Reward, $r(s, a)$	9
12	Expected Reward, $r(s, a)$	9
13	Value Function, $v(s)$	10
14	Consistency Condition of the Value Function, $v(s)$	10
15	Action-Value Function, $q(s, a)$	10
16	Consistency Condition of the Action-Value Function, $q(s, a)$	10
17	Derivation of the Incremental Mean and Incremental Monte Carlo, $\mu_k, V(s)$	13
18	n-Step Return, $G_t^{t+n}(s)$	14

List of Algorithms

1	Iterative Policy Evaluation	11
2	Policy Iteration	11
3	Value Iteration	11
4	Monte Carlo Method $v(s, a)$	12
5	Monte Carlo Method $q(s, a)$	12
6	Constant- α Monte Carlo Method $v(s, a)$	13
7	Temporal Difference Learning	14
8	One Step Q-Learning	15

1 (Evolutionary) Game Theory

1.1 Basics in Game Theory

Symbol	Name	Description
N	number of agents	the number of agents in an N-player game
S_i	strategies of player i	each player has a set of strategies, which he can choose to play
s_i	chosen strategy of player i	
m_i	number of strategies in S_i	
π_i	payoff function of player i	provides a reward to agent i after each agent chose his action
A	payoff matrix player 1 (or both)	represents the payoff matrix of the first player, or of both players in case the payoff is symmetric
B	payoff matrix player 2	

1.2 General 2-Player Games

Symbol	Name	Description
C	strategy "Cooperate"	often used in standard literature to describe cooperative behavior, which comes with additional cost for the playing agent
D	strategy "Defect"	often used in standard literature to describe cooperative behavior, which comes without a cost for the playing agent
T	temptation	Reward for defecting, when the other player is Cooperating
R	reward for mutual cooperation	reward in case both players are cooperating
S	suckers payoff	reward for the player who cooperated against a player who defected him
P	punishment	reward for the player when both players chose defect

1.3 Nash Equilibria

Symbol	Name	Description
\mathcal{S}	strategy profile	the chosen strategies per player
x	mixed strategy	probability distribution on all possible strategies in S_i
$E(s_i, y)$	expected payoff of a pure strategy	expected payoff of pure strategy s_i against mixed strategy y . See Equation (1)
$E(x, y)$	expected payoff of a mixed strategy	expected payoff of mixed strategy x against mixed strategy y . See Equation (2)

Expected payoff of pure strategy s_i , $E(s_i, y)$

$$A = \begin{bmatrix} \pi_{s_1, s'_1} & \pi_{s_1, s'_2} \\ \pi_{s_2, s'_1} & \pi_{s_2, s'_2} \end{bmatrix}, \quad E(s_i, y) = \sum_{j=1}^{|S_j|} \pi_{s_i, s'_j} y_j \quad (1)$$

Expected payoff of mixed strategy x , $E(x, y)$

$$A = \begin{bmatrix} \pi_{s_1, s'_1} & \pi_{s_1, s'_2} \\ \pi_{s_2, s'_1} & \pi_{s_2, s'_2} \end{bmatrix}, \quad E(x, y) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \pi_{s_1, s'_1} & \pi_{s_1, s'_2} \\ \pi_{s_2, s'_1} & \pi_{s_2, s'_2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (2)$$

1.4 Evolutionary Game Theory

Symbol	Name	Description
x	frequency of strategies	similar to the mixed strategy, but here the population frequency is given by x
s_i	strategy with index i	index corresponds to the frequency vector x
$f_i(x)$	fitness of strategy s_i	See Equation (3)
$\Phi(x)$	average population fitness	See Equation (4)
\dot{x}	replicator equation of x	shows the time and fitness dependent development of each strategies frequency, is computed for every s_i separately, See Equation (5)

Evolutionary Stable Strategy: A strategy σ is an evolutionary stable strategy if:

- $\pi_{\sigma,\sigma} > \pi_{\tau,\sigma}$ for all possible mutant strategies τ
- or, $\pi_{\sigma,\sigma} = \pi_{\tau,\sigma}$ and $\pi_{\sigma,\tau} > \pi_{\tau,\tau}$

Fitness of strategy s_i , $f_i(x)$

$$fitness(s_i) = f_i(x) = payoff(s_i) = \sum_{j=1}^n x_j \pi_{ij} \quad (3)$$

Average population fitness, $\Phi(x)$

$$\Phi(x) = \sum_{j=1}^n x_j f_j(x) \quad (4)$$

Replicator Equation for infinite populations, \dot{x}

$$\dot{x} = x_i [f_i(x) - \Phi(x)] \quad (5)$$

There seems to be something wrong with our current definition of replicator equations for finite populations. Please ignore this part for now! We will update the slides and this overview after a careful revision of this part.

1.5 Additional Information on Evolutionary Game Theory

Symbol	Name	Description
β	selection intensity	used to apply weak selection, See Equation (6)
ρ	fixation probability	probability that a mutant strategy can take over the whole population, See Equation (7)
Δf	fitness difference of two strategies	
b	benefit	benefit for playing with a cooperating player, See Equation (8)
c	cost	cost for playing cooperation

Weak Selection, \dot{x}

$$f_A(i) = 1 - \beta + \beta \pi_A(i)$$

$$\beta = 0 \Rightarrow \text{neutral selection} \quad (6)$$

$$\beta \ll 1 \Rightarrow \text{weak selection}$$

$$\beta = 1 \Rightarrow \text{selection by payoff}$$

Probability for replacing a strategy using weak selection ρ

$$\rho = \frac{1}{1 + \exp^{-\beta \Delta f}} \quad (7)$$

Comparison of using T,R,S,P or b,c for describing a payoff matrix

$$\begin{array}{cc} C' & D' \\ C & \begin{bmatrix} b-c & -c \end{bmatrix} \\ D & \begin{bmatrix} b & 0 \end{bmatrix} \end{array} \quad \begin{array}{cc} C' & D' \\ C & \begin{bmatrix} R & S \end{bmatrix} \\ D & \begin{bmatrix} T & P \end{bmatrix} \end{array} \quad (8)$$

2 Reinforcement Learning

2.1 General Notation

Symbol	Name	Description
t	timestep $0 \dots T$	
T	final timestep T	
\mathcal{S}	set of possible states	
s_t	state at time t	$s_t \in \mathcal{S}$, the t is dropped in case the equation is independent of the actual time
$\mathcal{A}(s_t)$	available actions in s_t	
a_t	action at time t	$a_t \in \mathcal{A}(s_t)$, the t is dropped in case the equation is independent of the actual time
r_t or R_t	reward at time t	after the agent left state s_t by executing action a_t , he receives the reward r_{t+1} , where $r_{t+1} \in \mathbb{R}$, depending on the context we use upper-case R to not confuse the single reward with the probability distribution at the bottom of this table
π	policy function	a probability distribution on the actions depending on the state
$\pi(a s)$	policy	probability of choosing action a if in state s
G_t, G_s	(expected) return	sum of rewards from s or timestep t till the end T (undiscounted return, See Equation (9)) or a non-ending episode (discounted return, See Equation (10))
γ	discount rate	affects the influence of future rewards on the return calculation (discounted return, See Equation (10))
$p(s', r s, a)$	state and reward probability depending on a and s	also known as the markov property, probability of transitioning to state s' and receiving reward r is only dependent on choosing action a in state s ,
$p(s' s, a)$	transition probability	probability of transitioning into s' after the agent executed its action a in state s
$r(s, a)$	expected reward	valid for all Markov Decision Processes, See Equation (11)
$r(s, a, s')$	expected reward after the follow-up state is known	valid for all Markov Decision Processes, See Equation (12)

Simple (undiscounted) Return, G_t

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T \quad (9)$$

Discounted Return, G_t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^T \gamma^k R_{t+k+1}, \quad \gamma \in [0, 1] \quad (10)$$

Expected Reward, $r(s, a)$

$$r(s, a) = \mathbb{E}[R_{t+1} | s_t = s, a_t = a] = \sum_{r \in \mathbb{R}} \sum_{s' \in \mathbb{S}} r(s, a, s') \cdot p(s' | s, a) \quad (11)$$

Expected Reward, $r(s, a)$

$$r(s, a, s') = \mathbb{E}[R_{t+1} | s_t = s, a_t = a, s_{t+1} = s'] = \frac{\sum_{r \in \mathbb{R}} r \cdot p(s', r | s, a)}{p(s' | s, a)} \quad (12)$$

2.2 Value and Action-Value Functions

Symbol	Name	Description
$v_\pi(s)$	value function	rates the value/expected return of a state following policy π , See Equation (13) and ??
π^*	optimal policy	policy with the best possible expected return for all states
v^*	optimal value function	value function of the optimal policy π^*
$q_\pi(s, a)$	action-value function	rates the value/expected return of choosing an action in a given state following policy π , See Equation (15) and Equation (16)
$V(s)$	approximation of $v(s)$	used in multiple iterative algorithms
$Q(s, a)$	approximation of $q(s, a)$	used in multiple iterative algorithms

Value Function, $v(s)$

$$v_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right] \quad (13)$$

Consistency Condition of the Value Function, $v(s)$

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | s_t = s] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')] \end{aligned} \quad (14)$$

Action-Value Function, $q(s, a)$

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right] \quad (15)$$

Consistency Condition of the Action-Value Function, $q(s, a)$

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')] \end{aligned} \quad (16)$$

2.3 Dynamic Programming

Algorithm 1: Iterative Policy Evaluation

Input: policy π to be evaluated

Initialize an array $V(s) = 0$, for all $s \in S$

repeat

$\Delta \leftarrow 0$

foreach $s \in S$ **do**

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s, r|s, a)[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (*a small positive number*)

Output: $V \approx v_\pi$

Algorithm 2: Policy Iteration

Input: policy π to be evaluated

Initialize an array $V(s) = 0$, for all $s \in S$

repeat

 apply Iterative Policy Evaluation given π

$policy_stable \leftarrow true$

foreach $s \in S$ **do**

$a \leftarrow \pi(s)$

$\pi(s) \leftarrow \arg \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

If $a \neq \pi(s)$ **then** $policy_stable \leftarrow false$

until $policy_stable$

Output: $V \approx v^*$, $\pi \approx \pi^*$

Algorithm 3: Value Iteration

Initialize an array $V(s) = 0$, for all $s \in S$

repeat

$\Delta \leftarrow 0$

foreach $s \in S$ **do**

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta \leftarrow 0$ (*a small positive number*)

Output: a deterministic policy π such that

$\pi(s) = \arg \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

2.4 Monte Carlo Method

Algorithm 4: Monte Carlo Method $v(s, a)$

Input: policy π to be evaluated

Initialize an array $V(s) = 0$, for all $s \in S$

Initialize an empty list $Returns(s)$, for all $s \in S$

while *true* **do**

 Generate an episode using π

foreach *state s appearing in the episode* **do**

$G \leftarrow$ collected return after the first occurrence of s

$Returns(s).append(G)$

$V(s) \leftarrow average(Returns(s))$

Output: $V \approx v_\pi$

Algorithm 5: Monte Carlo Method $q(s, a)$

Input: policy π to be evaluated

Initialize an array $V(s) = 0$, for all $s \in S$

Initialize an empty list $Returns(s, a)$, for all $s \in S$ and $a \in A$

while *true* **do**

 Choose $s_0 \in S$ and $a_0 \in A(s_0)$

 Generate an episode starting from s_0, a_0 following π

foreach *(s, a) in the episode* **do**

$G \leftarrow$ return following the first occurrence of (s, a)

$Returns(s, a).append(G)$

$Q(s, a) \leftarrow \arg \max_a Q(s, a)$

foreach *s in the episode* **do**

$\pi(s) \leftarrow \arg \max_a Q(s, a)$

Output: $Q \approx q_\pi, \quad \pi \approx \pi^*$

Derivation of the Incremental Mean and Incremental Monte Carlo, $\mu_k, V(s)$

$$\begin{aligned}
\mu_k &= \frac{1}{k} \sum_{j=1}^k x_j \\
&= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\
&= \frac{1}{k} (x_k + (k-1)\mu_{k-1}) \\
&= \mu_{k-1} \frac{1}{k} (x_k - \mu_{k-1})
\end{aligned} \tag{17}$$

$$\begin{aligned}
V(s) &\leftarrow \frac{1}{k} \sum_{i=1}^k G_s(i) \\
&\leftarrow \frac{1}{k} \left(G_s(k) + \sum_{i=1}^{k-1} G_s(i) \right) \\
&\leftarrow \frac{1}{k} (G_s(k) + (k-1) G_s(k-1)) \\
&\leftarrow G_s(k-1) + \frac{1}{k} (G_s(k) - G_s(k-1)) \\
&\leftarrow V(s) + \frac{1}{k} (G_s(k) - V(s))
\end{aligned} \tag{18}$$

Algorithm 6: Constant- α Monte Carlo Method $v(s, a)$

Input: policy π to be evaluatedInitialize an array $V(s) = 0$, for all $s \in S$ Initialize an empty list $Returns(s)$, for all $s \in S$ **while** *true* **do** Generate an episode using π **foreach** *state s appearing in the episode* **do** $G_s \leftarrow$ collected return after the first occurrence of s $V(s) \leftarrow V(s) + \alpha[G_s - V(s)]$ **Output:** $V \approx v_\pi$

2.5 Temporal Difference Learning

Symbol	Name	Description
G_t^{t+n}	n-step return	See Equation (19)
h	horizon $= t + n$	number of future reward steps used for return calculation, if $h > T$ the return is the sum of rewards till the end of the episode
V_t	approximation of v at timestep t	See Equation (19)
$\Delta_t(s_t)$	error based adaptation	Weighted error used in temporal difference learning, See Equation (19)

n-Step Return, $G_t^{t+n}(s)$

$$G_t^{t+n}(s) = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_h + \gamma^n V_t(s_{t+n})$$

$$\Delta_t(s_t) = \alpha [G_t^{t+n}(V_t(s_{t+n})) - V_t(s_t)] \quad (19)$$

$$V_{t+1}(s) = V_t(s) + \Delta_t(s), \quad \forall s \in \mathcal{S}$$

Algorithm 7: Temporal Difference Learning

Input: policy π to be evaluated

Initialize an array $V(s)$ arbitrarily, for all $s \in S$

foreach for each episode **do**

Initialize S

foreach $s \in S$ **do**

$A \leftarrow$ action given by π for S

Take action A ; observe reward R , and next state, S'

$V(S) \leftarrow V(S) + \alpha [R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

Output: $V \approx v_\pi$

Algorithm 8: One Step Q-Learning

Initialize a matrix $Q(s, a)$ arbitrarily, for all $s \in S$, and $a \in A$

foreach *terminal state* s **do** $Q(s, \cdot) = 0$

foreach *episode* **do**

foreach *state* s_t *in episode* **do**

 Choose a_t from s_t using policy derived from Q

 take action a_t , observe reward r , and follow-up state s_{t+1}

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$

$s \leftarrow s_{t+1}$

Output: $Q(s_t, a_t)$
